



Documentação módulo estratégias

Todos os tópicos



Sumário

1. **Introdução**
2. **Estrutura de uma estratégia**
3. **Fluxo de execução de uma estratégia**
4. **Variáveis, tipos de dados e constantes**
5. **Controle de fluxo**
6. **Operadores**
7. **Funções**
8. **Funções de biblioteca**
9. **Back-Testing**
10. **Abrir estratégia**
11. **Gerenciador de estratégias**
12. **Exportar/importar estratégia**
13. **Criar regra de coloração**
14. **Criar regra de alarme**
15. **Criar regra de execução**
16. **Screening**
17. **Inserir regra de coloração**
18. **Condições de coloração**
19. **Editor de estratégias**
20. **Propriedades do editor de estratégias**
21. **Nova estratégia**
22. **Lista de funções**
 - 22.1 **Alarme**
 - Alert(Cor : Integer)
 - 22.2 **Back-Testing**
 - BuyAtMarket
 - BuyLimit(Preco : Float)
 - BuyPosition
 - BuyPrice
 - BuyStop(Stop : Float, Limite : Float)
 - BuyToCoverAtMarket
 - BuyToCoverLimit(Preco : Float)
 - BuyToCoverStop(Stop : Float, Limite : Float)
 - CancelPendingOrders
 - ClosePosition



HasPendingOrders
IsBought
IsSold
ReversePosition
SellPosition
SellPrice
SellToCoverAtMarket
SellToCoverLimit(Preco : Float)
SellToCoverStop(Stop : Float, Limite : Float)
SellShortAtMarket
SellShortLimit(Preco : Float)
SellShortStop(Stop : Float, Limite : Float)
SendOrder(Lado : Integer, Tipo : Integer, Quantidade : Integer, Limite : Float, Stop : Float)

22.3 Calendário

BarDuration
CalcDate(DataReferencia : Integer, DiasDeslocamento : Integer)
CalcTime(HoraReferencia : Integer, MinutosDeslocamento : Integer)
CloseD(QuantidadeDiasAnteriores : Integer)
CloseM(QuantidadeMesesAnteriores : Integer)
CloseW(QuantidadeSemanasAnteriores : Integer)
CloseY(QuantidadeAnosAnteriores : Integer)
CurrentDate
CurrentTime
Date
DayOfMonth(Data : Integer)
DayOfWeek(Data : Integer)
DaysToExpiration(Mes : Integer, Ano : Integer)
ELDate(Ano : Integer, Mes : Integer, Dia : Integer)
ELDate_Consol(Data : Integer)
FindBar(Data : Integer, Hora : Integer)
Friday
HighD(QuantidadeDiasAnteriores : Integer)
HighM(QuantidadeMesesAnteriores : Integer)
HighW(QuantidadeSemanasAnteriores : Integer)
HighY(QuantidadeAnosAnteriores : Integer)
LowD(QuantidadeDiasAnteriores : Integer)
LowM(QuantidadeMesesAnteriores : Integer)



LowW(QuantidadeSemanasAnteriores : Integer)

LowY(QuantidadeAnosAnteriores : Integer)

LastCalcDate

LastCalcTime

LastDayOfMonth(MesReferencia : Integer)

Monday

Month(Date : Integer)

Next3rdFriday(Mes : Integer)

OpenD(QuantidadeDiasAnteriores : Integer)

OpenM(QuantidadeMesesAnteriores : Integer)

OpenW(QuantidadeSemanasAnteriores : Integer)

OpenY(QuantidadeAnosAnteriores : Integer)

RS_BarsPerDay

Saturday

Sunday

Thursday

BarAnnualization

Bartype

Time

TimeToMinutes(Hora : Integer)

Tuesday

VolumeD(QuantidadeDiasAnteriores : Integer)

VolumeM(QuantidadeMesesAnteriores : Integer)

VolumeW(QuantidadeSemanasAnteriores : Integer)

VolumeY(QuantidadeAnosAnteriores : Integer)

Wednesday

Year(Date : Integer)

22.4 **Candlestick**

C_3WhSolds_3BlkCrows(Comprimento : Integer, Fator : Integer, var o3WhiteSoldiers : Integer, var o3BlackCrows : Integer)

C_BullEng_BearEng(Comprimento : Integer, var oBullishEngulfing: Integer, var oBearishEngulfing : Integer)

C_Doji(Percentual : Integer)

C_Hammer_HangingMan(Comprimento : Integer, Fator : Integer, var oHammer : Integer, var oHangingMan : Integer)

C_MornDoji_EveDoji(Comprimento : Integer, Percentual : Float, var oMorningDojiStar : Integer, var oEveningDojiStar : Integer)

C_MornStar_EveStar(Comprimento : Integer, var oMorningStar : Integer, var oEveningStar : Integer)



C_PierceLine_DkCloud(Comprimento : Integer, var oPiercingLine : Integer, var oDarkCloud : Integer)

C_ShootingStar(Comprimento : Integer, Fator : Integer)

22.5 Exemplos

DiMaisDiMenos(Periodo : Integer)

IFR(Periodo : Integer)

Media(Periodo : Integer, TipoSerie : Serie)

MediaExp(Periodo : Integer, TipoSerie : Serie)

PaintVar

WellesSum(Periodo : Integer, SerieReferencia : Serie, Offset : Integer)

22.6 Gráficas

AvgPrice

CurrentBar

GetPlotColor(NumeroPlot : Integer)

GetPlotWidth(NumeroPlot : Integer)

Highest(SerieDeDados : Serie, Periodo : Integer)

HighestBar(SerieDeDados : Serie, Periodo : Integer)

LastBarOnChart

Leader

MaxBarsForward

MaxBarsBack

NoPlot(NumeroPlot : Integer)

PaintBar(Cor : Integer)

Lowest(SerieDeDados : Serie, Periodo : Integer)

LowestBar(SerieDeDados : Serie, Periodo : Integer)

MedianPrice

Plot(Dado : Float)

Plot2(Dado : Float)

Plot3(Dado : Float)

Plot4(Dado : Float)

Range

RangeLeader

RateOfChange(SerieDados : Serie, Periodo : Integer)

RGB(Red : Integer, Green : Integer, Blue : Integer)

SetPlotColor(NumeroPlot : Integer, Cor : Integer)

SetPlotWidth(NumeroPlot : Integer, Espessura : Integer)

TrueHigh

TrueLow



TrueRange

TrueRangeCustom(Maxima : Float, Minima : Float, Fechamento : Float)

TypicalPrice

WeightedClose

22.7 Indicadores

AvgSeparation(Periodo : Integer, TipoMedia : Integer)

AvgTrueRange(Periodo : Integer, TipoMedia : Integer)

AccAgressSaldo(TipoVolume : Integer)

AccuDistr

AccuDistrW

AdaptiveMovingAverage(Periodo : Integer, FastSC : Integer, SlowSC : Integer)

ADX(Periodo : Integer, PeriodoMedia : Integer)

AgressionVolBalance

AgressionVolBuy

AgressionVolSell

ArmsEaseOfMov(Media : Integer, TipoMedia : Integer)

AroonLin(Periodo : Integer)|Linha Desejada|

AroonOsc(Periodo : Integer)

AvgAgrBuySell(AlertaVariacoes : Integer, TipoVolume : Integer, TipoDesenho: Integer)|Linha : Integer|

AvgAgrTotal(AlertaVariacoes : Integer, TipoVolume : Integer, TipoDesenho: Integer)|Linha : Integer|

BalanceOfPower(Media : Integer, TipoMedia : Integer)

BearPower(Periodo : Integer)

BollingerBands(Desvio : Float, Media : Integer, TipoMedia : Integer)|Linha : Integer|

BollingerBandW(Desvio : Float, Media : Integer, TipoMedia : Integer)

BollingerBPerc(Desvio : Float, Media : Integer, TipoMedia : Integer)

BullPower(Periodo : Integer, PeriodoMedia : Integer, TipoMedia : Integer)

Carmine(Risco : Integer, ModoCalculo : Integer, Período : Integer, Desvio : Float, UsarVWAP : Boolean, UsarAtr : Boolean)

CCI(Periodo : Integer)

ChaikinMoneyFlow(Periodo : Integer)

ChainSetup

ChaikinOsc(MediaLonga : Integer, MediaCurta : Integer)

DarvasBox|Linha : Integer|

DecisionPoints(Tipo : Integer, Linha : Integer)

DiDiIndex(MedRef : Integer, TipoMedRef : Integer, Med1 : Integer, TipoMed1 : Integer, Med2 : Integer, TipoMed2 : Integer)|Linha : Integer|

DiPDIM(Periodo : Integer)|Linha : Integer|



Documentação Módulo Estratégias

DonchianCH(Periodo : Integer)|Linha : Integer|

DTOscillator(PeriodoEstocastico : Integer, PeriodoSK : Integer, TipoSK : Integer, PeriodoSD : Integer, TipoSD : Integer)|Linha : Integer|

Envelope(Percentual : Float, PeriodoMedia : Integer, TipoMedia : Integer)|Linha : Integer|

Euroinvest(Risco: Integer, ModoCalculo : Integer, Periodo : Integer, Desvio : Float, UsarVWAP : Boolean, UsarAtr : Boolean)

FastStochastic(Periodo : Integer)

FinancialVol(VolumeProjetado : Boolean, Agressores : Boolean)

ForceIndex(Periodo : Integer, TipoMedia : Integer)

FrassonATR(Fator : Float, PeriodoMaxMin : Integer, PeriodoATR : Integer)|Linha : Integer|

FrassonVH(Fator : Float, PeriodoMaxMin : Integer, PeriodoVH : Integer)|Linha : Integer|

FullStochastic(Periodo : Integer)

FuraChao(Coeficiente : Float, Deslocamento : Integer)

FuraTeto(Coeficiente : Float, Deslocamento : Integer)

HeikinAshi(Media : Integer, TipoMedia : Integer)|Dado : Integer|

HiLoActivator(Periodo : Integer)|Linha : Integer|

HistVolatility(Periodo : Integer, TipoMedia : Integer)

HSI(Periodo : Integer)

HullMovingAverage(Periodo : Integer)

IchimokuCloud(TenkanSen : Integer, KijunSen : Integer, SenkouSpanB : Integer)|Linha : Integer|

IFR(Periodo : Integer)

ImpliedVolatility(ModeloTeorico : Boolean, TipoOpcao : Boolean)

KeltnerCH(Desvio : Float, Periodo : Integer, TipoMedia : Integer)|Linha : Integer|

KVO(MediaLonga : Integer, MediaCurta : Integer, Sinal : Integer)|Dado : Integer|

LSVolatilityIndex

MACD(MediaLonga : Integer, MediaCurta : Integer, Sinal : Integer)|Dado : Integer|

Media(Periodo : Integer, TipoSerie : Serie)

MediaExp(Periodo : Integer, TipoSerie : Serie)

MFI

MIMA(Periodo : Integer)

Momentum(Periodo : Integer, Media : Integer, TipoMedia : Integer)

MomentumStochastic(Periodo : Integer)

MoneyFlow

MoneyFlowIndex(Periodo : Integer)

NelogicaBottomFinder|Dado : Integer|

NelogicaPullBackFinder|Dado : Integer|

NelogicaWeisWave(Periodo : Integer)



Documentação Módulo Estratégias

OBV
OBVAvg
OnBalanceTR
OpenInterest
ParabolicSAR(Fator : Float, Limite : Float)
Phibo(Periodo : Integer)
Pivot(Normal : Boolean, TresLinhas : Boolean)|Linha : Integer|
PriceOsc(Media1 : Integer, TipoMedia1 : Integer, Media2 : Integer, TipoMedia2 : Integer)
PriceOscillator(SerieDados : Serie, ComprimentoRapido : Integer, ComprimentoLento : Integer)
PriceVolumeTrend
PriorCote(Dado : Integer)
PTAX
PTAXFuturo
QuantityVol(VolumeProjetado : Boolean, Agressores : Boolean)
Rafi
Ravi(MediaCurta : Integer, MediaLonga : Integer)
RenkoVTwo(Periodo : Integer, Abertura : Float, Deslocamento : Integer)
RSI(Periodo : Integer, Tipo : Integer)
RsiStochastic(Periodo : Integer)
ROC(Periodo : Integer, Media : Integer, TipoMedia : Integer)
SafeZoneDownTrend(Multiplicador : Float, Periodo : Integer, Deslocamento : Integer)
SafeZoneUpTrend(Multiplicador : Float, Periodo : Integer, Deslocamento : Integer)
Santo(Periodo : Integer)|Linha : Integer|
SlowStochastic(Periodo : Integer)
StopATR(Desvio : Float, Periodo : Integer, TipoMedia : Integer)|Dado : Integer|
Tilson(Fator : Float, Media : Integer)
TimeAgrBuySell(AlertaVariacoes : Integer)
TimeAgrTotal(AlertaVariacoes : Integer)
TRIX(Media : Integer, TipoMedia : Integer)
TRIXM(Media : Integer, TipoMedia : Integer)
TopBottomDetector(Periodo : Integer)
Trades
TwoMVAggression
TwoMVPower(Periodo1 : Integer, Periodo2 : Integer, Periodo3 : Integer, Media : Integer)
TwoMVStandard
VSS(Multiplicador : Float, Media : Integer, Deslocamento : Integer)
VWAP(Periodo : Integer)



VWAPMonthly

VWAPWeekly

VWMA(Periodo : Integer)

WAverage(TipoSerie : SeriePeriodo, Periodo : Integer)

WellesSum(Periodo : Integer, SerieReferencia : Serie, Offset : Integer)

Williams(Periodo : Integer)

xAverage(TipoSerie : SeriePeriodo, Periodo : Integer)

22.8 **Livro**

AskPrice

AskSize

BidPrice

BidSize

BookSpread

IsBMF

Lote

MinPriceIncrement

22.9 **Matemáticas**

ABS(Valor : Float)

Arctangent(Numero : Float)

Ceiling(Numero : Float)

Combination(Numero : Integer, QtdGrupos : Integer)

Cos(Valor : Float)

Cosine(Valor : Float)

Cotangent(Valor : Float)

Cum(SerieDeDados : Serie)

Exp(Valor : Float)

ExpValue(Valor : Float)

ExtremePriceRatio

Factorial(Valor : Float)

FastD(Periodo : Integer)

FastK(Periodo : Integer):

FastKCustom(PrecosH : Serie, PrecosL : Serie, PrecosC : Serie, Periodo : Integer)

Floor(Valor : Float)

FracPortion(Valor : Float)

GCD(Valor1 : Float, Valor2 : Float)

HarmonicMean(SerieDados : Serie, Periodo : Integer)

IntPortion(Valor : Float)



Log(Valor : Float)
MidPoint(SerieDados : Serie, Período : Integer)
MinutesIntoWeek(DiaLimite : Integer, HoraLimite : Integer)
MinutesToTime(Minutos : Integer)
Mod(Dividendo : Integer, Divisor : Integer)
MyPrice
Neg(Numero : Float)
NumUnits(Amnt : Integer, MinLot : Integer)
PercentChange(SerieDados : Serie, Período : Integer)
PercentR(Comprimento : Integer)
Permutation(Numero : Integer, NumeroObjetos : Integer)
Pos(Valor : Float)
Power(Base : Float, Expoente : Integer)
Random(Limite : Integer)
RateOfChange(SerieDados : Serie, Período : Integer)
Round(Valor : Float)
Round2Fraction(Valor : Float)
Sign(Valor : Float)
Sin(Valor : Float)
Sine(Valor : Float)
SlowD(Período : Integer)
SlowK(Período : Integer)
Sqrt(Valor : Float)
Square(Valor : Float)
StdDevs(SerieDados : Serie, Período : Integer)
Summation(SerieDados : Serie, Período : Integer)
Tangent(Valor : Float)
TriAverage(SerieDados : Serie, Período : Integer)
UlcerIndex(SerieDados : Serie, Período : Integer)
UltimateOscillator(PeríodoCurto : Integer, PeríodoMedio : Integer, PeríodoLongo : Integer)
Volatilidade(Período : Integer)
VolumeOsc(PeríodoMediaRapida : Integer, PeríodoMediaLenta : Integer)
VolumeROC(Período : Integer)

22.10 Opções

Delta(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)
Gamma(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)



Rho(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

Theta(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

Vega(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

22.11 **Screening**

Select



Introdução

A NTSL (Nelogica Trading System Language) é uma poderosa linguagem criada com um único propósito: permitir o desenvolvimento dos melhores e mais eficientes sistemas algorítmicos de operação. A Nelogica oferece juntamente com a NTSL um ambiente de criação pioneiro e revolucionário chamado AlgoTools. Com o AlgoTools é possível codificar, testar e simular com grande agilidade qualquer estratégia de operação.

Na NTSL, o usuário encontra uma grande facilidade no momento de criação de suas estratégias, a possibilidade de criar toda ela em português, havendo assim, a facilidade e fácil entendimento da estratégia criada.



Estrutura de uma Estratégia

Observe o trecho de código abaixo que representa o indicador média móvel. Esse código apresenta as três áreas que definem a estrutura de uma estratégia. São elas: área de declaração de parâmetros de entrada, área de declaração de variáveis e funções e área de código.

```
1  input
2    Preco(Close);
3    Periodo(2);
4
5  var
6    sResult : Float;
7    nIndex  : Integer;
8
9  begin
10     sResult := 0;
11
12     For nIndex := 0 to Periodo -1 do
13     begin
14         sResult := sResult + Preco[nIndex];
15     end;
16
17     Plot(sResult / Periodo);
18 end;
```

PARSE SUCCESSFULL!

NoName2*

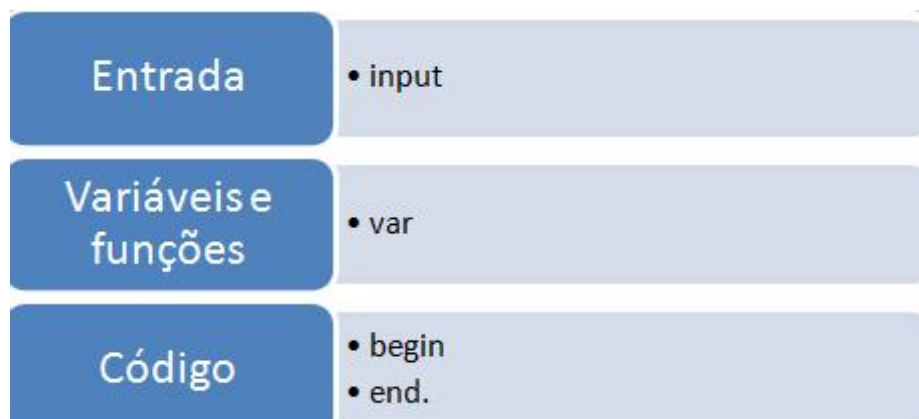


```
1 Parametro
2   Preco(Fechamento);
3   Periodo(2);
4
5 var
6   sResult : Real;
7   nIndex  : Inteiro;
8
9 Inicio
10  sResult := 0;
11
12  para nIndex := 0 ate Periodo -1 faca
13  inicio
14  sResult := sResult + Preco[nIndex];
15  fim;
16
17  Plot(sResult / Periodo);
18  Fim;
```

PARSE SUCCESFULL!

NoName2*

A área de parâmetros de entrada compreende toda a região entre a palavra reservada **input (parametro)** e a palavra reservada **var**. A área de variáveis e funções começa com a palavra **var** e estende-se até a palavra **begin (inicio)**. Finalmente, a região de código inicia-se com a palavra reservada **begin (inicio)** e finaliza na palavra **end (fim)**, conforme imagem abaixo:





Área de declaração de parâmetros de entrada

Na área de declaração de parâmetros de entrada informamos todos os parâmetros externos que a estratégia usará. Esses parâmetros são fundamentais, pois:

- Definem a interface com o mundo externo, ou seja, é onde usuário poderá alterar e o que servirá como parâmetro de chamada caso a estratégia seja utilizada em outra interface/estratégia.
- Define os itens que serão analisados no processo de otimização.

Para definir um parâmetro: NOME_DO_PARÂMETRO (VALOR_DE_INICIALIZAÇÃO)

Exemplo

Input	Definição
Preco (Close);	Define um parâmetro inicializado com o valor de fechamento de preços da série de dados.
Período (2);	Define um parâmetro chamado Período com o valor 2.

Área de declaração de variáveis e funções

Na área de declaração de variáveis informamos todas as variáveis que serão utilizadas na estratégia. Nesta região também descrevemos funções que desejamos usar no código. As funções devem ser sempre codificadas após a declaração das variáveis.

Para declarar uma variável: NOME_DA_VARIAVEL: TIPO

Exemplo:

Variável	Definição
sResult : Float;	Cria uma variável chamada sResult do tipo ponto flutuante.
nIndex: Integer;	Cria uma variável chamada nIndex para armazenar números inteiros.

Área de código

Nesta parte descreve-se o código propriamente dito, ou seja, as regras que utilizam parâmetros, variáveis e outros dados para calcular sinais e indicadores.



Fluxo de Execução de uma Estratégia

O código de uma estratégia, naturalmente, é executado de maneira sequencial. Dentro desse contexto, observe novamente o indicador de média móvel na figura 1. Não existe explicitamente um laço que faça com que o sistema percorra toda a base de dados de um ativo. No entanto, essa é exatamente a ação realizada pelo software.

Assim, a sequencia de comandos existente na área de código será executada uma vez para cada elemento de dados existente. Em um exemplo hipotético, mas ilustrativo imagine uma série de dados diários composta unicamente pelos 21 pregões de um determinado mês de janeiro. Para cada dado, será executado o código da estratégia em um loop implícito. A regra em pseudocódigo é, portanto:

- Tomando uma série de dados chamada aqui de X de tamanho N;
- Para cada item $X[i]$, com i variando de 0 até $N-1$, executa-se a área de código da estratégia.

```
1  input
2  Preco(Close);
3  Período(2);
4
5  var
6  sResult : Float;
7  nIndex : Integer;
8
9  begin
10 sResult := 0;
11
12 For nIndex := 0 to Período -1 do
13 begin
14 sResult := sResult + Preco[nIndex];
15 end;
16
17 Plot(sResult / Período);
18 end;
19
20
21
22
```

PARSE SUCCESFULL!

NoName2*



Documentação Módulo Estratégias

```
1 Parametro
2   Preço(Fechamento);
3   Período(2);
4
5 var
6   sResult : Real;
7   nIndex  : Inteiro;
8
9  Inicio
10 •   sResult := 0;
11
12 •   para nIndex := 0 ate Período -1 faça
13  inicio
14 •   sResult := sResult + Preço[nIndex];
15 •   fim;
16
17 •   Plot(sResult / Período);
18 •   Fim;
19
20
21
22
```

PARSE SUCCESFULL!

NoName2*



Variáveis, tipos de dados e constantes

As variáveis são utilizadas para armazenar dados, conforme seu tipo específico.

Tipos de dados:

Float ou Real - Representa números de ponto flutuante.

Integer ou Inteiro - Representa números inteiros.

Boolean ou Booleano - Representação lógica: **True** ou **Verdadeiro** e **False** ou **Falso**

Serie - Representa uma série de dados.

As variáveis na linguagem oferecem uma flexibilidade muito maior do que na maioria dos sistemas de programação conhecidos.

Na verdade, todas as variáveis são séries de dados, o que significa que é possível navegar entre valores atuais e passados, mediante indexação.

Séries de dados

As séries de dados são utilizadas para obter dados específicos de cada Candle.

Séries:

Open ou Abertura - Retorna o valor de abertura de cada candle.

Close ou Fechamento - Retorna o dado de fechamento.

Low ou Mínima - Retorna o valor de mínima

High ou Máxima - Retorna o dado de máxima.

Quantity ou Quantidade - Retorna o valor de contratos negociados de cada candle

Volume - Retorna o dado de volume financeiro de cada candle

Acessando dados anteriores

O dado atual de uma variável pode ser descrito por Variavel[0]. O número entre colchetes indica o dado de quantos períodos anteriores deseja-se acessar (sendo 0, portanto, da barra corrente). Para ilustrar melhor, vamos a um exemplo:

```
sResult := Preco[1];
```

A linha de código acima está atribuindo o valor da variável Preco[1] para a variável sResult. Imagine que Preco corresponde a todos os valores de fechamento da série de dados de um certo ativo, como na tabela abaixo:



Data	Posição	Valor
05/10/2010	Preco[0]	12,43
04/10/2010	Preco[1]	12,40
01/10/2010	Preco[2]	11,39
30/09/2010	Preco[3]	12,51
29/09/2010	Preco[4]	12,38
dd/mm/aaaa	Preco[n]	xx,xx

Portanto, Preco[1] refere-se ao valor de ontem do preço de fechamento (sResult vale então 12,40 em nosso exemplo). Dessa forma, o número inteiro que especificarmos entre colchetes indica ao sistema quantos períodos no passado deve-se acessar a informação.

Constantes

São utilizadas para a inserção de parâmetros de determinadas funções, estes valores não podem ser alterados pelo usuário.

Lado da Ordem:

osBuy ou **osCompra**: Ordem de compra.

osSell ou **osVenda**: Ordem de venda.

Opções:

optCall - Opção de compra.

optPut - Opção de venda.

Tipo de Ordem:

otLimit - Tipo limite.

otMarket - À mercado.

otStopLimit - Tipo stop.

Cores:

clAqua ou **clAzulClaro** - Cor azul-claro

clBlack ou **clPreto** - Cor preta

clBlue ou **clAzul** - Cor azul

clCream ou **clCreme** - Cor creme

clDkGray ou **clCinzaEscuro** - Cor cinza-escuro

clFuschia ou **clFucsia** - Cor fúcsia

clGray ou **clCinza** - Cor cinza

clGreen ou **clVerde** - Cor verde

clLime ou **clVerdeLimao** - Cor verde-limão

clLtGray ou **clCinzaClaro** - Cor cinza-claro

clMaroon ou **clMarrom** - Cor marrom

clMedGray ou **clCinzaMedio** - Cor Cinza médio

clMoneyGreen ou **clVerdeClaro** - Cor verde-claro

clNavy ou **clAzulMarinho** - Cor azul-marinho



clOlive ou clVerdeOliva - Cor verde-oliva
clPurple ou clPurpura - Cor púrpura
clRed ou clVermelho - Cor vermelha
clSilver ou clPrata - Cor prata
clSkyBlue ou clAzulClaro - Cor azul-claro
clTeal - Cor Verde-azulado
clWhite ou clBranco - Cor branca
clYellow ou clAmarelo - Cor amarela



Controle de Fluxo

As instruções de fluxo são utilizadas para administrar a sequência de execução das instruções dentro de um programa. A NTSL apresenta três tipos desse tipo:

IF THEN ELSE (SE ENTAO SENAO)

Se (condição) for verdadeiro executa-se a listagem (comandos 1), caso (condição) seja falso executa-se (comandos 2). Tanto (comandos 1) quanto (comandos 2) podem ser sequências de instruções.

A expressão (condição) pode ser qualquer tipo de teste que resulte em verdadeiro ou falso, alguns exemplos:

- IF (Close > Close[1])
- IF (nIndex = 10)
- IF ((bCond = false) and (Volume > 1000)

A seguir um exemplo de coloração de candles de acordo com a condição presente no **If then else**.



```
1  begin
2  •  if(Close = Close[1]) then
3  •  begin
4  •  PaintBar(clYellow);
5  •  end
6  •  else if(Close > Close[1]) then
7  •  begin
8  •  PaintBar(clGreen);
9  •  end
10 •  else
11 •  begin
12 •  PaintBar(clRed);
13 •  end;
14 •
15 •  Plot(Close);
16 •  end;
17
18
19
20
21
22
```

PARSE SUCCESFULL!

NoName3*



```
1 Inicio
2 Se(Fechamento = Fechamento[1]) entao
3 inicio
4 PaintBar(clAmarelo);
5 fim
6 senao se(Fechamento > Fechamento[1]) entao
7 inicio
8 PaintBar(clVerde);
9 fim
10 senao
11 inicio
12 PaintBar(clVermelho);
13 fim;
14
15 Plot(Fechamento);
16 Fim;
17
```

PARSE SUCCESFULL!

NoName3*

Se o valor de fechamento da barra atual (Close equivale a Close[0]) for igual ao fechamento da barra anterior (Representado por Close[1]) executa-se o código que segue a palavra reservada **THEN (ENTÃO)**. Caso contrário, o sistema executa o código subsequente até chegar na palavra reservada **ELSE (SENÃO)**.

FOR TO (PARA ATE)

O comando FOR é utilizado para definir um loop controlado, ou seja, (comandos) é executado repetidamente até que a (variável de contagem) saia do (valor inicial) e atinja (valor final). A cada iteração a (variável de contagem) é incrementada em 1.

Exemplo:

Observe o código da média móvel na figura 1. A variável de contagem nIndex começa valendo 0 e deve chegar ao valor de Período - 1. Período é um parâmetro de entrada, ele é usado para definir o tamanho da média. Assim, se Período valer 9, o comando FOR criará um loop de 9 iterações (de 0 até 8) para cada barra, calculando assim o valor médio para a posição atual.



Documentação Módulo Estratégias

```
1  input
2  Preco(Close);
3  Periodo(2);
4
5  var
6  sResult : Float;
7  nIndex  : Integer;
8
9  begin
10  sResult := 0;
11
12  For nIndex := 0 to Periodo - 1 do
13  begin
14  sResult := sResult + Preco[nIndex];
15  end;
16
17  Plot(sResult / Periodo);
18  end;
19
20
21
22
```

PARSE SUCCESFULL!

NoName2*



```
1 Parametro
2 Preco(Fechamento);
3 Periodo(2);
4
5 var
6 sResult : Real;
7 nIndex : Inteiro;
8
9 Inicio
10 sResult := 0;
11
12 para nIndex := 0 ate Periodo -1 faca
13 inicio
14 sResult := sResult + Preco[nIndex];
15 fim;
16
17 Plot(sResult / Periodo);
18 Fim;
19
20
21
22
```

PARSE SUCCESFULL!

NoName2*

WHILE DO (ENQUANTO FACAO)

A execução da estratégia ao chegar no comando WHILE testa o resultado de (condição). Caso (condição) seja verdadeiro (true) a listagem (comandos) é executada. Após a execução a (condição) volta a ser testada, assim, o loop apenas irá se encerrar quando (condição) deixar de ser verdadeira.

Exemplo

No código a seguir, reescrevemos o indicador média móvel utilizando a instrução **WHILE (ENQUANTO)** ao invés de **FOR (PARA)**.



```
1  input
2  Preço (Close) ;
3  Período (2) ;
4
5  var
6  sResult : Float;
7  nIndex  : Integer;
8
9  begin
10 •   sResult := 0;
11 •   nIndex  := 0;
12
13 •   while(nIndex <= Período -1) do
14 •   begin
15 •       sResult := sResult + Preço[nIndex];
16 •       nIndex  := nIndex + 1;
17 •   end;
18
19 •   Plot(sResult / Período);
20 • end;
21
22
< >
```

PARSE SUCCESFULL!

NoName2 MM2*



```
1 Parametro
2   Preco (Fechamento);
3   Periodo (2);
4
5 var
6   sResult : Real;
7   nIndex  : Inteiro;
8
9 Inicio
10  • sResult := 0;
11  • nIndex  := 0;
12
13  • enquanto (nIndex <= Periodo - 1) faça
14  inicio
15  •   sResult := sResult + Preco[nIndex];
16  •   nIndex  := nIndex + 1;
17  • fim;
18
19  • Plot (sResult / Periodo);
20  • Fim;
21
22
```

PARSE SUCCESFULL!

NoName2 MM2*



Operadores

Os operadores constituem os símbolos matemáticos e lógicos usados em cálculos e comparações.

Operadores matemáticos

Os operadores matemáticos são:

Operador	Descrição	Exemplo	Resultado
+	Adição	5+4	9
-	Subtração	5-4	1
*	Multiplicação	5*4	20
/	Divisão	5/4	1,25

O operador de divisão possui a maior força de precedência, seguido por multiplicação. Assim, como consta na imagem abaixo:

Expressão	Resultado
$10*10/5+2$	22
$10*10+2$	102
$50+100*10/2-1$	549

Operadores lógico

Os Operadores lógicos são utilizados principalmente para comparações.

"E" lógico

Representado pela palavra reservada **and (e)**, retornará TRUE somente quando as duas condições de teste forem verdadeiras conforme Tabela Verdade abaixo:



Condição 1	AND	Condição 2	Resultado
FALSE	AND	FALSE	FALSE
FALSE	AND	TRUE	FALSE
TRUE	AND	FALSE	FALSE
TRUE	AND	TRUE	TRUE

"OU" lógico

Representado pela palavra reservada **or (ou)**, retornará TRUE (verdadeiro) sempre que pelo menos uma das condições de teste for verdadeira, conforme Tabela Verdade abaixo:

Condição 1	OR	Condição 2	Resultado
FALSE	OR	FALSE	FALSE
FALSE	OR	TRUE	TRUE
TRUE	OR	FALSE	TRUE
TRUE	OR	TRUE	TRUE



Funções

Conforme visto, funções são declaradas e descritas na área de declaração de variáveis e funções, abaixo um exemplo de funções:

```
1  input
2  BreakPeriodo (5);
3  HiLoPeriodo (3);
4
5  var
6  bBuy      : Boolean;
7  bShor     : Boolean;
8  sMax      : Float;
9  sMin      : Float;
10 sMinHiLo  : Float;
11 sMaxHiLo  : Float;
12 fPlot     : Float;
13
14 function Max(Values : Float; nPeriodo : Integer): Float;
15 var
16     nIndex : Integer;
17     sAux   : Float;
18 begin
19     sAux := Values;
20     Result := sAux;
21
22     For nIndex := 1 to nPeriodo -1 do
23     begin
24         if(sAux[nIndex] > Result) then
25         begin
26             Result := sAux[nIndex];
27         end;
28     end;
29 end;
30
31 begin
32     fPlot := Max(Close, 5);
33     Plot(fPlot);
34 end;
```

Pro Editor de Estratégias

Editor Gráfico Misto

NoName4*



```
Pro Editor de Estratégias
Editor Gráfico Misto
[Icons]
1 Parametro
2 BreakPeriodo (5);
3 HiLoPeriodo (3);
4
5 var
6 bBuy : Booleano;
7 bShor : Booleano;
8 sMax : Real;
9 sMin : Real;
10 sMinHiLo : Real;
11 sMaxHiLo : Real;
12 fPlot : Real;
13
14 funcao Max(Values : Real; nPeriodo : Inteiro): Real;
15 var
16 nIndex : Inteiro;
17 sAux : Real;
18 Inicio
19 sAux := Values;
20 Result := sAux;
21
22 Para nIndex := 1 ate nPeriodo -1 faca
23 inicio
24 se(sAux[nIndex] > Result) entao
25 inicio
26 Result := sAux[nIndex];
27 fim;
28 fim;
29 Fim;
30
31 Inicio
32 fPlot := Max(Fechamento, 5);
33 Plot(fPlot);
34 Fim;
~
<
PARSE SUCCESSFUL!
NoName4*
```

Observe que primeiro são declaradas as seis variáveis usadas na área principal. É sempre interessante manter o código o mais claro e organizado possível e as funções desempenham um papel fundamental nessa tarefa.



Criando Funções(Sintaxe)

Function (funcao) Nome da Função ((parâmetro 1 : TIPO); (parâmetro 2: TIPO:);(parâmetro n: TIPO)): Tipo de Retorno

Begin(inicio)

Comandos

End (fim);



Funções de biblioteca

Além do usuário poder criar seus próprios indicadores, é possível utilizar a biblioteca do sistema, ou seja, o usuário pode utilizar estratégias já criadas em novas.

Dentro das funcionalidades de bibliotecas, o usuário poderá colorir os gráficos de acordo com as condições determinadas pelo seu indicador.

Funções Gráficas

Para criar um gráfico de linha o usuário deverá utilizar a função **Plot**, onde o sistema irá efetuar a interligação dos pontos criados pelo indicador.

Funções matemáticas

As funções matemáticas tem como finalidade implementar as seguintes funcionalidades:

- **Power(valor,potência):** Tem como funcionalidade, gerar valores elevados em determinada potência;
- **Round(valor):** Tem como funcionalidade, arredondamento de números quebrados, caso o valor após a vírgula seja menor do que cinco, arredonda para baixo, caso contrário, arredonda para cima;
- **Sqrt(valor):** Tem como funcionalidade mostrar a raiz quadrada de valores desejados pelo usuário;

Funções Gráficas

Como visto anteriormente, a função **Plot** realiza a ligação dos valores gerados na estratégia e cria gráficos de linhas, mas caso haja a necessidade, o usuário também poderá colorir o gráfico de acordo com o desejado.

Esta funcionalidade denominada **PaintBar(cor)** permite ao usuário, colorir o gráfico com cores em determinadas situações do indicador, como na imagem abaixo(as cores podem ser determinadas por **Strings**, ou a partir da função **RGB**):



```
1  begin
2  •  if(Close = Close[1]) then
3  •  begin
4  •  PaintBar(clYellow);
5  •  end
6  •  else if(Close > Close[1]) then
7  •  begin
8  •  PaintBar(clGreen);
9  •  end
10 •  else
11 •  begin
12 •  PaintBar(clRed);
13 •  end;
14 •
15 •  Plot(Close);
16 •  end;
17
18
19
20
21
22
```

PARSE SUCCESFULL!

NoName3*



```
1 Inicio
2 Se(Fechamento = Fechamento[1]) entao
3 início
4 PaintBar(clAmarelo);
5 fim
6 senao se(Fechamento > Fechamento[1]) entao
7 início
8 PaintBar(clVerde);
9 fim
10 senao
11 início
12 PaintBar(clVermelho);
13 fim;
14
15 Plot(Fechamento);
16 Fim;
17
```

PARSE SUCCESFULL!

NoName3*



Back-Testing

A funcionalidade de Back-testing permite ao usuário avaliar uma determinada estratégia, teoria ou modelo através de uma análise de dados históricos.

Lista de funcionalidades utilizadas para Back-Testing:

- [Lista de funções](#) ;
- [Criar regra de execução](#) ;
- [Execução da estratégia](#)

Após criada a estratégia de Back-Testing, para ser adicionado diretamente no gráfico, clique no botão direito sobre o mouse e selecione a opção "Inserir Regra de Execução".



Abrir Estratégias

Na opção de "Abrir Estratégias", o usuário terá acesso a três abas, elas são:

- **Todas:** O usuário poderá ver todas as estratégias dentro do seu ProfitChart;
- **Minhas Estratégias:** O usuário irá filtrar para somente exibir todas as estratégias criadas por ele dentro do ProfitChart;
- **Exemplos:** O usuário irá filtrar para exibir exemplos de estratégias que já vem como padrão no ProfitChart.

Além das abas, o usuário também poderá pré-visualizar o seu código de estratégia para confirmar informações.

Abrir Estratégia

Todas | Minhas Estratégias | Exemplos

Estratégias

Nome ▲	Indicador	Coloração	Execução
Early-Onset Trend	-	-	-
Gaps	-	-	-
IFR	-	-	-
MACD2	-	-	-
Media	Sim	-	-
MediaExp	Sim	-	-
MM	Sim	-	-
MM2	Sim	-	-
MyRSI	-	-	-
PaintBar	-	Sim	-
PaintVar	-	-	-
RSI	-	-	-
Testando grande	-	-	-
Teste	-	-	-
TradeByForce	-	-	-

Descrição

Preview

```
1 Input
2 LPPeriod (30);
3 K (0.85);
4 Var
5 alpha1 (0),
6 HP (0),
7 a1 (0),
8 b1 (0),
9 c1 (0),
10 c2 (0),
11 c3 (0),
12
```

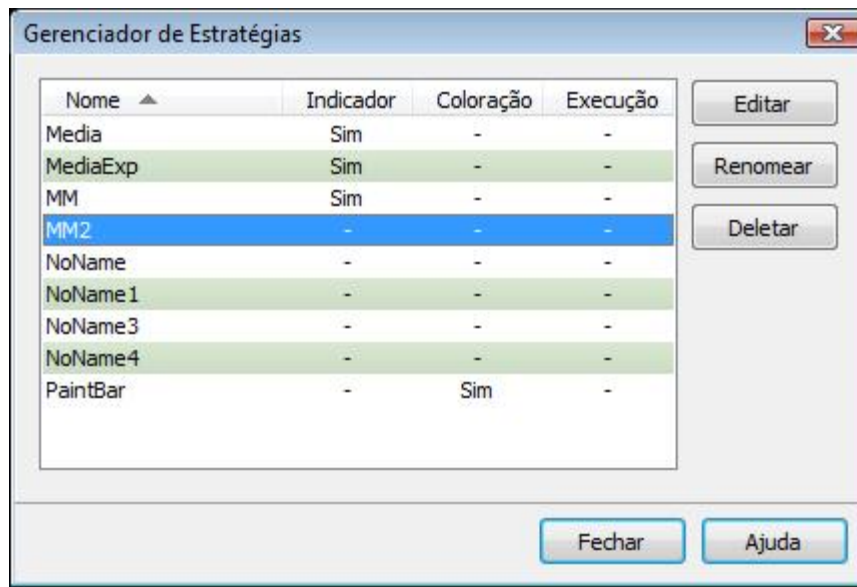
Abrir Cancelar



Gerenciador de Estratégias

A opção de gerenciador de estratégias, permite ao usuário escolher uma determinada estratégia criada para edição, fazendo com que o Editor de estratégias carregue a estratégia determinada, ao clicar em "Editar".

O usuário também poderá excluir as estratégias desejadas, selecionando as mesmas e clicando no botão "Excluir", além da funcionalidade de renomear a estratégia através do botão "Renomear".

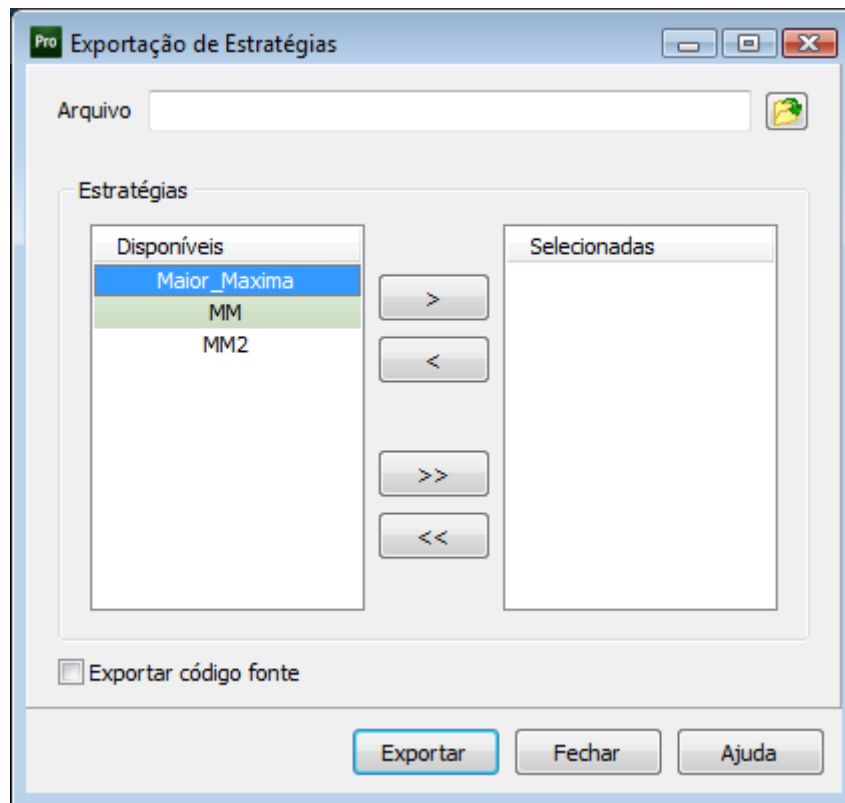




Exportar/Importar Estratégias

Nesta funcionalidade, permite ao usuário exportar as estratégias criadas por eles para que possam ser importadas novamente.

O usuário também tem a possibilidade de querer exportar o código fonte da estratégia ou apenas o arquivo executável.



Na importação, o usuário tem a funcionalidade de escolher quais estratégias serão carregadas e adicionadas junto ao ProfitChart.



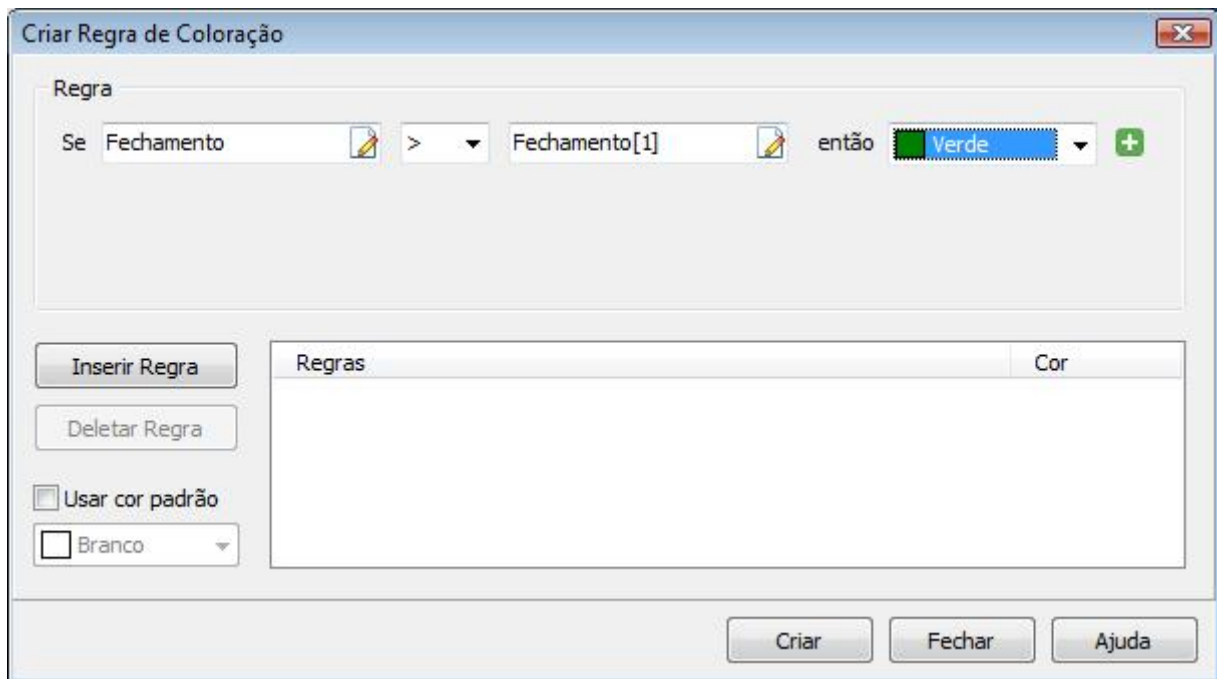


Criar Regra de Coloração

A funcionalidade de criar regra de coloração, mostra de forma visual, como criar uma regra de coloração com as condições desejadas.

A cada vez que for clicado no botão "+" irá criar uma condição para que se Condição 1 e Condição 2 sejam verdadeiras irá colorir de acordo com a cor desejada, na cor padrão será se caso as condições não retornem verdadeiro irá pintar em determinada cor.

Caso o usuário deseje utilizar outras informações, ele irá poder clicar no botão "Mais" que se encontra ao lado da variável para selecionar outras condições.





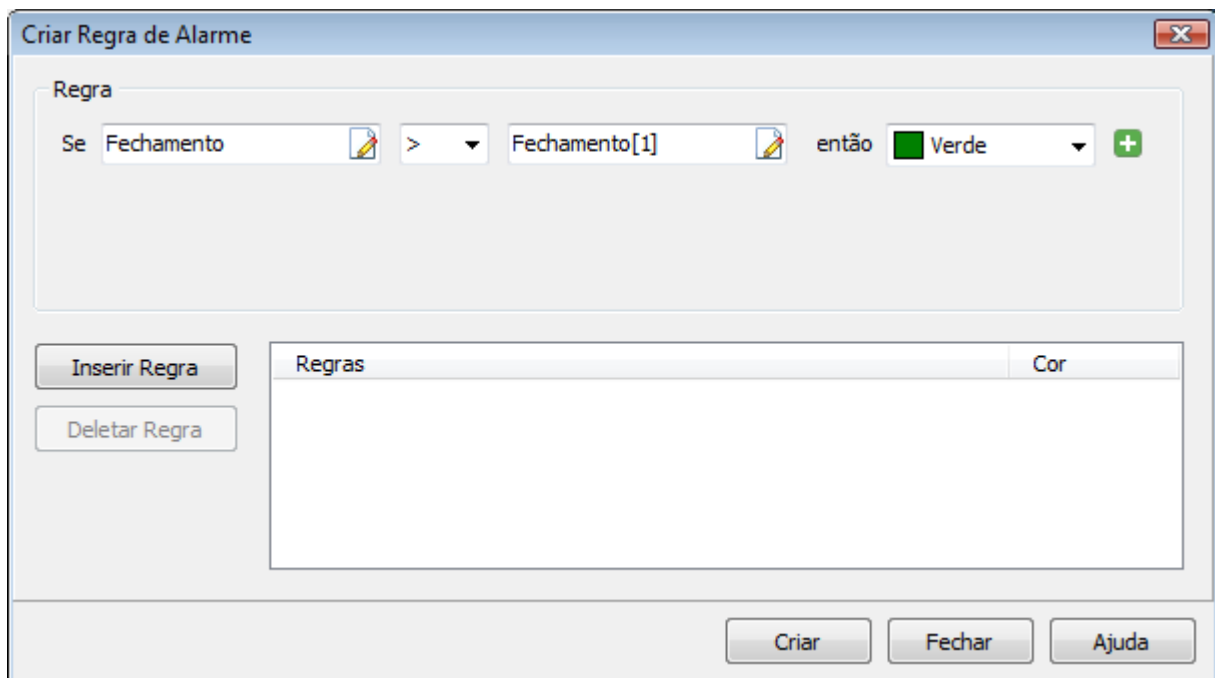
Criar Regra de Alarme

A funcionalidade do Criar Regra de Alarme mostra de forma visual, como criar uma regra de alarme de acordo com as condições desejadas.

A cada vez que for clicado no botão "+", será criada uma condição para que se Condição 1 e Condição 2 sejam verdadeiras irá acionar o alarme desejado, e o pop-up de alarme será emitido de acordo com a cor estabelecida.

Caso o usuário deseje utilizar outras informações, ele poderá clicar no botão "Mais" que se encontra ao lado da variável para selecionar outras condições.

Após a criação da regra de alarme, a estratégia deverá ser ativada, através do menu "Ferramentas > Gerenciador de Alarmes > Estratégias", selecione "Novo Alarme".





Criar Regra de Execução

A funcionalidade do Criar Regra de Execução mostra de forma visual, como criar uma regra de execução de acordo com as condições desejadas.

A cada vez que for clicado no botão "+" irá criar uma condição para que se Condição 1 e Condição 2 sejam verdadeiras irá acionar o alarme desejado e irá colorir o pop-up de alarme de acordo com a cor estabelecida.

Caso o usuário deseje utilizar outras informações, ele irá poder clicar no botão "Mais" que se encontra ao lado da variável para selecionar outras condições.

Para saber qual tratamento deve ser dado quando a condição for satisfeita, as funções de execução podem ser vistas [aqui](#).

The screenshot shows a software window titled "Criar Estratégia de Execução". The main area is labeled "Regra" and contains a visual rule builder. The rule is: "Se Fechamento > Fechamento[1] então BuyAtMarket". Each element in the rule (variables and operators) has a small icon next to it. Below the rule builder, there are two buttons: "Inserir Regra" and "Deletar Regra". To the right of these buttons is a table with two columns: "Regras" and "Ação". The table is currently empty. At the bottom of the window, there are three buttons: "Criar", "Fechar", and "Ajuda".



Screening

A funcionalidade de Screening mostra de forma visual, os ativos que se encontram na base de dados e que satisfazem as condições da estratégia.

A cada vez que for clicado no botão "+" irá criar uma condição para que se Condição 1 e Condição 2 sejam verdadeiras irá mostrar o ativo dentro da aba selecionada.

Caso o usuário deseje utilizar outras informações, ele irá poder clicar no botão "Mais" que se encontra ao lado da variável para selecionar outras condições.

Ao clicar no botão "Aplicar" a estratégia criada é aplicada a grade e irá mostrar os ativos que satisfazem a condição.

Ao clicar no botão "Desfazer" a estratégia irá retornar para a última estratégia aplicada a grade.

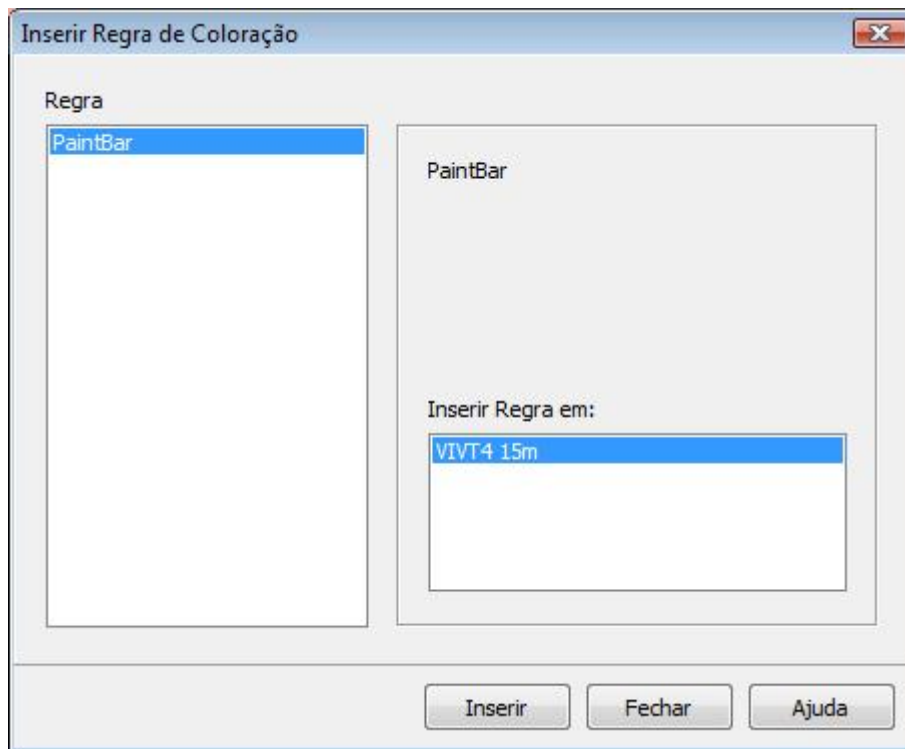
Ativo	Último	Hora	Variação	Negócios	Máximo	Mínimo	Fechament...	Abertura	Of. Compra
▼ Screening 1									
PETR4	13,11	14:38:06	1,39%	44.429	13,24	12,52	12,93	12,61	13,10



Inserir Regra de Coloração

A funcionalidade de inserir regra de coloração permite ao usuário colorir o gráfico de acordo como ele determinou os parâmetros nos gráficos dos ativos.

As regras de coloração serão feitas seguindo a ideologia de um indicador ou seja, respeitando o layout atual da janela.





Condições de Coloração

As condições de coloração permitem ao usuário, quais valores ele irá utilizar no momento em que irá criar uma nova regra de coloração, elas podem ser:

- Numérico: O usuário poderá utilizar números inteiros ou reais;
- Cotação: O usuário poderá utilizar os valores presente nas cotações, sendo elas: Abertura, Máxima, Mínima, Fechamento, Quantidade;
- Indicador: O usuário poderá utilizar os valores presentes nos indicadores criados por ele, e alterar os parâmetros presentes para de acordo com a vontade para coloração;
- Cotações Anteriores: Permite ao usuário utilizar os valores presentes nas cotações anteriores, conforme mostra na guia [Variáveis e séries de dados](#).

Condição

Valor

Numérico 1

Cotação Fechamento

Indicador Media(Close,21,)

Cotações Anteriores

Deslocar 1 período(s)

OK Cancelar Ajuda

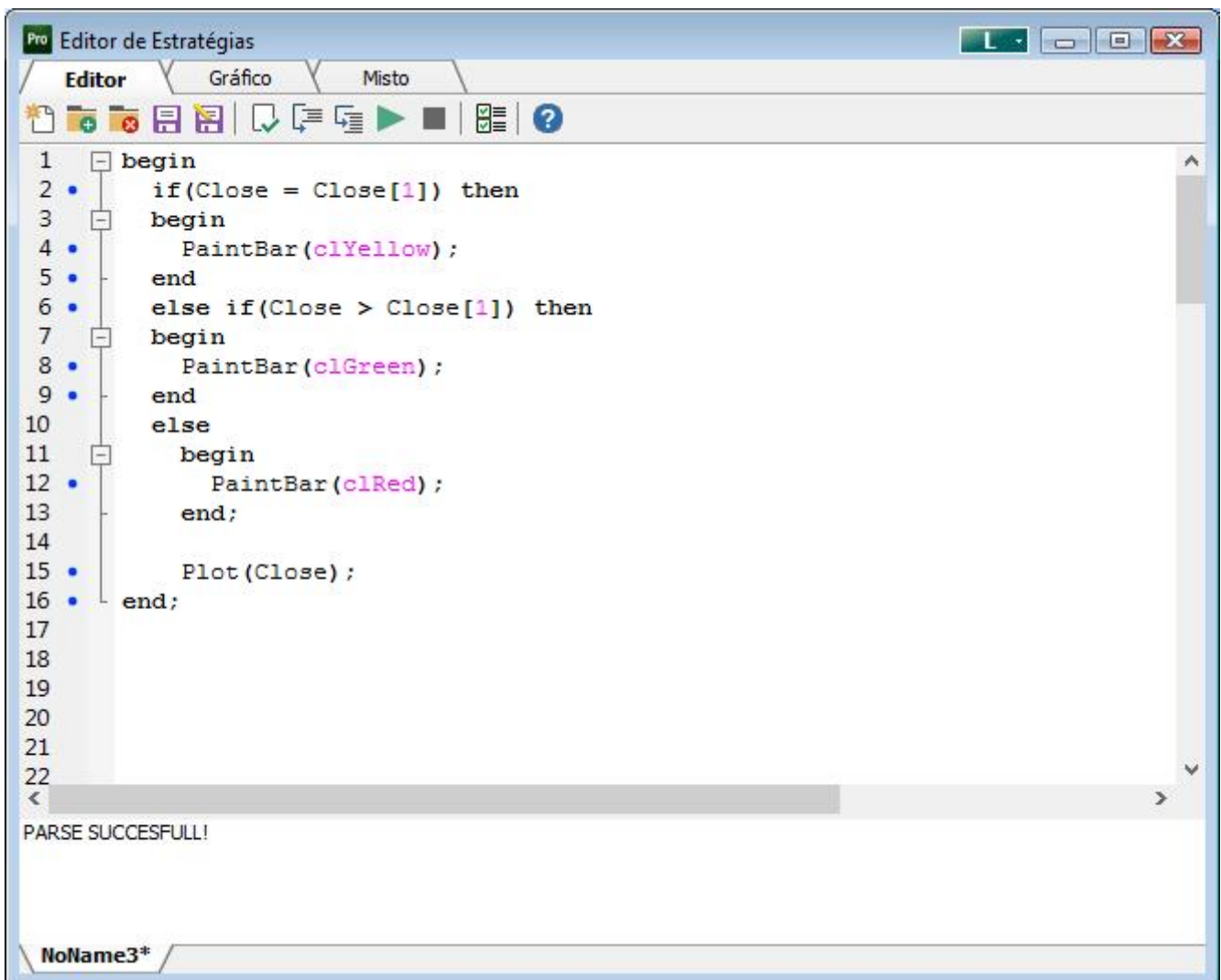


Editor de Estratégias

A janela de Editor de estratégias é onde o usuário irá poder criar suas próprias estratégias juntamente com a funcionalidade de visualizar diretamente no gráfico a estratégia criada.

O Editor de Estratégias possui três abas:










- Editor: Onde o usuário escreve a estratégia seguindo as instruções da linguagem NTSL.
- Gráfico: Onde o usuário visualiza a estratégia após executada diretamente no gráfico;
- Misto: É onde o usuário visualiza as informações da aba Editor e a aba Gráfico em uma só, onde a cada vez que ele execute o código, já irá aparecer diretamente no gráfico.







Documentação Módulo Estratégias

Dentro do Editor de estratégias o usuário irá possuir as seguintes opções:

-  Nova Estratégia: Onde o usuário irá poder criar uma nova estratégia;
-  Abrir Estratégia: Permite ao usuário abrir estratégias já criadas e editá-las
-  Fechar: Fecha a aba da estratégia atual;
-  Salvar: Salva a estratégia atual;
-  Salvar Como: Salva a estratégia atual, podendo ser adicionado uma descrição da mesma;
-  Verificar Sintaxe: Realiza a leitura do código verificando se há erros e a transforma em uma estratégia executável;
-  Trace: Mostra passo-a-passo o que o código da estratégia está realizando e mostrando os valores naquele momento;
-  Trace Into: Semelhante ao Trace, mostra passo a passo o que a estratégia está realizando no momento de criação, porém, quando há funções no código ele irá abrir a função para mostrar ao usuário que a função está executando;
-  Executar: Após apertar o botão "Compilar", o botão executar irá executar a estratégia criada e a mostra no gráfico;



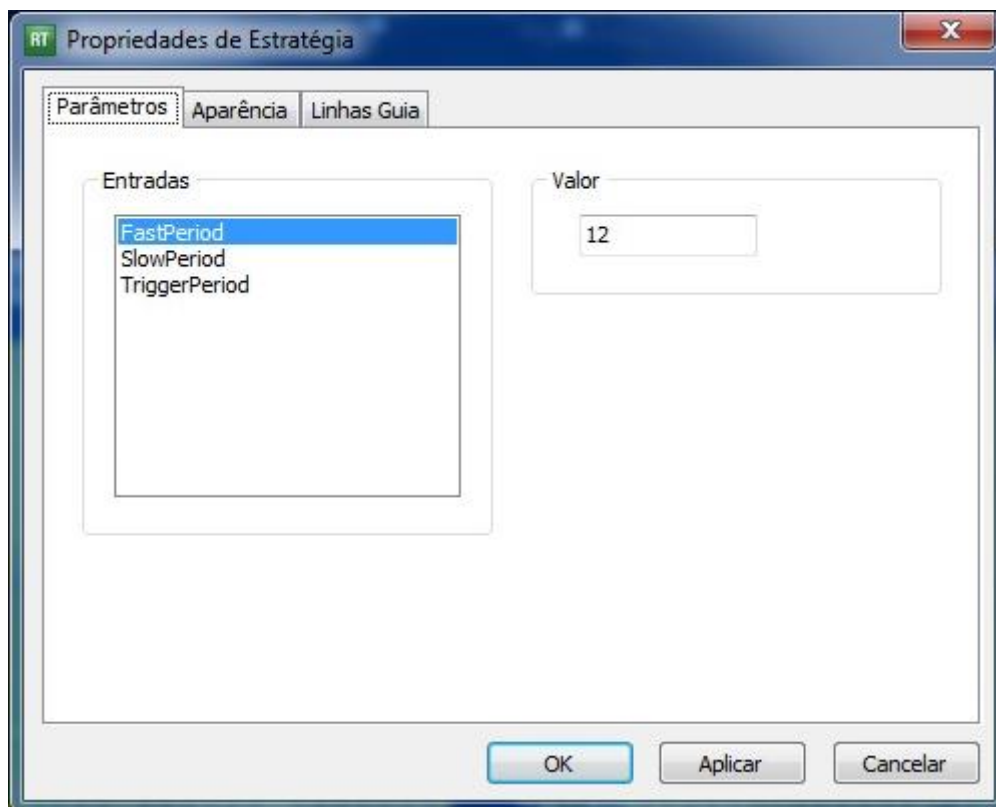
-  Parar: Tem como funcionalidade parar a estratégia para que o usuário pare a execução da estratégia criada;
-  Propriedades: Tem como funcionalidade mostrar as propriedades que irão constituir a estratégia, como desenho no gráfico, linhas guias e escala.



Propriedades do Editor de Estratégias

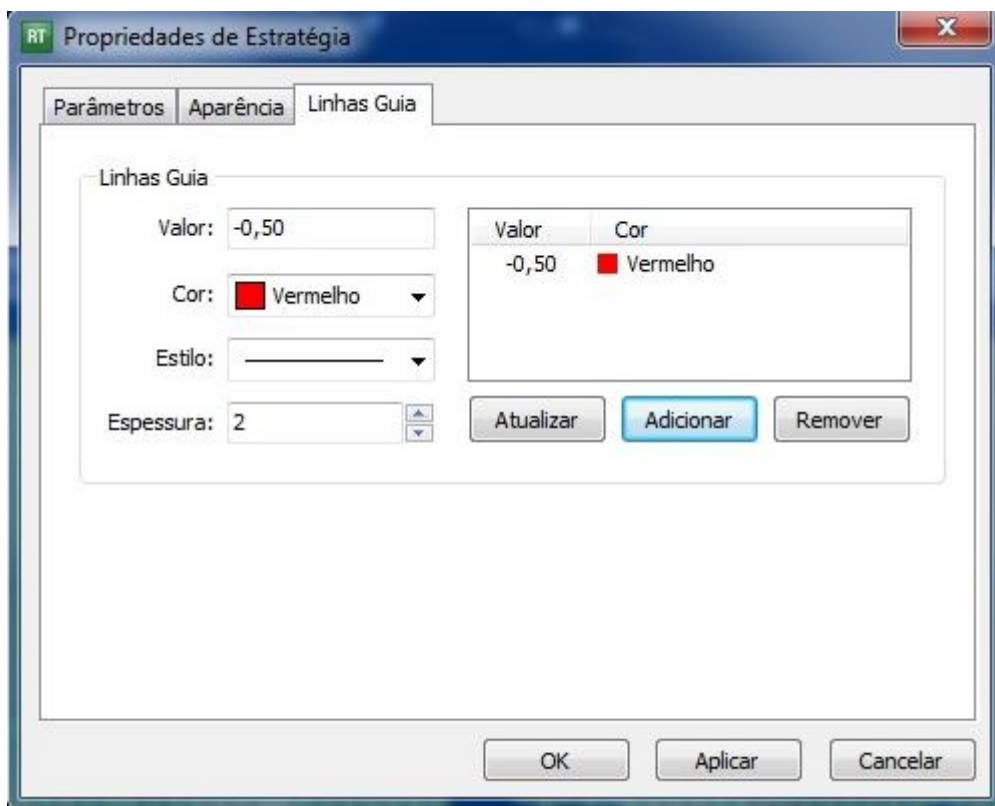
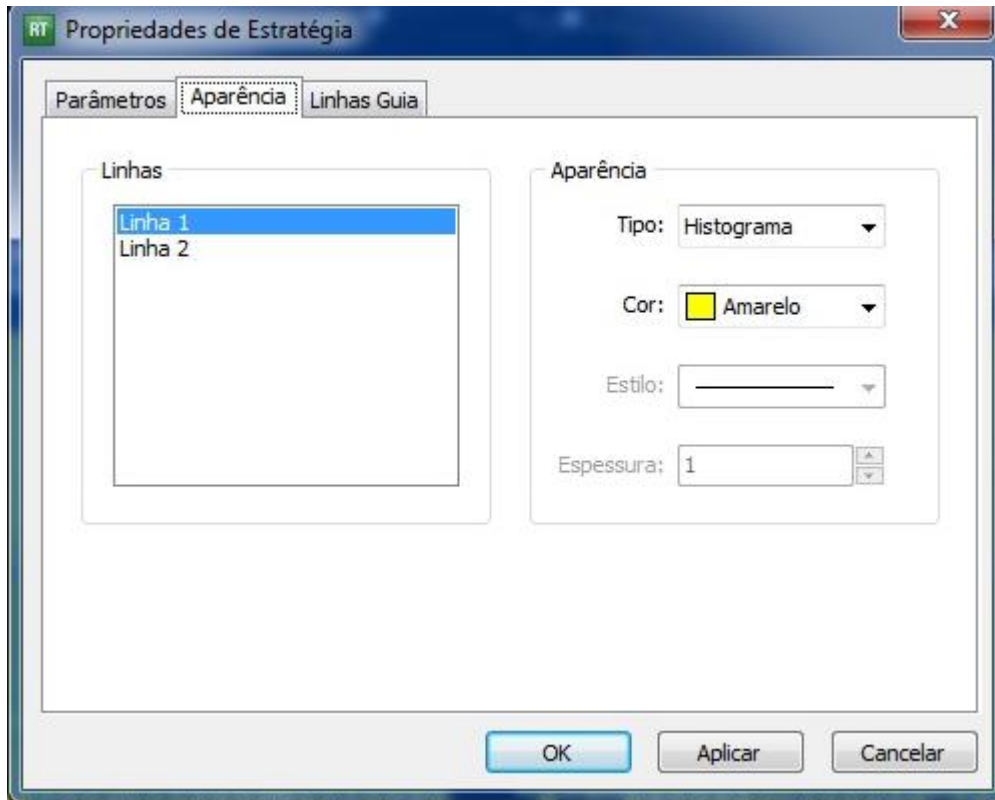
Nas propriedades do Editor de estratégia o usuário irá poder utilizar valores e informações adicionais no momento da criação da estratégia, elas são:

- **Parâmetros:** Permite ao usuário utilizar estratégias já criadas como parâmetros para uma nova estratégia junto com o valor desejado para a mesma;
- **Aparência:** Permite ao usuário determinar se deseja que a estratégia seja mostrada em linha ou em histograma;
- **Linhas Guia:** Permite ao usuário criar linhas para se basear como exemplos de linha de suporte e resistência.





Documentação Módulo Estratégias

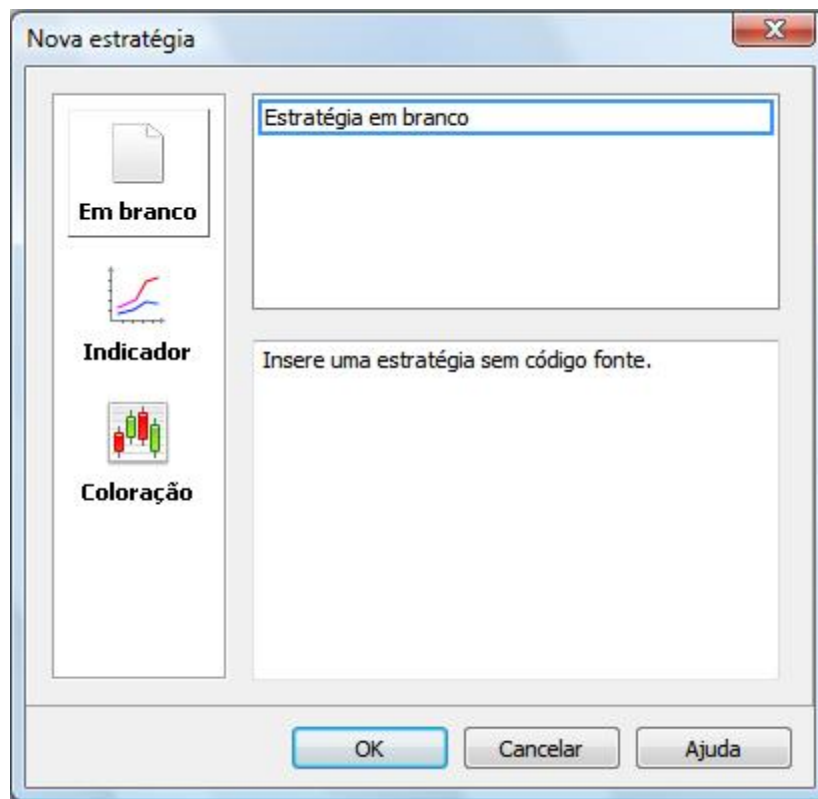




Nova Estratégia

Ao clicar no botão de Nova Estratégia, o usuário poderá escolher entre uma das três opções abaixo:

- Em branco: Ao escolher desta maneira, o usuário irá criar uma estratégia em branco;
- Indicador: Ao abrir a estratégia, irá carregar um exemplo de estratégia de indicador para o usuário;
- Coloração: Ao abrir a estratégia, irá carregar um exemplo de regra de coloração para o usuário.





Lista de Funções

A seguir de forma descritiva, segue a lista de funções presentes no Profit:

Alarme

Alert(Cor : Integer): Dispara uma notificação de alarme.

Back-Testing

BuyAtMarket: Realiza envio de ordem de compra à mercado.

BuyLimit(Preco : Float): Realiza envio de uma ordem limite de compra.

BuyPosition: Retorna o tamanho da posição da compra.

BuyPrice: Retorna o preço de compra da posição.

BuyStop(Stop : Float, Limite : Float): Realiza envio de ordem de compra stop.

BuyToCoverAtMarket: Realiza o fechamento de uma operação de venda.

BuyToCoverLimit(Preco : Float): Envia uma ordem de compra limite para fechar a operação.

BuyToCoverStop(Stop : Float, Limite : Float): Realiza o envio de ordem de compra stop para fechar posição.

CancelPendingOrders: Cancela as ordens pendentes.

ClosePosition: Envia ordens para encerrar todas as posições.

HasPendingOrders: Retorna verdadeiro (True) caso existam ordens pendentes.

IsBought: Retorna verdadeiro (True) caso exista posição de compra.

IsSold: Retorna verdadeiro (True) caso exista posição venda.

ReversePosition: Envia ordens para inverter a posição.

SellPosition: Retorna o tamanho da posição de venda.

SellPrice: Retorna o preço de venda da posição.

SellToCoverAtMarket: Realiza uma ordem de venda à mercado caso exista uma posição de compra.

SellToCoverLimit(Preco : Float): Envia uma ordem de venda limite para fechar a operação.

SellToCoverStop(Stop : Float, Limite : Float): Envia uma ordem stop caso exista uma posição de compra.

SellShortAtMarket: Envia ordem de venda à mercado para abrir posição.

SellShortLimit(Preco : Float): Envia ordem de venda do tipo limite.

SellShortStop(Stop : Float, Limite : Float): Abre uma posição de venda enviando uma ordem do tipo Stop.

SendOrder(Lado : Integer, Tipo : Integer, Quantidade : Integer, Limite : Float, Stop : Float)
: Envia uma ordem customizada.



Calendário

BarDuration: Retorna, em minutos, a duração da barra atual.

CalcDate(DataReferencia : Integer, DiasDeslocamento : Integer): Retorna o resultado ao efetuar a adição ou subtração de datas.

CalcTime(HoraReferencia : Integer, MinutosDeslocamento : Integer): Retorna o cálculo de adição ou subtração de horas.

CloseD(QuantidadeDiasAnteriores : Integer): Retorna o valor de fechamento de determinado dia anterior.

CloseM(QuantidadeMesesAnteriores : Integer): Retorna o valor de fechamento de determinado mês anterior.

CloseW(QuantidadeSemanasAnteriores : Integer): Retorna o valor de fechamento de determinada semana anterior.

CloseY(QuantidadeAnosAnteriores : Integer): Retorna o valor de fechamento de determinado ano anterior.

CurrentDate: Retorna a data atual do sistema.

CurrentTime: Retorna a hora atual do sistema.

Date: Retorna a data da barra que está sendo analisada.

DayOfMonth(Data : Integer): Retorna o dia do mês de uma data específica.

DayOfWeek(Data : Integer): Retorna o dia da semana de uma data específica.

DaysToExpiration(Mes : Integer, Ano : Integer): Retorna a quantidade de dias úteis restantes até a terceira sexta-feira de um determinado mês e ano.

ELDate(Ano : Integer, Mes : Integer, Dia : Integer): Retorna a data em EasyLanguage format(YYYYMMDD).

ELDate_Consol(Data : Integer): Transforma uma data YYYYMMDD em EasyLanguage format(YYYYMMDD).

FindBar(Data : Integer, Hora : Integer): Localiza uma barra de uma data e hora específica.

Friday: Retorna o número 5 referente ao dia da semana: sexta-feira.

HighD(QuantidadeDiasAnteriores : Integer): Retorna o valor de máxima de determinado dia retroativo.

HighM(QuantidadeMesesAnteriores : Integer): Retorna o valor de máxima de determinado mês retroativo.

HighW(QuantidadeSemanasAnteriores : Integer): Retorna o valor de máxima de determinada semana anterior.

HighY(QuantidadeAnosAnteriores : Integer): Retorna o valor de máxima de determinado ano anterior.

LowD(QuantidadeDiasAnteriores : Integer): Retorna o valor de mínima de determinado dia anterior.

LowM(QuantidadeMesesAnteriores : Integer): Retorna o valor de mínima de determinado mês anterior.

LowW(QuantidadeSemanasAnteriores : Integer): Retorna o valor de mínima de determinada semana anterior.



Documentação Módulo Estratégias

LowY(QuantidadeAnosAnteriores : Integer): Retorna o valor de mínima de determinado ano anterior.

LastCalcDate: Retorna a data do último candle completo do gráfico.

LastCalcTime: Retorna o horário de fechamento do último candle.

LastDayOfMonth(MesReferencia : Integer): Retorna o valor do último dia do mês de referência.

Monday: Retorna o número 1 referente ao dia da semana: segunda-feira.

Month(Date : Integer): Retorna o mês de uma data específica.

Next3rdFriday(Mes : Integer): Retorna quantos dias úteis faltam para a terceira sexta-feira de determinado mês.

OpenD(QuantidadeDiasAnteriores : Integer): Retorna o valor de abertura de determinado dia anterior.

OpenM(QuantidadeMesesAnteriores Integer): Retorna o valor de abertura de determinado mês anterior.

OpenW(QuantidadeSemanasAnteriores : Integer) Retorna o valor de abertura de determinado semana anterior.

OpenY(QuantidadeAnosAnteriores : Integer): Retorna o valor de abertura de determinado ano anterior.

RS_BarsPerDay: Obtém o número estimado de barras de determinada periodicidade(em minutos).

Saturday: Retorna o número 6 referente ao dia da semana: sábado.

Sunday: Retorna o número 0 referente ao dia da semana: domingo.

Thursday: Retorna o número 4 referente ao dia da semana: quinta-feira.

BarAnnualization: Retorna o fator de anualização baseado no intervalo da barra.

Bartype: Retorna um número referente à periodicidade setada.

Time: Retorna a hora de fechamento da barra atual.

TimeToMinutes(Hora : Integer): Converte um horário em minutos.

Tuesday: Retorna o número 2 referente ao dia da semana: Terça-feira.

VolumeD(QuantidadeDiasAnteriores : Integer): Retorna o volume financeiro de determinado dia retroativo.

VolumeM(QuantidadeMesesAnteriores : Integer): Retorna o volume financeiro de determinado mês retroativo.

VolumeW(QuantidadeSemanasAnteriores : Integer): Retorna o volume financeiro de determinada semana anterior.

VolumeY(QuantidadeAnosAnteriores : Integer): Retorna o volume financeiro de determinado ano anterior.

Wednesday: Retorna o número 3 referente ao dia da semana: Quarta-feira.

Year(Date : Integer): Retorna o ano de uma data específica.



Candlestick

C_3WhSolds_3BlkCrows(Comprimento : Integer, Fator : Integer, var o3WhiteSoldiers : Integer, var o3BlackCrows : Integer): Identifica a ocorrência de dois tipos de candles: 3 White Soldiers e 3 Black Crows.

C_BullEng_BearEng(Comprimento : Integer, var oBullishEngulfing: Integer, var oBearishEngulfing : Integer): Identifica a ocorrência de dois tipos de candles: Bullish Engulfing e Bearish Engulfing.

C_Doji(Percentual : Integer): Identifica a ocorrência de um candle: Doji.

C_Hammer_HangingMan(Comprimento : Integer, Fator : Integer, var oHammer : Integer, var oHangingMan : Integer): Identifica a ocorrência de dois tipos de candles: Hammer e Hanging Man.

C_MornDoji_EveDoji(Comprimento : Integer, Percentual : Float, var oMorningDojiStar : Integer, var oEveningDojiStar : Integer): Identifica a ocorrência de dois tipos de candles: Morning Doji Star e Evening Doji Star.

C_MornStar_EveStar(Comprimento : Integer, var oMorningStar : Integer, var oEveningStar : Integer): Identifica a ocorrência de dois tipos de candles: Morning Star e Evening Star.

C_PierceLine_DkCloud(Comprimento : Integer, var oPiercingLine : Integer, var oDarkCloud : Integer): Identifica a ocorrência de dois tipos de candles: Piercing Line e Dark Cloud.

C_ShootingStar(Comprimento : Integer, Fator : Integer): Identifica a ocorrência de candles: C_ShootingStar .

Exemplos

DiMaisDiMenos(Periodo : Integer): Exemplo de implementação do indicador DI+/DI-.

IFR(Periodo : Integer): Exemplo de implementação do indicador IFR.

Media(Periodo : Integer, TipoSerie : Serie): Exemplo de implementação do indicador Média Móvel(Aritmética).

MediaExp(Periodo : Integer, TipoSerie : Serie): Exemplo de implementação do indicador Média Móvel(Exponencial).

PaintVar: Exemplo de implementação de estratégia de coloração.

WellesSum(Periodo : Integer, SerieReferencia : Serie, Offset : Integer): Exemplo de implementação do indicador WellesSum.

Gráficas

AvgPrice: Retorna a média entre Abertura, Máxima, Mínima, Fechamento de determinado candle.

CurrentBar: Retorna o índice atual da barra(candle).

GetPlotColor(NumeroPlot : Integer): Retorna o valor numérico da cor de determinado Plot.

GetPlotWidth(NumeroPlot : Integer): Retorna o valor da espessura de determinado Plot.

Highest(SerieDeDados : Serie, Periodo : Integer): Retorna o maior valor da série dentro do período.

HighestBar(SerieDeDados : Serie, Periodo : Integer): Retorna o índice do maior valor da série no período.



Documentação Módulo Estratégias

- LastBarOnChart:** Retorna se é a última barra do gráfico.
- Leader:** Retorna se ponto médio é maior que mínima ou maior que máxima de candle anterior.
- MaxBarsForward:** Percorre a lista da série, a partir do candle atual(índice 0).
- MaxBarsBack:** Percorre a lista da série, a partir do primeiro candle criado(índice 0).
- NoPlot(NumeroPlot : Integer):** Remove determinado Plot.
- PaintBar(Cor : Integer):** Colore os candles e indicadores do gráfico.
- Lowest(SerieDeDados : Serie, Período : Integer):** Retorna o menor valor da série dentro no período.
- LowestBar(SerieDeDados : Serie, Período : Integer):** Retorna o índice da barra com o menor valor da série no período.
- MedianPrice:** Retorna a média entre a máxima e a mínima de determinado candle.
- Plot(Dado : Float):** Desenha o indicador de acordo com o gráfico.
- Plot2(Dado : Float):** Desenha o indicador de acordo com o gráfico.
- Plot3(Dado : Float):** Desenha o indicador de acordo com o gráfico.
- Plot4(Dado : Float):** Desenha o indicador de acordo com o gráfico.
- Range:** Retorna o valor de Máxima menos Mínima do determinado candle.
- RangeLeader:** Verifica se a barra atual é Range Leader.
- RGB(Red : Integer, Green : Integer, Blue : Integer):** Coloração a partir dos parâmetros RGB.
- SetPlotColor(NumeroPlot : Integer, Cor : Integer):** Altera a coloração de determinado Plot.
- SetPlotWidth(NumeroPlot : Integer, Espessura : Integer):** Altera a espessura de determinado Plot.
- TrueHigh:** Retorna o maior entre o máximo da barra e o fechamento da barra anterior.
- TrueLow:** Retorna o menor entre o mínimo da barra e o fechamento da barra anterior.
- TrueRange:** Retorna o valor do indicador TrueRange.
- TrueRangeCustom(Maxima : Float, Mínima : Float, Fechamento : Float):** Retorna o TrueRange de acordo com os dados informados pelo usuário.
- TypicalPrice:** Retorna o valor médio entre a máxima, mínima e fechamento de determinado candle.
- WeightedClose:** Retorna a média entre o ponto médio da barra e dois fechamentos.

Indicadores

- AvgSeparation(Período : Integer, TipoMedia : Integer):** Retorna o valor do indicador Afastamento Médio.
- AvgTrueRange(Período : Integer, TipoMedia : Integer):** Retorna o valor do indicador True Range.
- AccAgressSaldo(TipoVolume : Integer):** Retorna o valor do indicador TR - Acúmulo de Agressão - Saldo.
- AccuDistr:** Retorna o valor do indicador Acumulação/Distribuição.
- AccuDistrW:** Retorna o valor do indicador Acumulação/Distribuição Williams.



Documentação Módulo Estratégias

AdaptiveMovingAverage(Período : Integer, FastSC : Integer, SlowSC : Integer): Retorna o valor do indicador Adaptive Moving Average(AMV).

ADX(Período : Integer, PeríodoMedia : Integer): Retorna o valor do indicador ADX.

AgressionVolBalance: Retorna o valor do indicador TR - Volume de Agressão - Saldo.

AgressionVolBuy: Retorna o valor do indicador TR - Volume de Agressão - Compra.

AgressionVolSell: Retorna o valor do indicador TR - Volume de Agressão - Venda.

ArmsEaseOfMov(Media : Integer, TipoMedia : Integer): Retorna o valor do indicador Arms Ease of Movement.

AroonLin(Período : Integer)|Linha Desejada|: Retorna o valor do indicador Aroon Linha.

AroonOsc(Período : Integer): Retorna o valor do indicador Aroon Oscilador.

AvgAgrBuySell(AlertaVariacoes : Integer, TipoVolume : Integer, TipoDesenho: Integer)|Linha : Integer|: Retorna o valor do indicador TR - Agressão Média - Compra e Venda.

AvgAgrTotal(AlertaVariacoes : Integer, TipoVolume : Integer, TipoDesenho: Integer)|Linha : Integer|: Retorna o valor do indicador TR - Agressão Média - Total.

BalanceOfPower(Media : Integer, TipoMedia : Integer): Retorna o valor do indicador Balança do poder.

BearPower(Período : Integer): Retorna o valor do indicador Bear Power.

BollingerBands(Desvio : Float, Media : Integer, TipoMedia : Integer)|Linha : Integer|: Retorna o valor da linha da Banda de Bollinger de acordo com a linha desejada.

BollingerBandW(Desvio : Float, Media : Integer, TipoMedia : Integer): Retorna o valor do indicador Bollinger Band Width.

BollingerBPerc(Desvio : Float, Media : Integer, TipoMedia : Integer): Retorna o valor do indicador Bollinger b%.

BullPower(Período : Integer, PeríodoMedia : Integer, TipoMedia : Integer): Retorna o valor do indicador Bull Power.

Carmine(Risco : Integer, ModoCalculo : Integer, Período : Integer, Desvio : Float, UsarVWAP : Boolean, UsarAtr : Boolean): Retorna o dado do indicador Carmine.

CCI(Período : Integer): Retorna o valor do indicador CCI.

ChaikinMoneyFlow(Período : Integer): Retorna o valor do indicador Chaikin Money Flow.

ChainSetup: Retorna o dado do indicador ChainSetup.

ChaikinOsc(MediaLonga : Integer, MediaCurta : Integer): Retorna o valor do indicador Oscilador Chaikin.

DarvasBox|Linha : Integer|: Retorna o valor do indicador Darvas Box.

DecisionPoints(Tipo : Integer, Linha : Integer): Retorna o valor do indicador Pontos de Decisão.

DiDiIndex(MedRef : Integer, TipoMedRef : Integer, Med1 : Integer, TipoMed1 : Integer, Med2 : Integer, TipoMed2 : Integer)|Linha : Integer|: Retorna o valor do indicador DiDi Index.

DiPdIM(Período : Integer)|Linha : Integer|: Retorna o valor do indicador DI+/DI- de acordo com a linha desejada.

DonchianCH(Período : Integer)|Linha : Integer|: Retorna o valor do indicador Canal Donchian de



Documentação Módulo Estratégias

acordo com a linha desejada.

DTOscillator(PeriodoEstocastico : Integer, PeriodoSK : Integer, TipoSK : Integer, PeriodoSD : Integer, TipoSD : Integer)|Linha : Integer|: Retorna o valor do indicador DT Oscillator, de acordo com a linha desejada.

Envelope(Percentual : Float, PeriodoMedia : Integer, TipoMedia : Integer)|Linha : Integer|: Retorna o valor do indicador Envelope.

Euroinvest(Risco: Integer, ModoCalculo : Integer, Periodo : Integer, Desvio : Float, UsarVWAP : Boolean, UsarAtr : Boolean) Retorna o valor do indicador Euroinvest.

FastStochastic(Periodo : Integer): Retorna o valor do indicador Estocástico Rápido.

FinancialVol(VolumeProjetado : Boolean, Agressores : Boolean): Retorna o Valor do volume financeiro.

ForceIndex(Periodo : Integer, TipoMedia : Integer): Retorna o valor do indicador Force Index.

FrassonATR(Fator : Float, PeriodoMaxMin : Integer, PeriodoATR : Integer)|Linha : Integer|: Retorna o valor do indicador Frasson ATR.

FrassonVH(Fator : Float, PeriodoMaxMin : Integer, PeriodoVH : Integer)|Linha : Integer|: Retorna o valor do indicador Frasson VH.

FullStochastic(Periodo : Integer): Retorna o valor do indicador Estocástico Pleno.

FuraChao(Coeficiente : Float, Deslocamento : Integer): Retorna o valor do indicador Fura Chão.

FuraTeto(Coeficiente : Float, Deslocamento : Integer): Retorna o valor do indicador Fura Teto.

HeikinAshi(Media : Integer, TipoMedia : Integer)|Dado : Integer|: Retorna o valor do indicador HeikinAshi.

HiLoActivator(Periodo : Integer)|Linha : Integer|: Retorna o valor do HiLo Activator de acordo com a linha desejada.

HistVolatility(Periodo : Integer, TipoMedia : Integer): Retorna o valor do indicador Volatilidade Histórica.

HSI(Periodo : Integer): Retorna o valor do indicador IFR Índice de Força Harmônico (HSI).

HullMovingAverage(Periodo : Integer): Retorna o valor do indicador Hull Moving Average.

IchimokuCloud(TenkanSen : Integer, KijunSen : Integer, SenkouSpanB : Integer)|Linha : Integer|: Retorna o valor do indicador Ichimoku Cloud.

IFR(Periodo : Integer): Retorna o valor do indicador IFR.

ImpliedVolatility(ModeloTeorico : Boolean, TipoOpcao : Boolean): Retorna o cálculo do indicador Volatilidade Implícita, dependendo do tipo de cálculo utilizado.

KeltnerCH(Desvio : Float, Periodo : Integer, TipoMedia : Integer)|Linha : Integer|: Retorna o valor do indicador Keltner Channels, conforme com a linha desejada.

KVO(MediaLonga : Integer, MediaCurta : Integer, Sinal : Integer)|Dado : Integer|: Retorna o dado desejado do indicador KVO Linha & Histograma.

LSVolatilityIndex Retorna o dado desejado do indicador L&S Volatility Index.

MACD(MediaLonga : Integer, MediaCurta : Integer, Sinal : Integer)|Dado : Integer|: Retorna o dado desejado do indicador KVO Linha & Histograma.

Media(Periodo : Integer, TipoSerie : Serie): Retorna o dado do indicador Média Móvel(Aritmética).



Documentação Módulo Estratégias

MediaExp(Período : Integer, TipoSerie : Serie): Retorna o dado do indicador Média Móvel(Exponencial).

MFI: Retorna o valor do indicador Market Facilitation Index.

MIMA(Período : Integer): Retorna o dado do indicador PhiCube - MIMA.

Momentum(Período : Integer, Media : Integer, TipoMedia : Integer): Retorna o valor do indicador Momentum.

MomentumStochastic(Período : Integer): Retorna o valor do indicador Momento Estocástico.

MoneyFlow: Retorna o valor do indicador Money Flow.

MoneyFlowIndex(Período : Integer): Retorna o valor do indicador Money Flow Index de acordo com o período utilizado.

NelogicaBottomFinder|Dado : Integer|: Retorna o valor do indicador Nelogica Bottom Finder de acordo com a linha Desejada.

NelogicaPullBackFinder|Dado : Integer|: Retorna o valor do indicador Nelogica PullBack Finder de acordo com a linha desejada.

NelogicaWeisWave(Período : Integer): Retorna o valor do indicador Nelogica Weis Wave, conforme o período desejado.

OBV: Retorna o valor do indicador OBV.

OBVAvg: Retorna o valor do indicador OBV Ponderado.

OnBalanceTR: Retorna o valor do indicador On Balance True Range.

OpenInterest: Retorna o valor do indicador Contratos em aberto.

ParabolicSAR(Fator : Float, Limite : Float): Retorna o valor do indicador SAR Parabólico.

Phibo(Período : Integer): Retorna o valor do indicador PhiCube - Phibo Line.

Pivot(Normal : Boolean, TresLinhas : Boolean)|Linha : Integer|: Retorna o valor do indicador Pivot, de acordo com a linha selecionada.

PriceOsc(Media1 : Integer, TipoMedia1 : Integer, Media2 : Integer, TipoMedia2 : Integer) : Retorna o valor do indicador Oscilador de Preços.

PriceOscillator(SerieDados : Serie, ComprimentoRápido : Integer, ComprimentoLento : Integer): Retorna o valor do indicador Price Oscillator.

PriceVolumeTrend: Retorna o valor do indicador Tendência Preço/Volume.

PriorCote(Dado : Integer): Retorna o valor do indicador Prior Cote, de acordo com o dado desejado.

PTAX: Retorna o valor do indicador TR - PTAX.

PTAXFuturo: Retorna o valor do indicador TR - PTAX Futuro.

QuantityVol(VolumeProjetado : Boolean, Agressores : Boolean): Retorna o valor do indicador Volume Quantidade.

Rafi: Retorna o valor do indicador Rafi.

Ravi(MediaCurta : Integer, MediaLonga : Integer): Retorna o valor do indicador Ravi, de acordo com os períodos das médias desejadas.

RenkoVTwo(Período : Integer, Abertura : Float, Deslocamento : Integer): Retorna o dado do



Documentação Módulo Estratégias

indicador RenkoV2, conforme a linha desejada.

RSI(Período : Integer, Tipo : Integer): Retorna o valor do indicador IFR(RSI).

RsiStochastic(Período : Integer): Retorna o valor do indicador IFR Estocástico.

ROC(Período : Integer, Média : Integer, TipoMédia : Integer): Retorna o valor do indicador ROC.

SafeZoneDownTrend(Multiplicador : Float, Período : Integer, Deslocamento : Integer)
: Retorna o valor do indicador Stop SafeZone DownTrend.

SafeZoneUpTrend(Multiplicador : Float, Período : Integer, Deslocamento : Integer): Retorna o valor do indicador Stop SafeZone UpTrend.

Santo(Período : Integer)|Linha : Integer|: Retorna o valor do indicador PhiCube - Santo.

SlowStochastic(Período : Integer): Retorna o valor do indicador Estocástico Lento.

StopATR(Desvio : Float, Período : Integer, TipoMédia : Integer)|Dado : Integer|: Retorna o valor do indicador Stop ATR, de acordo com o dado desejado.

Tilson(Fator : Float, Média : Integer): Retorna o valor do indicador Tillson's T3 Moving Average.

TimeAgrBuySell(AlertaVariacoes : Integer): Retorna o valor do indicador TR - Tempo Agressão - Compra.

TimeAgrTotal(AlertaVariacoes : Integer): Retorna o valor do indicador TR - Tempo Agressão - Total.

TRIX(Média : Integer, TipoMédia : Integer): Retorna o valor do indicador TRIX.

TRIXM(Média : Integer, TipoMédia : Integer): Retorna o valor do indicador TRIXM.

TopBottomDetector(Período : Integer): Retorna o valor do indicador Detector de Topos e Fundos.

Trades: Retorna o valor do indicador Negócios.

TwoMVAggression: Retorna o dado do indicador 2MV Agressão.

TwoMVPower(Período1 : Integer, Período2 : Integer, Período3 : Integer, Média : Integer)
: Retorna o valor do indicador 2MV Power.

TwoMVStandard: Retorna o dado do indicador 2MV Padrão.

VSS(Multiplicador : Float, Média : Integer, Deslocamento : Integer): Retorna o valor do indicador VSS.

VWAP(Período : Integer): Retorna o valor do indicador VWAP, conforme a periodicidade desejada.

VWAPMonthly: Retorna o dado do indicador VWMA Mensal.

VWAPWeekly: Retorna o dado do indicador VWMA Semanal.

VWMA(Período : Integer): Retorna o dado do indicador VWMA, conforme o período desejado.

WAverage(TipoSerie : SeriePeríodo, Período : Integer): Retorna o dado do indicador Média Móvel(Ponderada).

WellesSum(Período : Integer, SerieReferencia : Serie, Offset : Integer): Retorna o dado do indicador WellesSum.

Williams(Período : Integer): Retorna o valor do indicador Williams %R, de acordo com o período desejado.

xAverage(TipoSerie : SeriePeríodo, Período : Integer): Retorna o dado do indicador Média Móvel(Exponencial).



Livro

- AskPrice:** Retorna o preço da melhor oferta de venda.
- AskSize:** Retorna a quantidade da melhor oferta de venda.
- BidPrice:** Retorna o preço da melhor oferta de compra.
- BidSize:** Retorna a quantidade da melhor oferta de compra.
- BookSpread:** Retorna a diferença entre o topo do livro.
- IsBMF:** Verifica se o ativo pertence ao segmento BMF.
- Lote:** Retorna a quantidade de contratos referente ao lote.
- MinPriceIncrement:** Retorna o incremento mínimo do preço.

Matemáticas

- ABS(Valor : Float):** Retorna o valor absoluto(sem sinal).
- Arctangent(Numero : Float):** Retorna o arcotangente(em graus) de um número.
- Ceiling(Numero : Float):** Retorna o menor inteiro maior que um número específico.
- Combination(Numero : Integer, QtdGrupos : Integer):** Retorna a quantidade de combinações, considerando um conjunto específico de números.
- Cos(Valor : Float):** Retorna o valor de um Cosseno em radianos.
- Cosine(Valor : Float):** Retorna o valor de um Cosseno em graus.
- Cotangent(Valor : Float):** Retorna o valor de uma Cotangente em graus.
- Cum(SerieDeDados : Serie):** Acumula o valor de uma série de dados.
- Exp(Valor : Float):** Retorna a enésima potência do número de Euler.
- ExpValue(Valor : Float):** Retorna o valor exponencial de um determinado número(e^x).
- ExtremePriceRatio:** Obtém o ratio das extremidades de um número determinado de barras.
- Factorial(Valor : Float):** Retorna o fatorial do valor estabelecido ($1*2*...n$).
- FastD(Periodo : Integer):** Retorna o valor de FastD do Oscilador Estocástico.
- FastK(Periodo : Integer):** Retorna o valor de FastK do Oscilador Estocástico.
- FastKCustom(PrecoH : Serie, PrecoL : Serie, PrecoC : Serie, Periodo : Integer):** Retorna o valor de FastK a partir de preços específicos.
- Floor(Valor : Float):** Retorna o maior inteiro, menor que um número específico.
- FracPortion(Valor : Float):** Retorna a parte fracionário de um número.
- GCD(Valor1 : Float, Valor2 : Float):** Retorna o maior divisor comum entre dois números.
- HarmonicMean(SerieDados : Serie, Periodo : Integer):** Retorna a média harmônica de uma série de dados baseada em um período.
- IntPortion(Valor : Float):** Retorna a parte inteira de um número.
- Log(Valor : Float):** Retorna o logaritmo natural(ln) de um número de um número.



Documentação Módulo Estratégias

- MidPoint(SerieDados : Serie, Período : Integer):** Retorna a média entre o maior e o menor valor encontrados no período.
- MinutesIntoWeek(DiaLimite : Integer, HoraLimite : Integer):** Retorna o valor de minutos desde a meia noite em horas.
- MinutesToTime(Minutos : Integer):** Retorna o valor de minutos desde a meia noite em horas.
- Mod(Dividendo : Integer, Divisor : Integer):** Retorna o resto da divisão entre dois números.
- MyPrice:** Retorna a média entre a máxima, mínima e fechamento.
- Neg(Numero : Float):** Retorna o valor negativo de um determinado número.
- NumUnits(Amnt : Integer, MinLot : Integer):** Retorna o número de contratos/ações de um certo investimento.
- PercentChange(SerieDados : Serie, Período : Integer):** Retorna a alteração percentual no preço do candle atual.
- PercentR(Comprimento : Integer):** Retorna uma porcentagem de onde o preço atual está, relacionado com a faixa de negociação avaliada.
- Permutation(Numero : Integer, NumeroObjetos : Integer):** Calcula um número de permutações.
- Pos(Valor : Float):** Retorna o valor absoluto(sem sinal).
- Power(Base : Float, Expoente : Integer):** Eleva valores nas determinadas potências.
- Random(Limite : Integer):** Retorna um valor aleatório de 0 até o limite.
- RateOfChange(SerieDados : Serie, Período : Integer):** Retorna a variação percentual de uma série.
- Round(Valor : Float):** Arredondamento de número.
- Round2Fraction(Valor : Float):** Arredonda o número para o valor mais próximo de um múltiplo do incremento mínimo de um ativo.
- Sign(Valor : Float):** Retorna um número inteiro, baseado no sinal de um número.
- Sin(Valor : Float):** Retorna o valor de Seno em radianos.
- Sine(Valor : Float):** Retorna o valor de Seno em graus.
- SlowD(Período : Integer):** Retorna o valor SlowD do Oscilador Estocástico.
- SlowK(Período : Integer):** Retorna o valor SlowK do Oscilador Estocástico.
- Sqrt(Valor : Float):** Retorna raiz quadrada dos valores.
- Square(Valor : Float):** Retorna o valor de um número ao quadrado.
- StdDevs(SerieDados : Serie, Período : Integer):** Calcula o desvio padrão de uma série de dados em um determinado período.
- Summation(SerieDados : Serie, Período : Integer):** Acumula o valor do preço de um determinado número de barras.
- Tangent(Valor : Float):** Retorna a tangente de um número em graus.
- TriAverage(SerieDados : Serie, Período : Integer):** Calcula a média triangular de uma série de dados, dentro de um certo período.
- UlcerIndex(SerieDados : Serie, Período : Integer):** Retorna o valor do indicador UlcerIndex.



UltimateOscillator(PeríodoCurto : Integer, PeríodoMedio : Integer, PeríodoLongo : Integer)
: Retorna o valor do Ultimate Oscillator desenvolvido por Larry Williams.

Volatilidade(Período : Integer): Retorna a volatilidade de determinado período.

VolumeOsc(PeríodoMediaRapida : Integer, PeríodoMediaLenta : Integer): Retorna a diferença entre a média aritmética rápida e a média aritmética lenta da série Volume.

VolumeROC(Período : Integer): Retorna VolumeROC baseado em um número de barras.

Opções

Delta(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer): Retorna o valor de Delta.

Gamma(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer): Retorna o valor de Gamma.

Rho(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer): Retorna o valor de Gamma.

Theta(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer): Retorna o valor de Theta.

Vega(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer): Retorna o valor de Vega.

Screening

Select: Seleciona um ativo para mostrar no Screening.



Lista de Funções

Lista com a descrição da sintaxe de cada função disponível na linguagem NTSL (Nelógica Trading System Language):

Alarme

Função Alert

Descrição:

A função **Alert** tem como finalidade gerar um alarme ao usuário.

Sintaxe:

Alert(Cor : Integer)

Parâmetros:

Cor: Determina a cor desejada para o popup de notificação, no momento de execução do alarme.

Observação: Uma cor pode ser determinada a partir de uma função **RGB**, ou através de uma **String** com o nome da cor.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo a seguir, será emitido um alarme(e um popup com coloração vermelha) caso a média móvel aritmética de 9 períodos for menor que a média de 21.

```
if(Media(9) < Media(21))  
  
    then Alert(clRed);
```



Back-Testing

Função BuyAtMarket

Descrição:

A função **BuyAtMarket** tem como funcionalidade realizar uma ordem de compra à mercado.

Sintaxe:

BuyAtMarket

Parâmetros:

Sem parâmetros.

Retorno:

Void: Sem retorno.

Exemplos

No exemplo, caso a mínima do candle atual for igual a do candle anterior, será realizada uma simulação de ordem à mercado.

```
if(Low = Low[1])
```

```
    then BuyAtMarket;
```



Função BuyLimit

Descrição:

A função **BuyLimit** possui como finalidade enviar uma ordem de compra do tipo limite.

Sintaxe:

BuyLimit(Preço : Float)

Parâmetros:

Preço: Preço para a inserção da ordem.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso a média exponencial de 9 períodos for maior que a de 21, será criada a ordem de compra considerando o último preço.

```
if(MediaExp(9, Close) > MediaExp(21, Close))  
    then BuyLimit(Close);
```



Função BuyPosition

Descrição:

A função **BuyPosition** retorna o tamanho da posição da compra.

Sintaxe:

BuyPosition

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "buyPos" irá receber o total da posição de compra.

```
buyPos := BuyPosition;
```



Função BuyPrice

Descrição:

A função **BuyPrice** retorna o preço de compra da posição.

Sintaxe:

BuyPrice

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "pCompra" o valor de compra da posição.

```
pCompra := BuyPrice;
```



Função BuyStop

Descrição:

A função **BuyStop** tem como funcionalidade realizar uma operação de Ordem Stop, igual à utilizada na [Boleta Stop](#).

Sintaxe:

BuyStop(Stop : Float, Limite : Float)

Parâmetros:

Stop: Valor do tipo Float que será o gatilho da ordem;

Limite: Valor do tipo Float que será o limite de preço aceito para execução.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso a mínima do candle atual for igual a do candle anterior, será realizada uma simulação de Ordem Stop.

```
if(Low = Low[1])
```

```
    then BuyStop(10.00, 10.02);
```



Função BuyToCoverAtMarket

Descrição:

A função **BuyToCoverAtMarket** tem como funcionalidade realizar o envio de uma ordem de compra à mercado, caso exista uma posição de venda.

Sintaxe:

BuyToCoverAtMarket

Parâmetros:

Sem parâmetros.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo abaixo, caso a mínima atual seja menor que a mínima anterior e exista uma posição de venda, será executada uma ordem à mercado, utilizando a função **BuyToCoverAtMarket** para fechar a operação.

```
if(Low > Low[1] and Issold)
```

```
    then BuyToCoverAtMarket;
```



Função BuyToCoverLimit

Descrição:

A função **BuyToCoverLimit** possui como finalidade enviar uma ordem de compra, do tipo limite, para finalizar a operação.

Sintaxe:

BuyToCoverLimit(Preço : Float)

Parâmetros:

Preço: Preço para a inserção da ordem.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso exista uma posição de venda, será inserida a ordem no preço de fechamento para zerar a posição.

```
if(IsSold) then
```

```
    BuyToCoverLimit(Close);
```




Função BuyToCoverStop

Descrição:

A função **BuyToCoverStop** tem como funcionalidade enviar uma ordem do tipo Stop de compra, caso exista uma posição de venda no determinado ativo.

Sintaxe:

BuyToCoverStop(Stop : Float, Limite : Float)

Parâmetros:

Stop: Valor que será o gatilho da ordem.

Limite: Valor que será o limite do preço aceito para execução.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso a mínima do candle atual for igual a do candle anterior e exista uma posição de venda, será realizada uma simulação de ordem Stop utilizando a função BuyToCoverStop.

```
if(Low = Low[1] and IsSold) then
```

```
    BuyToCoverStop(10.00, 10.02);
```



Função CancelPendingOrders

Descrição:

A função **CancelPendingOrders** possui como recurso efetuar o cancelamento de ordens pendentes.

Sintaxe:

CancelPendingOrders

Parâmetros:

Sem parâmetros.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso existam ordens pendentes, ocorrerá o cancelamento.

```
if(HasPendingOrders) then
```

```
    CancelPendingOrders;
```



Função ClosePosition

Descrição:

A função **ClosePosition** envia ordens para encerrar todas as posições.

Sintaxe:

ClosePosition

Parâmetros:

Sem parâmetros.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo a seguir, se o valor armazenado na variável **x** for maior que o valor armazenado na variável **y**, todas as posições serão fechadas.

```
if(x > y) then
```

```
    ClosePosition;
```



Função HasPendingOrders

Descrição:

A função **HasPendingOrders** retorna se há ordens pendentes.

Sintaxe:

HasPendingOrders

Parâmetros:

Sem parâmetros.

Retorno:

Boolean

Exemplos:

No exemplo, caso existam ordens pendentes, ocorrerá o cancelamento.

```
if(HasPendingOrders) then
```

```
    CancelPendingOrders;
```



Função IsBought

Descrição:

A função **IsBought** tem como funcionalidade determinar se há uma posição de compra em aberto.

Sintaxe:

IsBought

Parâmetros:

Sem parâmetros.

Retorno:

Boolean:

False - **Não há posição em aberto.**

True - **Há posição em aberto.**

Exemplos:

No exemplo, caso exista uma posição de compra, haverá a aplicação de uma coloração(verde).

```
if(IsBought) then
```

```
    PaintBar(clGreen);
```



Função IsSold

Descrição:

A função **IsSold** tem como funcionalidade determinar se há uma posição de venda em aberto.

Sintaxe:

IsSold

Parâmetros:

Sem parâmetros.

Retorno:

Boolean:

False - **Não há posição em aberto.**

True - **Há posição em aberto.**

Exemplos:

No exemplo, caso exista uma posição de venda, haverá a aplicação de uma coloração(vermelha).

```
if(IsSold) then
```

```
    PaintBar(clRed);
```



Função ReversePosition

Descrição:

A função **ReversePosition** tem como funcionalidade realizar o envio de ordens, a fim de inversão da posição.

Sintaxe:

ReversePosition

Parâmetros:

Sem parâmetros.

Retorno:

Void: Sem retorno.

Exemplos

No exemplo, caso a mínima do candle atual for menor a do candle anterior, e exista uma posição de compra, será realizada uma simulação de inversão.

```
if(Low < Low[1] and (IsBought = True)  
  
    then ReversePosition;
```



Função SellPosition

Descrição:

A função **SellPosition** retorna o tamanho da posição de venda.

Sintaxe:

SellPosition

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "sellPos" irá receber o total da posição de venda

```
sellPos := SellPosition;
```




Função SellPrice

Descrição:

A função **SellPrice** retorna o preço de venda da posição.

Sintaxe:

SellPrice

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "pVenda" o valor de venda da posição.

```
pVenda := SellPrice;
```



Função SellToCoverAtMarket

Descrição:

A função **SellToCoverAtMarket** tem como funcionalidade realizar o envio de uma ordem de venda à mercado, caso exista uma posição de compra.

Sintaxe:

SellToCoverAtMarket

Parâmetros:

Sem parâmetros.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo abaixo, caso a máxima atual seja maior que a máxima anterior e exista uma posição de compra, será executada uma ordem à mercado, utilizando a função **SellToCoverAtMarket** para fechar a operação.

```
if(High > High[1]) and (Isbought)
```

```
    then SellToCoverAtMarket;
```



Função SellToCoverLimit

Descrição:

A função **SellToCoverLimit** possui como finalidade enviar uma ordem de venda, do tipo limite, para finalizar a operação.

Sintaxe:

SellToCoverLimit(Preço : Float)

Parâmetros:

Preço: Preço para a inserção da ordem.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso exista uma posição de compra, será inserida a ordem no preço de fechamento para zerar a posição.

```
if(IsBought) then
```

```
    SellToCoverLimit(Close);
```



Função SellToCoverStop

Descrição:

A função **SellToCoverStop** tem como funcionalidade enviar uma ordem do tipo Stop de venda, caso exista uma posição de compra no determinado ativo.

Sintaxe:

SellToCoverStop(Stop : Float, Limite : Float)

Parâmetros:

Stop: Valor que será o gatilho da ordem.

Limite: Valor que será o limite do preço aceito para execução.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso a máxima do candle atual for maior que a do candle anterior e exista uma posição de compra, será realizada uma simulação de ordem Stop utilizando a função SellToCoverStop.

```
if(High > High[1] and Isbough) then
```

```
    SellToCoverStop(9.98, 9.95);
```



Função SellShortAtMarket

Descrição:

A função **SellShortAtMarket** tem como funcionalidade o envio de uma ordem à mercado de venda, caso não exista posição em aberto.

Sintaxe:

SellShortAtMarket

Parâmetros:

Sem Parâmetros.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo abaixo, será criada uma posição de venda caso a média móvel de 9 períodos for menor que a média de 21.

```
if(Media(9) < Media(21))  
  
    then sellshortatmarket;
```



Função SellShortLimit

Descrição:

A função **SellShortLimit** possui como finalidade enviar uma ordem de venda do tipo limite.

Sintaxe:

SellShortLimit(Preço : Float)

Parâmetros:

Preço: Preço para a inserção da ordem.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso a média exponencial de 9 períodos for menor que a de 21, será criada a ordem de venda considerando o último preço.

```
if(MediaExp(9, Close) < MediaExp(21, Close))  
    then SellShortLimit(Close);
```



Função SellShortStop

Descrição:

A função **SellShortStop** tem como funcionalidade enviar uma ordem de venda do tipo Stop, assim como utilizada na Boleta Stop, a fim de abrir a posição de venda em determinado ativo.

Sintaxe:

SellShortStop(Stop : Float, Limite : Float)

Parâmetros:

Stop: Valor do tipo Float que será o gatilho da ordem;

Limite: Valor do tipo Float que será o limite de preço aceito para execução.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, caso a máxima atual for maior do que a anterior, será realizada uma simulação de ordem Stop utilizando a função SellShortStop.

```
if(High > High[1])
```

```
    then SellShortStop(5.28, 5.20);
```



Função SendOrder

Descrição:

A partir da função **SendOrder**, pode-se enviar ordens customizadas, determinando o lado, tipo e quantidade.

Sintaxe:

SendOrder(Lado : Integer, Tipo : Integer, Quantidade : Integer, Limite : Float, Stop : Float)

Parâmetros:

Lado: Determina o lado:

osBuy - **Compra**

osSell - **Venda**

Tipo: Tipo da ordem:

otMarket - **À mercado**

otLimit - **Limite**

otStopLimit - **Stop**

Quantidade: Quantidade de contratos.

Limite: Limite do preço aceito para execução.

Stop: Valor que será o gatilho da ordem.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, será efetuada a inserção de uma ordem de compra, tipo Stop, no preço 17.44, com limite de execução até o nível 17.50.

```
SendOrder(osBuy, otStopLimit, 5, 17.50, 17.44);
```




Calendário

Função BarDuration

Descrição:

A função **BarDuration** retorna, em minutos, a duração da barra atual.

Sintaxe:

BarDuration

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "temp" irá receber o dado da função BarDuration.

```
temp := BarDuration;
```



Função CalcDate

Descrição:

A função **CalcDate** retorna um valor o qual representa uma data deslocada, obtida ao adicionar ou subtrair dias de uma data de referência.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

CalcDate(DataReferencia : Integer, DiasDeslocamento : Integer)

Parâmetros:

DataReferencia: Determina a data que será utilizada como base para o deslocamento.

DiasDelocamento: Determina quantos dias serão adicionados ou subtraídos da data de referência.

Retorno:

Integer

Exemplos:

No exemplo abaixo, será visualizada, graficamente, a data resultante ao subtrair 14 dias da data de 26/10/2018.

```
Plot(CalcDate(1181026, -14));
```



Funcao CalcTime

Descrição:

A função **CalcTime** retorna um valor o qual representa uma hora deslocada, obtida ao adicionar ou subtrair minutos de uma hora de referência.

Observação: Horas são representadas pelo tipo de dado "Integer", no formato(24 horas): HorasMinutos.

Sintaxe:

CalcTime(HoraReferencia : Integer, MinutosDeslocamento : Integer)

Parâmetros:

HoraReferencia: Determina a hora que será utilizada como base para o deslocamento;

MinutosDeslocamento: Determina quantos minutos serão adicionados ou subtraídos da hora de referência.

Retorno:

Integer

Exemplos:

No exemplo abaixo, será visualizada, graficamente, a hora resultante ao deslocar 65 minutos a partir das 14h(Resultado: 1505).

```
Plot(CalcTime(1400, 65));
```



Função CloseD

Descrição:

A função **CloseD** tem como finalidade retornar o valor de fechamento de um número determinado de dias atrás.

Sintaxe:

CloseD(QuantidadeDiasAnteriores : Integer)

Parâmetros:

QuantidadeDiasAnteriores: Determina a quantidade desejada de dias anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "nFechamento" o valor de fechamento de dois dias anteriores ao dia atual.

```
nFechamento := CloseD(2);
```



Função CloseM

Descrição:

A função **CloseM** tem como finalidade retornar o valor de fechamento de um número determinado de meses atrás.

Sintaxe:

CloseM(QuantidadeMesesAnteriores : Integer)

Parâmetros:

QuantidadeMesesAnteriores: Determina a quantidade desejada de meses anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "mFechamento" o valor de fechamento de três meses anteriores ao mês atual.

```
mFechamento := CloseM(3);
```



Função CloseW

Descrição:

A função **CloseW** tem como finalidade retornar o valor de fechamento de um número determinado de semanas atrás.

Sintaxe:

CloseW(QuantidadeSemanasAnteriores : Integer)

Parâmetros:

QuantidadeSemanasAnteriores: Determina a quantidade desejada de semanas anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "wFechamento" o valor de fechamento de duas semanas anteriores à semana atual.

```
wFechamento := CloseW(2);
```



Função CloseY

Descrição:

A função **CloseY** tem como finalidade retornar o valor de fechamento de um número determinado de anos atrás.

Sintaxe:

CloseY(QuantidadeAnosAnteriores : Integer)

Parâmetros:

QuantidadeAnosAnteriores: Determina a quantidade desejada de anos anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "yFechamento" o valor de fechamento do ano anterior.

```
yFechamento := CloseY(1);
```



Função CurrentDate

Descrição:

A função **CurrentDate** possui como finalidade retornar a data atual do sistema.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

CurrentDate

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "nData" a data do dia atual.

```
nData := CurrentDate;
```




Função CurrentTime

Descrição:

A função **CurrentTime** possui como finalidade retornar a hora atual do sistema.

Observação: Horas são representadas pelo tipo de dado "Integer", no formato: HHMM.

Sintaxe:

CurrentTime

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "nHora" a hora atual.

```
nHora := CurrentTime;
```



Função Date

Descrição:

A função **Date** possui como finalidade retornar a data do candle que está sendo analisado.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

Date

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "nData" a data do candle atual.

```
nData := Date;
```



Função DayOfMonth

Descrição:

A função **DayOfMonth** retorna o dia do mês de uma data específica.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

DayOfMonth(Data : Integer)

Parâmetros:

Data: Data para obter o dia.

Retorno:

Integer

Exemplos:

No exemplo, a variável "dAtual" irá receber o dia atual.

```
dAtual := DayOfMonth(CurrentDate);
```



Função DayOfWeek

Descrição:

A função **DayOfWeek** retorna o dia da semana de uma data específica.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

DayOfWeek(Data : Integer)

Parâmetros:

Data: Data para obter o dia da semana.

Retorno:

Integer:

Referência de dias da semana:

- 0 - **Domingo**
- 1 - **Segunda**
- 2 - **Terça**
- 3 - **Quarta**
- 4 - **Quinta**
- 5 - **Sexta**
- 6 - **Sábado**

Exemplos:

No exemplo, a variável "dAtual" irá receber o dia da semana atual.

```
dAtual := DayOfWeek(CurrentDate);
```



Função DaysToExpiration

Descrição:

A função **DaysToExpiration** é uma função a quantidade de dias úteis restantes até a terceira sexta-feira de um determinado mês e ano.

Sintaxe:

DaysToExpiration(Mes : Integer, Ano : Integer)

Parâmetros:

Mes: Determina o mês que se deseja a informação:

- 1 - **Janeiro**
- 2 - **Fevereiro**
- 3 - **Março**
- 4 - **Abril**
- 5 - **Maio**
- 6 - **Junho**
- 7 - **Julho**
- 8 - **Agosto**
- 9 - **Setembro**
- 10 - **Outubro**
- 11 - **Novembro**
- 12 - **Dezembro**

Ano: Determina o ano desejado para análise, onde deverá estar no formato: 1AnoDesejado.

Retorno:

Integer

Exemplos:

No exemplo abaixo, a variável "nQtdDias" irá receber a quantidade de dias úteis até a terceira sexta-feira de dezembro/2016.

```
nQtdDias := DaysToExpiration(12, 116);
```



Função ELDate

Descrição:

A função **ELDate** possui como finalidade retorna uma data em EasyLanguage format(YYYYMMDD).

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

ELDate(Ano : Integer, Mes : Integer, Dia : Integer)

Parâmetros:

Ano: Ano no formato YYYY.

Mes: Mes no formato MM.

Dia: Dia no formato DD.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "nData" a data de 13/11/2018 no formato: 1181113.

```
nData := ELDate(2018, 11, 13);
```



Função ELDate_Consol

Descrição:

A função **ELDate_Consol** possui como finalidade converter uma data YYYYMMDD em EasyLanguage format(YYYYMMDD)..

Sintaxe:

ELDate_Consol(Data : Integer)

Parâmetros:

Data: Data no formato YYYYMMDD.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "nData" a data de 2018/11/13 no formato: 1181113.

```
nData := ELDate_Consol(20181113);
```



Função FindBar

Descrição:

A função **FindBar** retorna o índice de uma barra através de uma data e hora, onde a contagem é iniciada a partir do candle atual(índice 0).

Observações:

Representação de datas: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Representação de horas: Horas são representadas pelo tipo de dado "Integer", no formato: HHMM.

Sintaxe:

FindBar(Data : Integer, Hora : Integer)

Parâmetros:

Data: Data do candle.

Hora: Hora do candle específico.

Retorno:

Integer

Exemplos:

No exemplo, a variável "n" irá receber o índice do candle relacionado com o horário 11h10 do dia atual.

```
n := FindBar(CurrentDate, 1110);
```




Função Friday

Descrição:

A função **Friday** retorna o número 5, representando o dia da semana: sexta-feira.

Sintaxe:

Friday

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "nDia" irá receber número 5, referente ao dia específico da semana.

```
nDia := Friday;
```



Função HighD

Descrição:

A função **HighD** tem como finalidade retornar o valor de máxima de um número determinado de dias atrás.

Sintaxe:

HighD(QuantidadeDiasAnteriores : Integer)

Parâmetros:

QuantidadeDiasAnteriores: Determina a quantidade desejada de dias anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "dMaxima" o valor de máxima de dois dias anteriores ao dia atual.

```
dMaxima := HighD(2);
```



Função HighM

Descrição:

A função **HighM** tem como finalidade retornar o valor de máxima de um número determinado de meses atrás.

Sintaxe:

HighM(QuantidadeMesesAnteriores : Integer)

Parâmetros:

QuantidadeMesesAnteriores: Determina a quantidade desejada de meses anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "mMaxima" o valor de máxima de cinco meses anteriores ao mês atual.

```
mMaxima := HighM(5);
```



Função HighW

Descrição:

A função **HighW** tem como finalidade retornar o valor de máxima de um número determinado de semanas atrás.

Sintaxe:

HighW(QuantidadeSemanasAnteriores : Integer)

Parâmetros:

QuantidadeSemanasAnteriores: Determina a quantidade desejada de semanas anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "wMaxima" o valor de máxima de sete semanas anteriores à semana atual.

```
wMaxima := HighW(7);
```



Função HighY

Descrição:

A função **HighY** tem como finalidade retornar o valor de máxima de um número determinado de anos atrás.

Sintaxe:

HighY(QuantidadeAnosAnteriores : Integer)

Parâmetros:

QuantidadeAnosAnteriores: Determina a quantidade desejada de anos anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "yMaxima" o valor de máxima de dois anos anteriores ao atual.

```
yMaxima := HighY(2);
```



Função LowD

Descrição:

A função **LowD** tem como finalidade retornar o valor de mínima de um número determinado de dias atrás.

Sintaxe:

LowD(QuantidadeDiasAnteriores : Integer)

Parâmetros:

QuantidadeDiasAnteriores: Determina a quantidade desejada de dias anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "dMinima" o valor de mínima de três dias anteriores ao dia atual.

```
dMinima := LowD(3);
```



Função LowM

Descrição:

A função **LowM** tem como finalidade retornar o valor de mínima de um número determinado de meses atrás.

Sintaxe:

LowM(QuantidadeMesesAnteriores : Integer)

Parâmetros:

QuantidadeMesesAnteriores: Determina a quantidade desejada de meses anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "mMinima" o valor de mínima do mês anterior.

```
mMinima := LowM(1);
```



Função LowW

Descrição:

A função **LowW** tem como finalidade retornar o valor de mínima de um número determinado de semanas atrás.

Sintaxe:

LowW(QuantidadeSemanasAnteriores : Integer)

Parâmetros:

QuantidadeSemanasAnteriores: Determina a quantidade desejada de semanas anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "wMinima" o valor de mínima de sete semanas anteriores à semana atual.

```
wMinima := LowM(7);
```




Função LowY

Descrição:

A função **LowY** tem como finalidade retornar o valor de mínima de um número determinado de anos atrás.

Sintaxe:

LowY(QuantidadeAnosAnteriores : Integer)

Parâmetros:

QuantidadeAnosAnteriores: Determina a quantidade desejada de anos anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "yMinima" o valor de mínima de dois anos anteriores ao atual.

```
yMinima := LowY(2);
```



Função LastCalcDate

Descrição:

A função **LastCalcDate** retorna a data do último candle completo formado dentro do gráfico.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

LastCalcDate

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo abaixo, será colocado diretamente no gráfico a data dos candles de acordo com o andamento do mercado.

Plot>LastCalcDate);



Função LastCalcTime

Descrição:

A função **LastCalcTime** retorna a hora do último candle completo formado dentro do gráfico, no formato 24h(HHMM).

Sintaxe:

LastCalcTime

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo abaixo, será colocado diretamente no gráfico a hora de fechamento dos candles, de acordo com o andamento do mercado.

```
Plot>LastCalcTime);
```



Função LastDayOfMonth

Descrição:

A função **LastDayOfMoth** retorna o último dia do mês de referência.

Sintaxe:

LastDayOfMonth(MesReferencia : Integer)

Parâmetros:

MesReferencia: Determina o mês de referência, de Janeiro(1) a Dezembro(12).

Retorno:

Integer

Exemplos:

No exemplo abaixo, será atribuído à variável "nDia" o último dia(30) do mês de Setembro.

```
nDia := LastDayOfMonth(9);
```



Função Monday

Descrição:

A função **Monday** retorna o número 1, representando o dia da semana: segunda-feira.

Sintaxe:

Monday

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "nDia" irá receber número 1, referente ao dia específico da semana.

```
nDia := Monday;
```



Função Month

Descrição:

A função **Month** retorna o mês de uma data específica.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

Month(Data : Integer)

Parâmetros:

Data: Data para obter o mês.

Retorno:

Integer

Exemplos:

No exemplo, a variável "mAtual" irá receber o mês atual.

```
mAtual := Month(CurrentDate);
```



Função Next3rdFriday

Descrição:

A função **Next3rdFriday** retorna a quantidade de dias úteis restantes até a terceira sexta-feira de determinado mês subsequente.

Sintaxe:

Next3rdFriday(Mes : Integer)

Parâmetros:

Mes: Determina o mês que se deseja a informação, onde a contagem inicia-se a partir do mês atual(0).

Retorno:

Integer

Exemplos:

No exemplo abaixo, a variável "nSexta" irá receber a quantidade de dias úteis até a terceira sexta-feira do mês seguinte.

```
nSexta := Next3rdFriday(1);
```



Função OpenD

Descrição:

A função **OpenD** tem como finalidade retornar o valor de abertura de um número determinado de dias atrás.

Sintaxe:

OpenD(QuantidadeDiasAnteriores : Integer)

Parâmetros:

QuantidadeDiasAnteriores: Determina a quantidade desejada de dias anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "dAbertura" o valor de abertura de seis dias anteriores ao dia atual.

```
dAbertura := OpenD(6);
```




Função OpenM

Descrição:

A função **OpenM** tem como finalidade retornar o valor de abertura de um número determinado de meses atrás.

Sintaxe:

OpenM(QuantidadeMesesAnteriores : Integer)

Parâmetros:

QuantidadeMesesAnteriores: Determina a quantidade desejada de meses anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "mAbertura" o valor de abertura do mês anterior ao atual.

```
mAbertura := OpenM(1);
```



Função OpenW

Descrição:

A função **OpenW** tem como finalidade retornar o valor de abertura de um número determinado de semanas atrás.

Sintaxe:

OpenW(QuantidadeSemanasAnteriores : Integer)

Parâmetros:

QuantidadeSemanasAnteriores: Determina a quantidade desejada de semanas anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "wAbertura" o valor de abertura de cinco semanas anteriores à semana atual.

```
wAbertura := OpenW(5);
```



Função OpenY

Descrição:

A função **OpenY** tem como finalidade retornar o valor de abertura de um número determinado de anos atrás.

Sintaxe:

OpenY(QuantidadeAnosAnteriores : Integer)

Parâmetros:

QuantidadeAnosAnteriores: Determina a quantidade desejada de anos anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "yAbertura" o valor de abertura do ano anterior ao atual.

```
yAbertura := OpenY(1);
```



Função RS_BarsPerDay

Descrição:

A função **RS_BarsPerDay** retorna o número estimado de barras de determinada periodicidade(em minutos). Caso o período seja diário, será obtido 1 como retorno, e, para períodos em minutos, ocorrerá a divisão do número total de minutos em um dia(1440) pela periodicidade selecionada.

Sintaxe:

RS_BarsPerDay

Parâmetros:

Sem parâmetros.

Retorno:

Inteiro

Exemplos:

No exemplo, será atribuído à variável "n" o retorno da chamada de função.

```
n := RS_BarsPerDay;
```



Função Saturday

Descrição:

A função **Saturday** retorna o número 6, representando o dia da semana: sábado.

Sintaxe:

Saturday

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "nDia" irá receber número 6, referente ao dia específico da semana.

```
nDia := Saturday;
```



Função Sunday

Descrição:

A função **Sunday** retorna o número 0, representando o dia da semana: domingo.

Sintaxe:

Sunday

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "nDia" irá receber número 0, referente ao dia específico da semana.

```
nDia := Sunday;
```



Função Thursday

Descrição:

A função **Thursday** retorna o número 4, representando o dia da semana: quinta-feira.

Sintaxe:

Thursday

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "nDia" irá receber número 4, referente ao dia específico da semana.

```
nDia := Thursday;
```



Função BarAnnualization

Descrição:

A função **BarAnnualization** retorna o fator de anualização(raiz quadrada) baseado no intervalo da barra(diário = 365, semanal = 52, mensal = 12).

Sintaxe:

BarAnnualization

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nBA" irá receber o dado de retorno da função BarAnnualization.

```
nBA := BarAnnualization;
```




Função Bartype

Descrição:

A função **Bartype** retorna um código numérico referente ao período utilizado.

Sintaxe:

Bartype

Parâmetros:

Sem parâmetros.

Retorno:

Inteiro

Retornos:

- 1 - **Outros**
- 1 - **Intraday**
- 2 - **Diário**
- 3 - **Semanal**
- 4 - **Mensal**

Exemplos:

No exemplo, a variável "n" irá receber um inteiro referente ao tempo determinado.

```
n := Bartype;
```



Função Time

Descrição:

A função **Time** possui como finalidade retornar a hora de fechamento do candle analisado.

Observação: Horas são representadas pelo tipo de dado "Integer", no formato: HHMM.

Sintaxe:

Time

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "nHora" a hora de fechamento do candle.

```
nHora := Time;
```



Função TimeToMinutes

Descrição:

A função **TimeToMinutes** possui como finalidade efetuar a conversão de um horário em minutos.

Sintaxe:

TimeToMinutes(Hora : Integer)

Parâmetros:

Hora: Hora para a conversão.

Retorno:

Integer

Exemplos:

No exemplo, a variável "minutos" irá receber a conversão em minutos(750) da hora: 12h30.

```
minutos := TimeToMinutes(1230);
```



Função Tuesday

Descrição:

A função **Tuesday** retorna o número 2, representando o dia da semana: terça-feira.

Sintaxe:

Tuesday

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "nDia" irá receber número 2, referente ao dia específico da semana.

```
nDia := Tuesday;
```



Função VolumeD

Descrição:

A função **VolumeD** tem como finalidade retornar o volume financeiro de um número determinado de dias atrás.

Sintaxe:

VolumeD(QuantidadeDiasAnteriores : Integer)

Parâmetros:

QuantidadeDiasAnteriores: Determina a quantidade desejada de dias anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "dVolume" o volume de seis dias anteriores ao dia atual.

```
dVolume := VolumeD(6);
```



Função VolumeM

Descrição:

A função **VolumeM** tem como finalidade retornar o volume financeiro de um número determinado de meses atrás.

Sintaxe:

VolumeM(QuantidadeMesesAnteriores : Integer)

Parâmetros:

QuantidadeMesesAnteriores: Determina a quantidade desejada de meses anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "mAbertura" o volume do mês anterior ao atual.

```
mVolume := VolumeM(1);
```



Função VolumeW

Descrição:

A função **VolumeW** tem como finalidade retornar o volume financeiro de um número determinado de semanas atrás.

Sintaxe:

VolumeW(QuantidadeSemanasAnteriores : Integer)

Parâmetros:

QuantidadeSemanasAnteriores: Determina a quantidade desejada de semanas anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "wVolume" o valor volume de cinco semanas anteriores à semana atual.

```
wVolume := VolumeW(5);
```



Função VolumeY

Descrição:

A função **VolumeY** tem como finalidade retornar o volume financeiro de um número determinado de anos atrás.

Sintaxe:

VolumeY(QuantidadeAnosAnteriores : Integer)

Parâmetros:

QuantidadeAnosAnteriores: Determina a quantidade desejada de anos anteriores.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "yVolume" o volume do ano anterior ao atual.

```
yVolume := VolumeY(1);
```




Função Wednesday

Descrição:

A função **Wednesday** retorna o número 3, representando o dia da semana: quarta-feira.

Sintaxe:

Wednesday

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, a variável "nDia" irá receber número 3, referente ao dia específico da semana.

```
nDia := Wednesday;
```



Função Year

Descrição:

A função **Year** retorna o ano de uma data específica.

Observação: Datas são representadas pelo tipo de dado "Integer", no formato: 1AnoMêsDia.

Sintaxe:

Year(Data : Integer)

Parâmetros:

Data: Data para obter o ano.

Retorno:

Integer

Exemplos:

No exemplo, a variável "yAtual" irá receber o ano atual.

```
yAtual := Year(CurrentDate);
```



Candlestick

Função C_3WhSolds_3BlkCrows

Descrição:

A função **C_3WhSolds_3BlkCrows** identifica a ocorrência de dois tipos de candles: **3 White Soldiers** e **3 Black Crows**.

Sintaxe:

C_3WhSolds_3BlkCrows(Comprimento : Integer, Fator : Integer, var o3WhiteSoldiers : Integer, var o3BlackCrows : Integer)

Parâmetros:

Comprimento: Tamanho utilizado para calcular a média do corpo do candle.

Fator: Determina quantas vezes a sombra do candle deve ser maior que o seu corpo.

o3WhiteSoldiers: Variável para identificação de padrão(3 White Soldiers).

o3BlackCrows: Variável para identificação de padrão(3 Black Crows).

Retorno:

Integer:

Identificação(retorno função):

0 - **Algum dos padrões foi identificado.**

1 - **Nenhuma padrão identificado.**

Identificação(retorno variável: o3WhiteSoldiers):

0 - **Padrão 3 White Soldiers não foi identificado.**

1 - **Padrão 3 White Soldiers identificado.**

Identificação(retorno variável: o3BlackCrows):

0 - **Padrão 3 Black Crows não foi identificado.**

1 - **Padrão 3 Black Crows identificado.**

Exemplos:

No exemplo, caso seja identificado algum dos padrões(3 White Soldiers ou 3 Black Crows), considerando 9(Comprimento) e 2(Fator), será aplicada uma coloração(vermelha).

```
aux := C_3WhSolds_3BlkCrows(9, 2, o3WhiteSoldiers, o3BlackCrows);
```

```
if(aux = 1) then
```

```
    PaintBar(clRed);
```



Função C_BullEng_BearEng

Descrição:

A função **C_BullEng_BearEng** identifica a ocorrência de dois tipos de candles: **Bullish Engulfing** e **Bearish Engulfing**.

Sintaxe:

C_BullEng_BearEng(Comprimento : Integer, var oBullishEngulfing: Integer, var oBearishEngulfing : Integer)

Parâmetros:

Comprimento: Tamanho utilizado para calcular a média do corpo do candle.
oBullishEngulfing: Variável para identificação de padrão(Bullish Engulfing).
oBearishEngulfing: Variável para identificação de padrão(Bearish Engulfing).

Retorno:

Integer:

Identificação(retorno função):

0 - **Algum dos padrões foi identificado.**

1 - **Nenhuma padrão identificado.**

Identificação(retorno variável: oBullishEngulfing):

0 - **Padrão Bullish Engulfing não foi identificado.**

1 - **Padrão Bullish Engulfing identificado.**

Identificação(retorno variável: oBearishEngulfing):

0 - **Padrão Bearish Engulfing não foi identificado.**

1 - **Padrão Bearish Engulfing identificado.**

Exemplos:

No exemplo, caso o padrão Bullish Engulfing seja identificado, considerando 13(Comprimento), será aplicada uma coloração(amarela).

```
aux := C_BullEng_BearEng(13, oBullishEngulfing, oBearishEngulfing);
```

```
if(oBullishEngulfing = 1) then
```

```
    PaintBar(clYellow);
```



Função C_Doji

Descrição:

A função **C_Doji** identifica a ocorrência de um candle tipo **Doji**.

Sintaxe:

C_Doji(Percentual : Integer)

Parâmetros:

Percentual: Limiar para(Abertura - Fechamento) que seria uma porcentagem do intervalo do candle.

Retorno:

Integer:

Identificação:

0 - **Padrão não identificado.**

1 - **Padrão identificado.**

Exemplos:

No exemplo, caso o padrão seja identificado, considerando percentual de 5%, será aplicada uma coloração(verde).

```
if(C_Doji(5) = 1)
```

```
    then PaintBar(clGreen);
```



Função C_Hammer_HangingMan

Descrição:

A função **C_Hammer_HangingMan** identifica a ocorrência de dois tipos de candles: **Hammer** e **Hanging Man**.

Sintaxe:

C_Hammer_HangingMan(Comprimento : Integer, Fator : Integer, var oHammer : Integer, var oHangingMan : Integer)

Parâmetros:

Comprimento: Tamanho utilizado para calcular a média do corpo do candle.

Fator: Determina quantas vezes a sombra do candle deve ser maior que o seu corpo.

oHammer: Variável para identificação de padrão(Hammer).

oHangingMan: Variável para identificação de padrão(Hanging Man).

Retorno:

Integer:

Identificação(retorno função):

0 - **Algum dos padrões foi identificado.**

1 - **Nenhuma padrão identificado.**

Identificação(retorno variável: oHammer):

0 - **Padrão Morning Hammer não foi identificado.**

1 - **Padrão Morning Hammer identificado.**

Identificação(retorno variável: oHangingMan):

0 - **Padrão Hanging Man não foi identificado.**

1 - **Padrão Hanging Man identificado.**

Exemplos:

No exemplo, caso seja identificado algum dos padrões(Hammer ou Hanging Man), considerando 14(Comprimento) e 2(Fator), será aplicada uma coloração(amarela).

```
aux := C_Hammer_HangingMan(14, 2, oHammer, oHangingMan);
```

```
if(aux = 1) then
```

```
    PaintBar(cIYellow);
```



Função C_MornDoji_EveDoji

Descrição:

A função **C_MornDoji_EveDoji** identifica a ocorrência de dois tipos de candles: **Morning Doji Star** e **Evening Doji Star**.

Sintaxe:

C_MornDoji_EveDoji(Comprimento : Integer, Percentual : Float, var oMorningDojiStar : Integer, var oEveningDojiStar : Integer)

Parâmetros:

Comprimento: Tamanho utilizado para calcular a média do corpo do candle.

Percentual: Doji limiar para o (abrir - fechar) como uma porcentagem do intervalo da barra.

oMorningDojiStar: Variável para identificação de padrão(Morning Doji Star).

oEveningDojiStar: Variável para identificação de padrão(Evening Doji Star).

Retorno:

Integer:

Identificação(retorno função):

0 - **Algum dos padrões foi identificado.**

1 - **Nenhuma padrão identificado.**

Identificação(retorno variável: oMorningDojiStar):

0 - **Padrão Morning Doji Star não foi identificado.**

1 - **Padrão Morning Doji Star identificado.**

Identificação(retorno variável: oEveningDojiStar):

0 - **Padrão Evening Doji Star não foi identificado.**

1 - **Padrão Evening Doji Star identificado.**

Exemplos:

No exemplo, caso seja identificado algum dos padrões(Morning Doji Star ou Evening Doji Star), considerando 9(Comprimento) e 3(Percentual), será aplicada uma coloração(branca).

```
aux := C_MornDoji_EveDoji(9, 3.0, oMorningDojiStar, oEveningDojiStar);
```

```
if(aux = 1) then
```

```
    PaintBar(clWhite);
```



Função C_MornStar_EveStar

Descrição:

A função **C_MornStar_EveStar** identifica a ocorrência de dois tipos de candles: **Morning Star** e **Evening Star**.

Sintaxe:

C_MornStar_EveStar(Comprimento : Integer, var oMorningStar : Integer, var oEveningStar : Integer)

Parâmetros:

Comprimento: Tamanho utilizado para calcular a média do corpo do candle.

oMorningStar: Variável para identificação de padrão(Morning Star).

oEveningStar: Variável para identificação de padrão(Evening Star).

Retorno:

Integer:

Identificação(retorno função):

0 - **Algum dos padrões foi identificado.**

1 - **Nenhuma padrão identificado.**

Identificação(retorno variável: oMorningStar):

0 - **Padrão Morning Star não foi identificado.**

1 - **Padrão Morning Star identificado.**

Identificação(retorno variável: oEveningStar):

0 - **Padrão Evening Star não foi identificado.**

1 - **Padrão Evening Star identificado.**

Exemplos:

No exemplo, caso o padrão Evening Star seja identificado, considerando 6(Comprimento), será aplicada uma coloração(verde).

```
aux := C_MornStar_EveStar(6, oMorningStar, oEveningStar);
```

```
if(oEveningStar = 1) then
```

```
    PaintBar(clGreen);
```




Função C_PierceLine_DkCloud

Descrição:

A função **C_PierceLine_DkCloud** identifica a ocorrência de dois tipos de candles: **Piercing Line** e **Dark Cloud**.

Sintaxe:

C_PierceLine_DkCloud(Comprimento : Integer, var oPiercingLine : Integer, var oDarkCloud : Integer)

Parâmetros:

Comprimento: Tamanho utilizado para calcular a média do corpo do candle.

oPiercingLine: Variável para identificação de padrão(Piercing Line).

oDarkCloud: Variável para identificação de padrão(Dark Cloud).

Retorno:

Integer:

Identificação(retorno função):

0 - **Algum dos padrões foi identificado.**

1 - **Nenhuma padrão identificado.**

Identificação(retorno variável: oPiercingLine):

0 - **Padrão Piercing Line não foi identificado.**

1 - **Padrão Piercing Line identificado.**

Identificação(retorno variável: oDarkCloud):

0 - **Padrão Dark Cloud não foi identificado.**

1 - **Padrão Dark Cloud identificado.**

Exemplos:

No exemplo, caso o padrão Dark Cloud seja identificado, considerando 5(Comprimento), será aplicada uma coloração(amarela).

```
aux := C_PierceLine_DkCloud(5, oPiercingLine, oDarkCloud);
```

```
if(oDarkCloud = 1) then
```

```
    PaintBar(c1Yellow);
```



Função C_ShootingStar

Descrição:

A função **C_ShootingStar** identifica a ocorrência de candles tipo **Shooting Star**.

Sintaxe:

C_ShootingStar(Comprimento : Integer, Fator : Integer)

Parâmetros:

Comprimento: Tamanho utilizado para calcular a média do corpo do candle.

Fator: Determina quantas vezes a sombra do candle deve ser maior que o seu corpo.

Retorno:

Integer:

Identificação:

0 - **Padrão não identificado.**

1 - **Padrão identificado.**

Exemplos:

No exemplo, caso o padrão seja identificado, considerando 10(Comprimento) e 2(Fator), será aplicada uma coloração(verde).

```
if(C_ShootingStar(10, 2) = 1) then
```

```
    PaintBar(cIVerde);
```



Exemplos

Função DiMaisDiMenos

Descrição:

O exemplo **DiMaisDiMenos** possui a implementação do indicador **DI+/DI-**, onde foi determinado um parâmetro (input) para o período.

Observação: o exemplo com o código fonte está disponível no editor de estratégias, para visualizá-lo, acesse o menu: "abrir > exemplos".

Sintaxe:

DiMaisDiMenos(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "aux" o retorno do indicador criado.

```
aux := DiMaisDiMenos(14);
```



Função IFR

Descrição:

A função **IFR** retorna o valor(tipo clássico) do indicador **IFR**, de acordo com o período.

Observação: o exemplo com o código fonte está disponível no editor de estratégias, para visualizá-lo, acesse o menu: "abrir > exemplos".

Sintaxe:

IFR(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o valor do indicador IFR, considerando 2 períodos para o cálculo.

```
aux := IFR(2);
```



Função Media

Descrição:

A função **Media** retorna o valor do indicador Média Móvel, tipo aritmética.

Observação: o exemplo com o código fonte está disponível no editor de estratégias, para visualizá-lo, acesse o menu: "abrir > exemplos".

Sintaxe:

Media(Período : Integer, TipoSerie : Serie)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

TipoSerie: Série que será considerada para o cálculo.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vMed" o valor do indicador Média Móvel(Aritmética), considerando 100 períodos, e a série de fechamento(Close) para o cálculo.

```
vMed := Media(100, Close);
```



Função MediaExp

Descrição:

A função **MediaExp** retorna o valor do indicador Média Móvel, tipo exponencial.

Observação: o exemplo com o código fonte está disponível no editor de estratégias, para visualizá-lo, acesse o menu: "abrir > exemplos".

Sintaxe:

MediaExp(Período : Integer, TipoSerie : Serie)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

TipoSerie: Série que será considerada para o cálculo.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vMed" o valor do indicador Média Móvel(Exponencial), considerando 100 períodos, e a série de fechamento(Close) para o cálculo.

```
vMed := MediaExp(100, Close);
```



Função(Regra de coloração) PaintVar

Descrição:

A função **PaintVar** possui a implementação de uma estratégia de coloração, a qual compara se o fechamento do último candle é positivo ou negativo em relação ao fechamento do(candle) anterior, e, como indicador, apenas retorna o último preço do ativo.

Observação: o exemplo com o código fonte está disponível no editor de estratégias, para visualizá-lo, acesse o menu: "abrir > exemplos".

Sintaxe:

PaintVar

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, será plotado o último preço, onde a regra de coloração correspondente poderá ser aplicada sobre o indicador.

Plot(PaintVar);



Função WellesSum

Descrição:

A função **WellesSum** retorna o valor do indicador **WellesSum**, de acordo com os parâmetros desejados.

Observação: o exemplo com o código fonte está disponível no editor de estratégias, para visualizá-lo, acesse o menu: "abrir > exemplos".

Sintaxe:

WellesSum(Período, Integer, SerieReferencia : Serie, Offset : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

SerieReferencia: Série de dados.

Offset: Referente ao parâmetro "Offset" do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nWS" o dado do indicador WellesSum, considerando 20(Período), série de fechamento(SerieReferencia) e 0(Offset) para o cálculo.

```
nWS := WellesSum(20, Close, 0);
```




Gráficas

Função AvgPrice

Descrição:

A função **AvgPrice** retorna o valor da média entre Abertura, Fechamento, Máxima e Mínima, de determinado candle.

Sintaxe:

AvgPrice

Parâmetros:

Sem parâmetros

Retorno:

Float

Exemplos:

Será atribuído à variável "nAvg" o retorno da função AvgPrice.

```
nAvg := AvgPrice;
```



Função CurrentBar

Descrição:

A função **CurrentBar** tem como finalidade retornar ao usuário o índice do candle atual.

Sintaxe:

CurrentBar

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, o candle de número 100 será identificado com a coloração vermelha.

```
if(CurrentBar = 100)  
    then PaintBar(clRed);
```



Função GetPlotColor

Descrição:

A função **GetPlotColor** possui como funcionalidade retornar o valor numérico da cor de determinado Plot.

Sintaxe:

GetPlotColor(NumeroPlot : Integer)

Parâmetros:

NumeroPlot: Número do Plot para obter a cor.

Retorno:

Integer

Exemplos:

No exemplo, será aplicada a coloração do Plot no Plot2.

```
SetPlotColor(1, RGB(200, 200, 200));
```

```
SetPlotColor(2, GetPlotColor(1));
```



Função GetPlotWidth

Descrição:

A função **GetPlotWidth** possui como finalidade retornar o valor da espessura de determinado Plot.

Sintaxe:

GetPlotWidth(NumeroPlot : Integer)

Parâmetros:

NumeroPlot: Número do Plot para obter a espessura.

Retorno:

Integer

Exemplos:

No exemplo, será aplicada a espessura do Plot no Plot2.

```
SetPlotWidth(1, 5);
```

```
SetPlotWidth(2, GetPlotWidth(1));
```



Função Highest

Descrição:

A função **Highest** tem como funcionalidade retornar ao usuário o maior valor da série estipulada por ele, dentro de um período determinado.

Sintaxe:

Highest(SerieDeDados : Serie, Período : Integer)

Parâmetros:

SerieDeDados: Série de dados desejada, podendo ser a abertura, máxima, mínima, fechamento, ou até mesmo indicadores.

Período: Determina o período que será considerado para a pesquisa.

Retorno:

Float

Exemplos:

No exemplo abaixo, usamos a função Highest para retornar a maior abertura dentro de 9 períodos.

```
Plot(Highest(Open, 9));
```



Função HighestBar

Descrição:

A função **HighestBar** tem como funcionalidade retornar ao usuário o índice do maior valor da série estipulada por ele, dentro de um período determinado.

Sintaxe:

HighestBar(SerieDeDados : Serie, Período : Integer)

Parâmetros:

SerieDeDados: Série de dados desejada, podendo ser a abertura, máxima, mínima, fechamento, ou até mesmo indicadores.

Período: Determina o período que será considerado para a pesquisa.

Retorno:

Float

Exemplos:

No exemplo abaixo, usamos a função HighestBar para retornar o índice da maior mínima dentro de 20 períodos.

```
mMinima := Highest(Open, 9);
```



Função LastBarOnChart

Descrição:

A função **LastBarOnChart** tem como função retornar um valor Booleano mostrando se o candle atual é o último candle do gráfico.

Sintaxe:

LastBarOnChart

Parâmetros:

Sem parâmetros.

Retorno:

Boolean

Exemplos:

No exemplo a seguir, o candle atual será identificado com a coloração amarela.

```
if(LastBarOnChart)  
    then PaintBar(c1Yellow);
```



Função Leader

Descrição:

A função **Leader** retorna o valor de 0 quando o ponto médio for menor que a mínima anterior ou 1 quando o ponto médio for maior que máxima anterior.

Sintaxe:

Leader

Parâmetros:

Sem parâmetros

Retorno:

Integer

Exemplos:

No exemplo abaixo, caso a função Leader seja igual a um, o candle analisado será identificado pela cor verde.

```
if(Leader = 1)  
    then PaintBar(clVerde);
```




Função MaxBarsForward

Descrição:

A função **MaxBarsForward** tem como finalidade percorrer a lista da série, iniciando(índice 0) a partir do último candle criado(atual).

Sintaxe:

MaxBarsForward

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, será aplicada uma coloração no candle anterior ao atual.

```
if(MaxBarsForward = 1) then
```

```
    PaintBar(clGreen);
```



Função MaxBarsBack

Descrição:

A função **MaxBarsBack** tem como finalidade percorrer a lista da série, iniciando(índice 0) a partir do primeiro candle criado.

Sintaxe:

MaxBarsBack

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos:

No exemplo, será aplicada uma coloração no segundo candle criado.

```
if(MaxBarsBack = 1) then
```

```
    PaintBar(clGreen);
```



Função NoPlot

Descrição:

A função **NoPlot** tem como finalidade efetuar a remoção de determinado Plot.

Sintaxe:

NoPlot(NumeroPlot : Integer)

Parâmetros:

NumeroPlot: Número do Plot(1, 2, 3 e 4) para a remoção.

Retorno:

Void: Sem retorno.

Exemplos:

Conforme no exemplo a seguir, será removida a linha referente ao Plot, e será plotada somente a linha vinculada ao Plot2.

```
Plot(Close);
```

```
Plot2(Open);
```

```
NoPlot(1);
```



Função PaintBar

Descrição:

A função **PaintBar** permite a aplicação de colorações, em indicadores ou candles.

Sintaxe:

PaintBar(Cor : Integer)

Parâmetros:

Cor: Determina a coloração, podendo-se passar por parâmetro uma String ou a chamada da função RGB:

clNomeCor - **String para aplicação, conforme a seguinte lista:**

- clBlack
- clMarrom
- clGreen
- clOlive
- clNavy
- clPurple
- clTeal
- clGray
- clSilver
- clRed
- clLime
- clYellow
- clBlue
- clFuchsia
- clAqua
- clWhite
- clMoneyGreen
- clSkyBlue

RGB - Função para aplicação.

Retorno:

Float

Exemplos:

No exemplo, será aplicada uma coloração(verde) quando o histograma de MACD for maior que zero.

```
if (MACD(23, 12, 9)|1| > 0) then
```

```
    PaintBar(clGreen);
```



Função Lowest

Descrição:

A função **Lowest** tem como funcionalidade retornar ao usuário o menor valor da série estipulada por ele, dentro de um período determinado.

Sintaxe:

Lowest(SerieDeDados : Serie, Período : Integer)

Parâmetros:

SerieDeDados: Série de dados desejada, podendo ser a abertura, máxima, mínima, fechamento, ou até mesmo indicadores.

Período: Determina o período que será considerado para a pesquisa.

Retorno:

Float

Exemplos:

No exemplo abaixo, usamos a função Lowest para retornar o menor fechamento dentro de 50 períodos.

```
Plot(Lowest(Close, 50));
```



Função LowestBar

Descrição:

A função **LowestBar** tem como funcionalidade retornar ao usuário o índice do menor valor da série estipulada por ele, dentro de um período determinado.

Sintaxe:

LowestBar(SerieDeDados : Serie, Período : Integer)

Parâmetros:

SerieDeDados: Série de dados desejada, podendo ser a abertura, máxima, mínima, fechamento, ou até mesmo indicadores.

Período: Determina o período que será considerado para a pesquisa.

Retorno:

Float

Exemplos:

No exemplo abaixo, usamos a função **LowestBar** para retornar o índice da menor mínima dentro de 26 períodos.

```
mMinima := LowestBar(Low, 26);
```



Função MedianPrice

Descrição:

A função **MedianPrice** retorna a média entre a máxima e a mínima de cada candle.

Sintaxe:

MedianPrice

Parâmetros:

Sem parâmetros

Retorno:

Float

Exemplos:

No exemplo a seguir, caso o dado de **MedianPrice** do candle atual for maior que o do anterior, os candles serão destacados com a coloração verde.

```
if(MedianPrice > MedianPrice[1])  
    then PaintBar(clGreen);
```



Função Plot

Descrição:

A função **Plot** realiza a ligação dos valores passados por parâmetro e cria gráficos de linhas. É possível efetuar a inserção de no máximo quatro linhas, onde deverá ser utilizada a função Plot numerada: Plot, Plot2, Plot3 e Plot4.

Sintaxe:

Plot(Dado : Float)
Plot2(Dado : Float)
Plot3(Dado : Float)
Plot4(Dado : Float)

Parâmetros:

Dado: Pode-se utilizar variáveis, funções ou constantes para realizar o desenho do indicador.

Retorno:

Void: Sem retorno.

Exemplos:

Conforme no exemplo a seguir, o usuário poderá desenhar até quatro linhas, em uma mesma estratégia.

Plot(Close);

Plot2(Open - Open[1]);

Plot3(Ifr(9) / variavel);

Plot4(Close + High(1));



Range

Descrição:

A função **Range** retorna a diferença entre a máxima e a mínima do candle.

Sintaxe:

Range

Parâmetros:

Sem parâmetros

Retorno:

Float

Exemplos:

No exemplo, a variável "nRange" irá receber o retorno do dado da função Range.

```
nRange := Range;
```



Função RangeLeader

Descrição:

A função **RangeLeader** verifica se a barra atual é Range Leader.

Sintaxe:

RangeLeader

Parâmetros:

Sem parâmetros

Retorno:

Float

Exemplos:

No exemplo, a variável "nRL" irá receber o retorno do dado da função RangeLeader.

```
nRL := RangeLeader;
```



Função RGB

Descrição:

A função **RGB** permite ao usuário customizar cores, a partir da aplicação de intensidade dos parâmetros vermelho, verde e azul.

Sintaxe:

RGB(Red : Integer, Green : Integer, Blue : Integer**)**

Parâmetros:

Red: Intensidade cor vermelha, variando de 0 a 255;

Green: Intensidade cor verde, variando de 0 a 255;

Blue: Intensidade cor azul, variando de 0 a 255.

Retorno:

Integer

Exemplos:

No exemplo a seguir, será aplicada a coloração azul, conforme os parâmetros de RGB.

PaintBar(RGB(0, 0, 230)**);**



Função SetPlotColor

Descrição:

A função **SetPlotColor** possui como finalidade alterar a coloração de determinado Plot(1 a 4)

Sintaxe:

SetPlotColor(NumeroPlot : Integer, Cor : Integer)

Parâmetros:

NumeroPlot: Número do Plot específico:

- 1 - **Plot**
- 2 - **Plot2**
- 3 - **Plot3**
- 4 - **Plot4**

Cor: Determina a coloração, podendo-se passar por parâmetro uma String ou a chamada da função RGB:

clNomeCor - **String para aplicação, conforme a seguinte lista:**

- clBlack
- clMarrom
- clGreen
- clOlive
- clNavy
- clPurple
- clTeal
- clGray
- clSilver
- clRed
- clLime
- clYellow
- clBlue
- clFuchsia
- clAqua
- clWhite
- clMoneyGreen
- clSkyBlue

RGB - Função para aplicação.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, será alterada a coloração(vermelha) referente à linha do Plot.



Plot(Close);

SetPlotColor(1, clRed);



Função SetPlotWidth

Descrição:

A função **SetPlotWidth** possui como finalidade alterar a espessura de determinado Plot(1 a 4)

Sintaxe:

SetPlotWidth(NumeroPlot : Integer, Espessura : Integer)

Parâmetros:

NumeroPlot: Número do Plot específico:

- 1 - **Plot**
- 2 - **Plot2**
- 3 - **Plot3**
- 4 - **Plot4**

Espessura: Número para a nova espessura.

Retorno:

Void: Sem retorno.

Exemplos:

No exemplo, será alterada a espessura referente à linha do Plot.

```
Plot(Close);
```

```
SetPlotWidth(1, 3);
```



Função TrueHigh

Descrição:

A função **TrueHigh** retorna o maior entre o máximo da barra e o fechamento da barra anterior.

Sintaxe:

TrueHigh

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "TH" irá receber o dado da função TrueHigh.

```
TH := TrueHigh;
```



Função TrueLow

Descrição:

A função **TrueLow** retorna o menor entre a mínima da barra e o fechamento da barra anterior.

Sintaxe:

TrueLow

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "TL" irá receber o dado da função TrueLow.

```
TL := TrueLow;
```




Função TrueRange

Descrição:

A função **TrueRange** retorna a diferença entre TrueHigh e TrueLow.

Sintaxe:

TrueRange

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "TR" irá receber o valor do indicador True Range.

```
TR := TrueRange;
```



Função TrueRangeCustom

Descrição:

A função **TrueRangeCustom** retorna o TrueRange de acordo com os dados informados pelo usuário.

Sintaxe:

TrueRangeCustom(Maxima : Float, Minima : Float, Fechamento : Float)

Parâmetros:

Maxima: Valor de máxima para o cálculo do indicador.

Minima: Valor de mínima para o cálculo do indicador.

Fechamento: Valor de fechamento de referência.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "TRC" o dado da função TrueRangeCustom.

```
TRC := TrueRangeCustom(3763.5, 3761, 3761);
```



Função TypicalPrice

Descrição:

A função **TypicalPrice** retorna a média entre a máxima, mínima e fechamento do candle.

Sintaxe:

TypicalPrice

Parâmetros:

Sem parâmetros

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o retorno do dado da função TypicalPrice.

```
aux := TypicalPrice;
```



Função WeightedClose

Descrição:

A função **WeightedClose** retorna a média entre o ponto médio da barra e dois fechamentos.

Sintaxe:

WeightedClose

Parâmetros:

Sem parâmetros

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o retorno do dado da função **WeightedClose**.

```
aux := WeightedClose;
```



Indicadores

Função AvgSeparation

Descrição:

A função **AvgSeparation** retorna o valor do indicador **Afastamento Médio**, de acordo com o período e tipo de média desejados.

Sintaxe:

AvgSeparation(Período : Integer, TipoMedia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "avgSep" irá receber o valor do indicador Afastamento Médio, considerando 21 períodos e tipo exponencial para o cálculo.

```
avgSep := AvgSeparation(21, 1);
```



Função AvgTrueRange

Descrição:

A função **AvgTrueRange** retorna o valor do indicador **True Range**, de acordo com o período e tipo de média desejados.

Sintaxe:

AvgTrueRange(Período : Integer, TipoMedia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nATR" irá receber o valor do indicador True Range, considerando 5 períodos, e tipo de média ponderada para o cálculo.

```
nATR := AvgTrueRange(5, 3);
```



Função AccAgressSaldo

Descrição:

A função **AccAgressSaldo** retorna o valor do indicador **TR - Acúmulo de Agressão - Saldo**.

Sintaxe:

AccAgressSaldo(TipoVolume : Integer)

Parâmetros:

TipoVolume: Tipo de volume para o cálculo:

- 0 - **Financeiro**
- 1 - **Quantidade**
- 2 - **Negócios**

Retorno:

Float

Exemplos:

No exemplo, a variável "aac" irá receber o volume de quantidade do indicador TR - Acúmulo de Agressão - Saldo.

```
aac := AccAgressSaldo(1);
```



Função AccuDistr

Descrição:

A função **AccuDistr** retorna o valor do indicador **Acumulação/Distribuição** .

Sintaxe:

AccuDistr

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nAcc" irá receber o valor do indicador Acumulação/Distribuição.

```
nAcc := AccuDistr;
```




Função AccuDistrW

Descrição:

A função **AccuDistrW** retorna o valor do indicador **Acumulação/Distribuição Williams**.

Sintaxe:

AccuDistrW

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nADW" irá receber o valor do indicador Acumulação/Distribuição Williams.

```
nADW := AccuDistr;
```



Função AdaptiveMovingAverage

Descrição:

A função **AdaptiveMovingAverage** retorna o valor do indicador **Adaptive Moving Average**, de acordo com o períodos específicos desejados.

Sintaxe:

AdaptiveMovingAverage(Período : Integer, FastSC : Integer, SlowSC : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

FastSC: Determina o período para o indicador FastStochastic.

SlowSC: Determina o período para o indicador SlowStochastic.

Retorno:

Float

Exemplos:

No exemplo, a variável "amv" irá receber o valor do indicador Adaptive Moving Average, considerando 10 períodos para o cálculo, com 2 períodos para FastStochastic, e 30 para SlowStochastic.

```
amv := AdaptiveMovingAverage(10, 2, 30);
```



Função ADX

Descrição:

A função **ADX** retorna o valor do indicador **ADX**, de acordo com os períodos desejados.

Sintaxe:

ADX(Período : Integer, PeríodoMedia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

PeríodoMedia: Período utilizado no momento do cálculo da média utilizada no indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nADX" irá receber o valor do indicador ADX, considerando `Período=9` e `PeríodoMedia=9` para o cálculo.

```
nADX := ADX(9,9);
```



Função AgressionVolBalance

Descrição:

A função **AgressionVolBalance** retorna o valor do indicador **TR - Volume de Agressão - Saldo**.

Sintaxe:

AgressionVolBalance

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "avb" irá receber o valor do indicador **TR - Acúmulo de Agressão - Saldo**.

```
avb := AgressionVolBalance;
```



Função AgressionVolBuy

Descrição:

A função **AgressionVolBuy** retorna o valor do indicador **TR - Volume de Agressão - Compra**.

Sintaxe:

AgressionVolBuy

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "avb" irá receber o valor do indicador TR - Volume de Agressão - Compra.

```
avb := AgressionVolBuy;
```



Função AgressionVolSell

Descrição:

A função AgressionVolSell retorna o valor do indicador **TR - Volume de Agressão - Venda**.

Sintaxe:

AgressionVolSell

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "avs" irá receber o valor do indicador AgressionVolSell.

```
avs := AgressionVolSell;
```



Função ArmsEaseOfMov

Descrição:

A função **ArmsEaseOfMov** retorna o valor do indicador **Arms Ease of Movement**, de acordo com o período e tipo de média desejados.

Sintaxe:

ArmsEaseOfMov(Media : Integer, TipoMedia : Integer)

Parâmetros:

Media: Período da média utilizada no momento do cálculo do indicador.

TipoMedia: Determina o tipo da média utilizada, onde "0" é **Aritmética**, "1" é **Exponencial**, "2" é **Welles Wilder** e "3" é **Ponderada**.

Retorno:

Float

Exemplos:

No exemplo, a variável "nATR" irá receber o valor do indicador Arms Ease of Movement, considerando 9 períodos para o tipo de média exponencial.

```
nATR := ArmsEaseOfMov(9, 1);
```



Função AroonLin

Descrição:

A função **AroonLin** retorna o valor do indicador **Aroon Linha**, de acordo com o período desejado.

Sintaxe:

AroonLin(Período : Integer)|Linha : Integer|

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Linha: Determina qual linha será obtida:

0 - **Aroon Up**

1 - **Aroon Down**

Retorno:

Float

Exemplos:

No exemplo, a variável "fAroon" irá receber o valor da linha "Aroon Down", considerando 9 períodos para o cálculo.

```
fAroon := AroonLin(9)|1|;
```




Função AroonOsc

Descrição:

A função **AroonOsc** retorna o valor do indicador **Aroon Oscilador**, de acordo com o período desejado.

Sintaxe:

AroonOsc(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "fAroonOsc" irá receber o valor do indicador Aroon Oscilador, considerando 9 períodos para o cálculo.

```
fAroonOsc := AroonOsc(9);
```



Função AvgAgrBuySell

Descrição:

A função **AvgAgrBuySell** retorna o valor do indicador **TR - Agressão Média - Compra e Venda**, de acordo com os parâmetros desejados.

Sintaxe:

AvgAgrBuySell(AlertaVariacoes : Integer, TipoVolume : Integer, TipoDesenho : Integer) | Linha : Integer |

Parâmetros:

AlertaVariacoes: Número de variações.

TipoVolume: Determina qual tipo de volume será obtido:

- 0 - Financeiro
- 1 - Quantidade

TipoDesenho: Relação entre compra e venda:

- 0 - **Compra e Venda**
- 1 - **Compra/Venda**
- 2 - **Compra-Venda**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "aux" o retorno do indicador, considerando 3 variações, a quantidade e tipo de desenho .

```
aux := AvgAgrBuySell(3, 1, 0);
```



Função AvgAgrTotal

Descrição:

A função **AvgAgrTotal** retorna o valor do indicador **TR - Agressão Média - Total**, de acordo com os parâmetros específicos.

Sintaxe:

AvgAgrTotal(AlertaVariacoes : Integer, TipoVolume : Integer, TipoDesenho : Integer) | Linha : Integer |

Parâmetros:

AlertaVariacoes: Número de variações.

TipoVolume: Determina qual tipo de volume será obtido:

- 0 - Financeiro
- 1 - Quantidade

TipoDesenho: Relação entre compra e venda:

- 0 - **Compra e Venda**
- 1 - **Compra/Venda**
- 2 - **Compra-Venda**

Linha: Determina qual linha será obtida:

- 0 - **Volume indicador**
- 1 - **Avaliar**

Retorno:

Float

Exemplos:

No exemplo, a variável "n" irá receber o valor da função AvgAgrTotal.

```
n := AvgAgrTotal(3, 1, 0);
```



Função BalanceOfPower

Descrição:

A função **BalanceOfPower** retorna o valor do indicador **Balança do Poder**, de acordo com o período desejado.

Sintaxe:

BalanceOfPower(Media : Integer, TipoMedia : Integer)

Parâmetros:

Media: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nBalance" irá receber o valor do indicador Balança do Poder, considerando 14 períodos, e o tipo de média exponencial para o cálculo.

```
nBalance := BalanceOfPower(14,1);
```



Função BearPower

Descrição:

A função **BearPower** retorna o valor do indicador **Bear Power**, conforme o período desejado.

Sintaxe:

BearPower(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nBear" irá receber o dado do indicador Bear Power, considerando 13 períodos.

```
nBear := BearPower(13);
```



Função BollingerBands

Descrição:

A função **BollingerBands** retorna o valor do indicador **Bandas de Bollinger**, de acordo com o período e tipo de média desejados.

Sintaxe:

```
BollingerBands(Desvio : Float, Media : Integer, TipoMedia : Integer)|Linha : Integer|
```

Parâmetros:

Desvio: Desvio utilizado no momento do cálculo do indicador.

Media : Período da média utilizada no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Linha: Determina qual linha será obtida:

- 0 - **Superior**
- 1 - **Inferior**

Retorno:

Float

Exemplos:

No exemplo, a variável "fBool" irá receber o valor da linha inferior do indicador Bandas de Bollinger, considerando 2.0 como desvio, 20 períodos e tipo de média aritmética.

```
fBool := BollingerBands(2.0, 20, 0)|1|;
```



Função BollingerBandW

Descrição:

A função **BollingerBandW** retorna o valor do indicador **Bollinger Band Width**, de acordo com o período e tipo de média desejados.

Sintaxe:

BollingerBandW(Desvio : Float, Media : Integer, TipoMedia : Integer)

Parâmetros:

Desvio: Desvio utilizado no momento do cálculo do indicador.

Media : Período da média utilizada no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "fBoolinBW" irá receber o valor do indicador Bollinger Band Width, considerando 2.0 como desvio, 20 períodos e tipo de média exponencial.

```
fBoolinBW := BollingerBandW(2.0,20,1);
```



Função BollingerBPerc

Descrição:

A função **BollingerBPerc** retorna o valor do indicador **Bollinger b%**, de acordo com o período e tipo de média desejados.

Sintaxe:

BollingerBPerc(Desvio : Float, Media : Integer, TipoMedia : Integer)

Parâmetros:

Desvio: Desvio utilizado no momento do cálculo do indicador.

Media : Período da média utilizada no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "fBool" irá receber o valor do indicador Bollinger Band Width, considerando 2.0 como desvio, 20 períodos e tipo de média Welles Wilder.

```
fBool := BollingerBPerc(2.0,20,3);
```




Função BullPower

Descrição:

A função **BullPower** retorna o valor do indicador **Bull Power**, de acordo com o período e tipo de média desejados.

Sintaxe:

BullPower(Período : Integer, PeríodoMédia : Integer, TipoMédia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador;

PeríodoMédia: Período utilizado no momento do cálculo da média utilizada no indicador.

TipoMédia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nBull" irá receber o valor do indicador Bull Power, considerando 21 períodos, e 9 períodos para a média aritmética.

```
nBull := BullPower(21,9,0)
```



Função Carmine

Descrição:

A função **Carmine** retorna o valor do indicador **Carmine**, de acordo com os parâmetros estabelecidos.

Sintaxe:

Carmine(Risco : Integer, ModoCalculo : Integer, Período : Integer, Desvio : Float, UsarVWAP : Boolean, UsarAtr : Boolean)

Parâmetros:

Risco: Informa o perfil que será utilizado:

- 0 - **Zero**
- 1 - **Um**
- 2 - **Dois**
- 3 - **Três**

ModoCalculo: Tipo de média para o cálculo:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Período: Período utilizado no momento do cálculo do indicador.

Desvio: Informa o desvio que será considerado.

UsarVWAP: Determina se o VWAP será utilizado.

UsarAtr: Determina a ativação do StopATR.

Retorno:

Float

Exemplos:

No exemplo, a variável "DecisionP" irá receber o dado referente à máxima.

```
DecisionP := DecisionPoints(0, 1);
```



Função CCI

Descrição:

A função **CCI** retorna o valor do indicador **CCI**, de acordo com o período desejado.

Sintaxe:

CCI(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nCCI" irá receber o valor do indicador CCI, considerando 14 períodos para o cálculo.

```
nCCI := CCI(14);
```



Função ChaikinMoneyFlow

Descrição:

A função **ChaikinMoneyFlow** retorna o valor do indicador **Chaikin Money Flow**, de acordo com o período desejado.

Sintaxe:

ChaikinMoneyFlow(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nCMF" irá receber o valor do indicador Chaikin Money Flow, considerando 21 período para o cálculo.

```
nCMF := ChaikinMoneyFlow(21);
```



Função ChainSetup

Descrição:

A função **ChainSetup** retorna o valor do indicador **ChainSetup** .

Sintaxe:

ChainSetup

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "cs" irá receber o dado do indicador ChainSetup.

```
cs := ChainSetup;
```



Função ChaikinOsc

Descrição:

A função **ChaikinOsc** retorna o valor do indicador **Oscilador Chaikin**, de acordo com as médias desejadas.

Sintaxe:

ChaikinOsc(MediaLonga : Integer, MediaCurta : Integer)

Parâmetros:

MediaLonga: Determina o período da Média Longa para formação do cálculo.

MediaCurta: Determina o período da Média Curta.

Retorno:

Float

Exemplos:

No exemplo, a variável "nCo" irá receber o valor do indicador Oscilador Chaikin, considerando 10 períodos para a média longa, e 3 para a curta.

```
nCo := ChaikinOsc(10, 3);
```



Função DarvasBox

Descrição:

A função **DarvasBox** retorna o valor do indicador **Darvas Box**.

Sintaxe:

DarvasBox|Linha : Integer|

Parâmetros:

Linha: Determina qual dado(Compra ou Venda) será obtido:

0 - **Compra**

1 - **Venda**

Retorno:

Float

Exemplos:

No exemplo, a variável "nDB" irá receber os valores(Dado: Venda) do indicador Darvas Box.

```
nDB := DarvasBox|1|;
```



Função DecisionPoints

Descrição:

A função **DecisionPoints** retorna o valor do indicador **Pontos de Decisão**.

Sintaxe:

DecisionPoints(Tipo : Integer, Linha : Integer)

Parâmetros:

Tipo: Determina o tipo: Preço, Volume, Faixas de Volume e Variação:

0 - **Tipo Preço:**

Linha: Dado da série que será obtido

0 - **Abertura**

1 - **Máxima**

2 - **Mínima**

3 - **Fechamento**

4 - **Ajuste**

1 - **Tipo Volume:**

Linha: Três maiores volumes dos períodos

0 - **Dado volume**

1 - **Dado volume**

2 - **Dado volume**

1 - **Tipo Faixas de Volume:**

Linha:

0 - **Retorna o dado específico ao tipo**

1 - **Tipo Variação:**

Linha:

0 - **Dado variação linha superior**

1 - **Dado variação linha inferior**

Retorno:

Float

Exemplos:

No exemplo, a variável "DecisionP" irá receber o dado referente à máxima.

```
DecisionP := DecisionPoints(0, 1);
```




Função DiDiIndex

Descrição:

A função **DiDiIndex** retorna o valor do indicador **Didi Index**, de acordo com o período e tipos de médias desejados.

Sintaxe:

DiDiIndex(MediaReferencia : Integer, TipoMediaReferencia : Integer, Media1 : Integer, TipoMedia1 : Integer, Media2 : Integer, TipoMedia2 : Integer)|Linha : Integer|

Parâmetros:

MediaReferencia: Parâmetro para o período utilizado no cálculo da média de referência do indicador.

TipoMediaReferencia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Media1: Período utilizado no cálculo da média1 do indicador.

TipoMedia1: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Media2: Período utilizado no cálculo da média1 do indicador.

TipoMedia2: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Linha: Determina qual linha será obtida:

- 0 - **Linha**
- 1 - **Linha 2**

Retorno:

Float

Exemplos:

No exemplo, a variável "fdIndex " irá receber o valor da "linha 2", considerando 8(Média de Referência), 3(Média 1) e 20(Média 1) períodos, aplicando o tipo de média aritmética para o cálculo.

```
fdIndex := DidiIndex(8,0,3,0,20,0)|1|;
```



Função DiPDiM

Descrição:

A função **DiPDiM** retorna o valor do indicador **DI+/DI-**, de acordo com o período desejado.

Sintaxe:

DiPDiM(Período : Integer)|Linha : Integer|

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Linha: Determina qual linha será obtida:

0 - **DI+**

1 - **DI-**

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o valor DI- do indicador DI+/DI-, considerando 14 período para o cálculo.

```
aux := DiPDiM(14)|1|;
```



Função DonchianCH

Descrição:

A função **DonchianCh** retorna o valor do indicador **Canal Donchian**, de acordo com o período desejado.

Sintaxe:

DonchianCh(Período : Integer) | Linha : Integer |

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Linha: Determina qual linha será obtida:

- 0 - **Média**
- 1 - **Superior**
- 2 - **Inferior**

Retorno:

Float

Exemplos:

No exemplo, a variável "nDC" irá receber o valor da linha inferior do indicador Canal Donchian, considerando 20 períodos para o cálculo.

```
nDC := DonchianCh(20) | 2 |;
```



Função DT Oscillator

Descrição:

A função **DT Oscillator** retorna o valor do indicador **DT Oscillator**, conforme os parâmetros desejados.

Sintaxe:

DT Oscillator(PeriodoEstocastico : Integer, PeriodoSK : Integer, TipoSK : Integer, PeriodoSD : Integer, TipoSD : Integer) | Linha : Integer |

Parâmetros:

PeriodoEstocastico: Período utilizado no momento do cálculo do indicador.

PeriodoSK: Período referente ao parâmetro "Período SK".

TipoSK: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

PeriodoSD: Período referente ao parâmetro "Período SD".

TipoSD: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Linha: Determina qual linha será obtida:

- 0 - **Linha 1**
- 1 - **Linha 2**

Retorno:

Float

Exemplos:

No exemplo, a variável "dtOsc" irá receber o valor do indicador DT Oscillator, considerando 12(PeriodoEstocastico), 8(PeriodoSK), 5(PeriodoSD) períodos, e tipo aritmética para o cálculo.

```
dtOsc := DT Oscillator(12, 8, 0, 5, 0) | 1 |;
```



Função Envelope

Descrição:

A função **Envelope** retorna o valor do indicador **Envelope**, de acordo com o período e média desejados.

Sintaxe:

Envelope(Percentual : Float, PeríodoMédia : Integer, TipoMédia : Integer)|Linha : Integer|

Parâmetros:

Percentual: Percentual utilizado no momento do cálculo do indicador.

PeríodoMédia: Período utilizado para o cálculo da média.

TipoMédia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Linha: Determina qual linha será obtida:

- 0 - **Superior**
- 1 - **Inferior**

Retorno:

Float

Exemplos:

No exemplo, a variável "nEnv" irá receber o valor da linha inferior do indicador Envelope, considerando 5.0 como percentual, 20 períodos e tipo de média aritmética para o cálculo.

```
nEnv := Envelope(5.0, 20 , 0)|1|;
```



Função Euroinvest

Descrição:

A função **Euroinvest** retorna o valor do indicador **Euroinvest**, conforme os parâmetros determinados.

Sintaxe:

Euroinvest(Risco: Integer, ModoCalculo : Integer, Periodo : Integer, Desvio : Float, UsarVWAP : Boolean, UsarAtr : Boolean)

Parâmetros:

Risco: Determina o tipo de perfil:

- 0 - **Zero**
- 1 - **Um**
- 2 - **Dois**
- 3 - **Três**

ModoCalculo: Tipo de média:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Periodo: Período considerado para o cálculo da média.

Desvio: Desvio da média.

UsarVWAP: Determina se o VWAP será utilizado.

UsarAtr: Determina a habilitação do StopATR.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "fEuro" o retorno da função, considerando o perfil Zero, tipode média aritmética, 21 períodos, com 2 de desvio, onde o VWAP e StopAtr estão habilitados.

```
fEuro := Euroinvest(0, 0, 21, 2.0, True, True);
```



Função FastStochastic

Descrição:

A função **FastStochastic** retorna o valor do indicador **Estocástico Rápido**, de acordo com o período desejado.

Sintaxe:

FastStochastic(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nEstRap" irá receber o valor do indicador Estocástico Rápido, considerando 14 períodos para o cálculo.

```
nEstRap := FastStochastic(14);
```



Função `FinancialVol`

Descrição:

A função **FinancialVol** retorna o valor do indicador **Volume Financeiro**, podendo-se incluir ou desconsiderar os dados: "volume projetado" e "leilão e trades diretos".

Sintaxe:

FinancialVol(**VolumeProjetado** : **Boolean**, **Agressores** : **Boolean**)

Parâmetros:

VolumeProjetado: Determina se o volume irá considerar o dado projetado.

Agressores: Determina se o volume irá desconsiderar o leilão e trades diretos.

Retorno:

Float

Exemplos:

No exemplo, a variável "vFinanceiro" irá receber o valor do indicador Volume Financeiro, desconsiderando os dados de "volume projetado" e "leilão e trades diretos".

```
vFinanceiro := FinancialVol(False, False);
```




Função ForceIndex

Descrição:

A função **ForceIndex** retorna o valor do indicador **Force Index**, de acordo com o período e tipo de média desejados.

Sintaxe:

ForceIndex(Período : Integer, TipoMedia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nForce" o valor do indicador Force Index, considerando 13 períodos e tipo de média exponencial.

```
nForce := ForceIndex(13, 1);
```



Função FrassonATR

Descrição:

A função **FrassonATR** retorna o valor do indicador **Frasson ATR**, de acordo com o fator e períodos desejados.

Sintaxe:

FrassonATR(Fator : Float, PeríodoMaxMin : Integer, PeríodoATR : Integer)|Linha : Integer|

Parâmetros:

Fator: Fator de multiplicação do ATR utilizado no momento do cálculo do indicador.

PeríodoMaxMin Determina o período de Máxima de Mínima.

PeríodoATR Determina o período do cálculo do ATR.

Linha: Determina qual linha será obtida:

0 - **Superior**

1 - **Inferior**

Retorno:

Float

Exemplos:

No exemplo, a variável "nFrasson" irá receber o valor da linha inferior do indicador Frasson ATR, considerando 0,03(Fator), 15(Período Máxima/Mínima) e 50(Período ATR) para o cálculo.

```
nFrasson := FrassonATR(0.03, 15, 50)|1|;
```



Função FrassonVH

Descrição:

A função **FrassonVH** retorna o valor do indicador **Frasson VH**, de acordo com o fator e períodos desejados.

Sintaxe:

FrassonVH(Fator : Float, PeríodoMaxMin : Integer, PeríodoVH : Integer)|Linha : Integer|

Parâmetros:

Fator: Fator de multiplicação do ATR utilizado no momento do cálculo do indicador.

PeríodoMaxMin Determina o período de Máxima de Mínima.

PeríodoVH Determina o período do cálculo do VH.

Linha: Determina qual linha será obtida:

0 - **Superior**

1 - **Inferior**

Retorno:

Float

Exemplos:

No exemplo, a variável "nFrasson" irá receber o valor da linha superior do indicador Frasson VH, considerando 0,03(Fator), 15(Período Máxima/Mínima) e 50(Período VH) para o cálculo.

nFrasson := FrassonVH(0.03, 15, 50);



Função FullStochastic

Descrição:

A função **FullStochastic** retorna o valor do indicador **Estocástico Pleno**, de acordo com o período desejado.

Sintaxe:

FullStochastic(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nPlen" irá receber o valor do indicador Estocástico Pleno, considerando 14 períodos para o cálculo.

```
nPlen := FullStochastic(14);
```



Função FuraChao

Descrição:

A função **FuraChao** retorna o valor do indicador **Fura-Chão**, de acordo com o coeficiente e deslocamento desejados.

Sintaxe:

FuraChao(Coeficiente : Float, Deslocamento : Integer)

Parâmetros:

Coeficiente: Coeficiente utilizado no momento do cálculo do indicador.

Deslocamento: Determina quantos períodos anteriores serão utilizados como base no indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "fChao" irá receber o valor do indicador Fura-Chão, considerando 0.14(Coeficiente) e 1 período(Deslocamento) para o cálculo.

```
fChao := FuraChao(0.14, 1);
```



Função FuraTeto

Descrição:

A função **FuraTeto** retorna o valor do indicador **Fura-Teto**, de acordo com o coeficiente e deslocamento desejados.

Sintaxe:

FuraTeto(Coeficiente : Float, Deslocamento : Integer)

Parâmetros:

Coeficiente: Coeficiente utilizado no momento do cálculo do indicador.

Deslocamento: Determina quantos períodos anteriores serão utilizados como base no indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "fTeto" irá receber o valor do indicador Fura-Teto, considerando 0.14(Coeficiente) e 1 período(Deslocamento) para o cálculo.

```
fTeto := FuraTeto(0.14, 1);
```



Função HeikinAshi

Descrição:

A função **HeikinAshi** retorna o valor do indicador **Heikin Ashi**, de acordo com o período e tipo de média desejados.

Sintaxe:

HeikinAshi(Media : Integer, TipoMedia : Integer) | Dado : Integer |

Parâmetros:

Media: Média utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Linha: Determina qual dado será obtido:

- 0 - **Abertura**
- 1 - **Fechamento**
- 2 - **Máxima**
- 3 - **Mínima**

Retorno:

Float

Exemplos:

No exemplo, a variável "HeikinAshi" irá receber o valor de fechamento do indicador Heikin Ashi, considerando 1(Período) e 0(Aritmética) para o cálculo.

```
nHA := HeikinAshi(1, 0) | 1 |;
```



Função HiLoActivator

Descrição:

A função **HiloActivator** retorna o valor do indicador **HiLo Activator**, de acordo com o período desejado.

Sintaxe:

HiloActivator(Período : Integer) | Linha : Integer |

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Linha: Determina qual linha será obtida:

0 - **Valor Indicador**

1 - **Tendência**

Retorno para identificação da tendência:

0 - **Baixa**

1 - **Alta**

Retorno:

Float

Exemplos:

No exemplo, a variável "nHiLo" irá receber o valor do indicador HiLo Activator, considerando 3 períodos para o cálculo.

```
nHiLo := HiloActivator(3);
```




Função HistVolatility

Descrição:

A função **HistVolatility** retorna o valor do indicador **Volatilidade Histórica**, de acordo com o período e tipo de média desejados.

Sintaxe:

HistVolatility(Período : Integer, TipoMedia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**

Retorno:

Float

Exemplos:

No exemplo, a variável "nVH" irá receber o valor do indicador Volatilidade Histórica, considerando 22(Períodos) e tipo de média exponencial.

```
nVH := HistVolatility(22, 2);
```



Função HSI

Descrição:

A função **HSI** retorna o dado do indicador **IFR Índice de Força Harmônico (HSI)**, conforme o período desejado.

Sintaxe:

HSI(Período : Integer)

Parâmetros:

Período: Período para o cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vHSI" o retorno da função, considerando 3 períodos para o cálculo.

```
vHSI := HSi(3);
```



Função HullMovingAverage

Descrição:

A função **HullMovingAverage** retorna o valor do indicador **Hull Moving Average**, de acordo com o período desejado.

Sintaxe:

HullMovingAverage(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "vHMV" irá receber o valor do indicador Hull Moving Average, considerando 8 períodos para o cálculo.

```
vHMV := HullMovingAverage(8);
```



Função IchimokuCloud

Descrição:

A função **IchimokuCloud** retorna o valor do indicador **Ichimoku Cloud**, de acordo com os parâmetros desejados.

Sintaxe:

IchimokuCloud(TenkanSen : Integer, KijunSen : Integer, SenkouSpanB : Integer) | Linha : Integer |

Parâmetros:

TenkanSen: Utilizado no momento do cálculo do indicador.

KijunSen: Utilizado no momento do cálculo do indicador.

SenkouSpanB: Utilizado no momento do cálculo do indicador.

Linha: Determina qual linha será obtida:

- 0 - **Tenkan-Sen**
- 1 - **Kijun-Sen**
- 2 - **Chikou Span**
- 3 - **Senkou Span A**
- 4 - **Senkou Span B**

Retorno:

Float

Exemplos:

No exemplo, a variável "nIchimoku" irá receber o valor da linha "Senkou Span B" do indicador Ichimoku Cloud, considerando 9(Tenkan-Sen), 26(Kijun-Sen) e 52(Senkou Span B) para o cálculo.

```
nIchimoku := IchimokuCloud(9, 26, 52)|4|;
```



Função ImpliedVolatility

Descrição:

A função **ImpliedVolatility** retorna o valor do indicador **Volatilidade Implícita**, de acordo com o período desejado.

Sintaxe:

ImpliedVolatility(ModeloTeorico : Boolean, TipoOpcao : Boolean)

Parâmetros:

ModeloTeorico: Determina o modelo para o cálculo:

True - **Black & Scholes**

False - **Binomial**

TipoOpcao: Determina o tipo da opção.

True - **Americana**

False - **Européia**

Retorno:

Float

Exemplos:

No exemplo, a variável "nIV" irá receber o valor do indicador Volatilidade Implícita, utilizando o modelo Black & Scholes.

```
nIV := ImpliedVolatility(True, False);
```



Função KeltnerCH

Descrição:

A função **KeltnerCH** retorna o valor do indicador **Keltner Channels**, de acordo com o período e tipo de média desejados.

Sintaxe:

KeltnerCH(Desvio : Float, Período : Integer, TipoMedia : Integer)|Linha : Integer|

Parâmetros:

Desvio: Desvio utilizado no momento do cálculo do indicador.

Período: Período utilizado para o cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Linha: Determina qual linha será obtida:

- 0 - **Superior**
- 1 - **Inferior**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nCh" o valor da linha inferior do indicador Keltner Channels, considerando 2.0(Desvio), 20(Períodos) e tipo de média exponencial.

```
nCh := KeltnerCH(2.0, 20, 1)|1|;
```



Função KVO

Descrição:

A função **KVO** retorna o valor do indicador **KVO**, de acordo com os períodos desejados.

Sintaxe:

KVO(MediaLonga : Integer, MediaCurta : Integer, Sinal : Integer) | Dado : Integer |

Parâmetros:

MediaLonga: Determina o período da Média Longa para formação do cálculo.

MediaCurta: Determina o período da Média Curta.

Sinal: Determina o sinal para a formação do cálculo.

Linha: Determina qual linha será obtida:

0 - **Linha**

1 - **Histograma**

Retorno:

Float

Exemplos:

No exemplo, a variável "n" irá receber o valor do histograma do indicador KVO, considerando 55(Média Longa), 34(Média Curta), 13(Sinal) para o cálculo.

```
n := KVO(55, 34, 13) | 1 |;
```



Função LSVolatilityIndex

Descrição:

A função **LSVolatilityIndex** retorna o valor do indicador **L&S Volatility Index**.

Sintaxe:

LSVolatilityIndex

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "vLSV" irá receber o valor do indicador L&S Volatility Index.

```
vLSV := LSVolatilityIndex;
```




Função MACD

Descrição:

A função **MACD** retorna o valor do indicador **MACD Linha e Histograma**, de acordo com os períodos desejados.

Sintaxe:

MACD(MediaLonga : Integer, MediaCurta : Integer, Sinal : Integer)|Dado : Integer|

Parâmetros:

MediaLonga: Determina o período da Média Longa para formação do cálculo.

MediaCurta: Determina o período da Média Curta.

Sinal: Determina o sinal para a formação do cálculo.

Linha: Determina qual linha será obtida:

0 - **Linha**

1 - **Histograma**

Retorno:

Float

Exemplos:

No exemplo, a variável "fMACD" irá receber o valor do histograma, considerando 26(Média Longa), 12(Média Curta), 9(Sinal).

```
fMACD := MACD(26, 12, 9)|1|;
```



Função MFI

Descrição:

A função **MFI** retorna o valor do indicador **Market Facilitation Index**.

Sintaxe:

MFI

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nMFI" irá receber o valor do indicador MFI.

```
nMFI := ;
```



Função MIMA

Descrição:

A função **MIMA** possui como característica retornar o dado do indicador PhiCube - MIMA.

Sintaxe:

MIMA(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vMIMA" o valor do indicador MIMA, considerando 10 períodos para o cálculo.

```
vMIMA := MIMA(10);
```



Função Momentum

Descrição:

A função **Momentum** retorna o valor do indicador **Momentum**, de acordo com o período e tipo de média desejados.

Sintaxe:

Momentum(Período : Integer, Média : Integer, TipoMédia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Média: Média utilizada no momento do cálculo do indicador.

TipoMédia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nMomentum" irá receber o valor do indicador Momentum, considerando 10(Períodos), 3(Média) e tipo de média exponencial para o cálculo.

```
nMomentum := Momentum(10, 3, 1);
```



Função MomentumStochastic

Descrição:

A função **MomentumStochastic** retorna o valor do indicador **Momento Estocástico**, de acordo com o período desejado.

Sintaxe:

MomentumStochastic(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nMS" irá receber o valor do indicador Momento Estocástico, considerando 14 período para o cálculo.

```
nMS := MomentumStochastic(14);
```



Função MoneyFlow

Descrição:

A função **MoneyFlow** retorna o valor do indicador **Money Flow**.

Sintaxe:

MoneyFlow

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nMoney" o valor do indicador Money Flow.

```
nMoney := MoneyFlow;
```



Função MoneyFlowIndex

Descrição:

A função **MoneyFlowIndex** retorna o valor do indicador **Money Flow Index**, de acordo com o período desejado.

Sintaxe:

MoneyFlowIndex(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nMoneyF" irá receber o valor do indicador Money Flow Index, considerando 14 período para o cálculo.

```
nMoneyF := MoneyFlowIndex(14);
```



Função NelogicaBottomFinder

Descrição:

A função **NelogicaBottomFinder** retorna o valor do indicador **Nelogica Bottom Finder**, conforme o dado desejado (linha e histograma).

Sintaxe:

NelogicaBottomFinder | Dado : Integer |

Parâmetros:

Dado: Determina qual dado será obtido:

0 - **Linha**

1 - **Histograma**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nBF" o dado do histograma.

```
nBF := NelogicaBottomFinder | 1 |;
```




Função NelogicaPullBackFinder

Descrição:

A função **NelogicaPullBackFinder** retorna o valor do indicador **Nelogica Pullback Finder**, conforme o dado desejado (linha e histograma).

Sintaxe:

NelogicaPullBackFinder | Dado : Integer |

Parâmetros:

Dado: Determina qual dado será obtido:

0 - **Linha**

1 - **Histograma**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nPF" o dado do histograma.

```
nPF := NelogicaPullBackFinder | 1 |;
```



Função NelogicaWeisWave

Descrição:

A função **NelogicaWeisWave** retorna o valor do indicador **Nelogica Weis Wave**, de acordo com o período desejado.

Sintaxe:

NelogicaWeisWave(Período : Integer)

Parâmetros:

Período: Determina o período para o cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nWW" o retorno da função, considerando 3 períodos para o cálculo.

```
nWW := NelogicaWeisWave(3);
```



Função OBV

Descrição:

A função **OBV** retorna o valor do indicador **OBV**.

Sintaxe:

OBV

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nOBV" irá receber o valor do indicador OBV.

```
nOBV := OBV;
```



Função OBVAvg

Descrição:

A função **OBVAvg** retorna o valor do indicador **OBV Ponderado**.

Sintaxe:

OBVAvg(Período : Integer, TipoMedia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nOBV" o retorno da função OBVAvg, considerando 10(Períodos) e tipo de média ponderada(3).

```
nOBV := OBVAvg(10, 3);
```



Função OnBalanceTR

Descrição:

A função **OnBalanceTR** retorna o valor do indicador **On-Balance True Range**.

Sintaxe:

OnBalanceTR

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nOBTR" irá receber o valor do indicador On-Balance True Range.

```
nOBTR := OnBalanceTR;
```



Função `OpenInterest`

Descrição:

A função `OpenInterest` retorna o valor do indicador `Contratos em Aberto`.

Sintaxe:

`OpenInterest`

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, será atribuída à variável "nOpen" o dado do indicador.

```
nOpen := OpenInterest;
```



Função ParabolicSAR

Descrição:

A função **ParabolicSAR** retorna o valor do indicador **SAR Parabólico**, de acordo com os parâmetros desejados.

Sintaxe:

ParabolicSAR(Fator : Float, Limite : Float)

Parâmetros:

Fator: Determina o Fator de Aceleração para formação do cálculo.

Limite: Determina o Limite de Aceleração.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nSAR" o valor do indicador SAR Parabólico, considerando 0.02(Fator) e 0.2(Limite) para o cálculo.

```
nSAR := ParabolicSAR(0.02, 0.2);
```



Função Phibo

Descrição:

A função **Phibo** retorna o valor do indicador **PhiCube - Phibo Line**.

Sintaxe:

Phibo(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vPhibo" o retorno da função, considerando 17(Períodos) para o cálculo.

```
vPhibo := Phibo(17);
```




Função Pivot

Descrição:

A função **Pivot** retorna o valor do indicador **Pivot**, de acordo com os parâmetros específicos desejados.

Sintaxe:

Pivot(Normal : Boolean, TresLinhas : Boolean)|Linha : Integer|

Parâmetros:

Normal: Determina o tipo de cálculo que será efetuado para o Pivot:

True - **(Máxima + Mínima + Fechamento) / 3**

False - **(Abertura + Máxima + Mínima + Fechamento) / 4**

TresLinhas: Determina quantas linhas serão consideradas:

True - **Três Linhas**

False - **Duas Linhas**

Linha: Determina qual linha será obtida:

True(Três Linhas):

0 - **Pivot**

1 - **R3**

2 - **S3**

3 - **R2**

4 - **S2**

5 - **R1**

6 - **S1**

False(Duas Linhas):

0 - **Pivot**

1 - **R2**

2 - **S2**

3 - **R1**

4 - **S1**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "aux" a linha S3, considerando para o Pivot o tipo de cálculo: (Máxima + Mínima + Fechamento) / 3.

```
aux := Pivot(True, True)|2|;
```



Função PriceOsc

Descrição:

A função **PriceOsc** retorna o valor do indicador **Oscilador de Preços**, de acordo com os períodos e tipos de médias desejados.

Sintaxe:

PriceOsc(Media1 : Integer, TipoMedia1 : Integer, Media2 : Integer, TipoMedia2 : Integer)

Parâmetros:

Media1: Período utilizado para a média 1.

TipoMedia1: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Media2: Período utilizado para a média 2.

TipoMedia2: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nPrice" irá receber o valor do indicador Oscilador de Preços, considerando 12(Média 1) e 21(Média 2) períodos, e tipo exponencial para o cálculo.

```
nPrice := PriceOsc(12, 1, 21, 1);
```



Função PriceOscillator

Descrição:

A função **PriceOscillator** retorna o valor do indicador **Price Oscillator**, de acordo com os parâmetros desejados.

Sintaxe:

PriceOscillator(SerieDados : Serie, ComprimentoRapido : Integer, ComprimentoLento : Integer)

Parâmetros:

SerieDados: Série utilizada para o cálculo do indicador.

ComprimentoRapido: Referente ao parâmetro FastLength.

ComprimentoLento: Referente ao parâmetro SlowLength.

Retorno:

Float

Exemplos:

No exemplo, a variável "pOsc" irá receber o valor do indicador Price Oscillator, considerando a série de máxima(SerieDados), 9(ComprimentoRapido) e 18(ComprimentoLento) para o cálculo.

```
pOsc := PriceOscillator(High, 9, 18);
```



Função PriceVolumeTrend

Descrição:

A função **PriceVolumeTrend** retorna o valor do indicador **Tendência Preço/Volume** .

Sintaxe:

PriceVolumeTrend

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nVolTrend" o dado do indicador Tendência Preço/Volume.

```
nVolTrend := PriceVolumeTrend;
```



Função PriorCote

Descrição:

A função **PriorCote** retorna o valor do indicador **Prior Cote**, de acordo com o dado desejado.

Sintaxe:

PriorCote(Dado : Integer)

Parâmetros:

Dado: Determina o dado que será obtido:

- 0 - **Fechamento**
- 1 - **Abertura**
- 2 - **Máxima**
- 3 - **Mínima**
- 4 - **Ajuste**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nPrior" o valor de fechamento do período anterior.

```
nPrior := PriorCote(0);
```



Função PTAX

Descrição:

A função **PTAX** retorna os dados do indicador **TR - PTAX**.

Sintaxe:

PTAX[Dado : Integer]

Parâmetros:

Dado: Obtém o dado desejado do indicador:

- 0 - **Oficial**
- 1 - **P4**
- 2 - **P3**
- 3 - **P2**
- 4 - **P1**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vPTAX" a PTAX oficial.

```
vPTAX := PTAX;
```



Função PTAXFuturo

Descrição:

A função **PTAXFuturo** retorna os dados do indicador **TR - PTAX Futuro**.

Sintaxe:

PTAXFuturo[Dado : Integer]

Parâmetros:

Dado: Obtém o dado desejado do indicador:

- 0 - **Oficial**
- 1 - **P4**
- 2 - **P3**
- 3 - **P2**
- 4 - **P1**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vPTAX" a PTAXF oficial.

```
vPTAXF := PTAXFuturo;
```



Função QuantityVol

Descrição:

A função **QuantityVol** retorna o valor do indicador **Volume Quantidade**, podendo-se incluir ou desconsiderar os dados: "volume projetado" e "leilão e trades diretos".

Sintaxe:

QuantityVol(VolumeProjetado : Boolean, Agressores : Boolean)

Parâmetros:

VolumeProjetado: Determina se o volume irá considerar o dado projetado.

Agressores: Determina se o volume irá desconsiderar o leilão e trades diretos.

Retorno:

Float

Exemplos:

No exemplo, a variável "vQuantidade" irá receber o valor do indicador Volume Quantidade, desconsiderando os dados de "volume projetado" e "leilão e trades diretos".

```
vQuantidade := QuantityVol(False, False);
```




Função Rafi

Descrição:

A função **Rafi** retorna o valor do indicador **Rafi**.

Sintaxe:

Rafi

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nRafi" irá receber o dado do indicador específico.

```
nRafi := Rafi;
```



Função Ravi

Descrição:

A função **Ravi** retorna o valor do indicador **Ravi**, de acordo com os períodos desejados.

Sintaxe:

Ravi(MediaCurta : Integer, MediaLonga : Integer)

Parâmetros:

MediaCurta: Período utilizado na média curta para o cálculo do indicador.

MediaLonga: Período utilizado na média longa.

Retorno:

Float

Exemplos:

No exemplo, a variável "nRavi" irá receber o valor do indicador Ravi, considerando 7(Média Curta) e 65(Média Longa) períodos para o cálculo.

```
nRavi := Ravi(7, 65);
```



Função RenkoVTwo

Descrição:

A função **RenkoVTwo** retorna o valor do indicador **RenkoV2**, de acordo com os parâmetros desejados.

Sintaxe:

RenkoVTwo(Período : Integer, Abertura : Float, Deslocamento : Integer) | Linha : Integer |

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Abertura: Referente ao parâmetro "Abertura" do indicador.

Deslocamento: Relacionado ao parâmetro "Deslocamento" do indicador.

Linha: Determina qual linha será obtida:

0 - **Linha RenkoV2+**

1 - **Linha RenkoV2-**

Retorno:

Float

Exemplos:

No exemplo, a variável "renkoV" irá receber o valor da linha RenkoV2- do indicador RenkoV2, considerando 20(Período), 1.5(Abertura) e 0(Deslocamento) para o cálculo.

```
renkoV := RenkoVTwo(20, 1.5, 0)|1|;
```



Função RSI

Descrição:

A função **RSI** retorna o valor do indicador **IFR(RSI)**, de acordo com o período e tipo desejados.

Sintaxe:

RSI(Período : Integer, Tipo : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Tipo: Tipo de cálculo aplicado:

0 - **Clássico**

1 - **Simples**

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o valor do indicador IFR(RSI), considerando 2 períodos e tipo clássico para o cálculo.

```
aux := RSI(2, 0);
```



Função RsiStochastic

Descrição:

A função **RsiStochastic** retorna o valor do indicador **IFR Estocástico**, de acordo com o período desejado.

Sintaxe:

RsiStochastic(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nRS" o valor do indicador IFR Estocástico, considerando 2 períodos para o cálculo.

```
nRS := RsiStochastic(2);
```



Função ROC

Descrição:

A função **ROC** retorna o valor do indicador **ROC**, de acordo com os períodos e tipo de média desejados.

Sintaxe:

ROC(Período : Integer, Media : Integer, TipoMedia : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Media: Período da média utilizada.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nRoc" irá receber o valor do indicador ROC, considerando 3(Período), 9(Média) e tipo de aritmética.

```
nRoc := ROC(3, 9, 0);
```



Função SafeZoneDownTrend

Descrição:

A função **SafeZoneDownTrend** retorna o valor do indicador **Stop SafeZone DownTrend**, de acordo com os parâmetros desejados.

Sintaxe:

SafeZoneDownTrend(Multiplicador : Float, Período : Integer, Deslocamento : Integer)

Parâmetros:

Multiplicador: : Valor de Multiplicador utilizado no cálculo do indicador.

Período: Período considerado.

Deslocamento: Deslocamento de períodos.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "aux" o dado do indicador Stop SafeZone DownTrend, considerando 2.0(Multiplicador), 10(Período) e 0(Deslocamento) para o cálculo.

```
aux := SafeZoneDownTrend(2.0, 10, 0);
```



Função SafeZoneUpTrend

Descrição:

A função **SafeZoneUpTrend** retorna o valor do indicador **Stop SafeZone UpTrend**, de acordo com os parâmetros desejados .

Sintaxe:

SafeZoneUpTrend(Multiplicador : Float, Período : Integer, Deslocamento : Integer)

Parâmetros:

Multiplicador: : Valor de Multiplicador utilizado no cálculo do indicador.

Período: Período considerado.

Deslocamento: Deslocamento de períodos.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "aux" o dado do indicador Stop SafeZone UpTrend, considerando 2.0(Multiplicador), 10(Período) e 0(Deslocamento) para o cálculo.

```
aux := SafeZoneUpTrend(2.0, 10, 0);
```




Função Santo

Descrição:

A função **Santo** retorna o valor do indicador **PhiCube - Santo**, de acordo com o período desejado.

Sintaxe:

Santo(Período : Integer)|Linha : Integer|

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Linha: Determina qual linha será obtida:

- 0 - **Dado referente à linha Santo.**
- 1 - **Dado relacionado ao Sinal.**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vSanto" o dado do sinal, considerando 72(Períodos) para o cálculo.

```
vSanto := Santo(72)|1|;
```



Função SlowStochastic

Descrição:

A função **SlowStochastic** retorna o valor do indicador **Estocástico Lento**, de acordo com o período desejado.

Sintaxe:

SlowStochastic(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nEstLen" irá receber o valor do indicador Estocástico Lento, considerando 14 períodos para o cálculo.

```
nEstLen := FastStochastic(14);
```



Função StopATR

Descrição:

A função **StopATR** retorna o valor do indicador **Stop ATR**, de acordo com os parâmetros desejados.

Sintaxe:

StopATR(Desvio : Float, Período : Integer, TipoMedia : Integer) | Dado : Integer |

Parâmetros:

Desvio: Desvio utilizado para o cálculo do indicador.

Período: Período que será considerado.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Dado: Determina o dado que será obtido:

- 0 - **Valor indicador.**
- 1 - **Flag para informar a ocorrência(retorno):**
 - 0 - **ATR+**
 - 1 - **ATR-**

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o valor do indicador Stop ATR, considerando 2.0(Desvio), 20(Períodos) para o cálculo.

```
aux := StopATR(2.0, 20, 0);
```



Função Tilson

Descrição:

A função **Tilson** retorna o valor do indicador **Tilson's T3 Moving Average**, de acordo com os parâmetros desejados.

Sintaxe:

Tilson(Fator : Float, Media : Integer)

Parâmetros:

Fator: Determina o Fator para formação do cálculo.

Media: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nTilson" irá receber o valor do indicador **Tilson's T3 Moving Average**, considerando 0.7(Fator) e 3(Media) para o cálculo.

```
nTilson := Tilson(0.7, 3;
```



Função TimeAgrBuySell

Descrição:

A função **TimeAgrBuySell** retorna o valor do indicador **TR - Tempo Agressão - Compra**.

Sintaxe:

TimeAgrBuySell(AlertaVariacoes : Integer)

Parâmetros:

AlertaVariacoes: Quantidade de variações que serão consideradas.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "alertV" o retorno da função, considerando 3 variações.

```
alertV := TimeAgrBuySell(3);
```



Função TimeAgrTotal

Descrição:

A função **TimeAgrTotal** retorna o valor do indicador **TR - Tempo Agressão - Total**.

Sintaxe:

TimeAgrTotal(AlertaVariacoes : Integer)

Parâmetros:

AlertaVariacoes: Quantidade de variações que serão consideradas.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "alertT" o retorno da função, considerando 3 variações.

```
alertT := TimeAgrTotal(3);
```



Função TRIX

Descrição:

A função **TRIX** retorna o valor do indicador **TRIX**, de acordo com o período e tipo de média desejados.

Sintaxe:

TRIX(Media : Integer, TipoMedia : Integer)

Parâmetros:

Media: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nTrix" irá receber o valor do indicador TRIX, considerando 9(Média) períodos e tipo exponencial para o cálculo.

```
nTrix := TRIX(9, 1);
```



Função TRIXM

Descrição:

A função **TRIXM** retorna o valor do indicador **TRIXM**, de acordo com o período e tipo de média desejados.

Sintaxe:

TRIXM(Media : Integer, TipoMedia : Integer)

Parâmetros:

Media: Período utilizado no momento do cálculo do indicador.

TipoMedia: Determina qual média será considerada:

- 0 - **Aritmética**
- 1 - **Exponencial**
- 2 - **Welles Wilder**
- 3 - **Ponderada**

Retorno:

Float

Exemplos:

No exemplo, a variável "nTrixm" irá receber o valor do indicador TRIXM, considerando 9(Média) períodos e tipo exponencial para o cálculo.

```
nTrixm := TRIXM(9, 1);
```




Função TopBottomDetector

Descrição:

A função **TopBottomDetector** retorna o valor do indicador **Detector de Topos e Fundos**, de acordo com o período desejado.

Sintaxe:

TopBottomDetector(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "TBD" o valor do indicador Detector de Topos e Fundos, considerando 2 período para o cálculo.

```
TBD := TopBottomDetector(2);
```



Função Trades

Descrição:

A função **Trades** retorna o valor do indicador **Negócios**.

Sintaxe:

Trades

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "n" irá receber o dado do indicador Negócios.

```
n := Trades;
```



Função TwoMVAggression

Descrição:

A função **TwoMVAggression** retorna o dado do indicador **2MV Agressão**.

Sintaxe:

TwoMVAggression

Parâmetros:

Sem parâmetros

Retorno:

Float(Dado Obtido):

Exemplos:

No exemplo, será atribuído o retorno da função na variável "2mvAgressao".

```
2mvAgressao := TwoMVAggression;
```



Função TwoMVPower

Descrição:

A função **TwoMVPower** retorna o valor do indicador **2MV Power**, de acordo com os parâmetros desejados.

Sintaxe:

TwoMVPower(Periodo1 : Integer, Periodo2 : Integer, Periodo3 : Integer, Media : Integer)

Parâmetros:

Periodo1: Período em minutos utilizado no momento do cálculo do indicador.

Periodo2: Período em minutos.

Periodo3: Período em minutos.

Media: Período determinado para a média.

Retorno:

Float(Dado Obtido):

-1 - **Baixa**

0 - **Neutro**

1 - **Alta**

Exemplos:

No exemplo, caso ocorra a flag de alta, será aplicada uma coloração(verde).

```
if(TwoMVPower(2, 5, 15, 20) = 1) then
```

```
    PaintBar(clGreen);
```



Função TwoMVStandard

Descrição:

A função **TwoMVStandard** retorna o dado do indicador **2MV Padrão**.

Sintaxe:

TwoMVStandard

Parâmetros:

Sem parâmetros

Retorno:

Float(Dado Obtido):

Exemplos:

No exemplo, será atribuído o retorno da função na variável "2mvP".

```
2mvP := TwoMVStandard;
```



Função VSS

Descrição:

A função **VSS** retorna o valor do indicador **VSS**, de acordo com os parâmetros desejados.

Sintaxe:

VSS(Multiplicador : Float, Media : Integer, Deslocamento : Integer)

Parâmetros:

Multiplicador: : Valor de Multiplicador utilizado no cálculo do indicador.

Media: Período da média utilizada.

Deslocamento: Deslocamento de períodos.

Retorno:

Float

Exemplos:

No exemplo, a variável "nVSS" irá receber o valor do indicador VSS, considerando 1.5(Multiplicador), 5(Média) períodos e 0(Deslocamento) para o cálculo.

```
nVSS := VSS(1.5, 5, 0);
```



Função VWAP

Descrição:

A função **VWAP** retorna o valor do indicador **VWAP**, de acordo com a periodicidade desejada.

Sintaxe:

VWAP(Período : Integer)

Parâmetros:

Período: Período para obter o dado do indicador:

- 0 - **Barra**
- 1 - **Diário**
- 2 - **Semanal**
- 3 - **Mensal**

Retorno:

Float

Exemplos:

No exemplo, a variável "nVWAP" irá receber o valor do indicador VWAP, considerando a periodicidade diária.

```
nVWAP := VWAP(1);
```



Função VWAPMonthly

Descrição:

A função **VWAPMonthly** retorna o valor do indicador **VWAP Mensal**.

Sintaxe:

VWAPMonthly

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nVWAP" irá receber o valor do indicador VWAPMonthly.

```
nVWAP := VWAPMonthly;
```




Função VWAPWeekly

Descrição:

A função **VWAPWeekly** retorna o valor do indicador **VWAP Semanal**.

Sintaxe:

VWAPWeekly

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, a variável "nVWAP" irá receber o valor do indicador VWAPWeekly.

```
nVWAP := VWAPWeekly;
```



Função VWMA

Descrição:

A função **VWMA** retorna o valor do indicador **VWMA**, de acordo com o período desejado.

Sintaxe:

VWMA(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "nVWMA" irá receber o valor do indicador VWMA, considerando 10 períodos para o cálculo.

```
nVWMA := VWMA(10);
```



Função WAverage

Descrição:

A função **WAverage** retorna o valor do indicador Média Móvel, tipo ponderada.

Sintaxe:

WAverage(TipoSerie : SeriePeriodo, Período : Integer)

Parâmetros:

TipoSerie: Série que será considerada para o cálculo.

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vMed" o valor do indicador Média Móvel(Ponderada), considerando 100 períodos, e a série de fechamento(Close) para o cálculo.

```
vMed := WAverage(Close, 100);
```



Função Williams

Descrição:

A função **Williams** retorna o valor do indicador **Williams %R**, de acordo com o período desejado.

Sintaxe:

Williams(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "nW" o valor do indicador Williams %R, considerando 14 períodos para o cálculo.

```
nW := Williams(14);
```



Função xAverage

Descrição:

A função **xAverage** retorna o valor do indicador Média Móvel, tipo exponencial.

Sintaxe:

xAverage(TipoSerie : SeriePeriodo, Período : Integer)

Parâmetros:

TipoSerie: Série que será considerada para o cálculo.

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vMed" o valor do indicador Média Móvel(Exponencial), considerando 100 períodos, e a série de fechamento(Close) para o cálculo.

```
vMed := xAverage(Close, 100);
```



Livro

Função AskPrice

Descrição:

A função **AskPrice** retorna o preço da melhor oferta de venda.

Sintaxe:

AskPrice

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos

No exemplo, será atribuído à variável "ask" o valor do topo(melhor oferta de venda) do livro.

```
ask := AskPrice;
```



Função AskSize

Descrição:

A função **AskSize** retorna a quantidade da melhor oferta de venda.

Sintaxe:

AskSize

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos

No exemplo, será atribuído à variável "qtdAsk" a quantidade do topo(melhor oferta de venda) do livro.

```
qtdAsk := AskSize;
```



Função BidPrice

Descrição:

A função **BidPrice** retorna o preço da melhor oferta de compra.

Sintaxe:

BidPrice

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos

No exemplo, será atribuído à variável "bid" o valor do topo(melhor oferta de compra) do livro.

```
bid := BidPrice;
```




Função BidSize

Descrição:

A função **BidSize** retorna a quantidade da melhor oferta de compra.

Sintaxe:

BidSize

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos

No exemplo, será atribuído à variável "qtdBid" a quantidade do topo(melhor oferta de compra) do livro.

```
qtdBid := BidSize;
```



Função BookSpread

Descrição:

A função **BookSpread** retorna a diferença entre os melhores preços de compra e venda, no topo do livro.

Sintaxe:

BookSpread

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos

No exemplo, será atribuído à variável "spread" a diferença entre os valores das melhores ofertas de compra e venda.

```
spread := BookSpread;
```



Função IsBMF

Descrição:

A função **IsBMF** retorna se o ativo pertence ao segmento BMF.

Sintaxe:

IsBMF

Parâmetros:

Sem parâmetros.

Retorno:

Boolean

Exemplos

No exemplo, caso o ativo pertença ao BMF, será plotado o preço da melhor oferta de venda.

```
if(IsBMF) then
```

```
    Plot(AskPrice);
```



Função Lote

Descrição:

A função **Lote** retorna a quantidade mínima de contratos referente ao lote do ativo.

Sintaxe:

Lote

Parâmetros:

Sem parâmetros.

Retorno:

Integer

Exemplos

No exemplo, será atribuído à variável "qtd" a quantidade do lote.

```
qtd := Lote;
```



Função `MinPriceIncrement`

Descrição:

A função `MinPriceIncrement` retorna o incremento mínimo do preço do ativo.

Sintaxe:

`MinPriceIncrement`

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos

No exemplo, será atribuído à variável "incMin" o valor do incremento mínimo.

```
incMin := MinPriceIncrement;
```



Matemáticas

Função ABS

Descrição:

A função **ABS** retorna o valor absoluto de um número ponto flutuante.

Sintaxe:

ABS(Valor : Float)

Parâmetros:

Valor: Valor ou variável para obter o módulo.

Retorno:

Float

Exemplos:

No exemplo, a variável "n" irá receber o módulo(4) do valor -4.

```
n := ABS(-4);
```



Função Arctangent

Descrição:

A função **Arctangent** retorna o arcotangente(em graus) de determinado número.

Sintaxe:

Arctangent(Numero : Float)

Parâmetros:

Numero: Número que será convertido.

Retorno:

Tipodado:

Exemplos:

No exemplo, a variável "arc" irá receber o arcotangente, em graus, do número 12.

```
arc := Arctangent(12);
```



Função Ceiling

Descrição:

A função **Ceiling** efetua um arredondamento, retornando o menor inteiro maior que um número específico.

Sintaxe:

Ceiling(Numero : Float)

Parâmetros:

Numero: Número que será arredondado.

Retorno:

Integer

Exemplos:

No exemplo abaixo, usamos a função **Ceiling** para retornar o menor inteiro maior que o valor 2,3(Retorno: 3).

```
aux := Ceiling(2.3);
```




Função Combination

Descrição:

A função **Combination** calcula o número de grupos com combinação única, considerando um conjunto específico de números.

Sintaxe:

Combination(Numero : Integer, QtdGrupos : Integer)

Parâmetros:

Numero: Total de números, ou itens, a serem considerados;

QtdGrupos: Número de itens únicos em cada grupo.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "nComb" o número de grupos da combinação(4,2), onde o resultado será 6, conforme o cálculo: (1,2), (1,3), (1,4), (2,3), (2,4) e (3,4).

```
nComb := Combination(4,2);
```



Função Cos

Descrição:

A função **Cos** tem como objetivo retornar ao usuário o Cosseno de um valor em radianos.

Sintaxe:

Cos(Valor : Float)

Parâmetros:

Valor: Valor ou variável para obter o Cosseno.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "nCos" o Cosseno do valor 180 em radianos(-0,60).

```
nCos := Cos(180);
```



Função Cosine

Descrição:

A função **Cosine** tem como objetivo retornar ao usuário o Cosseno de um valor em graus.

Sintaxe:

Cosine(Valor : Float)

Parâmetros:

Valor: Valor ou variável para obter o Cosseno.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "nCos" o Cosseno do valor 45 em graus(0,71).

```
nCos := Cosine(45);
```



Função Cotangent

Descrição:

A função **Cotangent** tem como objetivo retornar ao usuário a Cotangente de um valor em graus.

Sintaxe:

Cotangent(Valor : Float)

Parâmetros:

Valor: Valor ou variável para obter a Cotangente.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "nCo" a Cotangente do valor 30 em graus.

```
nCo := Cotangent(30);
```



Função Cum

Descrição:

A função **Cum** acumula o valor de uma série de dados, desde a primeira barra até a atual.

Sintaxe:

Cum(SerieDeDados : Serie)

Parâmetros:

SerieDeDados: Série para efetuar o somatório.

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o somatório da diferença entre as séries Close e Open.

```
aux := Cum(Close - Open);
```



Função Exp

Descrição:

A função **Exp** tem como objetivo retornar ao usuário a enésima potência do número de Euler.

Sintaxe:

Exp(Valor : Float)

Parâmetros:

Valor: Valor ou uma variável para obter a enésima potência(Euler).

Retorno:

Float

Exemplos:

No seguinte exemplo, a função Exp recebe o valor de "2" e irá retornar o valor "7,39".

```
n := Exp(2);
```



Função ExpValue

Descrição:

A função **ExpValue** possui como finalidade retornar o valor exponencial de um determinado número(e^x).

Sintaxe:

ExpValue(Valor : Float)

Parâmetros:

Valor: Valor ou uma variável para obter o valor exponencial(e^x).

Retorno:

Float

Exemplos:

No seguinte exemplo, a função ExpValue recebe o valor de "2" e irá retornar o valor "7,39".

```
n := ExpValue(2);
```



Função ExtremePriceRatio

Descrição:

A função **ExtremePriceRatio** retorna o ratio das extremidades (divide o maior valor no período pelo menor valor) de um número determinado de barras.

Sintaxe:

ExtremePriceRatio(Length : Integer, UseLog : Boolean)

Parâmetros:

Length: O número de barras que serão considerados no cálculo.

UseLog: Determina se o logaritmo de 10 do resultado da divisão será aplicado.

True - **Não é calculado o LOG**

False - **É calculado o LOG**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "pRatio" o retorno da função, considerando 300 períodos para a divisão.

```
pRatio := ExtremePriceRatio(300, True);
```




Função Factorial

Descrição:

A função **Factorial** tem como finalidade efetuar o cálculo fatorial($n!$) de um número natural n .

Sintaxe:

Factorial(Valor : Float)

Parâmetros:

Valor: Número natural para o cálculo do factorial.

Retorno:

Float

Exemplos:

No exemplo, a variável "nFactorial" irá receber o fatorial do número 4.

```
nFactorial := Factorial(4);
```



Função FastD

Descrição:

A função **FastD** retorna o valor de **FastD** do Oscilador Estocástico, de acordo com o período desejado.

Sintaxe:

FastD(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "d" irá receber o retorno da função FastD, considerando 14 períodos para o cálculo.

```
d := FastD(14);
```



Função FastK

Descrição:

A função **FastK** retorna o valor de **FastK** do Oscilador Estocástico, de acordo com o período desejado.

Sintaxe:

FastK(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "k" irá receber o retorno da função FastK, considerando 14 períodos para o cálculo.

```
k := FastK(14);
```



Função FastKCustom

Descrição:

A função **FastKCustom** retorna o valor de **FastK** do Oscilador Estocástico, de acordo com os preços determinados por parâmetro, e período desejado.

Sintaxe:

FastKCustom(PrecoH : Serie, PrecoL : Serie, PrecoC : Serie, Perodo : Integer)

Parâmetros:

PrecoH: Série de referência para a máxima.

PrecoL: Série de referência para a mínima.

PrecoC: Série de referência para o fechamento.

Perodo: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "KCustom" o retorno da função FastKCustom, considerando as séries de máxima(PrecoH), mínima(PrecoL), fechamento(PrecoC), e 14 períodos para o cálculo.

```
KCustom := FastKCustom(High, Low, Close, 14);
```



Função Floor

Descrição:

A função **Floor** possui como finalidade retornar o maior valor inteiro menor que um número determinado.

Sintaxe:

Floor(Valor : Float)

Parâmetros:

Valor: Valor de referência para obter o dado específico.

Retorno:

Integer

Exemplos:

Nos exemplos, serão atribuídos, às variáveis "m" e "n", os valores -7 e 6, respectivamente.

```
m := Floor(-6.1);
```

```
n := Floor(6.1);
```



Função **FracPortion**

Descrição:

A função **FracPortion** tem como recurso retornar a parte fracionário de determinado número.

Sintaxe:

FracPortion(Valor : Float)

Parâmetros:

Valor: Número para obter a parte fracionária.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "dec" o valor -0.59.

```
dec := FracPortion(-6.59);
```



Função GCD

Descrição:

A função **GCD** retorna o maior denominador comum entre dois números

Sintaxe:

GCD(Valor1 : Float, Valor2 : Float)

Parâmetros:

Valor1: Primeiro valor a ser analisado;

Valor2: Segundo valor a ser analisado.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "nDem" o maior denominador comum entre 12 e 9.

```
nDem := GCD(12, 9);
```



Função HarmonicMean

Descrição:

A função **HarmonicMean** calcula a média harmônica de uma série de dados, baseada em um determinado período.

Sintaxe:

HarmonicMean(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série utilizada para o cálculo.

Período: Período utilizado no momento do cálculo.

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o retorno da função HarmonicMean, considerando a máxima(Close) e 20(Períodos) para o cálculo.

```
aux := HarmonicMean(High, 20);
```




Função IntPortion

Descrição:

A função **IntPortion** tem como recurso retornar a parte inteira de determinado número.

Sintaxe:

IntPortion(Valor : Float)

Parâmetros:

Valor: Número para obter a parte inteira.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "aux" o valor 7.

```
aux := IntPortion(7.52);
```



Função Log

Descrição:

A função **Log** retorna o logaritmo natural(ln) de um número.

Sintaxe:

Log(Valor : Float)

Parâmetros:

Log: Número para o logaritmo natural(ln).

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vLn" o ln de 5(1,6).

```
vLn := Log(5);
```



Função MidPoint

Descrição:

A função **MidPoint** retorna a média entre o maior e o menor valor encontrados no período.

Sintaxe:

MidPoint(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série de referência.

Período: Período utilizado no momento do cálculo.

Retorno:

Float

Exemplos:

No exemplo, a variável "midP" irá receber o valor da função MidPoint, considerando o fechamento(SerieDados) e 15(Períodos) para o cálculo.

```
midP := MidPoint(Close, 15);
```



Função MinutesIntoWeek

Descrição:

A função **MinutesIntoWeek** retorna o número de minutos entre domingo(Dia: 0 - Hora: 0h00) até o dia e hora determinados por parâmetro.

Sintaxe:

MinutesIntoWeek(DiaLimite : Integer, HoraLimite : Integer)

Parâmetros:

DiaLimite: Dia de limite para a conversão em minutos.

Referência de dias da semana:

- 0 - **Domingo**
- 1 - **Segunda**
- 2 - **Terça**
- 3 - **Quarta**
- 4 - **Quinta**
- 5 - **Sexta**
- 6 - **Sábado**

HoraLimite: Hora de limite para a conversão.

Retorno:

Integer

Exemplos:

No exemplo a seguir, será atribuído à variável "tMin" o total de minutos(8.640) entre domingo(Dia: 0 - Hora: 0h00) e sábado(Dia: 6 - Hora: 0h00).

```
tMin := MinutesIntoWeek(6, 0000);
```



Função MinutesToTime

Descrição:

A função **MinutesToTime** retorna a conversão de minutos em hora militar(contagem iniciada à meia noite).

Sintaxe:

MinutesToTime(Minutos : Integer)

Parâmetros:

Minutos: Minutos para a conversão em horas.

Retorno:

Integer

Exemplos:

No exemplo a seguir, a variável "nHora" irá receber a conversão de 600 minutos em horas, ou seja, será retornado o valor 1000, representando 10h.

```
nHora := MinutesToTime(600);
```



Função Mod

Descrição:

A função **Mod** possui como finalidade retornar o resto da divisão entre dois números inteiros.

Sintaxe:

Mod(Dividendo : Integer, Divisor : Integer)

Parâmetros:

Dividendo: Número referente ao Dividendo.

Divisor: Número que será o divisor.

Retorno:

Integer

Exemplos:

No exemplo, será atribuído à variável "res" o valor 1, referente ao resto da divisão: 10/3

```
res := Mod(10, 3);
```



Função MyPrice

Descrição:

A função **MyPrice** retorna a média entre a máxima, mínima e fechamento.

Sintaxe:

MyPrice

Parâmetros:

Sem parâmetros.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "mPrice", o valor da função MyPrice.

```
mPrice := MyPrice;
```



Função Neg

Descrição:

A função **Neg** retorna o valor negativo de um determinado número.

Sintaxe:

Neg(Numero : Float)

Parâmetros:

Numero: Valor para obter seu número negativo.

Retorno:

Float

Exemplos:

No exemplo, a variável "nNeg" irá receber o retorno(-5) da função.

```
nNeg := Neg(5);
```




Função NumUnits

Descrição:

A função **NumUnits** retorna o número de contratos/ações de um certo investimento.

Sintaxe:

NumUnits(Amnt : Integer, MinLot : Integer)

Parâmetros:

Amnt: Valor total de investimento, em reais, por trade.

MinLot: Tamanho mínimo desejado de lote por transação.

Retorno:

Integer

Exemplos:

No exemplo, assumindo que a ação selecionada está com preço de 65,00 por ação, se quiser investir 15500 em 100 ações, você poderia comprar 200 ações($\text{NumUnits}(15500, 100) = 200$).

```
numU := NumUnits(15500, 100);
```



Função PercentChange

Descrição:

A função **PercentChange** calcula a alteração percentual no preço do candle atual sobre determinado descolamento.

Sintaxe:

PercentChange(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série base de referência.

Período: Período anterior para a comparação com o dado da série atual.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "pc" o retorno da função PercentChange, considerando a série de fechamento(Série de dados) e 2 períodos para a comparação.

```
pc := PercentChange(Close, 2);
```



Função PercentR

Descrição:

A função **PercentR** retorna uma porcentagem de onde o preço atual está, relacionado com a faixa de negociação avaliada.

Sintaxe:

PercentR(Comprimento : Integer)

Parâmetros:

Comprimento: Comprimento considerado para o cálculo.

Retorno:

Float

Exemplos:

Será atribuído à variável "PercentR" o retorno da função PercentR, considerando 2 como comprimento.

```
vPercentR := PercentR(2);
```



Função Permutation

Descrição:

A função **Permutation** calcula o número de permutações para um determinado número de objetos.

Sintaxe:

Permutation(Numero : Integer, NumeroObjetos : Integer)

Parâmetros:

Numero: Determina o número de candles a serem analisados.

NumeroObjetos: Define o número de objetos dentro do intervalo de candles que podem ser selecionados.

Retorno:

Integer

Exemplos:

A variável "n" irá receber a combinação, considerando 4(Número de candle) e 2(Número de objetos).

```
n := Permutation(4,2);
```



Função Pos

Descrição:

A função **Pos** retorna o valor absoluto de um número ponto flutuante.

Sintaxe:

Pos(Valor : Float)

Parâmetros:

Valor: Valor ou variável para obter o módulo.

Retorno:

Float

Exemplos:

No exemplo, a variável "n" irá receber o módulo(4) do valor -4.

```
n := Pos(-4);
```



Função Power

Descrição:

A função **Power** tem como finalidade retornar a enésima potência de um valor.

Sintaxe:

Power(Base : Float, Expoente : Integer)

Parâmetros:

Base: Valor para a base da potênciação;

Expoente: Valor no qual a base será elevada.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "x" a potência da operação(2^3).

```
x := Power(2, 3);
```



Função Random

Descrição:

Com a função **Random**, o usuário poderá gerar um número aleatório(Inteiro), dentro de um intervalo(iniciado em zero) que possui como limite o valor determinado por parâmetro.

Sintaxe:

Random(Limite : Integer)

Parâmetros:

Limite: Recebe um valor ou uma variável para determinar o limite do intervalo, para geração do número.

Retorno:

Integer

Exemplos:

No exemplo a seguir, a função Random irá gerar números aleatórios entre 0 até 5, os quais serão atribuídos à variável "aux".

```
aux := Random(5);
```



Função RateOfChange

Descrição:

A função **RateOfChange** retorna a variação percentual de uma série de dados.

Sintaxe:

RateOfChange(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série base de referência.

Período: Índice do dado para a comparação com o último valor da série.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "rc" o retorno da função RateOfChange, considerando a série de máxima(Série de dados), e o candle anterior para calcular a variação.

```
rc := RateOfChange(High, 1);
```




Função Round

Descrição:

A função **Round** possui como finalidade efetuar o arredondamento de um número ponto flutuante.

Sintaxe:

Round(Valor : Float)

Parâmetros:

Valor: Número(variável ou constante) com casas decimais.

Retorno:

Integer

Exemplos:

Conforme no exemplo a seguir, será atribuído à variável "aux" o valor arredondado de 2.6, ao utilizar a função Round.

```
aux := Round(2.6);
```



Função Round2Fraction

Descrição:

A função **Round2Fraction** efetua o arredondamento de um número, para o valor mais próximo de um múltiplo do incremento mínimo de um ativo.

Sintaxe:

Round2Fraction(Valor : Float)

Parâmetros:

Valor: Valor desejado para o arredondamento de acordo com o incremento mínimo do ativo.

Retorno:

Float

Exemplos:

No exemplo abaixo(ativos Bovespa), ao aplicar a função para o valor 27.626, será atribuído o valor 27,63 para a variável "nRound2".

```
nRound2 := Round2Fraction(27.626);
```



Função Sign

Descrição:

A função **Sign** possui como definição retornar um número inteiro, baseado no sinal de um número.

Sintaxe:

Sign(Valor : Float)

Parâmetros:

Valor: Número para obter seu sinal.

Retorno:

Float: Determina qual linha será obtida:

Resultados possíveis:

-1 - **Número com sinal negativo.**

0 - **Sem sinal(zero).**

1 - **Número com sinal positivo.**

Exemplos:

No exemplo, será atribuído à variável "nSinal" o valor -1, tendo em vista o valor(-205) passado por parâmetro.

```
nSinal := Sign(-205);
```



Função Sin

Descrição:

A função **Sin** tem como objetivo retornar o Seno de um valor em radianos.

Sintaxe:

Sin(Valor : Float)

Parâmetros:

Valor: Valor para obter o Seno.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "sSeno" o Seno do valor 180 em radianos(-0,80).

```
nSeno := Sin(180);
```



Função Sine

Descrição:

A função **Sine** tem como objetivo retornar o Seno de um valor em graus.

Sintaxe:

Sine(Valor : Float)

Parâmetros:

Valor: Valor para obter o Seno.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "sSeno" o Seno do valor 45 em graus(0,71).

```
nSeno := Sine(45);
```



Função SlowD

Descrição:

A função **SlowD** retorna o valor do **SlowD** (Oscilador Estocástico), de acordo com o período desejado.

Sintaxe:

SlowD(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "sd" irá receber o valor do indicador SlowD, considerando 14 períodos para o cálculo.

```
sd := SlowD(14);
```



Função SlowK

Descrição:

A função **SlowK** retorna o valor do **SlowK** (Oscilador Estocástico), de acordo com o período desejado.

Sintaxe:

SlowK(Período : Integer)

Parâmetros:

Período: Período utilizado no momento do cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, a variável "sk" irá receber o valor do indicador SlowK, considerando 14 períodos para o cálculo.

```
sk := SlowK(14);
```



Função Sqrt

Descrição:

A função **Sqrt** tem como funcionalidade retornar ao usuário o valor da raiz quadrada de um número.

Sintaxe:

Sqrt(Valor : Float)

Parâmetros:

Valor: Valor para obter a raiz quadrada.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "nRaiz", a raiz quadrada do valor 25.

```
nRaiz := Sqrt(25);
```




Função Square

Descrição:

A função **Square** tem como funcionalidade retornar ao usuário o valor de um determinado número ao quadrado.

Sintaxe:

Square(Valor : Float)

Parâmetros:

Valor: Valor para elevar ao quadrado.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "n", o valor ao quadrado do número 5.

```
n := Square(5);
```



Função StdDevs

Descrição:

A função **StdDevs** retorna o desvio padrão de uma série de dados, em um determinado período.

Sintaxe:

StdDevs(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série utilizada para o cálculo.

Período: Período utilizado no momento do cálculo.

Retorno:

Float

Exemplos:

No exemplo, a variável "sd" irá receber o retorno da função StdDevs, considerando o fechamento(Close) e 20(Períodos) para o cálculo.

```
sd := StdDevs(Close, 20);
```



Função Summation

Descrição:

A função **Summation** efetua o somatório do valor do preço de um determinado número de barras.

Sintaxe:

Summation(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série utilizada para o cálculo.

Período: Período utilizado no momento do cálculo.

Retorno:

Float

Exemplos:

No exemplo, a variável "s" irá receber o retorno da função Summation, considerando o fechamento(Close) e 10(Períodos) para o cálculo.

```
s := Summation(Close, 10);
```



Função Tangent

Descrição:

A função **Tangent** tem como objetivo retornar ao usuário a Tangente de um valor em graus.

Sintaxe:

Tangent(Valor : Float)

Parâmetros:

Valor: Valor ou variável para obter a Tangente.

Retorno:

Float

Exemplos:

No exemplo a seguir, será atribuído à variável "nTan" a Tangente do valor 30 em graus(0,58).

```
nTan := Tangent(30);
```



Função TriAverage

Descrição:

A função **TriAverage** efetua a média triangular de uma série de dados, dentro de um determinado período.

Sintaxe:

TriAverage(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série utilizada para o cálculo.

Período: Período utilizado no momento do cálculo.

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o retorno da função TriAverage, considerando a máxima(Close) e 20(Períodos) para o cálculo.

```
aux := TriAverage(High, 20);
```



Função UlcerIndex

Descrição:

A função **UlcerIndex** mede o nível de estresse de acordo com as condições do mercado.

Sintaxe:

UlcerIndex(SerieDados : Serie, Período : Integer)

Parâmetros:

SerieDados: Série utilizada para o cálculo.

Período: Período utilizado no momento do cálculo.

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o retorno da função UlcerIndex, considerando a máxima(Close) e 3(Períodos) para o cálculo.

```
aux := UlcerIndex(High, 3);
```



Função UltimateOscillator

Descrição:

A função **UltimateOscillator** retorna o valor do Ultimate Oscillator desenvolvido por Larry Williams.

Sintaxe:

UltimateOscillator(PeriodoCurto : Integer, PeriodoMedio : Integer, PeriodoLongo : Integer)

Parâmetros:

PeriodoCurto: Período curto utilizado no momento do cálculo.

PeriodoMedio: Período médio para o cálculo.

PeriodoLongo: Período longo para o cálculo.

Retorno:

Float

Exemplos:

No exemplo, a variável "aux" irá receber o retorno da função UltimateOscillator, considerando 5(PeriodoCurto), 8(PeriodoMedio) e 12(PeriodoLongo) para o cálculo.

```
aux := UltimateOscillator(5, 8, 12);
```



Função Volatilidade

Descrição:

A função **Volatilidade** retorna o dado do indicador **Volatilidade**, segundo a periodicidade desejada.

Sintaxe:

Volatilidade(Período : Integer)

Parâmetros:

Período: Período para cálculo do indicador.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "vol" o retorno do indicador, considerando 10 períodos.

```
vol := Volatilidade(10);
```




Função VolumeOsc

Descrição:

A função **VolumeOsc** retorna a diferença entre a média aritmética rápida e a média aritmética lenta da série de **Volume** (financeiro).

Sintaxe:

VolumeOsc(PeriodoMediaRapida : Integer, PeriodoMediaLenta : Integer)

Parâmetros:

PeriodoMediaRapida: Período utilizado para a média aritmética rápida.

PeriodoMediaLenta: Período utilizado para a média aritmética lenta.

Retorno:

Float

Exemplos:

No exemplo, a variável "difVol" irá receber o valor do indicador VolumeOsc, considerando 9(Média Rápida) e 21(Média Lenta) períodos para o cálculo.

```
difVol := VolumeOsc(9, 21);
```



Função VolumeROC

Descrição:

A função **VolumeROC** retorna o VolumeROC baseado em um número de barras.

Sintaxe:

VolumeROC(Período : Integer)

Parâmetros:

Período: Período utilizado para a cálculo.

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "VoIR" o retorno da função VolumeROC, considerando 5(Períodos).

```
VoIR := VolumeROC(5);
```



Opções

Função Delta

Descrição:

A função **Delta** mede a variação do preço da opção com o preço da ação.

Sintaxe:

Delta(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

Parâmetros:

DaysLeft: Dias uteis até o vencimento da opção

StrikePr: Preço do exercício da opção

AssetPr: Preço da ação

Rate100: Taxa de juros em %

Volty100: Volatilidade em %

PutCall: Indica se é uma put ou uma call

optPut - **Opção de venda**

optCall - **Opção de compra**

Retorno:

Float

Exemplos:

No exemplo, a variável "opD" irá receber a variação, considerando 13 dias para o vencimento, 16.47 como Strike, 17.25 como preço da ação, 0.1 para taxa de juros e 0.25 para volatilidade, aplicados para uma Call.

```
opD := Delta(13, 16.47, 17.25, 0.1, 0.25, optCall);
```



Função Gamma

Descrição:

A função **Gamma** mede a variação do delta em relação ao preço da ação.

Sintaxe:

Gamma(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

Parâmetros:

DaysLeft: Dias uteis até o vencimento da opção

StrikePr: Preço do exercício da opção

AssetPr: Preço da ação

Rate100: Taxa de juros em %

Volty100: Volatilidade em %

PutCall: Indica se é uma put ou uma call

optPut - **Opção de venda**

optCall - **Opção de compra**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "opG", a variação, considerando 9 dias para o vencimento, 11.08 como Strike, 11.94 como preço da ação, 0.2 para taxa de juros e 0.3 para volatilidade, aplicados sobre uma Call.

```
opG := Gamma(9, 11.08, 11.94, 0.2, 0.3, optCall);
```



Função Rho

A função **Rho** retorna a variação da opção em relação à taxa de juros.

Sintaxe:

Rho(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

Parâmetros:

DaysLeft: Dias uteis até o vencimento da opção

StrikePr: Preço do exercício da opção

AssetPr: Preço da ação

Rate100: Taxa de juros em %

Volty100: Volatilidade em %

PutCall: Indica se é uma put ou uma call

optPut - **Opção de venda**

optCall - **Opção de compra**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "opR", a variação, considerando 9 dias para o vencimento, 11.00 como Strike, 12.92 como preço da ação, 0.2 para taxa de juros e 0.3 para volatilidade, aplicados para uma Call.

```
opR := Rho(9, 11.00, 12.92, 0.2, 0.3, optCall);
```



Função Theta

Descrição:

A função **Theta** retorna a variação do preço da opção com o tempo.

Sintaxe:

Theta(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

Parâmetros:

DaysLeft: Dias uteis até o vencimento da opção

StrikePr: Preço do exercício da opção

AssetPr: Preço da ação

Rate100: Taxa de juros em %

Volty100: Volatilidade em %

PutCall: Indica se é uma put ou uma call

optPut - **Opção de venda**

optCall - **Opção de compra**

Retorno:

Float

Exemplos:

No exemplo, será atribuído à variável "opT", a variação, considerando 9 dias para o vencimento, 17.22 como Strike, 11.94 como preço da ação, 0.1 para taxa de juros e 0.32 para volatilidade, aplicados sobre uma Put.

```
opT := Theta(9, 17.22, 17.94, 0.1, 0.32, optPut);
```



Função Vega

Descrição:

A função **Vega** retorna a variação da opção em relação à volatilidade.

Sintaxe:

Vega(DaysLeft : Integer, StrikePr : Float, AssetPr : Float, Rate100 : Float, Volty100 : Float, PutCall : Integer)

Parâmetros:

DaysLeft: Dias uteis até o vencimento da opção

StrikePr: Preço do exercício da opção

AssetPr: Preço da ação

Rate100: Taxa de juros em %

Volty100: Volatilidade em %

PutCall: Indica se é uma put ou uma call

optPut - **Opção de venda**

optCall - **Opção de compra**

Retorno:

Float

Exemplos:

No exemplo, será considerado 28 dias para o vencimento, 11.08 como Strike, 11.94 como preço da ação, 0.2 para taxa de juros e 0.36 para volatilidade, aplicados para uma Put.

```
opV := Vega(28, 11.08, 11.94, 0.2, 0.36, optPut);
```



Screening

Função Select

Descrição:

A função **Select**, apesar de estar disponível na linguagem, não possui recursos para utilização do usuário, sua implementação é específica para uso do sistema, a fim de habilitação do Screening.

Sintaxe:

Select

Parâmetros:

Sem parâmetros.

Retorno:

Void: Sem retorno.

Nelogica[®] 