VestiCode Handboek

18-12-2002

Table of Contents

1 De VestiCode programmeertaal

In Vestics is het mogelijk om als gebruiker eigen indicatoren te ontwikkelen en die vervolgens te gebruiken in grafieken en andere onderdelen.

In feite is het zo dat alle bij Vestics meegeleverde indicatoren op dezelfde manier zijn ontwikkeld en desgewenst door de gebruiker aangepast kunnen worden of als basis gebruikt worden voor nieuwe indicatoren.

De formules voor deze indicatoren dienen geprogrammeerd te worden in een speciale programmeertaal, die VestiCode heet.

De taal VestiCode is specifiek voor Vestics, maar lijkt sterk op meer algemene programmeertalen zoals Pascal. In feite is VestiCode afgeleid van Pascal, maar gezien het speciale karakter van VestiCode zijn allerlei minder toepasselijke functies van Pascal weggelaten uit VestiCode, waardoor een eenvoudige variant van Pascal is ontstaan, die beter aansluit bij de Vestics omgeving en ook gemakkelijker te leren is.

Naast de eigen taal VestiCode kunnen in Vestics ook indicatoren gebruikt worden die ontwikkeld zijn in de taal EasyLanguage voor het programma TradeStation van Omega Research . VestiCode en EasyLanguage hebben zeer veel overeenkomsten omdat ze beide van Pascal zijn afgeleid. VestiCode lijkt daarbij echter veel meer op het oorspronkelijke Pascal dan EasyLanguage. In de verdere beschrijving zullen we ons uitsluitend bezighouden met VestiCode.

In de nu volgende hoofdstukken wordt de programmeertaal VestiCode in detail besproken. Door rechtsboven op de verwijzing Volgende te klikken, kunt u steeds doorklikken naar de volgende bladzijde van de tekst

1.1 De basis bouwstenen

In dit hoofdstuk kijken we naar de opbouw van de programmeertaal VestiCode.

Dit hoofdstuk is de basis voor de verdere beschrijving van VestiCode en de toepassingen er van bij het ontwikkelen van indicatoren. Het is daarom belangrijk dat u deze onderwerpen doorleest voordat u zelf aan de slag gaat met indicatoren.

1.1.1 Numerieke- en tekstinformatie

In VestiCode wordt gebruik gemaakt van 2 soorten gegevens, namelijk numerieke waarden en teksten. Met numerieke waarden kan gerekend worden terwijl dat met tekst informatie uiteraard niet kan. De enige bewerking die men op tekst informatie kan loslaten is het aan elkaar plakken van 2 of meer stukken tekst tot een langere tekst.

Numerieke gegevens
Een numerieke waarde kan alleen uit cijfers bestaan. ABC en 1A2B zijn geen geldige waarden, maar 123 en 896543 zijn dat wel. Behalve de cijfers kan nog gebruik gemaakt worden van een minus teken (-) om aan te geven dat de waarde negatief is en kan gebruik gemaakt worden van een decimale punt (.) om decimalen aan te geven. Dus 123, -123, 1.23, en –12.3 zijn allemaal voorbeelden van geldige waarden.

Het minus teken moet altijd vóór het getal staan en er mag uiteraard maar 1 decimale punt in een getal voorkomen. Daarom zijn 123- en 1.2.3 beide ongeldige waarden. Een komma als decimale punt, of eventueel als duizendtallen teken, is niet toegestaan. Dus 12,3 is niet geldig en 123,456.78 is ook niet geldig.

Tekst gegevens
Omdat tekstgegevens uit meerdere woorden zouden kunnen bestaan, worden tekstgegevens altijd bij elkaar gehouden door middel van apostroffen. Enkele voorbeelden van geldige teksten zijn 'ABC' en 'Dit is een tekst'.

Naast hoofdletters en kleine letters mogen in teksten zowat alle tekens voorkomen die op het toetsenbord van de computer staan, dus ook leestekens en cijfers.

Omdat de apostrof gebruikt worden om tekst bij elkaar te houden, geeft dat een probleem als er een apostrof midden in de tekst staat, zoals in 'jan's fiets'. In dit geval zou de computer denken dat de tekst alleen bestaat uit het stuk 'jan', en de rest zou er buiten vallen. Daarom is afgesproken dat een apostrof binnen een tekst altijd verdubbeld wordt. We schrijven dan 'jan''s fiets' met twee apostroffen achter elkaar. Wordt een dergelijke tekst naar de printer gestuurd of op het scherm weergegeven, dan wordt automatisch van de 2 apostroffen één enkele gemaakt.

1.1.2 Variabele informatie

Tot nu hebben we gekeken naar constante waarden en teksten, maar in een echt programma zal men veel vaker gebruik maken van variabele waarden die bijvoorbeeld het resultaat van een berekening zijn.

Net als bij de constante informatie komen we ook bij de variabele informatie weer de vertrouwde types tegen, namelijk numerieke waarden en teksten.

Bij het werken met variabele informatie is het handig als zo'n gegeven een naam heeft, waarmee u naar die betreffende waarde kunt refereren.

Bij het programmeren noemen we dat het 'definiëren' van een 'variabele'.
Dat definiëren gaat als volgt:

```
value  Bedrag;
string  Fonds;
```

In de eerste instructie wordt met het commando **`value`** een variabele met de naam `Bedrag` gedefinieerd die gebruikt kan worden om er een waarde in op te slaan.

Bij de tweede instructie **`string`** wordt een tekstvariabele met de naam `Fonds` gedefinieerd.

Het zal duidelijk zijn dat het woord **`value`** aangeeft dat de variabele een waarde bevat. Het woord **`string`** is misschien wat minder direct te relateren naar een tekst, maar het begrip string wordt in vrijwel alle programmeertalen gebruikt om aan te geven dat het om een reeks (string) van tekens gaat, oftewel een tekst. In VestiCode hebben we ons aangepast aan die algemeen gebruikelijke naamgeving. U went er snel aan.

Overigens is ook het kiezen van de juiste namen van de variabelen uitermate belangrijk. Een variabele waarin u bijv. de kosten berekend kunt u beter 'Kosten' noemen dan bijvoorbeeld X1. Helemaal een ramp wordt het uiteraard als de namen misleidend zijn, bijv. een variabele die de kosten bevat mag natuurlijk nooit 'Winst' of 'Aantal' genoemd worden! Kort en goed... kies een korte naam die duidelijk aangeeft wat de functie is.

1.1.3 De opbouw van instructies

In VestiCode worden alle instructies afgesloten met het sluitteken punt-comma (;).

Daarentegen heeft het einde van een tekstregel geen enkele betekenis en kunnen lange instructies dan ook over meerdere tekstregels verspreid worden.

De instructie...

```
value  Bedrag;
```

... had net zo goed over 3 of meer regels verspreid kunnen worden...

```
value
Bedrag
;
```

Uiteraard is het voor de leesbaarheid beter om de indeling van de instructies aan te passen aan de voor ons mensen gebruikelijke regel-indeling.

1.1.4 Sleutelwoorden

Misschien is u ook opgevallen dat de woorden **value** en **string** dik gedrukt zijn in de diverse voorbeelden van VestiCode instructies.

Dat doen we om aan te geven dat dit vaste sleutelwoorden zijn die exact zo gebruikt moeten worden.

Dit in tegenstelling tot de woorden Bedrag en Fonds die we zelf gekozen hebben als naam voor de variabele gegevens. Voor die namen hadden we net zo goed iets anders mogen kiezen.

Binnen VestiCode vervullen deze sleutelwoorden letterlijk een sleutelrol, omdat Vestics aan deze woorden kan zien welke instructies bedoeld worden.

1.1.5 Commentaar

Een van de eerste dingen die een programmeur leert, is dat hij moet zorgen dat zijn programma's ook voor anderen te begrijpen zijn.

Daarvoor wordt gebruik gemaakt van stukjes tekst (commentaar genoemd) die tussen of achter de instructies worden geplaatst met wat uitleg.

Om te zorgen dat de computer herkent dat een dergelijk stukje tekst genegeerd moet worden, wordt dit commentaar omsloten met accolades { }.

Enkele voorbeelden van commentaar...

```
value  Bedrag;  { variabele  voor  het  bedrag  }
string  Fonds;  { variabele  voor  de  fondsnaam  }
```

Ook al schrijft u een stuk VestiCode uitsluitend voor eigen gebruik, dan zult u merken dat het toch zinvol is om het programma met behulp van commentaar aan u zelf uit te leggen. Als u over 1 of 2 jaar iets opzoekt in het programma dan bent u al snel dankbaar voor wat uitleg. Wilt u dat anderen ook gebruik kunnen maken van uw werkstuk, dan is het helemaal belangrijk dat u wat uitleg

opneemt bij de instructies.

1.1.6 Het gebruik van hoofdletters

In VestiCode is er geen verschil tussen kleine en grote letters.

Dus mag het sleutelwoord **`string`** ook geschreven worden als **`String`** of als **`STRING`** of zelfs als **`StRiNg`** .

Hetzelfde geldt voor eigen gekozen namen.

Toch is het zinvol om zelf enige lijn te brengen in het gebruik van hoofdletters en bij Vestico hanteren we intern de volgende richtlijnen...

- Alle sleutelwoorden worden met kleine letters geschreven, dus **string** en **`value`**
- Wij gebruiken voor functies en variabelen altijd engelse namen. Hiermee proberen we een beetje aan te sluiten bij de literatuur over indicatoren die vrijwel uitsluitend in het Engels is en bij wat in EasyLanguage gebruikelijk is.
- Namen van functies en variabelen worden met 1 of meerdere hoofdletters geschreven, waarbij bij samengestelde namen, zoals TotalDays of CurrentPosition, elk onderdeel van de samengestelde naam met een hoofdletter begint.

Wij adviseren u om deze richtlijnen ook over te nemen voor hun eigen indicatoren, waarbij het gebruik van Engelse of Nederlandse namen uiteraard iets van persoonlijke voorkeur heeft.

1.1.7 Naamgeving variabelen

Uiteraard is het verstandig om voor alle variabelen in een programma een betekenisvolle naam te kiezen. Later zullen we zien dat we ook namen moeten kiezen voor functies en daar geldt hetzelfde.

Zo hebben we in ons vorige voorbeeld gebruik gemaakt van de variabelenamen Bedrag en Fonds. Dat is veel duidelijker dan dat we bijvoorbeeld namen als X en Y of v1 en v2 hadden gebruikt.

De een vindt het prettiger om met Engelse namen te werken, omdat de vaste sleutelwoorden en functienamen van VestiCode ook in het Engels zijn, terwijl de ander liever Nederlandse namen gebruikt. Die keuze laten we graag aan u over.

Pas echter op.. als u namen gaat kiezen voor variabelen, zeker in het Engels, maar soms ook in het Nederlands, dan is het mogelijk dat u per ongeluk een naam kiest die ook gebruikt wordt door een bestaande functie in VestiCode. Of misschien wordt er in de toekomst een nieuwe functie aan VestiCode toe gevoegd, die een naam heeft die u gebruikt voor uw variabelen. Denk maar aan namen zoals Margin, Stop en Trades; voordat u het weet heeft u een conflict.

Om dit soort conflicten te vermijden gebruiken we bij Vestico een standaard naamgeving die er als volgt uit ziet...

- Alle namen van variabelen laten we beginnen met de letter x, gevolgd door een betekenisvolle (Engelse) naam
- Alle namen van eigen functies laten we beginnen met de letter z, gevolgd door een betekenisvolle naam
- Alle namen van standaard VestiCode functies beginnen met de letter v, gevolgd door een betekenisvolle Engelse naam

Alle namen die niet beginnen v, x of z zijn daardoor beschikbaar voor sleutelwoorden en voor

EasyLanguage functies. Uiteraard hebben wij geen controle over de namen die Omega Research kiest voor functies en door onze eigen functienamen vooraf te laten gaan door een letter v voorkomen we conflicten nu en in de toekomst.

De letters v, x en z zijn o.a. gekozen omdat a) er weinig woorden zijn die met die letters beginnen, en b) deze letters een laag 'profiel' hebben, zodat ze niet erg storen vóór de eerste hoofdletter van de echte naam. Dit ziet er als volgt uit...

```
value  xBedrag;
string  xFonds;
```

Wij adviseren u om deze standaard van ons over te nemen en dus al uw eigen variabelenamen te laten beginnen met een x en al uw eigen functienamen te laten beginnen met een z.

1.1.8 Het veranderen van variabelen

Men kan de inhoud van een variabele wijzigen door er een (nieuwe) waarde aan toe te kennen.

Daarbij moet de toegekende waarde van hetzelfde type zijn als de variabele zelf. Men kan dus geen tekst waarde toekennen aan een numerieke variabele, en omgekeerd kan men geen numerieke waarde toekennen aan een variabele van het tekst type.

Het toekennen van een waarde gaat via de speciale code := (wordt gelijk aan).

Enkele voorbeelden...

```
xBedrag  := 123.95;
xFonds  := 'Philips' ;
xBedrag  := xTotaal;
```

De eerste instructie kan gelezen worden als: "xBedrag wordt gelijk aan 123.95". De derde instructie is alleen geldig als ergens eerder in het programma de variabele xTotaal is gedefinieerd als zijnde van het type **value**. Zou xTotaal zijn gedefinieerd als **string** dan zou de derde instructie niet geldig zijn omdat men geen tekstwaarde mag toekennen aan een numerieke variabele.

1.1.9 De wiskundige operatoren

Voor het doen van berekeningen staan in VestiCode min of alle rekenkundige bewerkingen ter beschikking.

De volgende bewerkingen, ook wel *operatoren* genoemd, staan ter beschikking:

teken	betekenis	voorbeeld	resultaat
^	machtsverheffen	5^2	25
*	vermenigvuldigen	5*2	10
/	delen	5/2	2.5
\	delen, resultaat afkappen op heel getal	5\2	2
%	modulus (resultaat is rest na deling)	5%2	1
+	optellen	5+2	7
-	aftrekken	5-2	3

Voor het vermenigvuldigen wordt niet het maal-teken gebruikt, maar in plaats daarvan wordt het

sterretje gebruikt.

Er is geen speciale operator voor worteltrekken. Voor worteltrekken moet gebruik gemaakt van het feit dat machtsverheffen de tegengestelde functie van worteltrekken is. Zo kan men dus door een getal te verheffen tot de macht 0.5 om de 2-de machts wortel uit dat getal trekken. Deze omkering is vergelijkbaar met de omkering die men kan toepassen op vermenigvuldigen en delen. Immers, `2*5` is gelijk aan `2/(1/5)` oftewel `2/0.2`.

1.1.10 Meneer van Dalen

Net als in de gewone wiskunde geldt ook in de VestiCode berekeningen dat vermenigvuldigen voor optellen gaat.

Dus de formule `3+2*5` geeft 13 en niet 25, omdat eerst `2*5` wordt uitgerekend (geeft 10) en vervolgens `3+10` geeft 13.

Wil men persé eerst 3+2 doen en das pas vermenigvuldigen met 5, dan kan men de volgorde van berekenen beïnvloeden door gebruik te maken van haakjes om de gewenste volgorde aan te geven. De formule `(3+2)*5` levert 25 op.

1.1.11 Delen door nul

Als we straks aan de slag gaan met berekeningen, dienen we steeds in het achterhoofd te houden dat in de wiskunde delen door 0 verboden is.

Deze zelfde beperking geldt ook in VestiCode en berekeningen zoals `5/0` leveren dan ook een ongeldig resultaat op.

Dergelijke ongeldige of niet beschikbare informatie is dus niet uit te drukken in een getal, en daarom is er een speciale waarde _NA (Not Available) voor dit doel gedefiniëerd.

1.1.12 De waarde _NA

Zoals we gezien hebben ontstaat bij delen door 0 de waarde _NA om aan te geven dat er geen waarde beschikbaar is als resultaat van deze berekening.

Een ander voorbeeld is, wanneer we refereren naar de koers van een bepaalde dag (hoe dat gaat zien we later wel), en er is op die dag geen koers beschikbaar. Ook dan krijgen we de waarde _NA.

Als de waarde _NA eenmaal is opgedoken, dan 'besmet' deze alle andere onderdelen van een formule.

Voorbeeld:
Iets vermenigvuldigen met _NA geeft de waarde _NA op.
De waarde _NA bij iets optellen levert de waarde _NA op.
Testen of een waarde groter of kleiner dan _NA is, levert _NA op.
enz.

De enige uitzondering voor deze regel is het vergelijken op gelijk of ongelijk. In een dergelijke vergelijking mag de waarde _NA gebruikt worden zonder dat het resultaat van de vergelijking automatisch _NA wordt. In plaats daarvan krijgt men als resultaat de waarde 0 (onwaar) of 1 (waar).

Door deze uitzondering is het mogelijk om een waarde _NA alsnog om te zetten naar een geldige waarde, zodat de verdere berekeningen gewoon doorgang kan vinden.

```
if xBedrag=_NA  then xBedrag  := 0;
xBTW  := 0.19*xBedrag;
```

Een andere toepassing is het overslaan van een berekening in geval van _NA waarden.

```
if Close[ 200 ]<>_NA  then  begin
    ...
    end ;
```

1.1.13 Grootste en kleinste waarde

In VestiCode is er een speciale bewerking (*operator*) die gebruikt kan worden om de grootste of kleinste van 2 waarden te selecteren.

De operator >> resulteert in de grootste van 2 waarden.
De operator << resulteert in de kleinste van 2 waarden.

Voorbeeld:

```
Waarde1  := Waarde2  >> Waarde3;
Waarde1  := Waarde2  << Waarde3;
Waarde1  := Waarde2  >> Waarde3  >> Waarde4  >> Waarde5;
```

De operatoren grootste en kleinste hebben onderling dezelfde prioriteit en worden dus van links naar rechts afgehandeld als ze door elkaar heen gebruikt worden:

```
Waarde1  := Waarde2  >> Waarde3  << Waarde4;
Waarde1  := (Waarde2  >> Waarde3)  << Waarde4;
```

In bovenstaand voorbeeld worden eerst de Waarde2 en Waarde3 vergeleken en de grootste van die twee wordt vergeleken met Waarde4 om daarvan de kleinste te selecteren. In de tweede regel is die volgorde expliciet aangegeven door middel van haakjes. De beide regels leveren dus hetzelfde resultaat op.

Indien de operatoren grootste en kleinste gecombineerd worden met wiskundige operatoren zoals + en -, dan hebben grootste en kleinste de laagste prioriteit van alle. Ze komen dus na het rijtje van meneer van Dalen.

```
Waarde1  := Waarde2+Waarde3   << Waarde4-Waarde5;
Waarde1  := (Waarde2+Waarde3)   << (Waarde4-Waarde5);
```

De haakjes in de tweede regel geven aan hoe in de eerste regel de volgorde van de bewerkingen is. De beide regels leveren dus hetzelfde resultaat op.

Opmerking:
In veel programmeertalen is er een functie Max die gebruikt kan worden om de grootste van twee waardes te selecteren. In VestiCode is deze functie beschikbaar onder de naam MaxList , waarmee de grootste van tot 25 waardes geselecteerd kan worden. Naast de MaxList is er ook nog de MaxList2 , die de 2de hoogste waarde selecteert, en de NthMaxList die de Nde grootste waarde selecteert.
Op dezelfde manier zijn de functies MinList , MinList2 en NthMinList te gebruiken om de kleinste waarde uit een lijst van waardes te selecteren.

1.1.14 Conditionele verwerking met IF

Om uit twee mogelijke alternatieve acties te kunnen kiezen staat in VestiCode de if-instructie ter beschikking.

Deze instructie heeft de volgende algemene opbouw...

```
if conditie then actie1 else actie2;
```

Als er geen actie2 is, mag het sleutelwoord **else** ook weggelaten worden, en dan ontstaat de eenvoudigere vorm...

```
if conditie then actie;
```

1.1.15 De vergelijkings operatoren

De conditie in IF-instructies en dergelijke is altijd een vorm van vergelijking die resulteert in waar of onwaar. Bij deze vergelijking kunnen we gebruik maken van de volgende vergelijkings operatoren...

teken	betekenis	voorbeeld	resultaat
=	is gelijk	5=2	onwaar
<	kleiner dan	5<2	onwaar
<=	kleiner of gelijk	5<=2	onwaar
>	groter dan	5>2	waar
>=	groter of gelijk	5>=2	waar
<>	ongelijk	5<>2	waar

1.1.16 Het gebruik van waar en onwaar

In sommige programmeertalen bestaan er naast de numerieke en tekst variabelen nog een derde type variabelen, namelijk de waar/onwaar variabelen, ook wel *booleans* genoemd.

In VestiCode bestaan er geen speciale booleans, en wordt gebruik gemaakt van de waarden 0 en 1 voor **onwaar** en **waar** .

Of beter gezegd... 0 is onwaar en elke andere waarde geldt als waar. Dus elke numerieke waarde kan automatisch gebruikt worden als de conditie van een IF-instructie, maar dat wordt in de praktijk zelden gedaan omdat het nogal verwarrend is.

Omgekeerd kan een conditionele expressie dus ook in een berekening worden gebruikt, waarbij de waarde 0 of 1 op de plaats van de conditie komt. Ook deze constructie, hoewel toegestaan, kan beter vermeden worden omdat ze niet erg duidelijk is.

1.1.17 Het combineren van condities

Door 2 of meer condities onderling te combineren ontstaan samengestelde condities. Voor dit combineren wordt gebruik gemaakt van de zogenaamde 'logische' operatoren...

Bij de EN-relatie moeten beide condities waar zijn, oftewel in gewone spreektaal 'alleen als A waar is en B waar is'. De EN-relatie wordt aangegeven met het woordje 'and' of met het teken '&'.

Bij de OF-relatie moet één van de beide condities waar zijn, oftewel 'Als A of B waar is'. De OF-relatie wordt aangegeven met het woordje 'or' of met het teken '|'.

teken	betekenis	voorbeeld	resultaat
and	en	5>2 and 7>10	onwaar
&	en	5>2 & 7<10	waar
or	of	5>2 or 7>10	waar
|	of	5>2 | 7<10	waar

1.1.18 Voorbeeld gebruik if-instructie

Stel we willen een gemiddelde uitrekenen en hebben daartoe eerst alle waarden gesommeerd in een variabele Totaal en het aantal waarden geteld in de variabele Aantal.

Het uitrekenen van de gemiddelde waarde gaat dan als volgt:

```
xGemiddelde  := xTotaal/xAantal;
```

Als het echter kan voorkomen dat er 0 waarden zijn, dan moet voorkomen worden dat we delen door 0, en wel als volgt:

```
if xAantal= 0 then xGemiddelde  := 0
else xGemiddelde  := xTotaal/xAantal;
```

1.1.19 BEGIN - END blokken

Als we als actie bij een if-instructie meerdere instructies willen gebruiken, dan kan dat niet zo maar.

Onderstaande constructie werkt dus niet:

```
if xBedrag= 0 then xBTW := 0; xNetto := 0; xTotaal := 0;
```

Door de ; achter xBTW := 0 wordt tevens de if-instructie beëindigd.

Met andere woorden, de resterende twee instructies op de regel worden altijd uitgevoerd, ongeacht of xBedrag 0 is of niet. Het weglaten van de ; is ook geen oplossing want dan worden de instructies niet meer correct herkent door Vestics.

Een probleem dus? Nee hoor, met behulp van de woorden **begin** en **end** kunnen een aantal instructies samengevoegd worden tot één enkele instructie, ook wel een begin-end blok genoemd. Dat ziet er als volgt uit:

```
if xBedrag= 0 then begin
    xBTW  := 0;
    xNetto  := 0;
    xTotaal  := 0;
    end;
```

1.1.20 Het inspringen van instructies

Als we een aantal instructies willen uitvoeren bij een if-instructie, dan gebruiken we begin en end om de instructies te groepen in een begin-end blok.

```
if xBedrag= 0 then begin
    xBTW  := 0;
    xNetto  := 0;
    xTotaal  := 0;
    end;
```

Het zal u misschien opgevallen zijn dat we in bovenstaand voorbeeld de laatste 4 regels wat hebben laten inspringen ten opzichte van de eerste regel.

Dat is gedaan om duidelijk te laten uitkomen dat deze 4 regels afhankelijk zijn van de eerste regel. Ze hangen er als het ware aan vast.

In de praktijk komt het vaak voor binnen de actie weer een nieuwe if-constructie wordt gebruikt, eventueel ook weer met een begin-end blok. Door steeds verder in te springen houdt men het overzicht van welke instructie bij welk blok hoort.

Wie dit inspringen niet strikt toepast zal al snel vastlopen en z'n eigen programma niet meer begrijpen doordat de bij elkaar behorende begin en end instructies niet meer te volgen zijn.

1.1.21 De WHILE instructie

Met behulp van de while instructie kan men een groep instructies zo lang herhalen als een bepaalde conditie waar is.

```
xTeller  := 5;
while  xTeller> 0 do begin
    { doe iets }
    xTeller  := xTeller- 1;
    end;
```

In bovenstaand voorbeeld wordt de variabele xTeller op 5 gezet, en vervolgens steeds met 1 verminderd. De while instructie zorgt er voor dat de begin-end blok steeds weer opnieuw wordt uitgevoerd zolang de xTeller groter dan 0 is. In dit geval zullen de instructies in het begin-end blok dus precies 5 keer uitgevoerd worden.

1.1.22 De REPEAT - UNTIL instructie

Een variant op de while instructie is de repeat-until instructie.

```
xTeller  := 5;
do begin
    { doe iets }
    xTeller  := xTeller- 1;
    end until  xTeller  = 0;
```

Waar de while instructie door gaat omdat de conditie waar is zal de repeat-until instructie doorgaan totdat de conditie waar is. Zodra de conditie waar is wordt het herhalen afgebroken.

Omdat de while en de repeat-until vrijwel identiek zijn zou men in de praktijk kunnen volstaan met

alleen maar de while-instructie.

1.1.23 De FOR instructie

Een derde variant herhaal instructie is de for-instructie.

Deze instructie maakt het gemakkelijker om een teller bij te houden voor het aantal keer dat een groepje instructies herhaalt moet worden:

```
for xTeller := 1 to 5 do begin
    { doe iets }
    end;
```

In bovenstaand voorbeeld zal de variabele xTeller achtereenvolgens de waarden 1 tot en met 5 krijgen en voor elke waarde worden de instructies van de begin-end blok uitgevoerd.

Het is ook mogelijk om een teller te laten aftellen...

```
for xTeller := 5 downto 1 do begin
    { doe iets }
    end;
```

1.1.24 Skipwoorden

Een relict uit EasyLanguage is de mogelijkheid om op alle plaatsen in de instructies een aantal woorden te gebruiken die bij de interpretaties geheel overgeslagen worden.

De volgende woorden worden onder alle omstandigheden overgeslagen...
A, AN, AT, BASED, BY, DOES, FROM, IS, OF, ON, PLACE, THAN, THE en WAS.

Vestico adviseerd u dringend om geen gebruik te maken van deze mogelijkheid omdat onze ervaring is dat het alleen maar verwarrend werkt en niet bijdraagt tot de leesbaarheid van uw programma's.

Door in uw programma's altijd de geadviseerde naamgevingsstandaard toe te passen kunt u nooit conflicten krijgen tussen uw eigen namen en deze skipwoorden.

1.2 Werken met functies

Bij het programmeren van nieuwe indicatoren in VestiCode, is het mogelijk om gebruik te maken van standaard functies die allerlei veel voorkomende bewerkingen uitvoeren.

Als voorbeeld kan men daarbij denken aan het berekenen van het gemiddelde van de slotkoersen van de laatste 10 dagen. Men kan uiteraard die 10 slotkoersen bij elkaar optellen en vervolgens delen door 10, maar het is veel eenvoudiger om de functie Average aan te roepen met als argumenten de slotkoers en het aantal dagen. De functie Average rekent dan automatisch het gemiddelde uit.

In VestiCode zijn er op die manier tientallen functies die kant-en-klaar te gebruiken zijn en die allerlei handige calculaties e.d. uitvoeren zoals het berekenen van de standaard deviatie, een lineaire regressie, enz.

In Vestics is er voor gekozen om voor deze functies Engelse namen te gebruiken die een goede weergave zijn van de uitgevoerde berekening.

Verder hebben we als standaard besloten om alle specifieke VestiCode functies, die niet bekend zijn in EasyLanguage, een naam te geven die begint met een kleine letter v. Hierdoor weet u dat het om een specifiek VestiCode functie gaat en bovendien vermijden we eventuele conflicten met toekomstige uitbreidingen in EasyLanguage.

Dus als een functie de naam Average heeft, dan weet dat dit een functie is die ook in EasyLanguage bestaat.
Een functie met de naam vLinReg is door de beginletter v te herkennen als een VestiCode functie.

Let op: alle EasyLanguage functies zijn in VestiCode op exact dezelfde manier geïmplementeerd als in EasyLanguage, zodat u probleemloss de formules van EasyLanguage indicatoren uit tijdschriften e.d. kunt intikken in VestiCode.

1.2.1 Het schrijven van eigen functies

U als gebruiker kunt zelf uw eigen functies maken, die vervolgens in allerlei indicatoren en dergelijke gebruikt kunnen worden. Op die manier hoeft een bepaalde berekening slechts eenmalig bedacht en gecontroleerd te worden en kan deze daarna naar hartelust gebruikt worden.

De naam van een dergelijke functie mag u zelf bedenken. U mag zelf bepalen of u zich wil aansluiten bij de Vestics conventie van Engelse functienamen, of dat u de voorkeur geeft aan meer vertrouwde Nederlandse namen.

Als voor een functie een naam gekozen wordt die gelijk is aan de naam van een standaard Vestics functie, dan zal vanaf dat moment de door u gemaakte functie prevaleren boven de oorspronkelijke Vestics functie. Op die manier kunt u dus een aangepaste versie maken van een bestaande functie die dan vervolgens overal gebruikt zal worden waar vroeger de bestaande functie werd gebruikt.

Pas op: het omgekeerde kan natuurlijk ook gebeuren... u maakt een nieuwe functie die u Average noemt, en zonder dat u dat echt wilde wordt deze nieuwe functie door alle bestaande indicatoren gebruikt omdat u per ongeluk dezelfde naam heeft gekozen als die van de bestaande Average functie.

Door onze aanbeveling op het gebied van functienamen over te nemen, en al uw eigen functienamen te laten beginnen met een letter z voorkomt u veel problemen in de toekomst. U krijgt dan namen als zAverage en zRSI en die kunnen nooit conflicteren met namen van VestiCode functies (vAverage en vRSI) of EasyLanguage functies (Average en RSI).

1.2.2 De opzet van een functie

Als we een eigen functie maken, dan beginnen we als eerste met een instructie die aangeeft welk soort functie het betreft, hoe de functie heet, en wat de argumenten zijn.

```
value function zBerekenBTW ( value xBedrag ) begin
    { doe iets }
    end;
```

In bovenstaand stukje programma wordt een functie begonnen met als naam zBerekenBTW. Door het woord value voor de functie geeft u dat deze functie resulteert in een numerieke waarde.

Tussen de haken, achter de naam van de functie, worden de eventuele argumenten voor de functie een voor een opgesomd, en ook hier wordt voor elk argument opgegeven van welk type (value of string) het betreffende argument is.

De instructies die de werkelijke functie vormen, worden ingesloten in een begin-end blok.

1.2.3 Het resultaat van een functie

Nadat uitgerekend is welke waarde de functie zou moeten opleveren, moet deze waarde vervolgens op de een of andere manier als de eindwaarde worden vastgelegd.

Dat gebeurd door die waarde toe te kennen aan de functienaam.

```
value function zBerekenBTW (value xBedrag) begin
    value xResultaat;
    xResultaat := 0.19 * xBedrag;
    zBerekenBTW := xResultaat;
    end;
```

In bovenstaand voorbeeld wordt eerst het resultaat berekend en vervolgens wordt dat resultaat toegewezen aan de functienaam.

Als het een simpele berekening betreft, zou men deze tussenstap kunnen weglaten en het resultaat van de berekening in één keer kunnen toewijzen aan de functienaam.

```
value function zBerekenBTW (value xBedrag) begin
    zBerekenBTW := 0.19 * xBedrag;
    end;
```

Men zou de drie bovenstaande instructies zelfs op 1 regel kunnen plaatsen, zodat men een BTW-berekeningsfunctie heeft die uit slechts 1 programmaregel bestaat.

1.2.4 Het gebruiken van een functie

Als een functie eenmaal bekend is in Vestics, dan kan men deze functie overal gebruiken.

Vestics herkent automatisch de naam van de functie en zoekt zelf uit welk type functie het betreft en welke argumenten door de functie gebruikt worden.

Het 'aanroepen' van een functie gebeurd door middel van de functienaam, eventueel gevolgd door de argumenten tussen haakjes.

```
xBTW := zBerekenBTW(xPrijs);
xTotaalBedrag := xPrijs + zBerekenBTW(xPrijs);
```

In het eerste voorbeeld zien we hoe de functie zBerekenBTW gebruikt wordt om de BTW te berekenen van een bepaalde waarde die in de variabele Prijs zit.

Zoals te zien is, staat de naam van het meegegeven argument (xPrijs) geheel los van de naam die binnen de functie voor dat argument gebruikt wordt (xBedrag). Wat er namelijk gebeurd is dat de inhoud van de variabele xPrijs wordt overgenomen in de variabele xBedrag van de functie.

Het is dus ook mogelijk om als argument een constante of zelfs een expressie te gebruiken...

```
xBTW := zBerekenBTW( 100 );
xBTW := zBerekenBTW( 100+50.95+3*99.50 );
```

De meegegeven waarde of expressie wordt gewoon toegekend aan de variabele xBedrag die binnen de functie wordt gebruikt als basis voor de BTW berekening.

1.2.5 Functies zonder argumenten

Binnen VestiCode zijn er tal van standaardfuncties die helemaal geen argumenten hebben.

Een typisch voorbeeld is de functie CurrentDate die huidige datum teruggeeft. Bij het aanroepen van een functie zonder argumenten kunt u de haakjes achter de functienaam desgewenst helemaal weglaten, of u vermeldt gewoon de twee haakjes zonder iets er tussen.

```
Datum := CurrentDate;
Datum := CurrentDate();
```

N.B. aan het feit dat de functienaam CurrentDate niet met een v begint kunt u meteen zien dat dit een functienaam is die ook gebruikt wordt in EasyLanguage.

1.2.6 Functies met optionele argumenten

Het is ook mogelijk om een of meerdere argumenten van een functie optioneel te maken.

Worden de betreffende argumenten weggelaten bij het aanroepen van de functie dan gebruikt de functie automatisch een eigen default waarde.

```
value function zRente( value xKapitaal, value xPerc=5) begin
    {actie}
    end;
```

In bovenstaande functie definitie wordt aangegeven dat de functie de naam zRente heeft, en dat deze functie 2 argumenten verwacht, namelijk xKapitaal en xPerc(entage).

Doordat er bij het argument xPerc al een waarde is aangegeven (de zogenaamde default waarde), mag dit argument weggelaten worden bij het aanroepen van de functie zRente.

In dat geval zal het zijn alsof de functie aangeroepen werd met de waarde 5 als 2de argument.

```
xEindBedrag := xKapitaal + zRente(xKapitaal, 6);
xEindBedrag := xKapitaal + zRente(xKapitaal);
```

In de eerste instructie zal gerekend worden met het opgegeven percentage van 6%, terwijl in de tweede instructie de default rente van 5% wordt gehanteerd.

1.3 Tabellen en Series

We hebben tot nu toe steeds gewerkt met 'gewone' variabelen van het type **value** of **string** .

Al deze variabelen waren enkelvoudig in de zin dat ze steeds maar plaats hadden voor één bepaalde waarde.

In dit hoofdstuk gaan we kijken naar variabelen die meer dan 1 waarde kunnen bevatten.

1.3.1 Tabel variabelen

Tabel variabelen zijn variabelen waarin meer dan één waarde tegelijkertijd in kan worden opgeslagen.

Een dergelijke variabele is dus eigenlijk een verzameling van variabelen die onder één enkele naam worden aangeroepen. Om een variabele van het type tabel te definiëeren moet u aangeven uit hoeveel elementen de tabel bestaat:

```
value xBedrag[ 5];
```

De variabele xBedrag zal nu intern uit 5 afzonderlijke variabelen bestaan, die elk aangesproken kunnen worden door middel van hun volgnummer.

Dat volgnummer wordt ook vaak de tabelindex of gewoon index genoemd:

```
xBedrag[ 1] := 3;
```

Misschien is het nog niet zo duidelijk wat het voordeel is van één enkele tabel variabele boven een heleboel losse variabelen. Later zullen we allerlei voorbeelden zien waar dit voordeel duidelijker wordt.

1.3.2 Het begrip BAR

Voordat we verder gaan zullen we eerst even stilstaan bij een begrip dat belangrijk is bij het werken met indicatoren, namelijk het begrip bar.

Als we in een grafiek werken met dagkoersen, dan zal het duidelijk zijn dat er elke dag een koers is. We kunnen spreken van de 'koers van vandaag' en de 'koers van gisteren'.

Zouden we werken met weekkoersen dan moet die terminologie aangepast worden naar 'koers van deze week' en 'koers van vorige week'. Idem bij uurkoersen of kwartierkoersen en ga zo maar door.

We zouden dus liever een aanduiding willen gebruiken die altijd toepasbaar is, ongeacht de tijdsbasis van de koersgegevens.

In de technische analyse wordt vaak gebruik gemaakt van koersstaafjes (*price bars* in het engels) die het koersverloop binnen één zo'n tijdseenheid uitbeelden.

Het begrip *bar* is daardoor zo langzamerhand synoniem geworden met de basis tijdseenheid van de gebruikte koersgegevens. Dus als we met dagkoersen werken, dan correspondeert 1 bar met 1 dag, werken we met bijv. kwartierkoersen, dan is elk kwartier 1 bar.

En op die manier kunnen we vervolgens spreken over de 'koers van deze bar' en de 'koers van de vorige bar' zonder dat we hoeven te weten of het dagkoersen of weekkoersen betreft.

1.3.3 Series variabelen

Het begrip bar speelt een cruciale rol bij het soort variabelen dat we nu gaan bespreken, namelijk de data series oftewel series variabelen.

Net als bij de eerder besproken tabellen gaat het hierbij om een variabele die uit meerdere elementen bestaat en deze elementen kunnen via een index individueel worden geadresseerd.

Echter, bij een series variabele is het aantal elementen niet vast, maar is er voor elke bar een element. En die elementen worden opgenummerd naar het verleden toe. Element 0 is daarbij de huidige bar, element 1 is de vorige bar, enz.

De definitie van een serie variabele lijkt sterk op die van een tabel:

```
value Bedrag[];
```

In tegenstelling tot de definitie van een tabel, waar we aangaven hoeveel elementen er moesten zijn in de tabel, laten we bij de definitie van een series het aantal elementen weg. Het aantal elementen wordt dan automatisch aangepast aan het aantal bars in de beschikbare koersdata.

U heeft waarschijnlijk al geraden wat het doel van een series variabele is... namelijk het opslaan en bewerken van koersen en daaraan gerelateerde gegevens.

Sterker nog, als u in VestiCode aan het programmeren bent, dan zijn er altijd al een aantal series beschikbaar zonder dat u daar iets voor hoeft te doen. Deze 'ingebouwde' serie variabelen heten Open, High, Low, Close, Volume en OpenInt, en de namen spreken voor zich.

Via indexering kan de koers van elke bar in het verleden benaderd worden:

```
if Close[ 0]>Close[ 1] then xUpBar := xUpBar+ 1;
```

In bovenstaand voorbeeld wordt getest of de slotkoers van deze bar hoger is dan de slotkoers van de vorige bar en zo ja dan verhogen we de variabele xUpBar met 1.

Overigens mag men de index aanduiding [0] altijd weglaten, zodat Close gelijk is aan Close[0] en refereert aan de slotkoers van deze bar.

1.3.4 Het gebruik van tabellen en series

Doordat bij een tabel of series variabele een heleboel gegevens onder één naam beschikbaar zijn, wordt het heel gemakkelijk om in één klap al die data aan te spreken.

Stel we willen een gemiddelde koers berekenen voor de laatste 200 bars, dan is het ondoenlijk om al die 200 variabelen te noemen in één lange optel-instructie:

```
xTotaal := Close+Close[ 1]+...+Close[ 198]+Close[ 199];
xGemiddelde := xTotaal/ 200;
```

Dat zou wel een hele lange regel worden als we alle 200 index waarden expliciet moeten benoemen. Veel gemakkelijker is het gebruik van de repeterende for-instructie:

```
xTotaal := 0;
for xIndex := 0 to 199 do xTotaal := xTotaal+Close[xIndex];
xGemiddelde := xTotaal/ 200;
```

In deze opzet wordt door de for-instructie een variabele xIndex steeds met 1 verhoogt en vervolgens wordt die indexwaarde gebruikt om de slotkoers van de betreffende bar bij het totaal op te tellen.

1.3.5 Series als argument voor functies

Een ander voordeel van serie variabelen is het feit dat er tal van functies beschikbaar zijn die kunnen werken met series variabelen, zoals de Average functie die het gemiddelde berekent van een aantal waarden in een series variabele, zodat het berekenen van het gemiddelde van de laatste 200 bars ook via één enkele functie aanroep verkregen kan worden:

```
xGemiddelde  := Average(Close, 200);
```

1.3.6 Een functie als series variabele

Sommige functies gedragen zich zelf ook weer als een serie variabele, zodat het mogelijk is om de waarde van een vorige bar op te vragen aan de betreffende functie:

```
xVorigGemiddelde   := Average(Close, 200)[1];
```

Doordat Average een series functie is, worden alle waarden van voorgaande bars bewaard en kunnen deze voorgaande waarden via het toevoegen van een index opvraagbaar.

1.4 Eigen indicatoren maken

Nu we weten hoe we eenvoudige instructies in VestiCode kunnen schrijven, en deze kunnen groeperen tot functies, gaan we over tot het eigenlijke werk... het programmeren van eigen indicatoren en (verderop) handelssystemen.

1.4.1 Datum en tijd formaat

Binnen VestiCode gebruiken we hetzelfde formaat voor datum en tijd als gebruikt wordt in EasyLanguage. Op die manier kunnen we bestaande indicatoren, die gepubliceerd worden in tijdschriften e.d., probleemloos overnemen in VestiCode.

Voor datums wordt daarbij een ietwat vreemde notering gebruikt...
Datums worden in VestiCode opgeslagen als een getal van 6 of 7 cijfers, in het formaat YYYMMDD, waarbij YYY staat voor het jaartal minus 1900, MM staat voor de maand en DD staat voor de dag. Zo is 14 februari 1998 gecodeerd als 980214 terwijl 27 november 2002 gecodeerd wordt als 1021127. Toegegeven, het is even wennen om 1900 van het jaartal af te trekken, maar dit gebruik stamt nog uit de vorige eeuw, toen datums als 2 cijfers werden weergegeven. Volgens dat systeem volgt na 991231 (31 december 1999) als volgende datum 1000101 (1 januari 2000).

Voor tijden wordt gebruik gemaakt van de volgende notering...
Tijden worden in Vestics gecodeerd volgens het 24-uur systeem als een getal van maximaal 4 cijfers in het formaat HHMM, waarbij HH het uur is en MM de minuten. Volgens dit systeem staat 0000 (oftewel 0) gelijk aan middernacht en is 2359 gelijk aan 23 uur 59 oftewel 1 minuut voor middernacht.

Binnen VestiCode zijn tal van functies beschikbaar die hetzij gebruik maken van bovenstaande

datum en tijd gegevens, hetzij een resultaat opleveren in een van bovenstaande formaten.

1.4.2 Koersgegevens

Als we in VestiCode aan het programmeren zijn, dan bevinden ons vrijwel altijd in een omgeving waar we werken met de koersgegevens van één of meerdere fondsen. We zullen ons in eerste instantie beperken tot situaties waar we met één fonds werken. Later zullen we bekijken hoe het gaat met meerdere fondsen.

De koersgegevens van het fonds zijn ten alle tijde in alle functies van VestiCode direct toegankelijk via een aantal standaard series variabelen met de volgende namen...

- **Date** is een series variabele met per bar de datum in het speciale datum formaat .
- **Time** is een series variabele met per bar de slottijd van de betreffende bar in het tijdformaat HHMM
- **Close** is een series variabele met de slotkoersen van alle bars
- **Open** bevat de openingskoersen van alle bars
- **Low** bevat de laagste koersen
- **High** bevat de hoogste koersen
- **Volume** bevat de totale omzet per bar
- **OpenInt** bevat (alleen bij opties en futures) de open interest per bar
- **Ticks** bevat (alleen bij intradag koersen) het aantal tikken per bar
- **UpTicks** bevat het aantal tikken omhoog
- **DownTicks** bevat het aantal tikken omlaag

Binnen EasyLanguage mogen de namen van deze series ook afgekort worden tot de letters D, T, C, O, L, H, V, I. Wij vinden dat zeer verwarrend en onze ervaring is dat dit zeer onleesbare programma's oplevert. Ons dringende advies is dan ook om altijd de volledige namen te gebruiken. Het feit dat we ook in VestiCode deze afkortingen toestaan is uitsluitend bedoeld om het mogelijk te maken om bestaande EasyLanguage functies over te nemen in VestiCode zonder eerst een vertaalslag te moeten maken.

1.4.3 Bar na bar verwerking

Als we in VestiCode aan de slag gaan worden onze functies, de indicatoren en handelssystemen, een voor een doorgerekend in de volgorde dat we ze toegevoegd hebben aan onze Grafiek.

Nog belangrijker is dat deze berekening steeds weer voor elke volgende koersbar herhaald wordt.

Stel dat we een Grafiek hebben met daarin de koersen van Philips plus een indicator de een korte MA van 5 bars uitrekend en een tweede indicator die een lange MA van 200 bars uitrekend. Kijken we dan in de VestiCode Designer naar deze Grafiek dan zien we 3 formules, namelijk een formule om de koersen van Philips in te laden, een 2de formule voor de korte MA en een 3de formule voor de lange MA.

Bij het tekenen van de Grafiek zullen deze 3 formules steeds weer voor elke bar 'uitgerekend' worden. De waarden die dat oplevert worden vervolgens in de grafiek weergegeven. Als je dit proces zou kunnen vertragen, dan zou je eerst een puntje zien verschijnen voor de koers bij de eerste bar. Vervolgens verschijnt een 2de puntje recht boven of onder de koers die aangeeft waar de korte MA op die dag op uitkomt, en even later verschijnt ook weer boven of onder de koers een 3de puntje ter hoogte van de waarde van de lange MA.

Vervolgens schijft alles 1 bar naar rechts, en verschijnen wederom achtereenvolgens de 3 puntjes voor de koers en de twee MA's. Ook nu weer staan die 3 puntjes recht boven elkaar want ze hebben immers betrekking op dezelfde bar, en de X-as van de Vestics Grafiek is altijd de tijd van de

betreffende bar.

Dit proces wordt herhaald voor alle koersdata. Dus als we werken met 2 jaar dagkoersen (dus 1 bar = 1 dag) dan zal dat ongeveer 522 bars opleveren en bij elke bar horen de drie punten van de koers en de twee MA's. Alle koerspunten vormen voor het oog een doorlopende lijn, net zoals de MA-punten ook twee lijnen vormen. Afhankelijk van het ingestelde lijntype zal het Grafiekprogramma bovendien deze punten met elkaar verbinden als ze door verder inzoomen los van elkaar komen te staan.

Het proces van bar-na-bar doorrekenen stopt uiteraard als we bij de laatste bar van de koersdata gearriveerd zijn. Als we echter bezig zijn met een realtime intradag grafiek, dan kan het zijn dat elke bar 15 minuten is, en dat er dus elk kwartier een extra bar bijkomt. Voor elke nieuwe bar worden de drie nieuwe punten berekend en worden de drie lijnen verder doorgetrokken.

1.4.4 De huidige bar

Voor elke bar in de koersdata worden alle functies en formules in een Grafiek opnieuw uitgerekend. Het is dus als het ware alsof we voor elke bar opnieuw binnenkomen in de functie. Toch levert de functie voor elke bar een andere waarde op, anders zouden we immers alleen maar horizontale rechte lijnen op ons scherm zien.

Het feit dat dezelfde formules toch andere waarden opleveren, heeft twee oorzaken...

1) de belangrijkste oorzaak ligt in het feit dat een referentie naar een series-variabele alle relatief is ten opzichte van de huidige bar. Dus als een functie refereerd naar de standaard series variabele Close dan levert dat voor deze bar een andere waarde op dan bij dezelfde berekening tijdens de vorige bar. We refereren dus altijd impliciet naar de slotkoers van de bar die op dat moment aan de beurt is. En omdat de slotkoers elke dag anders is, is het resultaat van onze berekening ook elke dag anders.

Dat geldt overigens ook als we bijv. refereren naar de waarde van een series in het verleden. Immers, de waarde van Close[3] is de slotkoers van 3 bars geleden, maar ook die waarde schuift mee met elke nieuwe bar. Dus ook hier weer is het resultaat afhankelijk van de bar waar we op dat moment mee bezig zijn.

2) een andere factor is het feit dat alle eigen variabelen (zoals xBedrag, e.d.) bij elke nieuwe bar doorgaan met de waarde die deze variabele had bij de vorige bar, tenzij we ze expliciet in het begin van onze functie steeds weer op een vaste beginwaarde zetten.

Het feit dat variabelen hun oude waarde behouden, wordt vaak gebruikt om totalen bij te houden e.d., zoals in onderstaand voorbeeld...

```
value function zVoorbeeld begin
    value xTotaal := 0;
    xTotaal := xTotaal+Close;
    zVoorbeeld := xTotaal;
    end;
```

In dit voorbeeld wordt een totaal bijgehouden van alle slotkoersen en de som tot nu wordt teruggeven. Deze functie levert dus elke bar een ander resultaat op omdat 1) de series variabele Close voor elke bar een andere waarde heeft en, 2) de waarde xTotaal steeds verder groeit.

1.4.5 De Print functie

Een van de eerste dingen waar u bij het oefenen met VestiCode behoefte aan zult hebben is de mogelijkheid om het resultaat van uw programmeersels te kunnen toetsen.

Dat kan heel eenvoudig met behulp van de standaard functie Print, die als volgt gebruikt wordt...

```
Print(arg1,arg2,arg3,...);
```

Op de plaats van arg1, enz. kunt u de informatie opgeven die u wilt printen. U mag tot maximaal 25 argumenten opgeven, en elk argument mag een value of string waarde zijn in de vorm van een constante, variabele of een expressie.

Enkele voorbeelden...

```
value function zTest (value xGetal) begin
    Print('functie zTest aangeroepen met het getal ',xGetal);
    if xGetal< 0 then Print('dit getal is negatief')
    else if xgetal= 0 then print('dit getal is nul')
    else Print('dit getal is positief');
    end;
```

Als deze functie aangeroepen wordt met de waarde 12 dan worden de volgende 2 regels weergegeven...

```
functie zTest aangeroepen met het getal 12
dit getal is positief
```

Elke Print-functie resulteerd in een nieuwe regel. Door de Print functie aan te roepen zonder argumenten, dus Print of Print(), wordt een lege regel afgedrukt.

Normaal gaat deze informatie naar het Report-tabblad van de Grafiek of van de Designer.
Het is echter mogelijk om de gegevens naar een echte printer te sturen of naar een tekstbestand op schijf. Dat gaat door middel van een eerste argument warain de bestemming wordt opgegeven...

```
Print( Printer ,arg2,arg3,...);
Print( File ('c:\temp\bestand.txt'),arg2,arg3,...);
```

Door middel van het woord Printer als eerste argument wordt de informatie afgedrukt op de Standaard Printer van Windows.
Door middel van het woord File kunt u een bestandnaam opgeven waaronder de informatie op schijf wordt weggeschreven.

1.4.6 Het tekenen van grafieken

Bij het maken van indicatoren is een van de belangrijkste resultaten uiteraard een mooie grafiek waarin de waarde van uw indicator wordt weergegeven.

Een grafiek ontstaat in principe door voor elke bar een puntje op het scherm te zetten dat een bepaalde waarde representeerd. Bij een lijngrafiek worden al die puntjes onderling met elkaar verbonden, terwijl bijvoorbeeld bij een histogram de puntjes elk afzonderlijk verbonden worden met het nul-punt zodat de histogram-staafjes ontstaan.

Wat we dus nodig hebben is een series variabele waarin voor elke bar de weer te geven waarde wordt opgenomen. Een dergelijke series variabele, die tevens op de grafiek weergegeven wordt, noemen we een plot.

Voor het aanmaken van plot series wordt gebruik gemaakt van de Plot-instructie.

Deze instructie heeft de volgende opbouw...

```
Plotn ( waarde , naam );
```

Waarbij...
- Plot*n* gelijk kan zijn aan Plot1, Plot2, Plot3 of Plot4.
- waarde is de (vertikale) waarde van de grafiek voor de huidige bar.
- naam is de naam die in de legenda van de Grafiek wordt weergegeven voor deze plot

Voorbeeld, maak grafiek van 5-bar Moving Average...

```
Plot1(Average(Close, 5),'MA5');
```

Het is via Plot1-Plot4 mogelijk om per indicator maximaal 4 verschillende lijnen te tekenen, of om met behulp van maximaal 4 verschillende waarden complexe plots te maken, zoals candle-sticks en price-bars.

De series variabele die ontstaat door deze Plot-instructies kan ook als een gewone series variabele in de verdere berekeningen gebruikt worden...

```
xVorige = Plot1[1]; { pak plot-waarde vorige bar }
```

Hoe een plot series wordt weergegeven op de Grafiek kan worden ingesteld via de INFO-knop in de VestiCode Designer .

1.4.7 Speciale soorten plots

In EasyLanguage is het ook mogelijk om in de Grafiek bepaalde bars te markeren via een bepaald symbooltje boven of onder de koers, c.q. door de betreffende bar een andere kleur te geven.

In VestiCode is het uiteraard ook mogelijk dergelijke markeringen aan te brengen in de grafiek. Dit wordt met name gestuurd door het lijntype voor de betreffende plot op de gewenste markering in te stellen.
Belangrijk is het dan wel om op dagen dat er geen markering moet komen geen waarde in de series te hebben. Dit bereikt u door geconditioneerd de plot waarde in te vullen...

```
if Close>Close[1] and Close[1]>Close[2] and Close[2]>Close[3] then
    Plot1(High,'3xUp');
```

Door vervolgens deze plot in te stellen als 'Markering' zullen alle bars waarbij de koers 3 opeenvolgende dagen gestegen is gemarkeerd worden op de door u aangegeven manier.

Voor markeringen wordt meestal de High of de Low als waarde opgegeven, waardoor de markering net boven of net onder het koersstaafje wordt weergegeven.

1.4.8 Functies voor indicatoren

Bij het maken van nieuwe indicatoren kan gebruik gemaakt worden van een rijk arsenaal aan kant en klare functies. Denk hierbij aan het middelen van koerswaarden van de afgelopen tijd, Lineaire Regressies, enz.

Een volledige lijst van alle beschikbare functies, gegroepeerd naar toepassing, vindt u verderop in

dit handboek.

In een later hoofdstuk zullen we ook een aantal praktijkvoorbeelden doornemen, zodat u een duidelijker beeld krijgt van hoe het ontwikkelen van indicatoren in z'n werk gaat.

1.5 Eigen handelssystemen maken

1.5.1 Crosses above en below

In de ontwikkeling van handelssystemen komt het vaak voor dat men aktie moet nemen als een koers of een indicator kruist met een andere lijn of met een vaste waarde.

Dit kruisen van twee series wordt zo vaak gebruikt dat er aparte functies voor bestaan, namelijk de functies...

- `vCrosses (series1,series2)`
- `vCrossesAbove (series1,series2)`
- `vCrossesBelow (series1,series2)`

De functie **vCrosses** geeft als resultaat de waarde 0 als de beide series zich tussen vandaag en gisteren niet kruisen. Als er wel een kruising heeft plaatsgevonden, dan geeft een negatief resultaat aan dat de eerste series lager per de huidige bar lager is dan de tweede series. Omgekeerd geeft bij een kruising een positief resultaat aan dat de eerste series na de kruising boven de tweede series staat.

De functie **vCrossesAbove** geeft als resultaat de waarde 0 als beide series zich niet gekruisd hebben bij de huidige bar, of als er wel een kruising heeft plaatsgevonden, maar de eerste series is daarbij onder de tweede series gedoken. Heeft er wel een kruising plaatsgevonden, en de eerste series is na de kruising boven de tweede series terecht gekomen, dan resulteerd dat in een positieve waarde.

De functie **vCrossesBelow** is het spiegelbeeld van de CrossesAbove functie en resulteerd in een positief resultaat als er een kruising heeft plaatsgevonden en series1 is onder series2 gedoken. In alle andere gevallen wordt de waarde 0 teruggegeven.

Omdat de waarden 0 en ongelijk 0 als de condities onwaar en waar worden geïnterpreteerd in if-instructies en dergelijke, is het dus mogelijk om op eenvoudige manier geconditioneerde akties te programmeren die uitgevoerd moeten worden als een bepaalde kruising plaatsvindt...

```
value xMaKort;
xMaKort := Average(Close, 5);
if vCrossesAbove(Close,xMaKort) then {doe iets};
```

De positieve of negatieve waarde die bovenstaande functies genereren in geval van een kruising is bovendien een maatstaf voor de kracht van de kruising. Als namelijk een koers stijgt van 92,30 naar 92,50 en daarbij een voortschrijdend gemiddelde kruist dat op 92,40 lag, dan is dat een veel geleidelijkere beweging dan als de beweging was geweest van 90,70 naar 92,50 of van 92,30 naar 94,10. In eerste geval is de koers maar 0,2% cent gestegen terwijl er in de tweede situatie een stijging van bijna 2% was. De niet-nul waarde waarin bovenstaande drie functies resulteren is dan ook de procentuele stijging of daling die tot de kruising heeft geleid.

1.5.2 Het genereren van alarmeringen

Het doel van vrijwel elk handelssysteem is het genereren van alarmeringen of orders.

Als we alleen maar een alarmering genereren, dan moeten we zelf later nog beslissen of we ook daadwerkelijk iets met het betreffende waarschuwing doen.

Uiteraard is het ook goed mogelijk om alarmeringen te gebruiken als een soort waarschuwing vooraf dat de kans groot is dat er binnenkort een echte order komt.

Alarmmeldingen worden via het Signaleringsmechanisme van Vestics weergegeven op het beeldscherm, doorgestuurd naar een SMS-telefoon of via ICQ of eMail naar een bepaald adres

doorgegeven.

U kunt een dergelijke Alarmmelding vanuit VestiCode initieren door middel van de Alert functie. Deze ziet er als volgt uit...

```
Alert ('boodschap' );
```

Over het algemeen zult u een Alert willen verzenden onder bepaalde omstandigheden, zodat een Alert vaak in een conditionele if-constructie gebruikt wordt...

```
if Close<xStopLoss  then Alert( 'Stoploss-waarde  overschreden' );
```

Uiteraard is het ook mogelijk om in de te verzenden boodschap variabele informatie op te nemen, zoals...

```
string xTekst;
if Close<xStopLoss  then begin
    xTekst := 'Stoploss van '+NumToStr (xStopLoss)+ ' overschreden' ;
    Alert(xTekst);
    end;
```

In dit voorbeeld wordt de tekst die gezonden worden als signaal samengesteld uit een combinatie van vaste tekst en de stoploss-waarde die overschreden werd.

1.5.3 Het genereren van orders

Behalve het genereren van alarmeringen met behulp van de Alert functie is het in VestiCode ook mogelijk om orders te genereren voor het kopen of verkopen van aandelen.

Voor het het kopen of verkopen van aandelen wordt gebruik gemaakt van een viertal functies:

- vEnterLong Deze functie start een nieuwe long positie
- vExitLong Deze functie sluit een eventuele long positie
- vEnterShort Deze functie start een eventuele short positie
- vExitShort Deze functie sluit een eventuele short positie

Alle order functies hebben min of meer dezelfde argumenten, namelijk...

- xName Naam die aan deze positie wordt gehangen.
- xContracts Aantal aandelen of contracten die gekocht worden
- xWhen Code die aangeeft wanneer de positie geopend wordt
- xOrderType Code die aangeeft welk soort order wordt gebruikt (bijv. Bestens of limietorder)
- xLimit Limietprijs indien een limietorder wordt opgegeven

Met behulp van **xName** kan een naam of omschrijving aan een bepaalde openingsorder gehangen. Deze naam kan desgewenst weer gebruikt worden bij de sluitorder om aan te geven dat alleen de betreffende openingsorder ongedaan moet worden gemaakt. De naam van de order wordt tevens meegegeven in de transactie-regel, zodat later eventueel te herleiden is op grond van welk entry-signaal een bepaalde positie werd geopend.

In het argument **xContracts** geeft men aan hoeveel aandelen of contracten men wil kopen of verkopen. Als dit argument wordt weggelaten in een openingsorder (vEnterLong of vEnterShort) zal de waarde 1 gebruikt worden.

Met het argument **xWhen** kunt u sturen wanneer de order wordt geplaatst. Dat kan zijn op de slotkoers van de huidige bar of op de openingskoers van de volgende bar. Indien dit argument niet

wordt meegegeven dan zal gehandeld worden op basis van de slotkoers van de huidige bar. Met behulp van de instelling Slippage kan deze slotkoers eventueel gecorrigeerd worden.

Via het argument **xOrderType** kunt u aangeven of het een Bestens order of een Limiet order moet worden. Bij het opgeven van limietorders kan het zijn dat de betreffende positie niet ingenomen wordt omdat de limietprijs niet gehaald werd.

Met het argument **xLimit** kunt u in geval van een limietorder opgeven welke limiet gehanteerd wordt. Bij het opgeven van limietorders kan het zijn dat de betreffende positie niet ingenomen wordt omdat de limietprijs niet gehaald werd.

Indien een money management functie wordt gebruikt zal het aantal contracten berekend worden door deze functie en wordt de instelling **xContracts** genegeerd.

Indien gebruik gemaakt wordt van een instrument functie , zal de aan- of verkoopkoers altijd berekend worden door de Instrument functie en worden de argumenten **xWhen** , **xOrderType** en **xLimit** genegeerd.

De **vEnterLong** functie zal een eventuele short positie altijd sluiten voordat de long positie wordt ingenomen. Omgekeerd zal de **vEnterShort** functie een eventuele long positie sluiten voordat de short positie wordt ingenomen.

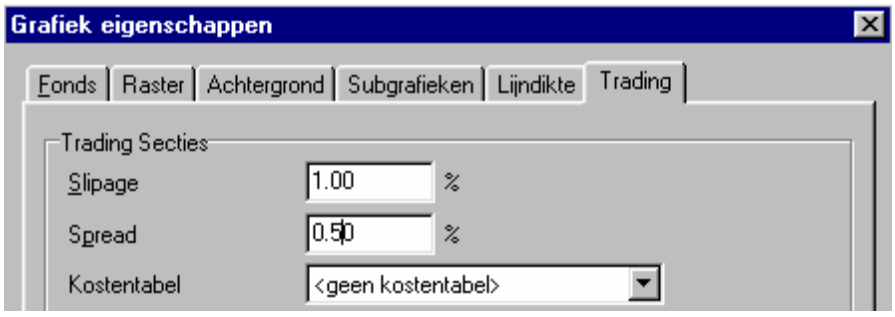
Een **vExitLong** functie doet niets als de huidige positie neutraal of short is. Hetzelfde geldt voor een **vExitShort** functie die niets doet in geval van een neutrale of long positie.

Indien bij een **vExitLong** of **vExitShort** functie een naam wordt meegegeven, en er is geen tegengestelde positie met dezelfde naam, dan wordt deze functie genegeerd.

Indien een **vEnterLong** functie wordt aangeroepen, en de huidige positie is reeds long, dan wordt de long positie verder vergroot, tenzij de instelling 'Positie Uitbreiden ' is gedeactiveerd. Hetzelfde geldt voor **vEnterShort** bij een bestaande short positie.

1.5.4 Het gebruik van Spread en Slippage

Via het **Trading** tabblad van de grafiekeigenschappen dialoog is het mogelijk om slippage en spread in te voeren die toegepast moeten worden bij het afhandelen van de transacties.



Slippage
Vooral trendvolgende systemen hebben last van het euvel dat de koers vaak sterk in beweging is op het moment dat een signaal gegeven wordt. Bijv. bij een uitbraak treedt meestal een versnelling op in de koersbeweging omdat een groot aantal beleggers die uitbraak als signaal gebruiken om een order in te leggen. Uiteraard is die beweging altijd een negatieve. Dus bij een uitbraak omhoog zal de koers versneld stijgen terwijl (of beter: omdat) de meeste beleggers dan een kooporder inleggen. Bij een uitbraak naar beneden zal de koers eerder versneld zakken. Het effect van deze slippage is

dat men altijd meer moet betalen of minder ontvangt dan men zou verwachten op basis van de koers op het moment dat de order wordt ingelegd.

Slippage wordt ook vaak gebruikt om een aanpassing te maken in een handelssysteem dat theoretisch handelt op de slotkoers van een dag terwijl in de praktijk gehandeld zal worden tegen de openingskoers van de volgende dag. Als er dan geen betrouwbare openingskoersen beschikbaar is kan men eventueel handelen op de slotkoers en met slippage een aanpassing maken.

Door een bepaald percentage slippage in te stellen, zal elke aan- en verkoopkoers automatisch (in negatieve zin) aangepast worden. Dus bij een kooporder wordt de aankoopkoers met dat percentage verhoogd, bij een verkooporder wordt de verkoopkoers met hetzelfde percentage verlaagd.

Spread
Dit is de welbekende bid-ask spread die bij elk beleggingsinstrument aanwezig is. Terwijl er slechts één koers is die gebruikt wordt in de simulaties, is er in de praktijk altijd sprake van 2 koersen, namelijk de bied-koers en de laat-koers. De spread is het verschil tussen deze beide koersen. Bij beleggingsinstrumenten die met grote omzetten verhandeld worden is de spread meestal relatief laag. Gaat men echter handelen in exotische beleggingsinstrumenten, dan kan de spread sterk oplopen.

Door een bepaald percentage spread in te stellen zal elke aan- en verkoopkoers automatisch (in negatieve zin) aangepast worden. Dus bij een kooporder wordt de aankoopkoers met de helft van dat percentage verhoogd, bij een verkooporder wordt de verkoopkoers met de helft van het percentage verlaagd.

1.5.5 Pyramiding oftewel positie uitbreiden

Op dit moment is pyramiding nog niet ondersteund in VestiCode

1.6 Eigen instrumenten maken

Een Instrument functie is een speciaal soort functie die bepaalt wat en tegen welke prijs er gekocht gaat worden.

De structuur van de functie is als volgt...

```
value function zMyInstrument() begin
    { bereken xPrijs }
    zMyInstrument := xPrijs;
    end;
```

Enkele mogelijke toepassingen van een instrument functie...

Berekenen theoretische prijs van een effect
Stel u wilt een backtest doen van handelen in FTI-futures, maar u heeft alleen de koersen van de AEX index die ten grondslag ligt aan de FTI-futures. Op basis van de resterende looptijd van de future op een willekeurige dag, kan altijd met redelijke nauwkeurigheid de prijs van de future op die betreffende dag berekend worden in een instrument functie waarna die prijs gehanteerd wordt bij alle orders.

Berekenen prijs van combinaties van effecten
Stel u wilt uw aandelen positie altijd afdekken met put-opties. Bij elke kooporder koopt u dus niet alleen het aandeel, maar tevens een putoptie. Bij elke verkooporder verkoopt u zowel het aandeel als de put-optie. Door steeds de theoretische prijs van de putoptie op te tellen bij de koers van het aandeel, kunt u bij elke order berekenen wat de kosten zijn van een dergelijk 'setje'.

Handelen in een ander effect

Stel u wilt signalen berekenen op basis van het aandeel Philips en vervolgens een positie innemen door steeds call of put opties te kopen. U kunt dan in de instrument functie hetzij de theoretische prijs van die calls en puts berekenen, of zelfs de actuele koers laden uit een koersbestand en die vervolgens gebruiken in de order.

Hoewel het principe van een instrument functie relatief eenvoudig is, moet men in de praktijk aan allerlei neven effecten denken, zoals het doorrollen van derivaten, het afronden naar hele contracten, enz.

Bij Vestics worden een aantal standaard instrumenten meegeleverd die bovenstaande situaties ondersteunen. Bij het ontwikkelen van eigen instrumenten kunt u terugvallen op deze standaard instrumenten als voorbeeld.

1.7 Eigen Money Management maken

Money management is de techniek waarbij de 'inzet' (in het engels de *betting size*) per signaal wordt aangepast.

Dat kan hele gewone oorzaken hebben, bijv. door het aantal aandelen te relateren aan het beschikbare kapitaal, maar kan ook gebaseerd zijn op een analyse van de marktsituatie (volatility, risco, enz.) of op basis van de resultaten in het recente verleden.

In VestiCode is de doelstelling van de Money management functie om te bepalen hoeveel aandelen gekocht moeten worden.

Standaard worden een 3-tal MM-functies meegeleverd...

FixedShares
Deze functie zorgt er voor dat bij elke transactie een vast aantal aandelen wordt gekocht. Per default is dat 1, maar het aantal kan zelf ingesteld worden via de rechter muisknop >> Instellingen wijzigen. Met name bij het handelen in futures worden systemen vaak doorgetest op basis van 1 contract per trade.

FixedCapital
Deze functie zorgt er voor dat bij elke transactie een vast bedrag geïnvesteerd wordt. Afhankelijk van de geldende koers worden er op die manier meer of minder aandelen gekocht.

Reinvest
Deze functie begint met een bepaald startkapitaal en naarmate er winsten en verliezen worden gemaakt zal het beschikbare kapitaal groeien of afnemen. Bij het begin van elke trade wordt belegd op basis van het beschikbare kapitaal. Hierbij kan men tevens aangeven welk percentage van het beschikbare kapitaal steeds wrdt ingezet.

- bij een inzet van 100% zal steeds al het beschikbare kapitaal ingezet worden
- bij een inzet <100% zal slechts een gedeelte van het kapitaal ingezet worden. Dit is met name van belang bij optie-strategieën waarbij de hele inzet verloren kan gaan. Men zou dan kunnen besluiten om per trade maximaal 30% van het beschikbare kapitaal in te zetten.
- bij een inzet >100% wordt gewerkt met leverage, oftewel geleend geld.

Eigen MM-functies
Naast deze standaard functies kunt u zelf uw eigen MM-functies toevoegen aan Vestics. Mogelijke toepassingen zijn...

- Aanpassen inzet per transactie aan de geldende markt omstandigheden. Hierbij moet u denken aan volatility en tradingrange achtige berekeningen. Vervolgens bepaald u dat, gegeven de huidige marktomstandigheden, u voorzichtiger of net agressiever in de markt stapt.

- Aanpassen inzet per transactie aan de recente resultaten. U kunt daarbij besluiten dat u na een aantal verliesgevende transacties voorlopig wat rustiger aan gaat doen. U zou ook kunnen besluiten om na een aantal verliesgevende transacties de inzet te verhogen in anticipatie op de omkeer die binnenkort zal komen.

U zou kunnen overwegen om uw MM-functie te voorzien van een aantal invoer-argumenten waarmee de richting van de bijstellingen en de mate van bijstelling van buiten af bestuurd kan worden. Vervolgens kunt u via optimaliseren bepalen wat het effect is van een bepaald soort bijstelling op het resultaat van uw trading systeem.

Via money management kunt u geen invloed uitoefenen op het feit of winst of verlies gemaakt wordt op individuele trades. Wat u wel kunt doen is...

- Het percentage winstgevende trades beïnvloeden door eventueel trades met een hoog risico geheel over te slaan
- Het rendement beïnvloeden doordat u bij positieve winstverwachting meer aandelen koopt dan bij riscante trades

1.8 Eigen rapportage functie maken

Een belangrijk element in met name een handelssimulatie met historische koersen (een zgn *back test*) is de rapportagefunctie die een zo duidelijk mogelijk beeld geeft van de performance van het geteste handelssysteem. Iedereen zal z'n eigen voorkeuren hebben qua beoordelingscriteria en kengetallen, en daarom worden er bij Vestics een aantal verschillende rapportagefuncties standaard meegeleverd.

Daarnaast is het mogelijk om een eigen rapportagefunctie te ontwikkelen, waarin men naar believe eigen kengetallen kan berekenen en de resultaten naar hartelust kan opsplitsen, sorteren en groeperen.

Voor het maken van de rapportagefunctie kan men op twee manieren te werk gaan...

1. Men kan geheel bouwen op de performance functies van Vestics, die getallen kunnen ophoesten als aantal winnende trades, aantal long trades, totale winst, enz.
2. Men kan ook bar voor bar zelf allerlei getallen verzamelen en deze vervolgens bij de laatste bar gebruiken om de eigen kengetallen te berekenen.

In de praktijk zal men bijna altijd een combinatie van beide methoden toepassen omdat men bepaalde eigen informatie wil verzamelen en rapporteren, maar uiteraard ook informatie wil rapporteren zoals die in elke trading report staan, zoals
totale winst, totaal aantal trades, enz. Voor die laatste gegevens hoeft men niet moeilijk zelf te gaan tellen en rekenen, maar kan men de performance functies van Vestics gebruiken.

Voor het uiteindelijk afdrukken van het rapport gebruikt men de Print functie.

Een typische rapportage functie heeft dan ook de volgende opbouw...

```
value function zMyReport() begin
    value xTeller1,xTeller2;
    if CurrentBar =1 then begin
        xTeller1 := 0;
        xTeller2 := 0;
        }
    if CurrentContracts <>0 then begin
        { doe tellingen als we 'n positie hebben }
        end;
    if vLastBar then begin
```

```
        Print( 'Totaal  aantal  trades  ',TotalTrades);
        Print( 'Totaal  aantal  xyz     ',xTeller1);
        end ;
end ;
```

In bovenstaand generiek voorbeeld worden bij de eerste bar (als de functie CurrentBar de waarde 1 oplevert) allerlei initiële zaken geregeld, terwijl bij de laatste bar (als de functie vLastBar de waarde *true* oplevert) de resultaten geprint worden. Daar tussen in worden allerlei tellingen gedaan, waarbij gebruik gemaakt wordt van functies zoals CurrentContracts e.d. om op te vragen wat de huidige positie.

1.9 EasyLanguage: korte inleiding

Het is mogelijk om in Vestics gebruik te maken van bestaande EasyLanguage indicatoren die overgenomen kunnen worden vanuit TradeStation of vanaf het Internet.

Normaliter hoeven er geen aanpassingen gemaakt te worden in deze bestaande EasyLanguage code, en kan de tekst integraal worden overgenomen in Vestics.

Mocht u aanpassingen willen maken in de bestaande EasyLanguage broncode, dan verwijzen we naar de EasyLanguage literatuur die rijkelijk aanwezig is op het Internet. Zo is er bijv. een compleet (engelstalig) handboek te downloaden van de website van Omerga Research.

1.9.1 Overzicht EasyLanguage begrippen

In dit overzicht worden de belangrijkste EasyLanguage sleutelwoorden en begrippen kort toegelicht. Voor een meer gedetailleerde beschrijving verwijzen wij naar de officiële EasyLanguage documentatie.

a	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
above	"crosses above" is een operator, in VestiCode beschikbaar als functie vCrossesAbove()
ago	"close 1 bar ago" is gelijk aan "close[1]"
alert	zowel functie als interne variabele. "alert = true;" activeert een signaal
all	"exitlong all shares" is gelijk aan "exitlong" en sluit hele positie
an	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
array(s)	"array: xTable[4](0);" definieert een tabel en is gelijk aan "value xTable[4]=0;"
at	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
at$	refereerd naar de entry-bar van een trade. (*niet ondersteund in Vestics*)
bar(s)	gebruikt in "next bar" of "1 bar ago"
based	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
below	"crosses below" is een operator, in VestiCode beschikbaar als functie vCrossesBelow()
black	kleur (waarde is 1) in sommige EasyLanguage functies
blue	kleur (waarde is 2) in sommige EasyLanguage functies
by	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
c	afkorting van close
cancel	"cancel alert" maakt eerdere alert ongedaan
contract	"buy 1 contract" specificeerd aantal aandelen of contracten
cross(es)	zie above en below

cyan	kleur (waarde is 3) in sommige EasyLanguage functies
d	afkorting voor date
darkblue	kleur (waarde is 9) in sommige EasyLanguage functies
darkbrown	kleur (waarde is 14) in sommige EasyLanguage functies
darkcyan	kleur (waarde is 10) in sommige EasyLanguage functies
darkgray	kleur (waarde is 15) in sommige EasyLanguage functies
darkgreen	kleur (waarde is 11) in sommige EasyLanguage functies
darkmagenta	kleur (waarde is 12) in sommige EasyLanguage functies
darkred	kleur (waarde is 13) in sommige EasyLanguage functies
day(s)	synoniem voor bar(s)
default	gebruikt in plot-statement (*niet ondersteund in Vestics*)
does	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
entry	"exitlong from entry ("xyz");"
false	in VestiCode gelijk aan de waarde 0
file	zie Print functie
friday	dag van de week (waarde is 5) in sommige EasyLanguage functies
from	"exitlong from entry ("xyz");"
green	kleur (waarde is 4) in sommige EasyLanguage functies
h	afkorting van high
higher	"exitlong next bar at 75 or higher;" (limiet of stop order)
i	afkorting voor openint
input(s)	"input: xArg(0);" wordt gebruikt om inputargumenten te definiëren
is	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
l	afkorting van low
limit	"buy next bar at 75 limit;"
lower	"buy next bar at 75 or lower;" (limiet of stop order)
magenta	kleur (waarde is 5) in sommige EasyLanguage functies
market	"buy next bar at market;" (bestens order)
mod	"25 mod 3" geeft rest na deling
monday	dag van de week (waarde is 1) in sommige EasyLanguage functies
newline	wordt gebruikt in print en fileappend om nieuwe regel te forceren
next	"buy next bar"
numeric...	"input: xArg(numeric);" geeft aan dat het een numerieke variabele is
o	afkorting van open
of	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
on	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
over	"crosses over" synoniem voor "crosses above"
place	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
pob	synoniem voor limit
point(s)	gebruik bij limiet om minimale koersstapjes van een aandeel aan te geven (meestal 0.01)
printer	zie Print functie
red	kleur (waarde is 6) in sommige EasyLanguage functies
saturday	dag van de week (waarde is 6) in sommige EasyLanguage functies
share(s)	"buy 5 shares;"
stop	"buy at 65 stop;" stop loss order: koopt op 65 of hoger
string...	"input: xArg1(string);" definieerd argument als string
t	afkorting van time
than	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
the	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien

this	"buy this bar on close;"
thursday	dag van de week (waarde is 4) in sommige EasyLanguage functies
ticks	synoniem met points
today	synoniem met "this bar"
tomorrow	synoniem met "next bar"
tool_xxx	waarbij xxx 'n kleur is. Synoniem met xxx zonder tool_
total	synoniem met all
true	in VestiCode gelijk aan waarde 1
truefalse...	"input: xArg1(truefalse);" definiëert een boolean variabele (is value in VestiCode)
tuesday	dag van de week (waarde is 2) in sommige EasyLanguage functies
under	"crosses under" synoniem met "crosses below"
v	afkorting van volume
var(s)	"var: xValue1(0);" gebruikt om variabele te definiëren
variable(s)	synoniem met var
was	zgn. skipwoord dat geen enkele betekenis heeft en als commentaar wordt gezien
wednesday	dag van de week (waarde is 3) in sommige EasyLanguage functies
white	kleur (waarde is 8) in sommige EasyLanguage functies
yellow	kleur (waarde is 7) in sommige EasyLanguage functies
yesterday	"close yesterday" is gelijk aan "close[1]"

1.10 Overzicht beschikbare functies

In dit overzicht worden alle VestiCode functies beschreven.

U kunt daarbij gebruik maken van een alfabetische lijst van alle functies om de gewenste functie op te zoeken.

Als u niet precies weet hoe de functie heet, dan kunt u gebruik maken van de lijst per categorie .

1.10.1 Overzicht per categorie

In deze lijst worden de verschillende VestiCode functies gegroepeerd per categorie.

Alle namen die met een kleine letter v beginnen, zoals vCrosses, zijn functies die alleen in Vestics beschikbaar zijn. Alle andere functies zijn standaard EasyLanguage functies die ook beschikbaar zijn in TradeStation e.d.

De functies zijn opgedeeld in een aantal categorieën. Omdat de categorieën tamelijk algemeen zijn, kan dezelfde functie ingedeeld zijn in meerdere categorieën. Dit zijn de beschikbare categorieën...

Bar gerelateerde functies geven informatie over huidige bar, aantal bars sinds..., enz.
Bestandsfuncties worden gebruik om uitvoer te realiseren
Datum en tijd functies maken het werken met datum en tijd gemakkelijker
Indicatorfuncties zijn functies die een indicator waarde berekenen
Informatiefuncties geven informatie over vaste informatie zoals fondsen, beurzen, enz.
Koersfuncties worden gebruikt om afgeleide gegevens van de koers te berekenen
Numerieke functies doen allerlei bewerkingen met getallen
Positiefuncties geven informatie over de trading positie
Performancefuncties geven informatie over het resultaat van trading
Pseudofuncties zijn helemaal geen functies maar dataseries zoals Close e.d.
Tekstfuncties helpen bij het werken met teksten

Trading- en signaleringsfuncties worden gebruikt om signalen en orders te genereren

1.10.1.1 Bar gerelateerde functies

Dit overzicht bevat alle functies die iets met barnummers doen.

AvgBarsLosTrade	Berekent gemiddelde aantal bars per slechte trade
AvgBarsWinTrade	Berekent gemiddelde aantal bars per goede trade
BarInterval	Geeft aantal minuten per bar
BarsSinceEntry	Geeft aan hoeveel bars geleden een bepaalde positie geopend is
BarsSinceExit	Geeft aan hoeveel bars geleden een bepaalde positie gesloten is
CurrentBar	Geeft het huidige bar nummer, tellend vanaf 1
vLastBar	Geeft het barnummer van de laatste bar in de koersdata
LastBarOnChart	Geeft aan of dit de laatste bar in de grafiek is
Sess1FirstBarTime	Eindtijd van de eerste bar in de 1ste of enige beurssessie
Sess2FirstBarTime	Eindtijd 1ste bar van de 2de beurssessie
TotalBarsLosTrades	Totale duur (in bars) van alle verliesgevende trades
TotalBarsWinTrades	Totale duur (in bars) van alle winnende trades

1.10.1.2 Bestandsfuncties

Deze lijst bevat alle functies die iets doen met bestanden of in en uitvoer naar het scherm

Alert	Stuurt boodschap naar beeldscherm of SMS
FileAppend	Schrijf tekstregel naar tekstbestand
FileDelete	Bestand verwijderen
PlaySound	Speel een geluidsfragment af via de luidsprekers van de computer
Plot1-4	Plot een lijn op de grafiek
Print	Tekstregel naar scherm, printer of bestand zenden

1.10.1.3 Datum en tijd functies

In dit overzicht staan alle functies die op de een of andere manier iets met datum of tijd doen.

BarInterval	Geeft aantal minuten per bar
CalcTime	Maakt het mogelijk om met een tijd te rekenen
CurrentDate	Geeft de huidige datum (op basis van de PC klok)
CurrentTime	geeft de huidige tijd (op basis van de PC klok)
Date	(*pseudo functie*) geeft datum huidige bar
DateToJulian	Converteert een datum naar een dagnummer sinds 1 januari 1900
DayOfMonth	Geeft het dagnummer uit een datum (DD van YYMMDD)
DayOfWeek	Geeft dag van de week voor een datum
DaysToExpiration	Berekent het aantal dagen tot aan de expiratie
EntryDate	Datum waarop een bepaalde positie geopend werd
EntryTime	Tijdstip waarop een bepaalde positie geopend werd
ExitDate	Datum waarop een bepaalde positie gesloten werd
ExitTime	Tijdstip waarop een bepaalde positie gesloten werd
FindBar	Geeft barnummer dat correspondeert met een datum en tijd
JulianToDate	Converteert een dagnummer naar een datum
LastCalcDate	Geeft datum van de laatst beschikbare bar in de koersdata
LastCalcJDate	Geeft dagnummer van de laatst beschikbare bar in de koersdata
LastCalcMMTime	Geeft eindtijd van de laatst beschikbare bar in de koersdata
LastCalcTime	Geeft eindtijd van de laatst beschikbare bar in de koersdata

LastHour	Geeft true indien in het laatste uur van sessie-1 van de beurs
MinutesToTime	Converteert een tijd naar minuten sinds middernacht
Month	Geeft de maand (1-12) uit een datum
Next3rdFriday	Aantal dagen tot volgende 3de vrijdag van de maand
Sess1EndTime	Eindtijd van de 1ste of enige beurssessie
Sess1FirstBarTime	Eindtijd van de eerste bar in de 1ste of enige beurssessie
Sess1StartTime	Begintijd van de 1ste of enige beurssessie
Sess2EndTime	Eindtijd van de 2de beurssessie
Sess2FirstBarTime	Eindtijd 1ste bar van de 2de beurssessie
Sess2StartTime	Begin tijd 2de beurs sessie
Time	(*pseudo functie*) geeft tijd huidige bar
Year	Geeft het jaartal van een datum

1.10.1.4 Indicatorfuncties

In deze lijst staan alle functies die gebruikt worden om indicatoren te berekenen.

De meeste van deze functies zijn hulpfuncties die gebruikt worden in Vestics indicatoren. Er is momenteel nog geen documentatie over deze indicator functies. Voor de preciese werking van de indicatoren verwijzen we naar de vele publicaties die hierover zowel in boekvorm als op het internet beschikbaar zijn. Voor de argumenten van elke indicatorfunctie verwijzen we naar de bronkode van de betreffende functie. Het ligt in de bedoeling om in de toekomst wel informatie over deze routines te verstrekken.

AccumDist	Accumulation / Distribution waarde
AccumSwingIndex	geaccumuleerde SwingIndex
ADX	ADX
AvgTrueRange	Average True Range
BollingerBand	Bollinger Band
CCI	Commodity Channel Index (CCI) indicator
ChaikinOsc	Chaikin Oscilator
CSI	Commodity Selection Index (CSI) indicator
DMI	Directional Movement Indicator (DMI)
DMIMinus	berekent dalende component van de DMI
DMIPlus	berekent de stijgende component van de DMI
EaseOfMovement	berekent de EaseOfMovement indicator
FastD	berekent de FastD waarde van de Stochastics indicator
FastK	berekent de FastK waarde van de Stochastics indicator
HPI	Herrick Payoff Index (HPI) indicator
MACD	MACD indicator
MassIndex	MassIndex
Momentum	Momentum indicator
OBV	On Balance Volume indicator
Parabolic	berekent de Parabolic (SAR) waarde
PercentR	Percent-R indicator
PriceVolTrend	Price Volume Trend indicator
RateOfChange	Rate of Change (ROC) indicator
RSI	Relative Strength Indicator (RSI)
SlowD	berekent de SlowD waarde van de Stochastics indicator
SlowK	berekent de SlowK waarde van de Stochastics indicator
SwingHigh	berekent de Swing High waarde
SwingIndex	berekent de Swing Index
SwingLow	berekent de Swing Low waarde
UltimateOsc	berekent de Ultimate Indicator
VolumeOsc	Volume Oscilator

VolumeROC	Volume Rate Of Change indicator
ZigZag	berekent de Zig Zag

1.10.1.5 Informatiefuncties

Deze lijst bevat alle functies die informatie verstrekken die los staat van de trades en de performance.

BarInterval	Geeft aantal minuten per bar
CurrentBar	Geeft het huidige bar nummer, tellend vanaf 1
CurrentDate	Geeft de huidige datum (op basis van de PC klok)
CurrentTime	geeft de huidige tijd (op basis van de PC klok)
Date	(*pseudo functie*) geeft datum huidige bar
DownTicks	(*pseudo functie*) geeft aantal tikken omlaag in huidige bar (alleen bij intradag koersen)
vLastBar	Geeft true of false al naar gelang dit wel of niet de laatste bar is in de koersdata
Open	(*pseudo functie*) geeft openingskoers huidige bar
OpenInt	(*pseudo functie*) geeft open interest huidige bar
Plot	Opvragen huidige waarde van een plot series
Random	Geeft een willekeurig getal
Sess1EndTime	Eindtijd van de 1ste of enige beurssessie
Sess1FirstBarTime	Eindtijd van de eerste bar in de 1ste of enige beurssessie
Sess1StartTime	Begintijd van de 1ste of enige beurssessie
Sess2EndTime	Eindtijd van de 2de beurssessie
Sess2FirstBarTime	Eindtijd 1ste bar van de 2de beurssessie
Sess2StartTime	Begin tijd 2de beurs sessie
SymbolName	Geeft de Fondsnaam van het huidige (data1) fonds
SymbolNumber	Geeft het Fondsnummer van het huidige (data1) fonds
SymbolRoot	Geeft de naam van het onderliggende fonds van een optie of future
Ticks	(*pseudo functie*) geeft aantal tikken in huidige bar (alleen bij intradag koersen)
Time	(*pseudo functie*) geeft tijd huidige bar
UpTicks	(*pseudo functie*) geeft aantal tikken omhoog in huidige bar (alleen bij intradag koersen)
Volume	(*pseudo functie*) geeft totale omzet huidige bar

1.10.1.6 Koersfuncties

Deze lijst bevat alle functies die gebruikt kunnen worden om allerlei afgeleiden van de koers te berekenen.

Average	het gemiddelde van een series over *n* Bars
AvgPrice	de gemiddelde koers voor vandaag
CloseD	de dag-slotkoers van enkele dagen geleden
CloseM	de maand-slotkoers van enkele maanden geleden
CloseW	de week-slotkoers van enkele weken geleden
Correlation	de correlatie tussen 2 series
Cum	de som van een data series
HighD	de hoogste koers van een bepaalde dag
Highest	de hoogste waarde in een series
HighestBar	het barnummer van de hoogste waarde in een series
HighM	de hoogste koers van een maand
HighW	de hoogste koers van een week
Leader	vergelijkt middenkoers met hoogste en laagste koers vorige bar

LowD	de laagste koers van een bepaalde dag
Lowest	de laagste waarde in een series
LowestBar	het barnummer van de laagste waarde in een series
LowM	de laagste koers van een maand
LowW	de laagste koers van een week
MedianPrice	berekent de middenkoers van een bar
MidPoint	berekent het gemiddelde van de hoogste en de laagste koers
OpenD	openingskoers van een bepaalde dag
OpenM	openingskoers van een bepaalde maand
OpenW	openingskoers van een bepaalde week
Range	hoog-laag voor huidige bar
TrueHigh	geeft de hoogste van Close[1] en High[0]
TrueLow	geeft de laagste van Close[1] en Low[0]
TrueRange	berekent de True Range waarde (= TrueHigh-TrueLow)
TypicalPrice	berekent de Typical Price (= (High+Low+Close)/3)

1.10.1.7 Logische functies

Deze lijst bevat alle logische functies.

IFF	geeft één van twee waarden afhankelijk van een conditie
LRO	minst recente (oudste) keer dat een conditie waar is
MRO	meest recente (jongste) keer dat een conditie waar is

1.10.1.8 Numerieke functies

In dit overzicht staan alle functies de werken met getallen

AbsValue	Geeft getal als positieve waarde
ArcTangent	Berekent tangens van een hoek van *n* graden
AvgList	Berekent gemiddelde van een aantal getallen
Ceiling	Geeft getal afgerond naar beneden
CoSine	Berekent cosinus van een hoek van *n* graden
CoTangent	Berekent cotanges van een hoek van *n* graden
ExpValue	Berekent de waarde *e* (basis natuurlijke logaritme) tot de opgegeven macht.
Floor	Geeft getal afgerond naar beneden
FracPortion	Geeft het gedeelte van een getal dat achter de comma staat
IntPortion	Geeft het gedeelte van een getal dat vóór de comma staat
LinearRegAngle	Bereken de stijgingshoek van een lineaire regressie door de data
LinearRegSlope	Bereken de stijging per bar van een lineaire regressie door de data
LinearRegValue	Bereken de waarde van een bepaalde bar op basis van lineaire regressie
Log	Berekent de natuurlijke logaritme van een getal
MaxList	Geeft het grootste getal uit een reeks van getallen
MaxList2	Geeft het 2de grootste getal uit een reeks van getallen
Median	Zoekt de Median (middelste) waarde van een series
MinList	Geeft het laagste getal uit een reeks van getallen
MinList2	Geeft het 2de laagste getal uit een reeks van getallen
Mod	Geeft de rest na deling van 2 getallen
Neg	Maakt een getal negatief
NthMaxList	Geeft het zoveelste hoogste getal uit een reeks getallen

NthMinList	Geeft het zoveelste laagste getal uit een reeks getallen
NumToStr	Converteer getal naar tekst
Pos	Maakt een getal positief
Power	Berekent de waarde van en getal verheven tot een bepaalde macht
Random	Geeft een willekeurig getal
Round	Rondt een getal af op het gewenste aantal decimalen
Sign	Geeft aan welk teken een getal heeft
Sine	Berekent de sinus van een hoek
Square	Berekent het kwadraat van een getal
SquareRoot	Berekent de wortel van een getal
StdDev	bereken de standaard deviatie van een series
StrToNum	Converteerd een tekst naar een getalswaarde
Tangent	Berekent de tanges van een hoek

1.10.1.9 Optie functies

Deze lijst bevat alle functies die optieberekeningen doen.

Call	geeft true of false al naar gelang het huidige fonds een Call optie is
Delta	berekent de Delta van een optie
Gamma	berekent de Gamma van een optie
Put	geeft true of false al naar gelang het huidige fonds een Put optie is
Theta	berekent de Theta van een optie
Vega	berekent de Vega van een optie
Volatility	berekent de volatility op basis van de True Range
zImpliedVolatility	berekent de implied volatility

1.10.1.10 Positiefuncties

In dit overzicht staan alle functies die informatie geven over een bepaalde positie c.q. een bepaalde trade.

AvgEntryPrice	Berekent gemiddelde aankoopprijs
BarsSinceEntry	Geeft aan hoeveel bars geleden een bepaalde positie geopend is
BarsSinceExit	Geeft aan hoeveel bars geleden een bepaalde positie gesloten is
CurrentContracts	Geeft het aantal aandelen of contracten in de huidige positie
CurrentEntries	Geeft aan hoeveel entry-signalen er zijn geweest in de huidige positie
EntryDate	Datum waarop een bepaalde positie geopend werd
EntryPrice	Geeft de (gemiddelde) instapprijs voor een bepaalde positie
EntryTime	Tijdstip waarop een bepaalde positie geopend werd
ExitDate	Datum waarop een bepaalde positie gesloten werd
ExitPrice	Geeft de (gemiddelde) uitstapprijs voor een bepaalde positie
ExitTime	Tijdstip waarop een bepaalde positie gesloten werd
MarketPosition	Geeft aan of de huidige positie long, neutraal of short is
MaxContracts	Geeft het maximale aantal aandelen of contracten gedurende een bepaalde trade
MaxContractsHeld	Geeft het maximale aantal aandelen of contracten gedurende de huidige trade
MaxEntries	Geeft het maximale aantal entries gedurende een bepaalde trade
MaxGain	Geeft de hoogste winst tot nu toe in de huidige trade
MaxLoss	Geeft het hoogste verlies tot nu toe in de huidige trade
MaxPositionLoss	Geeft het hoogste verlies gedurende het verloop van een trade
MaxPositionProfit	Geeft de hoogste winst gedurende het verloop van een trade
OpenPositionProfit	Geeft de winst in de huidige positie
PositionProfit	Geeft de winst van een bepaalde positie

1.10.1.11 Performancefuncties

In dit overzicht staan alle functies die informatie geven over de totale trading simulatie.

AvgBarsLosTrade	Berekent gemiddelde aantal bars per slechte trade
AvgBarsWinTrade	Berekent gemiddelde aantal bars per goede trade
DailyLosers	Berekent het aantal verliesgevende trades op een dag
DailyWinners	Berekent het aantal winstgevende trades op een dag
EntriesToday	Berekent het aantal entries (nieuwe posities) van vandaag
ExitsToday	Berekent het aantal exits van vandaag
GrossLoss	Totaal van alle verliesgevende trades
GrossProfit	Totaal van alle winstgevende trades
LargestLosTrade	Grootste verliesgevende trade
LargestWinTrade	Grootste winstgevende trade
MaxConsecLosers	Geeft het hoogste aantal opeenvolgende verliesgevende trades
MaxConsecWinners	Geeft het hoogste aantal opeenvolgende winnende trades
NetProfit	Geeft de totale winst van alle trades
NumLosTrades	Geeft het aantal verliesgevende trades
NumWinTrades	Geeft het aantal winnende trades
TotalBarsLosTrades	Totale duur (in bars) van alle verliesgevende trades
TotalBarsWinTrades	Totale duur (in bars) van alle winnende trades
TotalTrades	Totaal aantal trades

1.10.1.12 Pseudofuncties

Onder pseudo functies verstaan we functies uit de EasyLanguage literatuur die helemaal geen functies zijn, maar soms als zodanig worden voorgesteld, c.q. functies lijken te zijn.

Als voorbeeld noemen we Date, die staat voor de datum van de huidige bar, maar die helemaal niet als een functie benaderd wordt, maar een series variabele is. Voor het ophalen van de datum huidige bar maakt dat geen verschil, maar omdat het een series variabele is, mag u niet Date() gebruiken want de haakjes duiden op een functie referentie. Aan de andere kant, omdat het een series variabele is, zou u ook Date[0] mogen schrijven, of zelfs Date[5] om de datum van 5 bars geleden op te vragen.

De volgende pseudo functies zijn in werkelijkheid data series en moeten dus als data series benaderd worden...

- Close
- Date
- DownTicks
- High
- Low
- Open
- OpenInt
- Ticks
- Time
- UpTicks

1.10.1.13 Series functies

In dit overzicht staan alle functies die basis bewerkingen met dataseries uitvoeren.

De meeste van deze functies zijn hulpfuncties die gebruikt worden in Vestics indicatoren. Er is momenteel nog geen documentatie over deze indicator functies. Voor de preciese werking van de indicatoren verwijzen we naar de vele publicaties die hierover zowel in boekvorm als op het internet

beschikbaar zijn. Voor de argumenten van elke indicatorfunctie verwijzen we naar de bronkode van de betreffende functie. Het ligt in de bedoeling om in de toekomst wel informatie over deze routines te verstrekken.

Average	het gemiddelde van een series over *n* Bars
Cum	de som van een data series
Highest	de hoogste waarde in een series
HighestBar	het barnummer van de hoogste waarde in een series
LinearRegAngle	Bereken de stijgingshoek van een lineaire regressie door de data
LinearRegSlope	Bereken de stijging per bar van een lineaire regressie door de data
LinearRegValue	Bereken de waarde van een bepaalde bar op basis van lineaire regressie
Lowest	de laagste waarde in een series
LowestBar	het barnummer van de laagste waarde in een series
Median	Zoekt de Median (middelste) waarde van een series
NthHighest	zoekt de Nde hoogste waarde uit een series
NthHighestBar	geeft het barnummer van de Nde hoogste waarde uit een series
NthLowest	zoekt de Nde laagste waarde uit een series
NthLowestBar	geeft het barnummer van de Nde laagste waarde uit een series
StdDev	bereken de standaard deviatie van een series
Summation	bereken de sum van alle waardes in een serie
TimeSeriesForecast	berekent een lineaire regressie waarde
Volatility	berekent de volatility op basis van de True Range
WAverage	berekent een gewogen gemiddelde
XAverage	berekent een exponentieel gemiddelde

1.10.1.14 Tekstfuncties

Dit overzicht bevat alle functies die iets met tekst doen

FileAppend	Schrijf tekstregel naar tekstbestand
InStr	Zoekt een tekst op in een andere tekst
LeftStr	Geeft het linker gedeelte van een tekst
MidStr	Geeft een aantal tekens uit het midden van een tekst
NumToStr	Converteer getal naar tekst
Print	Tekstregel naar scherm, printer of bestand zenden
RightStr	Geeft de laatste tekens van een tekst
StrLen	Geeft het aantal tekens plus spaties in een tekst
StrToNum	Converteerd een tekst naar een getalswaarde
UpperStr	Geeft een tekst waarbij alle kleine letters hoofdletters worden gemaakt

1.10.1.15 Trading- en signaleringfuncties

Dit overzicht bevat alle functies die iets te maken hebben met het starten en stoppen van trading posities.

Alert	Stuurt boodschap naar beeldscherm of SMS
vCrosses	Test voor kruising van twee series
vCrossesAbove	Test of een series boven een andere series stijgt
vCrossesBelow	Test of een series onder een andere series daalt
CurrentEntries	Geeft aan hoeveel entry-signalen er zijn geweest in de huidige positie
vEnterLong	Plaats kooporder om long positie te openen
vEnterShort	Plaats verkooporder om short positie te openen
vExitLong	Sluit (eventuele) long positie geheel of gedeeltelijk
vExitShort	Sluit (eventuele) short positie geheel of gedeeltelijk

PlaySound	Speel een geluidsfragment af via de luidsprekers van de computer

1.10.2 Overzicht VestiCode functies

Om conflicten te voorkomen met toekomstige uitbreidingen van EasyLanguage beginnen alle functienamen van VestiCode functies altijd met een v. Deze v wordt klein geschreven, terwijl de echte naam van de functie met een hoofdletter begint en daardoor beter opvalt.

C

vCrosses	Test voor kruising van twee series
vCrossesAbove	Test of een series boven een andere series stijgt
vCrossesBelow	Test of een series onder een andere series daalt

E

vEnterLong	Plaats kooporder voor long positie
vEnterShort	Sluit (eventuele) long positie geheel of gedeeltelijk
vExitLong	Plaats verkooporder voor short positie
vExitShort	Sluit (eventuele) short positie geheel of gedeeltelijk

L

vLastBar	Geeft true of false al naar gelang dit wel of niet de laatste bar is in de koersdata

1.10.3 Alphabetische Lijst functies

In deze lijst worden de verschillende VestiCode functies alfabetisch gegroepeerd.

In de lijst komen 2 soorten functies voor...
- Alle namen die met een kleine letter v beginnen, zoals vCrosses, zijn functies die alleen in Vestics beschikbaar zijn.
- Alle andere functies zijn standaard EasyLanguage functies die ook beschikbaar zijn in TradeStation e.d.

De VestiCode worden gesorteerd op de 2de letter. De v wordt dus niet meegeteld en vLastBar staat dus onder de L.

A

AbsValue	Geeft een getal terug als een positieve waarde
Alert	Stuurt een boodschap naar het beeldscherm of SMS
ArcTangent	Berekent tangens van een hoek van *n* graden
Average	Berekent het gemiddelde van een series over *n* bars
AvgBarsLosTrade	Berekent gemiddelde aantal bars per slechte trade
AvgBarsWinTrade	Berekent gemiddelde aantal bars per goede trade
AvgEntryPrice	Berekent gemiddelde aankoopprijs
AvgList	Berekent gemiddelde van een aantal getallen

B

BarInterval	Geeft aantal minuten per bar
BarsSinceEntry	Geeft aan hoeveel bars geleden een bepaalde positie geopend is
BarsSinceExit	Geeft aan hoeveel bars geleden een bepaalde positie gesloten is

C

Ceiling	Geeft getal afgerond naar beneden
Close	(*pseudo functie*) geeft slotkoers huidige bar
CoSine	Berekent cosinus van een hoek van *n* graden

CoTangent	Berekent cotanges van een hoek van *n* graden
vCrosses	Test voor kruising van twee series
vCrossesAbove	Test of een series boven een andere series stijgt
vCrossesBelow	Test of een series onder een andere series daalt
CurrentBar	Geeft het huidige bar nummer, tellend vanaf 1
CurrentContracts	Geeft het aantal aandelen of contracten in de huidige positie
CurrentDate	Geeft de huidige datum (op basis van de PC klok)
CurrentEntries	Geeft aan hoeveel entry-signalen er zijn geweest in de huidige positie
CurrentTime	geeft de huidige tijd (op basis van de PC klok)

D

Date	(*pseudo functie*) geeft datum huidige bar
DateToJulian	Converteert een datum naar een dagnummer sinds 1 januari 1900
DayOfMonth	Geeft het dagnummer uit een datum (DD van YYMMDD)
DayOfWeek	Geeft dag van de week voor een datum
DownTicks	(*pseudo functie*) geeft aantal tikken omlaag in huidige bar (alleen bij intradag koersen)

E

vEnterLong	Plaats kooporder voor long positie
vEnterShort	Sluit (eventuele) long positie geheel of gedeeltelijk
EntryDate	Datum waarop een bepaalde positie geopend werd
EntryPrice	Geeft de (gemiddelde) instapprijs voor een bepaalde positie
EntryTime	Tijdstip waarop een bepaalde positie geopend werd
ExitDate	Datum waarop een bepaalde positie gesloten werd
vExitLong	Plaats verkooporder voor short positie
ExitPrice	Geeft de (gemiddelde) uitstapprijs voor een bepaalde positie
vExitShort	Sluit (eventuele) short positie geheel of gedeeltelijk
ExitTime	Tijdstip waarop een bepaalde positie gesloten werd
ExpValue	Berekent de waarde *e* (basis natuurlijke logaritme) tot de opgegeven macht.

F

FileAppend	Schrijf tekstregel naar tekstbestand
FileDelete	Bestand verwijderen
Floor	Geeft getal afgerond naar beneden
FracPortion	Geeft het gedeelte van een getal dat achter de comma staat

G

GrossLoss	Totaal van alle verliesgevende trades
GrossProfit	Totaal van alle winstgevende trades
InStr	Zoekt een tekst op in een andere tekst
IntPortion	Geeft het gedeelte van een getal dat vóór de comma staat

J

JulianToDate	Converteert een dagnummer naar een datum

L

LargestLosTrade	Grootste verliesgevende trade
LargestWinTrade	Grootste winstgevende trade
vLastBar	Geeft true of false al naar gelang dit wel of niet de laatste bar is in de koersdata
LastCalcJDate	Geeft dagnummer van de laatst beschikbare bar in de koersdata
LastCalcMMTime	Geeft eindtijd van de laatst beschikbare bar in de koersdata
LeftStr	Geeft het linker gedeelte van een tekst
Log	Berekent de natuurlijke logaritme van een getal

M

MarketPosition	Geeft aan of de huidige positie long, neutraal of short is
MaxConsecLosers	Geeft het hoogste aantal opeenvolgende verliesgevende trades
MaxConsecWinners	Geeft het hoogste aantal opeenvolgende winnende trades
MaxContracts	Geeft het maximale aantal aandelen of contracten gedurende een bepaalde trade
MaxContractsHeld	Geeft het maximale aantal aandelen of contracten gedurende de huidige trade
MaxEntries	Geeft het maximale aantal entries gedurende een bepaalde trade
MaxGain	Geeft de hoogste winst tot nu toe in de huidige trade
MaxList	Geeft het grootste getal uit een reeks van getallen
MaxList2	Geeft het 2de grootste getal uit een reeks van getallen
MaxLoss	Geeft het hoogste verlies tot nu toe in de huidige trade
MaxPositionLoss	Geeft het hoogste verlies gedurende het verloop van een trade
MaxPositionProfit	Geeft de hoogste winst gedurende het verloop van een trade
MidStr	Geeft een aantal tekens uit het midden van een tekst
MinList	Geeft het laagste getal uit een reeks van getallen
MinList2	Geeft het 2de laagste getal uit een reeks van getallen
Mod	Geeft de rest na deling van 2 getallen
Month	Geeft de maand (1-12) uit een datum

N

Neg	Maakt een getal negatief
NetProfit	Geeft de totale winst van alle trades
NthMaxList	Geeft het zoveelste hoogste getal uit een reeks getallen
NthMinList	Geeft het zoveelste laagste getal uit een reeks getallen
NumLosTrades	Geeft het aantal verliesgevende trades
NumToStr	Converteer getal naar tekst
NumWinTrades	Geeft het aantal winnende trades

O

Open	(*pseudo functie*) geeft openingskoers huidige bar
OpenInt	(*pseudo functie*) geeft open interest huidige bar
OpenPositionProfit	Geeft de winst in de huidige positie

P

PlaySound	Speel een geluidsfragment af via de luidsprekers van de computer
Plot	Opvragen huidige waarde van een plot series
Plot1-4	Plot een lijn op de grafiek
Pos	Maakt een getal positief
PositionProfit	Geeft de winst van een bepaalde positie
Power	Berekent de waarde van en getal verheven tot een bepaalde macht
Print	Tekstregel naar scherm, printer of bestand zenden

R

Random	Geeft een willekeurig getal
RightStr	Geeft de laatste tekens van een tekst
Round	Rondt een getal af op het gewenste aantal decimalen

S

Sess1EndTime	Eindtijd van de 1ste of enige beurssessie
Sess1FirstBarTime	Eindtijd van de eerste bar in de 1ste of enige beurssessie
Sess1StartTime	Begintijd van de 1ste of enige beurssessie
Sess2EndTime	Eindtijd van de 2de beurssessie
Sess2FirstBarTime	Eindtijd 1ste bar van de 2de beurssessie
Sess2StartTime	Begin tijd 2de beurs sessie
Sign	Geeft aan welk teken een getal heeft
Sine	Berekent de sinus van een hoek
Square	Berekent het kwadraat van een getal

SquareRoot	Berekent de wortel van een getal
StrLen	Geeft het aantal tekens plus spaties in een tekst
StrToNum	Converteerd een tekst naar een getalswaarde
SymbolName	Geeft de Fondsnaam van het huidige (data1) fonds
SymbolNumber	Geeft het Fondsnummer van het huidige (data1) fonds
SymbolRoot	Geeft de naam van het onderliggende fonds van een optie of future

T

Tangent	Berekent de tanges van een hoek
Ticks	(*pseudo functie*) geeft aantal tikken in huidige bar (alleen bij intradag koersen)
Time	(*pseudo functie*) geeft tijd huidige bar
TotalBarsLosTrades	Totale duur (in bars) van alle verliesgevende trades
TotalBarsWinTrades	Totale duur (in bars) van alle winnende trades
TotalTrades	Totaal aantal trades

U

UpperStr	Geeft een tekst waarbij alle kleine letters hoofdletters worden gemaakt
UpTicks	(*pseudo functie*) geeft aantal tikken omhoog in huidige bar (alleen bij intradag koersen)

V

vXxxxxx	Alle namen die met kleine v beginnen staan gesorteerd op de 2de letter.
Volume	(*pseudo functie*) geeft totale omzet huidige bar

Y

Year	Geeft het jaartal van een datum

1.10.3.1 AbsValue

Deze functie maakt van elk getal een positief getal.

Deze functie haalt als het ware het min-teken weg bij een negatief getal. Positieve getallen blijven onveranderd.

syntax:

```
value function AbsValue( value xGetal);
```

voorbeelden:

```
xResultaat  := AbsValue(xGetal);
xResultaat  := Pos(xGetal);
```

De functies AbsValue en Pos doen exact hetzelfde.

zie ook:

Numerieke functies

1.10.3.2 AccumDist

Deze functie berekent de Accumulation/Distribution Index.

Deze index is een variant op de bekende On Balance Volume (OBV) indicator. Waar bij de OBV indicator het hele volume van een dag wordt bij geteld of afgetrokken al naar gelang de koers die dag is gestegen of gedaald, wordt bij de Acuumulation/Distribution Index proportioneel bijgeteld of afgetrokken.

syntax:
```
value function AccumDist( value xSeries = Volume);
```

voorbeelden:
```
xAccumDist := AccumDist;
xAccumDist := AccumDist(Volume);
```
Als er geen argument wordt meegegeven wordt automatisch het Volume van Data1 gebruikt in de berekening.

zie ook:
Indicatorfuncties

1.10.3.3 AccumSwingIndex

Deze functie berekent de Accumulated Swing Index.

Deze index wordt berekent door de dagelijkse waarde van de SwingIndex bij elkaar op te tellen. Op die manier onstaat een gecummuleerde SwingIndex.

syntax:
```
value function AccumSwingIndex();
```

voorbeelden:
```
xAccumSwingIndex := AccumSwingIndex;
```

zie ook:
Indicatorfuncties

1.10.3.4 ADX

Deze functie berekent de ADX indicator.

syntax:
```
value function ADX();
```

voorbeelden:
```
xADX := ADX;
```

zie ook:
Indicatorfuncties

1.10.3.5 Alert

Deze functie genereert een Vestics Signalering op het scherm of via SMS e.d.

syntax:
```
value function Alert( string xText);
```

voorbeelden:
```
if Close<xStopLoss then Alert( 'Koers onder stoploss gezakt' );
```

Meer uitleg vindt u in het hoofdstuk Het genereren van alarmeringen .

zie ook:
Trading en Signaleringsfuncties

1.10.3.6 ArcTangent

Deze functie uit de goniometrie berekent de tangens van een hoek uitgedrukt in graden.

syntax:
```
value function ArcTangent( value xGraden);
```

voorbeelden:
```
xTangens := ArcTangent(45);
```

zie ook:
Numerieke functies

1.10.3.7 Average

Deze functie berekent het gemiddelde van een series over *n* Bars

syntax:
```
value function Average( value xSeries[], value xNumberOfBars);
```

voorbeelden:
```
xMA200 := Average(Close,200);
```

zie ook:
Koersfuncties

1.10.3.8 AvgBarsLosTrade

Deze functie geeft de gemiddelde duur (in bars) van een verliesgegevende trade.

Het aantal bars wordt altijd afgekapt op een geheel getal, dus als er 3 verliesgevende trades zijn, die in totaal 89 bars hebben geduurd, dan AvgBarLosTrade de waarde 29 opleveren en niet 29,67.

syntax:
```
value function AvgBarLosTrade();
```

voorbeelden:
```
if vLastBar then Print('Gemiddelde duur slechte trade
',AvgBarLosTrade);
```

opmerking:
Indien u het aantal bars per verliesgevende trade als een breuk wilt hebben, dan kunt u dit als volgt berekenen..
```
xBars := TotalBarsLosTrades/NumLosTrades;
```

zie ook:
Performance functies

1.10.3.9 AvgBarsWinTrade

Deze functie geeft de gemiddelde duur (in bars) van een winnende trade.

Het aantal bars wordt altijd afgekapt op een geheel getal, dus als er 3 winnende trades zijn, die in totaal 89 bars hebben geduurd, dan AvgBarWinTrade de waarde 29 opleveren en niet 29,67.

syntax:

```
value function AvgBarWinTrade();
```

voorbeelden:

```
if vLastBar then Print('Gemiddelde duur goede trade ',AvgBarWinTrade);
```

opmerking:

Indien u het aantal bars per winnende trade als een breuk wilt hebben, dan kunt u dit als volgt berekenen..

```
xBars := TotalBarsWinTrades/NumWinTrades;
```

zie ook:

Performance functies

1.10.3.10 AvgEntryPrice

Deze functie geeft de gemiddelde aankoopprijs van de huidige positie.

Als er geen positie is, wordt de waarde 0 gegeven.

syntax:

```
value function AvgEntryPrice();
```

voorbeelden:

```
xAankoopPrijs := AvgEntryPrice;
```

zie ook:

Positie functies

1.10.3.11 AvgList

Deze functie berekent het gemiddelde van maximaal 25 getallen.

syntax:

```
value function AvgList( value xValue1, ... ,value xValue25);
```

voorbeelden:

```
xAvgPrice := AvgList(Open,High,Low,Close);
xAvgPrice := (Open+High+Low+Close)/ 4;
```

Beide berekeningen leveren hetzelfde resultaat op.

zie ook:

Numerieke functies

1.10.3.12 AvgPrice

Deze functie berekent de gemiddelde koers voor vandaag

syntax:
```
value function AvgPrice();
```

voorbeeld:
```
xAvgPrice := AvgPrice;

xAvgPrice := AvgList(Open,High,Low,Close);
xAvgPrice := (Open+High+Low+Close)/ 4;
```
Alle 3 de berekeningen leveren hetzelfde resultaat op.

zie ook:
Koersfuncties

1.10.3.13 AvgTrueRange

Deze functie berekent een voortschrijdend gemiddelde van de True Range.

syntax:
```
value function AvgTrueRange( value xNumberOfBars );
```

voorbeeld:
```
xAvgTrueRange := AvgTrueRange(14);
```

zie ook:
Indicatorfuncties

1.10.3.14 BarInterval

Deze functie geeft het aantal minuten in een bar bij intradag grafieken.

syntax:
```
value function BarInterval();
```

voorbeelden:
```
xMinuten := BarInterval;
```
Bij een grafiek op uurbasis levert de functie BarInterval de waarde 60 op.

zie ook:
Bar functies

1.10.3.15 BarsSinceEntry

Deze functie geeft aan hoeveel bars geleden het is dat een bepaalde positie is ingenomen.

Met deze functie kan niet alleen opgevraagt worden hoe lang geleden de huidige positie is ingenomen, maar kan dat ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.

Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:
```
value function BarsSinceEntry(value xPositieNr);
```

voorbeelden:
```
xBarsGeleden := BarsSinceEntry(0);
```

zie ook:
Positie functies
Bar functies

1.10.3.16 BarsSinceExit

Deze functie geeft aan hoeveel bars geleden het is dat een bepaalde positie is gesloten.

Via het positienummer kan men opgeven van welke positie men wil weten hoe lang geleden deze gesloten is.
Indien het positienummer 0 is dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:
```
value function BarsSinceExit(value xPositieNr);
```

voorbeelden:
```
xBarsGeleden := BarsSinceExit(0);
```

zie ook:
Positie functies
Bar functies

1.10.3.17 Ceiling

Deze functie maakt van elk getal een heel getal, altijd afgerond naar boven.

Deze functie test of een getal cijfers achter de comma heeft.
Zo niet, dan is het resultaat gelijk aan het oorspronkelijke getal.
Als er wel decimalen zijn, dan worden deze weggehaald en er wordt 1 opgeteld bij het getal.

syntax:
```
value function Ceiling( value xGetal);
```

voorbeelden:
```
xResultaat := Ceiling(xGetal);
```
Als xGetal gelijk is aan 13.45 dan is xResultaat gelijk aan 14.

zie ook:
Numerieke functies

1.10.3.18 Cosine

Deze functie uit de goniometrie berekent de cosinus van een hoek uitgedrukt in graden.

syntax:
```
value function Cosine( value xGraden);
```

voorbeelden:
```
xCosinus := Cosine(45);
```

zie ook:
Numerieke functies

1.10.3.19 Cotangent

Deze functie uit de goniometrie berekent de cotangens van een hoek uitgedrukt in graden.

syntax:
```
value function Cotangent( value xGraden);
```

voorbeelden:
```
xCotangens := Cotangent(45);
```

zie ook:
Numerieke functies

1.10.3.20 vCrosses

Deze functie test of de 2 meegegeven series waarden zich vandaag gekruisd hebben.

syntax:
```
value function vCrosses( value xSeries1[], value xSeries2[]);
```

resultaat:

0	als er geen kruising plaats gevonden heeft.
< 0	wel kruising en xSeries 1 is nu onder xSeries2 terecht gekomen
> 0	wel kruising en xSeries 2 is nu boven xSeries2 terecht gekomen

De waarde > 0 is een uitdrukking (in procenten) van de stijging van series1.
De waarde < 0 is een uitdrukking (in procenten) van de daling van series1.

waarschuwing:
Deze functie gaat niet verder terug dan 1 bar. Indien de twee series gisteren exact dezelfde waarde hadden (wat niet vaak voorkomt) en vandaag zijn de waardes ongelijk, dan resulteert dit altijd in een kruising. Het kan echter zijn dat series1 twee dagen geleden boven series2 was, gisteren gelijk, en vandaag wederom boven series2 staat. In dat geval is er geen sprake van een kruising maar eerder van een 'aanraking'. Toch zal de functie vCrossesAbove dit signaleren als een kruising.

voorbeelden:
```
if Crosses(Close,xMaKort)> 0 then begin
  { aktie }
  end;
```

Meer uitleg over de Cross-functies vindt u in het hoofdstuk Crosses above en below .

EasyLanguage:
In EasyLanguage is **crosses** gedefinieerd als een sleutelwoord dat gebruikt kan worden in combinatie met het sleutelwoord **above** of **below** in de vorm van: "`if Close crosses above xMaLang then ...`". Deze constructie wordt ook in Vestics ondersteund om bestaande EasyLanguage functies te kunnen importeren.

zie ook...
Trading en Signalering functies

1.10.3.21 vCrossesAbove

Deze functie test of de eerste meegegeven series waarde vandaag de tweede series gekruisd heeft, en nu boven de tweede series waarde ligt.

syntax:
```
value function vCrossesAbove( value xSeries1[], value xSeries2[]);
```

resultaat:

0	als er geen kruising plaats gevonden heeft.
0	wel kruising en xSeries 1 is nu onder xSeries2 terecht gekomen
> 0	wel kruising en xSeries 2 is nu boven xSeries2 terecht gekomen

De waarde >0 is een uitdrukking (in procenten) van de stijging van series1.

waarschuwing:
Deze functie gaat niet verder terug dan 1 bar. Indien de twee series gisteren exact dezelfde waarde hadden (wat niet vaak voorkomt) en vandaag is series1 boven series2, dan resulteert dit altijd in een kruising. Het kan echter zijn dat series1 twee dagen geleden boven series2 was, gisteren gelijk, en vandaag wederom boven series2 staat. In dat geval is er geen sprake van een kruising maar eerder van een 'aanraking'. Toch zal de functie vCrosses dit signaleren als een kruising.

voorbeelden:
```
if vCrossesAbove(Close,xMaKort)    then begin
    { aktie }
    end;
```

uitleg:
Omdat de waarde 0 als onwaar wordt geïnterpreteerd in condities, en elke waarde ongelijk aan 0 als waar, hoeft niet expliciet op groter dan 0 getest te worden en volstaat de bovenstaande if-conditie als test.

Meer uitleg over de Cross-functies vindt u in het hoofdstuk Crosses above en below .

EasyLanguage:
In EasyLanguage is **above** gedefinieerd als een sleutelwoord dat gebruikt kan worden in combinatie met het sleutelwoord **crosses** in de vorm van: "`if Close crosses above xMaLang then ...`". Deze constructie wordt ook in Vestics ondersteund om bestaande EasyLanguage functies te kunnen importeren.

zie ook...
Trading en Signalering functies

1.10.3.22 vCrossesBelow

Deze functie test of de eerste meegegeven series waarde vandaag de tweede series gekruisd heeft, en nu onder de tweede series waarde ligt.

syntax:

```
value function vCrossesBelow( value xSeries1[], value xSeries2[]);
```

resultaat:

0	als er geen kruising plaats gevonden heeft.
> 0	wel kruising en xSeries 1 is nu onder xSeries2 terecht gekomen
0	wel kruising en xSeries 2 is nu boven xSeries2 terecht gekomen

De waarde >0 is een uitdrukking (in procenten) van de daling van series1.

waarschuwing:

Deze functie gaat niet verder terug dan 1 bar. Indien de twee series gisteren exact dezelfde waarde hadden (wat niet vaak voorkomt) en vandaag is series1 onder series2, dan resulteert dit altijd in een kruising. Het kan echter zijn dat series1 twee dagen geleden onder series2 was, gisteren gelijk, en vandaag wederom onder series2 staat. In dat geval is er geen sprake van een kruising maar eerder van een 'aanraking'. Toch zal de functie vCrossesBelow dit signaleren als een kruising.

voorbeelden:

```
if vCrossesBelow(Close,xMaKort)    then begin
    { aktie }
    end;
```

uitleg:

Omdat de waarde 0 als onwaar wordt geïnterpreteerd in condities, en elke waarde ongelijk aan 0 als waar, hoeft niet expliciet op groter dan 0 getest te worden en volstaat de bovenstaande if-conditie als test.

Meer uitleg over de Cross-functies vindt u in het hoofdstuk Crosses above en below .

EasyLanguage:

In EasyLanguage is **below** gedefinieerd als een sleutelwoord dat gebruikt kan worden in combinatie met het sleutelwoord **crosses** in de vorm van: "`if Close crosses below xMaLang then ...`". Deze constructie wordt ook in Vestics ondersteund om bestaande EasyLanguage functies te kunnen importeren.

zie ook...

Trading en Signalering functies

1.10.3.23 CurrentBar

Deze functie geeft het huidige bar nummer vanaf de start. De eerste bar is hierbij bar nummer 1. Deze functie wordt o.a. veel gebruikt om variabelen e.d. te initialiseren bij de start van een functie.

syntax:

```
value function CurrentBar();
```

voorbeelden:

```
if CurrentBar=1  then xTeller := 0;
```

zie ook:

Bar functies

1.10.3.24 CurrentContracts

Deze functie geeft het huidige aantal aandelen of contracten dat men in positie heeft.

Als de huidige positie neutraal is, wordt de waarde 0 teruggegeven.
Bij een long positie wordt het aantal aandelen of contracten als een positief getal teruggegeven.
Bij een short positie wordt het aantal aandelen of contracten als een negatief getal teruggegeven.

syntax:

```
value function CurrentContracts();
```

voorbeelden:

```
if CurrentContracts=0 then Print('Positie is neutraal');
```

zie ook:
Positie functies

1.10.3.25 CurrentDate

Deze functie geeft de datum van vandaag op basis van de PC-klok en is dus niet afhankelijk van het huidige barnummer.

De datum is in het speciale datumformaat YYYMMDD waarin het jaartal wordt weergegeven minus 1900.
Volgens deze notatie is 31 december 1998 gelijk aan 981231, en is 31 december 2004 gelijk aan 1041231 omdat 2004-1900 gelijk is aan 104.

syntax:

```
value function CurrentDate();
```

Voorbeelden:

```
xVandaag := CurrentDate;
```

opmerking:
De datum van de huidige bar is beschikbaar via de series Date die altijd aanwezig is en meeloopt met de bars.

zie ook:
Datum en tijd functies

1.10.3.26 CurrentEntries

Deze functie geeft het aantal entries (vEnterLong of vEnterShort) in de huidige positie.

Als de huidige positie neutraal is, wordt de waarde 0 teruggegeven.

syntax:

```
value function CurrentEntries();
```

voorbeelden:

```
if CurrentEntries<3 then vEnterLong; { max. 2 keer bijkopen }
```

zie ook:
Positie functies

1.10.3.27 CurrentTime

Deze functie geeft de tijd op basis van de PC-klok en is dus niet afhankelijk van het huidige barnummer.

De tijd is in het 24-uurs formaat HHMM .

syntax:
```
value function CurrentDate();
```

Voorbeelden:
```
xVandaag := CurrentDate;
```

opmerking:
De datum van de huidige bar is beschikbaar via de Date series.

zie ook:
Datum en tijd functies

1.10.3.28 DataCompression

Deze functie geeft aan op welke interval basis een grafiek werkt.

Mogelijke resultaten zijn...
waarde 0 Tickbasis
waarde 1 Intradag
waarde 2 Dagbasis
waarde 3 Weekbasis
waarde 4 Maandbasis
waarde 5 Point & Figure grafiek

syntax:
```
value function DataCompression();
```

voorbeelden:
```
if DataCompression=3 then Print('Dit is een weekgrafiek');
```

zie ook:
Datum en tijd functies

1.10.3.29 DateToJulian

Deze functie vertaalt een datum naar het dagnummer sinds 1 januar 1900.

Door datums te converteren naar een dagnummer, wordt het mogelijk om via optellen en aftrekken allerlei berekeningen uit te voeren met datums.

syntax:
```
value function DateToJulian(value xDatum);
```

voorbeelden:

```
xDagNummer  := DateToJulian(CurrentDate);
```

zie ook:
Datum en tijd functies

1.10.3.30 DayOfMonth

Deze functie haalt het dagnummer (1-31) uit een datum.

syntax:

```
value function DayOfMonth(value  xDatum);
```

voorbeelden:

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

zie ook:
Datum en tijd functies

1.10.3.31 DayOfWeek

Deze functie bepaalt de weekdag van een datum.

De mogelijke resultaten zijn..

waarde 0	Zondag
waarde 1	Maandag
waarde 2	Dinsdag
waarde 3	Woensdag
waarde 4	Donderdag
waarde 5	Vrijdag
waarde 6	Zaterdag

syntax:

```
value function DayOfWeek(value  xDatum);
```

voorbeelden:

```
if DayOfWeek(CurrentDate)=3   then Print('Woensdag,  gehaktdag');
```

zie ook:
Datum en tijd functies

1.10.3.32 vEnterLong

Deze functie start een *long* positie of een *hause* positie. Een long positie is een positie waarbij de belegger speculeert op een stijging van de koers van het fonds dat gekocht wordt.

syntax:

```
value function vEnterLong(
    string xName='',          { standaard is geen naam }
    value xContracts=1,             { standaard is 1 contract }
    value xWhen=0,            { standaard is slotkoers huidige bar plus
    slippage }
    value xOrderType=0,             { standaard is Bestens }
```

```
    value  xLimit=0);
```

argumenten:

xName	Naam die aan deze positie wordt gehangen.
xContracts	Aantal aandelen of contracten die gekocht worden
xWhen	Code die aangeeft wanneer de positie geopend wordt
xOrderType	Code die aangeeft welk soort order wordt gebruikt (bijv. Bestens of limietorder)
xLimit	Limietprijs indien een limietorder wordt opgegeven

waarschuwing:

Een eventuele *short* (*baise*) positie wordt automatisch gesloten omdat een long positie wordt geopend.

voorbeelden:

```
if vCrossesAbove(Close,xMaLang)    then vEnterLong;
```

Meer uitleg over de order-functies vindt u in het hoofdstuk Het genereren van orders .

zie ook...

Trading en Signalering functies

1.10.3.33 vEnterShort

Deze functie start een *short* positie of een *baise* positie. Een short positie is een positie waarbij de belegger speculeert op een daling van de koers van het fonds dat gekocht wordt.

syntax:

```
value function vEnterShort(
    string xName='',          { standaard is geen naam }
    value xContracts=1,             { standaard is 1 contract }
    value xWhen=0,            { standaard is slotkoers huidige bar plus
    slippage }
    value xOrderType=0,             { standaard is Bestens }
    value xLimit=0);
```

argumenten:

xName	Naam die aan deze positie wordt gehangen.
xContracts	Aantal aandelen of contracten die verkocht worden
xWhen	Code die aangeeft wanneer de positie geopend wordt
xOrderType	Code die aangeeft welk soort order wordt gebruikt (bijv. Bestens of limietorder)
xLimit	Limietprijs indien een limietorder wordt opgegeven

waarschuwing:

Een eventuele *long* (*hause*) positie wordt automatisch gesloten omdat een short positie wordt geopend.

voorbeelden:

```
if vCrossesBelow(Close,xMaLang)    then vEnterShort;
```

Meer uitleg over de order-functies vindt u in het hoofdstuk Het genereren van orders .

zie ook...

Trading en Signalering functies

1.10.3.34 EntryDate

Deze functie geeft de datum waarop dat een bepaalde positie is ingenomen.

Met deze functie kan niet alleen opgevraagt worden op welke datum de huidige positie is ingenomen, maar kan dat ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.
Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function EntryDate(value xPositieNr);
```

voorbeelden:

```
xStartDatum := EntryDate(0);
```

zie ook:

Positie functies
Datum en tijd functies

1.10.3.35 EntryPrice

Deze functie geeft de gemiddelde aankoopprijs van een bepaalde positie.

Met deze functie kan niet alleen de gemiddelde aankoopprijs van de huidige positie opgevraagd worden, maar ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.
Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function EntryPrice(value xPositieNr);
```

voorbeelden:

```
xAankoopPrijs := EntryPrice(0);
```

zie ook:

Positie functies

1.10.3.36 EntryTime

Deze functie geeft de tijd waarop dat een bepaalde positie is ingenomen. Meestal wordt deze functie gecombineerd met EntryDate om het exacte moment van de aankoop te bepalen.

Met deze functie kan niet alleen opgevraagt worden op welk tijdstip de huidige positie is ingenomen, maar kan dat ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.

Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function EntryTime(value xPositieNr);
```

voorbeelden:

```
xStartTijd := EntryTime(0);
```

zie ook:

Positie functies
Datum en tijd functies

1.10.3.37 ExitDate

Deze functie geeft de datum waarop dat een bepaalde positie is gesloten.

Via het positienummer kan men opgeven van welke positie men wil weten op welke datum deze gesloten is.
Indien het positienummer 0 is dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function ExitDate(value xPositieNr);
```

voorbeelden:

```
xEindDatum := ExitDate(0);
```

zie ook:

Positie functies
Datum en tijd functies

1.10.3.38 vExitLong

Deze functie sluit een eventuele *long* positie of een *hause* positie.

syntax:

```
value function vExitLong(
    string xName='',          { standaard is geen naam }
    value xContracts=0,             { standaard is hele positie sluiten
    }
    value xWhen=0,            { standaard is slotkoers huidige bar plus
    slippage }
    value xOrderType=0,             { standaard is Bestens }
    value xLimit=0);
```

argumenten:

xName	Naam van de positie die gesloten wordt.
xContracts	Aantal aandelen of contracten die verkocht worden
xWhen	Code die aangeeft wanneer de positie gesloten wordt
xOrderType	Code die aangeeft welk soort order wordt gebruikt (bijv. Bestens of limietorder)

xLimit	Limietprijs indien een limietorder wordt opgegeven

waarschuwing:
Als er op dit moment geen long positie is, wordt deze order genegeerd.

voorbeelden:
```
if vCrossesBelow(Close,xMaLang)    then vExitLong;
```

Meer uitleg over de order-functies vindt u in het hoofdstuk Het genereren van orders .

zie ook...
Trading en Signalering functies

1.10.3.39 ExitPrice

Deze functie geeft de gemiddelde verkoopprijs van een bepaalde positie.

Via het positienummer kan men opgeven van welke positie men wil weten tegen welke prijs de positie gesloten is.
Indien het positienummer 0 is dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:
```
value function ExitPrice(value xPositieNr);
```

voorbeelden:
```
xVerkoopPrijs := ExitPrice(0);
```

zie ook:
Positie functies

1.10.3.40 vExitShort

Deze functie sluit een eventuele *short* positie of een *baise* positie.

syntax:
```
value function vExitShort(
    string xName='',          { standaard is geen naam }
    value xContracts=0,             { standaard is hele positie sluiten
    }
    value xWhen=0,            { standaard is slotkoers huidige bar plus
    slippage }
    value xOrderType=0,             { standaard is Bestens }
    value xLimit=0);
```

argumenten:

xName	Naam van de positie die gesloten wordt.
xContracts	Aantal aandelen of contracten die gekocht worden
xWhen	Code die aangeeft wanneer de positie gesloten wordt
xOrderType	Code die aangeeft welk soort order wordt gebruikt (bijv. Bestens of limietorder)
xLimit	Limietprijs indien een limietorder wordt opgegeven

waarschuwing:
Als er op dit moment geen short positie is, wordt deze order genegeerd.

voorbeelden:
```
if vCrossesAbove(Close,xMaLang)    then vExitShort;
```

Meer uitleg over de order-functies vindt u in het hoofdstuk Het genereren van orders .

zie ook...
Trading en Signalering functies

1.10.3.41 ExitTime

Deze functie geeft de tijd waarop dat een bepaalde positie is gesloten. Meestal wordt deze functie gecombineerd met ExitDate om het exacte moment van de sluiting te bepalen.

Via het positienummer kan men opgeven van welke positie men wil weten om hoe laat deze gesloten is.
Indien het positienummer 0 is dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:
```
value function ExitTime(value  xPositieNr);
```

voorbeelden:
```
xEindTijd  := ExitTime(0);
```

zie ook:
Positie functies
Datum en tijd functies

1.10.3.42 ExpValue

Deze functie berekent *e* tot de opgegeven macht, waarbij *e* het basisgetal is voor de natuurlijke logaritme (2.718).

syntax:
```
value function ExpValue( value xMacht);
```

voorbeelden:
```
xResultaat  := ExpValue(3);
```
Deze berekening levert e tot de macht 3 op, zijnde 20.0855.

zie ook:
Numerieke functies

1.10.3.43 FileAppend

Deze functie schrijft een regel tekst achter een bestand.

syntax:
```
value function FileAppend( string xBestand,  string xRegel);
```

Voorbeelden:
```
FileAppend('c:\Temp\Test.txt','Dit     is een regel');
```

zie ook:
Bestands functies

1.10.3.44 FileDelete

Deze functie verwijdert een bepaald bestand van de harde schijf.

syntax:
```
value function FileDelete( string xBestand);
```

Voorbeelden:
```
FileDelete('c:\Temp\Test.txt');
```

zie ook:
Bestands functies

1.10.3.45 Floor

Deze functie maakt van elk getal een heel getal, altijd afgerond naar beneden.

Deze functie test of een getal cijfers achter de comma heeft.
Zo niet, dan is het resultaat gelijk aan het oorspronkelijke getal.
Als er wel decimalen zijn, dan worden deze weggehaald.

syntax:
```
value function Floor( value xGetal);
```

voorbeelden:
```
xResultaat  := Floor(xGetal);
```
Als xGetal gelijk is aan 13.45 dan is xResultaat gelijk aan 13.

zie ook:
Numerieke functies

1.10.3.46 FracPortion

Deze functie geeft het gedeelte achter de comma (de decimalen) van een getal.

Het resultaat van deze functie heeft hetzelfde teken als het oorspronkelijke getal.
Dus FracPortion(123.456) is 0.456, FracPortion(-7.89) is gelijk aan -0.89.

syntax:
```
value function FracPortion( value xGetal);
```

voorbeelden:
```
xResultaat  := FracPortion(xGetal);
```
Als xGetal gelijk is aan -13.45 dan is xResultaat gelijk aan -0.45.

zie ook:
Numerieke functies

1.10.3.47 GrossLoss

Deze functie geeft het totale verlies van alle afgesloten verliesgevende trades.

syntax:

```
value function GrossLoss();
```

voorbeelden:

```
if vLastBar then Print('Totaal verlies alle verliezers ',GrossLoss);
```

zie ook:

Performance functies

1.10.3.48 GrossProfit

Deze functie geeft de totale winst van alle afgesloten winstgevende trades.

syntax:

```
value function GrossProfit();
```

voorbeelden:

```
if vLastBar then Print('Totale winst alle winnaars ',GrossProfit);
```

zie ook:

Performance functies

1.10.3.49 InStr

Deze functie zoekt een bepaalde tekst op binnen een te onderzoeken tekst.

Als de gezochte tekst niet voorkomt is het resultaat 0.
Als de gezochte tekst wel voorkomt, dan is het resultaat de startpositie binnen de onderzochte tekst.

syntax:

```
value function InStr( string xTeOnderzoeken, string xGezocht);
```

voorbeelden:

```
xPositie := InStr('ABCDEFGHI JKL','EFG');
```

Het resultaat is 5 omdat de gezochte tekst 'EFG' op de 5de positie in de onderzochte tekst begint.

zie ook:

Tekst functies

1.10.3.50 IntPortion

Deze functie geeft het gedeelte vóór de comma van een getal.

Het resultaat van deze functie heeft hetzelfde teken als het oorspronkelijke getal.
Dus IntPortion(123.456) is 123, IntPortion(-7.89) is gelijk aan -7.

syntax:

```
value function IntPortion( value xGetal);
```

voorbeelden:

```
xResultaat := IntPortion(xGetal);
```

Als xGetal gelijk is aan -13.45 dan is xResultaat gelijk aan -13.

zie ook:

Numerieke functies

1.10.3.51 JulianToDate

Deze functie vertaalt een dagnummer sinds 1 januari 1900 naar een datum.

Deze functie kan gebruikt worden om een datum die eerder met behulp van de functie DateToJulian omgezet is naar een dagnummer, weer terug te converteren naar een datum.

syntax:

```
value function DateToJulian(value xDatum);
```

voorbeelden:

```
xDagNummer := DateToJulian(CurrentDate);
```

zie ook:

Datum en tijd functies

1.10.3.52 LargestLosTrade

Deze functie geeft het hoogste verlies dat ooit is voorgekomen bij een van de gesloten verliesgevende trades in een trading simulatie.

Het betreft hierbij het verlies aan het einde van de trade. Het kan goed zijn dat de bewuste trade (of een andere trade) tussentijds op een nog groter verlies heeft gestaan.

syntax:

```
value function LargestLosTrade();
```

voorbeelden:

```
if vLastBar then Print('Grootste verliezer ',LargestLosTrade);
```

zie ook:

Performance functies

1.10.3.53 LargestWinTrade

Deze functie geeft de hoogste winst dat ooit is voorgekomen bij een van de gesloten winstgevende trades in een trading simulatie.

Het betreft hierbij de winst aan het einde van de trade. Het kan goed zijn dat de bewuste trade (of een andere trade) tussentijds op een nog hogere winst heeft gestaan.

syntax:

```
value function LargestWinTrade();
```

voorbeelden:

```
if vLastBar then Print('Grootste winst ',LargestWinTrade);
```

zie ook:
Performance functies

1.10.3.54 vLastBar

Deze functie geeft het barnummer van de laatst beschikbare bar in de koersdata is.
Deze functie wordt o.a. veel gebruikt om totalen te printen en andere afsluitende zaken te doen bij een simulatie.

syntax:
```
value function vLastBar();
```

waarschuwing:
Indien deze grafiek is aangesloten op een data feed, dan zal de functie vLastBar bij elke nieuwe bar die afgesloten wordt, steeds weer het nieuwe laatste barnummer geven omdat elke nieuw binnengekomen koersbar op dat moment ook weer de laatst beschikbare bar is.

voorbeelden:
```
if CurrentBar=vLastBar then Print('Totaal is ',xTotaal);
```

zie ook:
Bar functies

1.10.3.55 LastCalcJDate

Deze functie geeft het dagnummer sinds 1 januari 1900 voor de laatste bar in de huidige koersdata.

Via de functie JulianToDate kan dit dagnummer geconverteerd worden naar een datum .

Vaak zal deze datum gelijk zijn aan CurrentDate, maar dat is niet het geval als de koersdata eerder stopt dan vandaag. Bijvoorbeeld gedurende het weekend, als de laatste koers van vrijdag is, terwijl het vandaag zaterdag of zondag is.

syntax:
```
value function LastCalcJDate();
```

voorbeelden:
```
xDatum := JulianToDate(LastCalcJDate);
```

Opmerking:
In VestiCode kan gebruik gemaakt worden van de functie vLastBar om te testen of dit de laatste bar in de koersdata is.

zie ook:
Datum en tijd functies

1.10.3.56 LastCalcMMTime

Deze functie geeft de tijd voor de laatste bar in de huidige koersdata, uitgedrukt in minuten sinds middernacht.

syntax:
```
value function LastCalcMMTime();
```

voorbeelden:

```
xMinuten  := LastCalcMMTime;
```

Opmerking:

In VestiCode kan gebruik gemaakt worden van de functie vLastBar om te testen of dit de laatste bar in de koersdata is.

zie ook:

Datum en tijd functies

1.10.3.57 LeftStr

Deze functie geeft de eerste (oftewel meest linkse) *n* tekens van een tekst.

Als er minder dan *n* tekens zijn in de tekst wordt de hele tekst teruggeven.

syntax:

```
value function  LeftStr( string xTekst, value xAantal);
```

voorbeelden:

```
xLinkerDeel  := LeftStr('ABCDEFGHI   JKL',4);
```

Het resultaat is de tekst 'ABCD'.

zie ook:

Tekst functies

1.10.3.58 Log

Deze functie berekent de *e*-log van een getal, waarbij *e* het basisgetal is voor de natuurlijke logaritme (2.718).

syntax:

```
value function  Log( value xGetal);
```

voorbeelden:

```
xResultaat  := Log(3);
```

Deze berekening levert de *e*-log van 3 op, zijnde 1.099.

zie ook:

Numerieke functies

1.10.3.59 MarketPosition

Deze functie geeft aan in welke richting (long of short) een bepaalde positie is ingenomen.

Met deze functie kan niet alleen opgevraagt worden wat de huidige positie is, maar kan dat ook voor alle voorgaande posities.

Het resultaat kan zijn...

waarde 0	neutraal (kan alleen bij de huidige positie, dus als positienummer=0)
waarde +1	long positie
waarde -1	short positie

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.
Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function MarketPosition(value xPositieNr);
```

voorbeelden:

```
xHuidigePositie := MarketPosition(0);
```

zie ook:
Positie functies

1.10.3.60 MaxConsecLosers

Deze functie geeft het hoogste aantal opeenvolgende verliezergevende trades dat ooit is voorgekomen in een trading simulatie.

syntax:

```
value function MaxConsecLosers();
```

voorbeelden:

```
if vLastBar then Print('Aantal opeenvolgende verliezers
',MaxConsecLosers);
```

zie ook:
Performance functies

1.10.3.61 MaxConsecWinners

Deze functie geeft het hoogste aantal opeenvolgende winnende trades dat ooit is voorgekomen in een trading simulatie.

syntax:

```
value function MaxConsecWinners();
```

voorbeelden:

```
if vLastBar then Print('Aantal opeenvolgende winnaars
',MaxConsecWinners);
```

zie ook:
Performance functies

1.10.3.62 MaxContracts

Deze functie geeft aan wat het hoogste aantal aandelen of contracten was gedurende een bepaalde positie.

Met deze functie kan niet alleen opgevraagt worden wat het hoogste aantal contracten was gedurende de huidige positie, maar ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.

Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function MaxContracts(value xPositieNr);
```

voorbeelden:

```
xAantalAandelen := MaxContracts(0);
```

zie ook:
Positie functies

1.10.3.63 MaxContractsHeld

Deze functie geeft het hoogste aantal aandelen of contracts dat ooit is voorgekomen in een trading simulatie.

syntax:

```
value function MaxContractsHeld();
```

voorbeelden:

```
if vLastBar then Print('Grootste positie ',MaxContractsHeld);
```

zie ook:
Performance functies

1.10.3.64 MaxEntries

Deze functie geeft aan wat het hoogste aantal entries was gedurende een bepaalde positie.

Met deze functie kan niet alleen opgevraagt worden wat het hoogste aantal entries was gedurende de huidige positie, maar ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.

Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function MaxEntries(value xPositieNr);
```

voorbeelden:

```
xAantalSignalen := MaxEntries(0);
```

zie ook:
Positie functies

1.10.3.65 MaxGain

Deze functie geeft aan wat de hoogste winst is geweest tot dus ver in de huidige positie.

Deze functie kan bijv. gebruikt worden om een trailing stop loss te implementeren op basis van de hoogste winst tot dus ver.

syntax:

```
value function MaxGain();
```

voorbeelden:

```
if MaxGain> 2000 and OpenPositionProfit< 0.75*MaxGain then vExitLong;
```

Deze instructie zal er voor zorgen dat als de winst eenmaal meer dan 2000 euro is geweest, deze daarna niet meer dan 25% kan dalen.

zie ook:

Positie functies

1.10.3.66 MaxList

Deze functie zoekt het hoogste getal uit een lijst van maximaal 25 getallen.

syntax:

```
value function MaxList( value xValue1, ... ,value xValue25);
```

voorbeelden:

```
x3BarHigh := MaxList(Close,Close[1],Close[2]);
x3BarHigh := Close>>Close[ 1]>>Close[ 2];
```

Beide berekeningen leveren hetzelfde resultaat op, namelijk de hoogste Close van de afgelopen 3 bars.

zie ook:

Numerieke functies

1.10.3.67 MaxList2

Deze functie zoekt het op een na hoogste (het 2de hoogste) getal uit een lijst van maximaal 25 getallen.

syntax:

```
value function MaxList2( value xValue1, ... ,value xValue25);
```

voorbeelden:

```
x2deHoogste := MaxList2(Close,Close[1],Close[2]);
```

zie ook:

Numerieke functies

1.10.3.68 MaxLoss

Deze functie geeft aan wat het hoogste verlies is geweest tot dus ver in de huidige positie.

syntax:

```
value function MaxGain();
```

voorbeelden:

```
xMaxVerlies  := MaxGain;
```

zie ook:

Positie functies

1.10.3.69 MaxPositionLoss

Deze functie geeft aan wat het hoogste verlies was gedurende een bepaalde positie.

Met deze functie kan niet alleen opgevraagt worden wat het hoogste verlies was gedurende de huidige positie, maar ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.

Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function MaxPositionLoss(value   xPositieNr);
```

voorbeelden:

```
xHoogsteVerlies  := MaxPositionLoss(0);
xHoogsteverlies  := MaxLoss;
```

Doordat bij de MaxPositionLoss functie gevraagd wordt maar positienummer 0 (de huidige positie), zal het resultaat van beide bovenstaande instructies hetzelfde zijn.

zie ook:

Positie functies

1.10.3.70 MaxPositionProfit

Deze functie geeft aan wat de hoogste winst was gedurende een bepaalde positie.

Met deze functie kan niet alleen opgevraagt worden wat de hoogste winst was gedurende de huidige positie, maar ook voor alle voorgaande posities.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.

Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:

```
value function MaxPositionProfit(value   xPositieNr);
```

voorbeelden:

```
xHoogsteWinst  := MaxPositionProfit(0);
xHoogsteWinst  := MaxGain;
```

Doordat bij de MaxPositionProfit functie gevraagd wordt maar positienummer 0 (de huidige positie), zal het resultaat van beide bovenstaande instructies hetzelfde zijn.

zie ook:
Positie functies

1.10.3.71 MidStr

Deze functie geeft de *n* tekens van een tekst te beginnen vanaf een bepaalde startpositie.

syntax:
```
value function MidStr( string xTekst, value xStart, value xAantal);
```

voorbeelden:
```
xMiddenDeel := MidStr('ABCDEFGHI JKL',8,4);
```
Het resultaat is de tekst 'HI J'.

zie ook:
Tekst functies

1.10.3.72 MinList

Deze functie zoekt het laagste getal uit een lijst van maximaal 25 getallen.

syntax:
```
value function MinList( value xValue1, ... ,value xValue25);
```

voorbeelden:
```
x3BarLow := MinList(Close,Close[1],Close[2]);
x3BarLow := Close<<Close[1]<<Close[2];
```
Beide berekeningen leveren hetzelfde resultaat op, namelijk de laagste Close van de afgelopen 3 bars.

zie ook:
Numerieke functies

1.10.3.73 MinList2

Deze functie zoekt het op een na laagste (het 2de laagste) getal uit een lijst van maximaal 25 getallen.

syntax:
```
value function MinList2( value xValue1, ... ,value xValue25);
```

voorbeelden:
```
x2deLaagste := MinList2(Close,Close[1],Close[2]);
```

zie ook:
Numerieke functies

1.10.3.74 Mod

Deze functie deelt twee getallen op elkaar en geeft de rest van de deling terug.

De rest heeft altijd hetzelfde teken als de deler van de deling.

syntax:

```
value function Mod(value xGetal, value xDeler);
```

voorbeelden:

```
xRest := Mod(10,xDeler);
```

Als xDeler gelijk is aan 4 dan is dat 10/4 = 2 rest 2, en dus is xRest gelijk aan 2.
Als xDeler gelijk is aan -3 dan is dat 10/-3 = -4 rest -2, dus is xRest gelijk aan -2.

zie ook:

Numerieke functies

1.10.3.75 Month

Deze functie haalt het maandnummer (1-12) uit een datum.

syntax:

```
value function Month(value xDatum);
```

voorbeelden:

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

zie ook:

Datum en tijd functies

1.10.3.76 Neg

Deze functie maakt van elk getal een negatief getal.

Deze functie zet als het ware een min-teken bij een positief getal. Negatieve getallen blijven onveranderd.

syntax:

```
value function Neg(value xGetal);
```

voorbeelden:

```
xResultaat := Neg(xGetal);
xResultaat := -AbsValue(xGetal);
```

Beide instructies geven hetzelfde resultaat.

zie ook:

Numerieke functies

1.10.3.77 NetProfit

Deze functie geeft de totale winst van alle afgesloten trades (zowel winnende als verliesgevende trades).

syntax:

```
value function NetProfit();
```

voorbeelden:

```
if vLastBar then Print('Totale winst ',NetProfit);
```

zie ook:

Performance functies

1.10.3.78 NthMaxList

Deze functie zoekt het op *n* na hoogste (het *n*-de hoogste) getal uit een lijst van maximaal 25 getallen.

syntax:

```
value function NthMaxList( value xNste, value xValue1, ... ,value
xValue25);
```

voorbeelden:

```
x2deHoogste := NthMaxList(2,Close,Close[1],Close[2]);
x2deHoogste := MaxList2(Close,Close[1],Close[2]);
```

Omdat de MaxList2 functie altijd de 2de hoogste zoekt, leveren beide instructies hetzelfde getal op.

zie ook:

Numerieke functies

1.10.3.79 NthMinList

Deze functie zoekt het op *n* na laagste (het *n*-de laagste) getal uit een lijst van maximaal 25 getallen.

syntax:

```
value function NthMinList( value xNste, value xValue1, ... ,value
xValue25);
```

voorbeelden:

```
x2deLaagste := NthMinList(2,Close,Close[1],Close[2]);
x2deLaagste := MinList2(Close,Close[1],Close[2]);
```

Omdat de MinList2 functie altijd de 2de laagste zoekt, leveren beide instructies hetzelfde getal op.

zie ook:

Numerieke functies

1.10.3.80 NumLosTrades

Deze functie geeft het totaal aantal gesloten verliesgevende trades in een trading simulatie.

syntax:

```
value function NumLosTrades();
```

voorbeelden:

```
if vLastBar then Print('Totaal aantal verliezers ',NumLosTrades);
```

zie ook:
Performance functies

1.10.3.81 NumToStr

Deze functie converteert een numerieke waarde naar een string waarde.

syntax:
```
string function NumToStr( value xValue, value xDecimals=- 1);
```

voorbeelden:
```
string xTekst;
xTekst := 'Koers van vandaag is '+NumToStr(Close, 2);
```

Als u het tweede argument (het aantal decimalen) weglaat, of u geeft de waarde -1 mee, dan wordt de waarde weergegeven met 0 of 2 decimalen, al naar gelang het een heel getal is of een getal met decimalen.

zie ook:
Tekst functies
Numerieke functies

1.10.3.82 NumWinTrades

Deze functie geeft het totaal aantal gesloten winstgevende trades in een trading simulatie.

syntax:
```
value function NumWinTrades();
```

voorbeelden:
```
if vLastBar then Print( 'Totaal aantal winners ',NumWinTrades);
```

zie ook:
Performance functies

1.10.3.83 OpenPositionProfit

Deze functie geeft aan wat de winst op dit moment is in de huidige positie.

Deze functie kan bijv. gebruikt worden om een trailing stop loss te implementeren op basis van de hoogste winst tot dus ver.

syntax:
```
value function OpenPositionProfit();
```

voorbeelden:
```
if MaxGain> 2000 and OpenPositionProfit< 0.75*MaxGain then vExitLong;
```
Deze instructie zal er voor zorgen dat als de winst eenmaal meer dan 2000 euro is geweest, deze daarna niet meer dan 25% kan dalen.

zie ook:
Positie functies

1.10.3.84 PlaySound

Deze functie speelt een geluidsfragment af via de luidsprekers van de computer.

syntax:

```
value function PlaySound( string xGeluidsBestand);
```

voorbeelden:

```
if Close<xStopLoss then PlaySound( 'c:\Geluid\Sirene.wav' );
```

zie ook:

Trading en Signalering functies
Bestands functies

1.10.3.85 Plot

Deze functie resulteert in de waarde van een plot series voor de huidige bar.

syntax:

```
value function Plot( value xPlotNummer);
```

waarbij xPlotnummer een waarde van 1 tot 4 is om aan te geven van welke plot de waarde gevraagd wordt.

voorbeelden:

```
xPlotWaarde := Plot(1);
```

Opmerking:

In VestiCode is de waarde van een Plot series op dezelfde manier toegankelijk als alle andere series variabelen, en dus kan de waarde van een plot rechtstreeks opgevraagd worden. Bovendien is het daardoor eenvoudig om de waarde van een plot voor voorgaande bars op te vragen.

```
xPlotWaarde := Plot1;       { waarde van huidige bar }
xPlotWaarde := Plot1[0];    { idem }
xPlotWaarde := Plot1[5];    { waarde 5 bars geleden }
if CrossesAbove(Close,Plot1)  then...    { gebruik Plot1 als series }
```

Meer uitleg over plot series vindt u in het hoofdstuk Het tekenen van grafieken .

zie ook:

Bestands functies

1.10.3.86 Plot1-4

Deze functie wordt gebruikt om een grafiek te tekenen.

Door bij elke bar aan te geven op welke waarde een puntje in de grafiek geplot moet worden ontstaat al naar gelang de gekoezen instelling een lijngrafiek, tikgrafiek, enz.

syntax:

```
value function Plot n(value xWaarde,string xName);
```

waarbij *n* een waarde van 1 tot 4 is om aan te geven van welke plot de waarde ingevuld wordt.

voorbeelden:

```
Plot3(Close-Close[9],'Mom10');
```

Meer uitleg over plot series vindt u in het hoofdstuk Het tekenen van grafieken .

zie ook:
Bestands functies

1.10.3.87 Pointvalue

Deze functie geeft als resultaat de waarde van 1 punt koersstijging of daling.

syntax:
```
value function PointValue();
```

voorbeelden:
```
xWinst := ( xVerkoopprijs - xAankoopprijs )* PointValue;
```

1.10.3.88 Pos

Deze functie maakt van elk getal een positief getal.

Deze functie haalt als het ware het min-teken weg bij een negatief getal. Positieve getallen blijven onveranderd.

syntax:
```
value function Pos( value xGetal);
```

voorbeelden:
```
xResultaat := Pos(xGetal);
xResultaat := AbsValue(xGetal);
```
De functies Pos en AbsValue doen exact hetzelfde.

zie ook:
Numerieke functies

1.10.3.89 PositionProfit

Deze functie geeft aan wat de winst was van een bepaalde positie.

Met deze functie kan niet alleen opgevraagt worden wat de winst in de huidige positie is, maar kan dat ook voor alle voorgaande posities.

Als de betreffende positie eindigde met verlies, dan wordt dit aangegeven door een negatieve winst. Dus als het verlies 150 euro was, dan is de winst -150 euro.

Indien het positienummer gelijk is aan 0, dan betreft het de huidige (open) positie, 1 is de vorige positie, enz.
Indien het positienummer 0 is, en er is momenteel geen open positie, dan is het resultaat 0.
Indien een positienummer wordt gebruikt dat hoger is dan het aantal trades tot nu toe, dan is het resultaat 0.

syntax:
```
value function PositionProfit(value xPositieNr);
```

voorbeelden:

```
xWinst := PositionProfit(1);
```

Resulteert in de winst van de vorige trade.

zie ook:

Positie functies

1.10.3.90 Power

Deze functie berekent een bepaald getal tot de opgegeven macht.

Deze functie kan ook gebruikt worden met gebroken machten en op die manier kan ook de n-de machts wortel uit een getal getrokken worden.

In Vestics is de functie machtsverheffen ook beschikbaar als de operator ^, zodat Power(3,3) gelijk is aan 3^3.

syntax:

```
value function Power( value xGetal, value xMacht);
```

voorbeelden:

```
xResultaat := Power(3,3);     { geeft 27 want 3x3x3 is 27 }
xResultaat := Power(8,1/3);   { geeft 2 want de 3de machts wortel uit 8
is 2 }
```

zie ook:

Numerieke functies

1.10.3.91 Print

Deze functie stuurt een tekstregel naar het Samenvatting tabblad van de Designer of naar de printer of een bestand op schijf.

syntax:

```
function Print(arg1,arg2,...);
function Print( Printer ,arg1,arg2,...);
function Print( File ('\Temp\Lijst.txt'),arg1,arg2,...);
```

waarbij maximaal 25 argumenten meegegeven kunnen worden

voorbeelden:

```
Print('De slotkoers is ',Close-Close[2],' hoger dan gisteren.');
Print(File('KLM.asc'),Date+19000000,',',Open,',',Low,',',High,',',Clos
e);
```

Meer uitleg over de print functie vindt u in het hoofdstuk De Print functie .

zie ook:

Bestands functies

1.10.3.92 Random

Deze functie resulteert in een willekeurig getal van 0 tot *n*, waarbij *n* meegegeven wordt als argument.

Deze functie kan ook gebruikt worden om toevalslogica toe te passen.

syntax:

```
value function Random( value xKansBereik);
```

voorbeelden:

```
if Random( 1)<0.25 then vEnterLong else vEnterShort;
```

In dit geval zal in ca. 25% van de keren een long positie worden gestart en in de resterende 75% van de gevallen een short positie.

zie ook:

Numerieke functies

1.10.3.93 RightStr

Deze functie geeft de laatste (oftewel meest rechtse) *n* tekens van een tekst.

Als er minder dan *n* tekens zijn in de tekst wordt de hele tekst teruggeven.

syntax:

```
value function RightStr( string xTekst, value xAantal);
```

voorbeelden:

```
xRechterDeel := LeftStr('ABCDEFGHI JKL',5);
```

Het resultaat is de tekst 'I JKL'.

zie ook:

Tekst functies

1.10.3.94 Round

Deze functie rond een getal af op het gegeven aantal decimalen.

syntax:

```
value function Round( value xGetal, value xDecimalen);
```

voorbeelden:

```
xResultaat := Round(xGetal,1);
```

Als xGetal gelijk is aan 13.45 dan is xResultaat gelijk aan 13.5.

zie ook:

Numerieke functies

1.10.3.95 Sess1EndTime

Deze functie geeft de sluitingstijd van de beurs waarop het huidige fonds wordt verhandeld.

Als een beurs 2 sessies per dag heeft, wordt de eindtijd van de eerste sessie gegeven.

De tijd is in het 24-uurs formaat HHMM .

syntax:

```
value function Sess1EndTime();
```

Voorbeelden:

```
xSlot := Sess1EndTime;
```

Geeft 1730 als de beurs sluit om 17:30 uur.

zie ook:

Datum en tijd functies

1.10.3.96 Sess1FirstBarTime

Deze functie geeft de eindtijd van de eerste bar bij intradag koersen.

De tijd is in het 24-uurs formaat HHMM .

syntax:

```
value function Sess1FirstBarTime();
```

Voorbeelden:

```
xEindeBar1 := Sess1FirstBarTime;
```

Als de beurs begint om 9:00 uur en de bar interval is 15 minuten, dan wordt de waarde 915 gegeven.

zie ook:

Datum en tijd functies

1.10.3.97 Sess1StartTime

Deze functie geeft de openingstijd van de beurs waarop het huidige fonds wordt verhandeld.

Als een beurs 2 sessies per dag heeft, wordt de openingstijd van de eerste sessie gegeven.

De tijd is in het 24-uurs formaat HHMM .

syntax:

```
value function Sess1StartTime();
```

Voorbeelden:

```
xOpening := Sess1StartTime;
```

Geeft 900 als de beurs begint om 9:00 uur.

zie ook:

Datum en tijd functies

1.10.3.98 Sess2EndTime

Deze functie geeft de sluitingstijd van de beurs waarop het huidige fonds wordt verhandeld.

Als een beurs 2 sessies per dag heeft, wordt de eindtijd van de tweede sessie gegeven.

De tijd is in het 24-uurs formaat HHMM .

syntax:

```
value function Sess2EndTime();
```

Voorbeelden:

```
xSlot := Sess2EndTime;
```

Geeft 1730 als de beurs sluit om 17:30 uur.

zie ook:

Datum en tijd functies

1.10.3.99 Sess2FirstBarTime

Deze functie geeft de eindtijd van de eerste bar bij intradag koersen.

De tijd is in het 24-uurs formaat HHMM .

syntax:

```
value function Sess1FirstBarTime();
```

Voorbeelden:

```
xEindeBar1 := Sess1FirstBarTime;
```

Als de beurs begint om 9:00 uur en de bar interval is 15 minuten, dan wordt de waarde 915 gegeven.

zie ook:

Datum en tijd functies

1.10.3.100 Sess2StartTime

Deze functie geeft de openingstijd van de beurs waarop het huidige fonds wordt verhandeld.

Als een beurs 2 sessies per dag heeft, wordt de openingstijd van de tweede sessie gegeven.

De tijd is in het 24-uurs formaat HHMM .

syntax:

```
value function Sess1StartTime();
```

Voorbeelden:

```
xOpening := Sess2StartTime;
```

Geeft 1300 als de 2de beurssessie begint om 13:00 uur.

zie ook:

Datum en tijd functies

1.10.3.101 Sign

Deze functie geeft de waarde +1 (indien positief), 0 (indien nul) of -1 (indien negatief) afhankelijk van het teken van een getal.

syntax:

```
value function Sign( value xGetal);
```

voorbeelden:

```
xPlusOfMin := Sign(xGetal);
```

zie ook:
Numerieke functies

1.10.3.102Sine

Deze functie uit de goniometrie berekent de sinus van een hoek uitgedrukt in graden.

syntax:

```
value function Sine( value xGraden);
```

voorbeelden:

```
xSinus := Sine(45);
```

zie ook:
Numerieke functies

1.10.3.103Spaces

Deze functie geeft een tekststring van het aantal gevraagde spaties.

syntax:

```
value function Spaces( value xLength);
```

voorbeelden:

```
xTekst := xTekst1+Spaces(5)+xTekst2;
```

zie ook:
Tekst functies

1.10.3.104Square

Deze functie berekent het kwadraat van een bepaald getal.

In Vestics is de functie machtsverheffen ook beschikbaar als de operator ^, zodat Square(3) gelijk is aan 3^2.

syntax:

```
value function Square( value xGetal);
```

voorbeelden:

```
xKwadraat := Square(3); { geeft 9 want 3x3=9 }
```

zie ook:
Numerieke functies

1.10.3.105SquareRoot

Deze functie berekent de (2de machts) wortel van een bepaald getal.

syntax:
```
value function SquareRoot( value xGetal);
```

voorbeelden:
```
xWortel := SquareRoot(9);  { geeft 3 want 3x3=9 }
```

zie ook:
Numerieke functies

1.10.3.106StrLen

Deze functie geeft het aantal tekens in een tekst string.

Dit is inclusief spaties en alle speciale tekens.

syntax:
```
value function StrLen( string xTekst);
```

voorbeelden:
```
xAantalTekens  := StrLen('ABCDEFGHI  JKL');
```
Het resultaat is 13 (12 letters plus 1 spatie).

zie ook:
Tekst functies

1.10.3.107StrToNum

Deze functie geeft de numerieke waarde van een tekst string.

Indien de tekst leeg is, of een ongeldig getal bevat, is het resultaat 0.
De tekst mag alleen een eventueel min-teken, cijfers en eventueel een decimale punt bevatten.

syntax:
```
value function StrToNum( string xTekst);
```

voorbeelden:
```
xGetal := StrToNum('-348.12');
```
Het resultaat is het getal -348.12.

zie ook:
Tekst functies
Numerieke functies

1.10.3.108SymbolName

Deze functie geeft de Fondsnaam van het huidige effect (aandeel, optie, enz).

syntax:
```
string function SymbolName();
```

voorbeelden:

```
if vLastBar then Print('Fondsnaam   ',SymbolName);
```

zie ook:

Informatie functies

1.10.3.109SymbolNumber

Deze functie geeft het Fondsnummer van het huidige effect (aandeel, optie, enz).

syntax:

```
value function SymbolNummer();
```

voorbeelden:

```
if vLastBar then Print('Fondsnummer   ',SymbolNumber);
```

zie ook:

Informatie functies

1.10.3.110SymbolRoot

Deze functie geeft de Fondsnaam van de onderliggende waarde van het huidige effect (alleen bij opties e.d.).

syntax:

```
string function SymbolRoot();
```

voorbeelden:

```
if vLastBar then Print('Fondsnaam   Onderliggend  Fonds  ',SymbolRoot);
```

zie ook:

Informatie functies

1.10.3.111Tangent

Deze functie uit de goniometrie berekent de tangens van een hoek uitgedrukt in graden.

syntax:

```
value function Tangent( value xGraden);
```

voorbeelden:

```
xTangens  := Tangent(45);
```

zie ook:

Numerieke functies

1.10.3.112TotalBarsLosTrades

Deze functie geeft de totale duur (in bars) van alle verliesgevende trades.

syntax:

```
value function TotalBarsLosTrades();
```

voorbeelden:

```
if vLastBar then Print('Totale duur verliezers ',TotalBarLosTrades);
```

zie ook:

Performance functies

1.10.3.113TotalBarsWinTrades

Deze functie geeft de totale duur (in bars) van alle winstgevende trades.

syntax:

```
value function TotalBarsWinTrades();
```

voorbeelden:

```
if vLastBar then Print('Totale duur winnaars ',TotalBarWinTrades);
```

zie ook:

Performance functies

1.10.3.114TotalTrades

Deze functie geeft het totaal aantal gesloten trades in een trading simulatie.

syntax:

```
value function TotalTrades();
```

voorbeelden:

```
if vLastBar then Print('Totaal aantal trades ',TotalTrades);
```

zie ook:

Performance functies

1.10.3.115UpperStr

Deze functie veranderd alle kleine letters in een tekst naar hoofdletters. Alle andere tekens blijven ongewijzigd.

syntax:

```
value function UpperStr( string xTekst);
```

voorbeelden:

```
xHoofdLetters := UpperStr('Dit is VestiCode');
```

Het resultaat in de variabele xHoofdLetters is de tekst 'DIT IS VESTICODE'.

zie ook:

Tekst functies

1.10.3.116Year

Deze functie haalt het jaartal (0-135) uit een datum .

Om het echte jaartal te krijgen moet de waarde 1900 bij het resultaat opgeteld worden, waarna het resulterende jaartal de waarde 1900 tot 2035 kan hebben.

syntax:

```
value function Year(value xDatum);
```

voorbeelden:

```
xJaar := 1900+Year(CurrentDate);
```

zie ook:

Datum en tijd functies

Index

- _ -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- R -

- S -

- T -

- U -

- V -

- W -

- Y -