Nieuwsbrief

18de jaargang maart 2011 **Nummer 1**

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Geen	

Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Jan van der Linden	071-3413679	voorz@basic-gg.hcc.nl
Secretaris	Gordon Rahman Tobias Asserstraat 6 2037 JA Haarlem	023-5334881	secr@basic-gg.hcc.nl
Penningmeester	Piet Boere	0348-473115	penm@basic-gg.hcc.nl
Bestuurslid	Titus Krijgsman	075-6145458	t.krijgsman8@upcmail.nl
Redacteur	M.A. Kurvers Schaapsveld 46 3773 ZJ Barneveld	0342-424452	m.a.kurvers@hccnet.nl
Ledenadministratie	Fred Luchsinger	0318-571187	f.luchsinger@kader.hcc.nl
Webmaster	Jan van der Linden	071-3413679	j.vd.linden@kader.hcc.nl

http://www.basic.hcc.nl

Redactioneel

VBA is weer terug van vakantie. In VBA laat ik u onderdelen zien die we kunnen gebruiken en hoe we daarmee zelf dialoogformulieren kunnen ontwerpen.

Het tekenen met grafische functies wordt wat ingewikkelder. We gaan eens de parametervorm bekijken. Deze mogelijkheid kan in elke Basic versie gebruikt worden. Het werkt zelfs beter dan wanneer we alleen de poolcoördinaat gebruiken.

In de Crimson Editor gaan we beginnen met de eerste pagina, index.html, en kunt u zien hoe verschillende tags in de pagina werken.
Er zal ook een eerste inleiding komen over stylesheets, de tekstopmaak met CSS. In de volgende nieuws-brieven ga ik uiteraard daar op verder.

Marco Kurvers

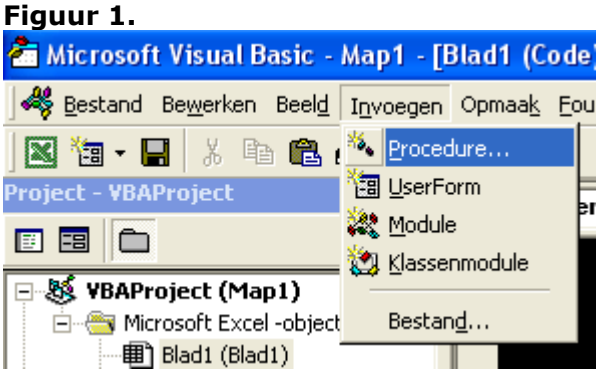
BASIC cursus: VBA met Excel (5).

Microsoft heeft veel vrijheid gegeven om gebruikers hun eigen werkboeken, of ook wel genoemd: Excel projecten, te laten automatiseren door gebruik te maken van VBA. Tot nu toe hebben we gezien dat we inderdaad veel kunnen doen met VBA. Werk dat we in Excel niet kunnen ontwikkelen. Maar VBA heeft nog veel meer mogelijkheden. We kunnen Excel uitbreiden zodat het bijna een eigen Excel applicatie wordt.

Terug naar VBA – de onderdelen.

In VBA kunnen we gebruik maken van onderdelen die lijken op onderdelen in Visual Basic 6. Figuur 1 laat het Invoegen menu zien van de VBA editor.

Figuur 1.



In plaats van zelf een procedure te schrijven kunt u in het Invoegen menu ook een procedure in de editor laten maken. De procedure zal alleen een leeg geraamte zijn.
De UserForm is een onderdeel waarmee u zelf dialoogformulieren kunt ontwerpen. Later kom ik daar op terug.
De module en klassenmodule zien er hetzelfde uit als in Visual Basic 6. U kunt dus uw eigen klassenobjecten maken.
In de gewone module kunt u de variabelen declareren en de macro's in maken en uiteraard uw zelfgemaakte procedures en functies.

Het is ook mogelijk bestanden uit een ander formaat in te voegen. In de volgende les kom ik daar op terug.

Een procedure invoegen.

Het menu-item Procedure... heeft drie punten. Er volgt dus nog een formulier met opties. Figuur 2 laat het formulier zien.

Figuur 2.

Zoals u Figuur 2 ziet zou eigenlijk het formulier anders moeten heten, namelijk niet 'Procedure invoegen' maar 'Methode invoegen', want er zijn mogelijkheden om ook functies en eigenschappen in te voegen en een methode is ook een functie. Een procedure niet.

Het type en het bereik dat u op kunt geven spreken voor zich, maar laten we eens het keuzevakje onderaan bekijken. Vinkt u het keuzevakje "Alle lokale variabelen als Static-waarden" aan dan zullen de lokale variabelen in de methode (Sub, Function of Property) statisch worden. De waarden zullen dan in de (klassen)module, dialoogformulier of werkblad herkenbaar blijven, ook al zou de methode verlaten worden. Deze mogelijkheid is niet altijd handig. Er kan een ongestructureerd gedrag ontstaan in de code.

Stel, u hebt de naam EigenSub ingetoetst en het vinkje aangezet om de lokale variabelen statisch te houden. Als u op de OK knop klikt, zal onderstaande procedure verschijnen:

```
Public Static Sub EigenSub()

End Sub
```

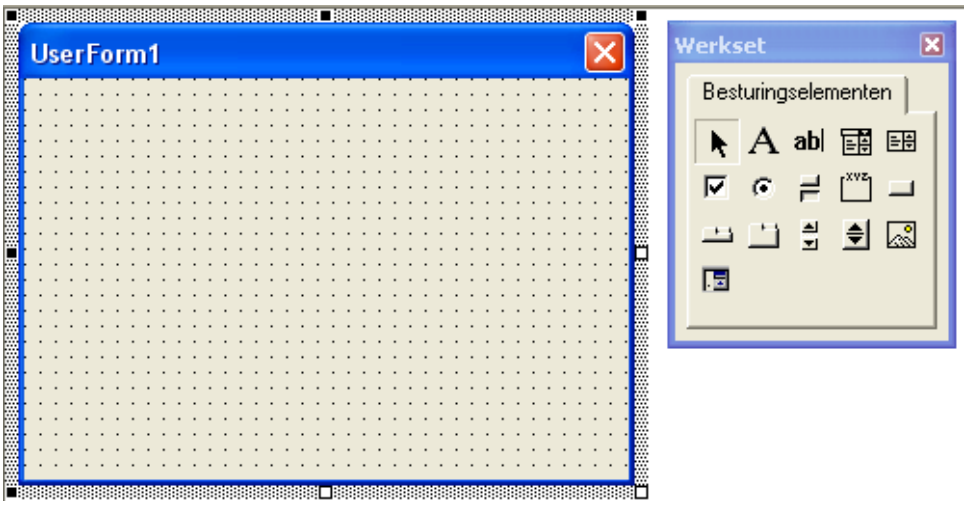
In QuickBASIC zal het Static sleutelwoord achter de parameterhaakjes worden geplaatst, zoals u dat in de Intermezzo listing 'Een Werknemers project' kunt terugzien.

Uw dialoogformulier in VBA.

Hoewel ik het over dialoogformulieren heb, hebben de formulieren toch de eigenschap ShowModal. We kunnen dan ervoor zorgen dat de gebruiker zowel met het formulier als met het werkblad kan werken als wij die optie op False zetten. Het formulier wordt dan een modeless formulier. Maak niet teveel van zulke formulieren in VBA. Het Excel project kan dan rommelig werken en bovendien zullen de knoppen OK en Annuleren geen nut meer hebben. Modal dialoogformulieren zijn ook altijd de verankerde formulieren die deze knoppen hebben met ook wel eens een Toepassen knop.

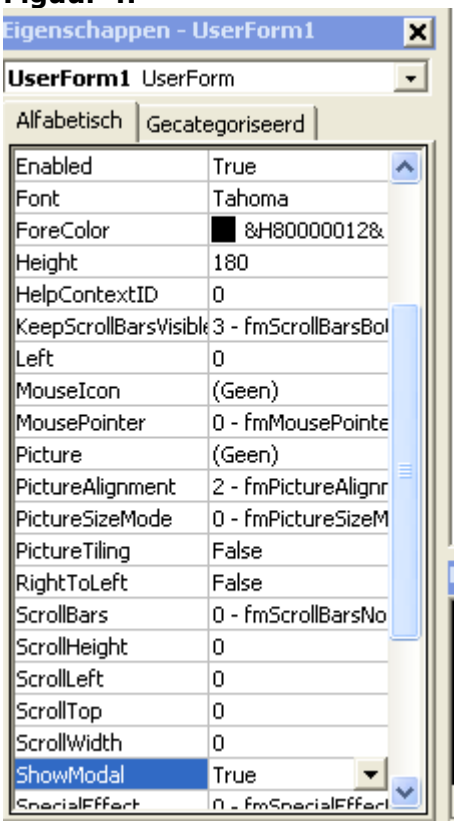
Figuur 3 laat een mini Visual Basic IDE zien in VBA stijl.

Figuur 3.



Zoals u ziet is de werkset niet uitgebreid. Er zijn maar enkele besturingselementen waarmee het formulier opgebouwd kan worden. Maar na u het UserForm menu-item gekozen heeft, is het hoofd-menu uitgebreid, zie hieronder. Zo is het opmaak menu uitgebreid met extra mogelijkheden voor het formulier en kunt u in het Extra menu extra besturingselementen toevoegen. U kunt ook verwijzen naar externe objecten, zoals Word objecten. Dat kan ook andersom.

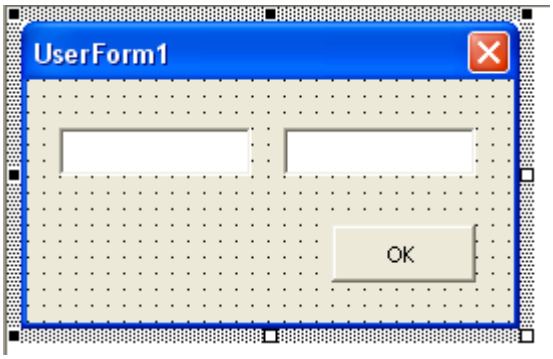
Figuur 4.



De eigenschap ShowModal is nog geselecteerd. Hier kunt u zien dat u die optie op True of False kunt zetten, zoals eerder uitgelegd is.

Als we het menu bekijken dan zult u zien dat we in een hele andere VBA zijn belandt. Dit is de IDE omgeving voor het maken van VBA projecten. We kunnen ook zonder dit projecten maken, maar dan gebruiken we alleen de VBA code.
Het Extra menu bevat veel mogelijkheden die zelf ook weer uit dialoogformulieren bestaan (zie de drie punten). We kunnen ook weer macro's opnemen tijdens het ontwikkelen van het VBA project.

Experimenteer maar eens met de formulieren. U kunt bijvoorbeeld twee tekstvakken en een knop op het formulier plaatsen. De tekstvakken zullen de invoer zijn voor een cel en de waarde die in de cel moet komen. Verander de Caption van de knop in OK die voor de doorvoer van de gegevens moet zorgen, zie het voorbeeld.



Als we nu het formulier starten, zien we dat ook Excel tevoorschijn komt, met het formulier dat we net gemaakt hebben. We kunnen wel waarden in gaan voeren en op OK klikken, maar er gebeurt niets. Ook op de werkbladen kunnen we niets doen, omdat het formulier Modal is.

Sluit het formulier door op het kruisje te klikken. Direct komt het ontwerpscherm van VBA weer terug.
Om de ingevoerde waarde in de opgegeven cel te kunnen zetten, moeten we wat code maken in de OK knop, zie onderstaand voorbeeld.

De code hieronder moet u in de gebeurtenis typen van de knop. Dubbelklik op de knop om de *cmdOK_Click()* subroutine in de editor te openen.

```
Blad1.Range(txtCel.Text).Value = txtWaarde.Text
```

Start het formulier en voer de waarden A1 en 50 in. Klik op OK en u ziet het getal 50 verschijnen in cel A1.

Sluit het formulier en start deze nogmaals. Voer nu de waarden A1:C5 en 10 in. Klik op OK. Nu ziet u het opgegeven gebied gevuld met de waarde 10.

We kunnen de invoer en uitvoer ook omdraaien. Verwijder tekstvak txtCel en verander de code in de OK knop.

```
txtWaarde.Text = ActiveCell.Value
```

Verander ook de waarde van de formuliereigenschap ShowModal in False en start het formulier.
Voer wat waarden in de cellen, kies elk een cel en klik dan op OK. De waarde van een cel zal dan in het tekstvak verschijnen.
Open ook eens andere bladen en doe weer hetzelfde. Geen probleem, de waarden komen in het tekstvak. Hadden we dit ook vanuit het formulier naar de bladen willen hebben dan hadden we de code anders moeten schrijven. Niet met gebruik van Blad1, maar met gebruik van: `ActiveCell.Range(...)`. De reden dat ik hierboven ActiveCell gebruik is dat Blad1 geen eigen Value eigenschap heeft. Door het opgeven van een blad kan VBA niet zelf bepalen welke cel actief is, maar ActiveCell kan dat wel omdat die bijhoudt in welke cel er gewerkt wordt, inclusief in welk blad.

Omdat ActiveCell ook een Range object heeft kunnen we ook celgebieden naar het formulier sturen en andersom.

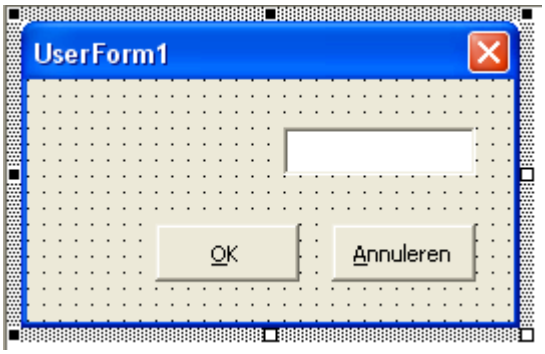
In plaats van de waarden kunnen we ook de adressen van de cellen weergeven zoals onderstaand voorbeeld doet.

```
txtWaarde.Text = ActiveCell.Address
```

Het adres zal met de dollartekens worden weergegeven.

Koppeling tussen Excel en VBA.

We kunnen dus werken met formulieren en werkbladen tegelijk. Tot nu toe hebben we steeds het formulier opgestart. Het zou beter zijn om vanuit het werkblad het formulier te openen, iets te wijzigen, en de wijziging als resultaat in de actieve cel weer te geven. Op die manier kunnen we altijd het werkblad als hoofdblad gebruiken met de dialoogformulieren die pas geopend worden als ergens op geklikt wordt of wijzigingen optreden.



Verander het formulier in de afbeelding zoals u hier links ziet. De knoppen OK en Annuleren moeten bepalen of de invoer doorgegeven mag worden of niet. Er moet daarom wat code in het formulier aangepast worden.

U ziet ook de streepjes onder de letter O van OK en onder de A van Annuleren. U kunt het teken & die normaal gesproken in Visual Basic in de Caption eigenschap ingetypt moet worden niet in VBA gebruiken. Het teken zal gewoon weergegeven worden.

Toch is er een mogelijkheid. Het CommandButton besturingselement heeft in VBA een andere eigenschap om de onderstreping weer te geven. U hoeft alleen maar in de eigenschap Accelerator de letter op te geven die u als sneltoets wilt laten werken.

Open het codevenster en declareer een public variabele bovenaan de andere code:

```
Public bOK As Boolean
```

Ga terug naar uw formulier en dubbelklik op de OK knop. Als u de vorige code nog daar in hebt staan verwijder die dan eerst. Typ onderstaande code in de subroutine:

```
bOK = True
Hide
```

Onthoud die twee regels ook voor als u werkt in Visual Basic 6. Gebruik altijd het Hide statement en niet het Unload statement. Het sluiten (unloaden) van het dialoogformulier moet altijd gebeuren in de code waar ook het dialoogformulier met Load geladen wordt.

Typ in de subroutine van de Annuleren knop ook die twee regels, maar gebruik de waarde False in plaats van de waarde True. De knop Annuleren moet er voor zorgen dat de invoer niet doorgegeven wordt.

Het dialoogformulier koppelen en de gegevens doorgeven.

De code in het UserForm is klaar, maar het doorgeven zal nog niet werken. Bovendien is er nog helemaal geen koppeling tussen het werkblad en het dialoogformulier.

We kunnen acties uit laten voeren als de gebruiker een handeling uitvoert. Dat kan een klik zijn op een knop of een gewijzigde selectie op het werkblad.
Onderstaand voorbeeld zijn acties die uitgevoerd worden zodra de gebruiker op een cel dubbelklikt.
Open Blad1 en kies uit de events de subroutine BeforeDoubleClick. Typ de code in de subroutine.

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
    Load UserForm1
    With UserForm1
        .txtWaarde.Text = Target.Value
        .Show vbModal
        If .bOK Then
            Target.Value = .txtWaarde.Text
        End If
    End With
    Unload UserForm1
End Sub
```

Als variabele bOK de waarde True heeft, dus de gebruiker op OK heeft geklikt, zal de waarde uit het geselecteerd gebied dat via het Target argument is ontvangen, vervangen worden door de waarde die in het tekstvak van het dialoogformulier is ingevoerd. Het tekstvak mag nog steeds worden gebruikt zolang UserForm1 niet ontladen is, ook al is in de OK of Annuleren knop het Hide statement al aangeroepen die alleen maar UserForm1 zal verbergen.

Nu de BeforeDoubleClick event klaar is hebben we een prima koppeling met de Target (het cellenbereik) en het dialoogformulier UserForm1.

Bewaar de wijzigingen en sluit VBA af. Terug in Excel ziet u nog steeds de waarden in de cellen. Een goede test om daar eens op te dubbelklikken. Als het goed is moet dan het dialoogformulier verschijnen en moet in het tekstvak de waarde staan van de cel die u gekozen hebt.

Wijzig de waarde en klik op OK. Direct sluit het dialoogformulier en ziet u uw wijziging in de cel verschijnen. Doe dit nogmaals, maar klik dan op Annuleren. De wijziging zal dan niet in de cel verschijnen en de oude waarde zal blijven staan.

Merk op als u op een cel dubbelklikt die een verwijzing heeft dat, na het klikken op Annuleren, plotseling de verwijzingsformule verschijnt. Dit gebeurt automatisch en dat is ook het geval met som functies. Het resultaat zal pas weer verschijnen als u de cel verlaat.

Marco Kurvers

Ik kan programmeren.

Kunt u programmeren? Dat is mooi. Ik kan dan niet zeggen dat u het fout doet, of toch wel?!
Tijdens het programmeren kunt u verrassende resultaten tegenkomen. U denkt dat wat u doet op die manier ook zal werken. De computer is helaas niet slim genoeg om te weten wat u aan het doen bent. Basic wil ook liever rechtdoor terwijl u vindt dat de code rechtsaf moet.

Meerdere wegen naar Rome.

Er zijn dus meer programmeermogelijkheden die voor hetzelfde doel dienen. Maakt dat wat uit? Natuurlijk, het is altijd prima als de code werkt, maar de gebruiker wil ook liever dat de gegevens zo goed mogelijk in tabellen worden weergegeven, de formulieren er netjes uit zien en de applicatie niet te rommelig wordt. Er zijn dus meerdere richtingen om hetzelfde te kunnen maken. Om te kiezen welke richting het beste is, is het maken van een ontwerp of een schets niet verkeerd.

Voor de duidelijkheid geef ik over dit onderwerp wat voorbeelden met uitleg. Het zijn 'foutjes' die niet echt fout zijn en gewoon werken. Om handig te zijn in slim programmeren is zo'n werkend foutje slecht te noemen.

Onderstaand voorbeeld laat een foutje zien waar echter niks mee aan de hand is.

```
IF A = 10 THEN B = TRUE ELSE B = FALSE
```

We kunnen in plaats van een IF ... THEN statement ook een IIF functie gebruiken. Controleer eerst of uw BASIC versie die functie kent! U kunt ook de appendix raadplegen in deze nieuwsbrief.

```
B = IIF(A = 10, TRUE, FALSE)
```

De code doet nog steeds hetzelfde, maar nu wordt het resultaat toegekend aan variabele B. Het is wat simpeler en teveel IF ... THEN statements die dit moeten doen maakt de code te druk.

Soms wil men dat er geen resultaat wordt gegeven als de voorwaarde onjuist is. Men heeft de neiging om dan een coderegel te schrijven als:

```
GETAL = IIF(N > 100, 100, GETAL)
```

De derde parameter in de IIF functie is verplicht. Tja, er zit dan niks anders op dan dezelfde variabele op te geven als de variabele die het resultaat krijgt. Als u goed kijkt is die manier van programmeren ook een slechte keuze. Als de voorwaarde onjuist is zal de waarde van GETAL toegekend worden aan GETAL, zodat we een rare situatie krijgen:

```
GETAL = GETAL
```

Om die situatie te vermijden is het beter wel een IF ... THEN statement te schrijven, maar dan zonder het ELSE statement.

Hebt u er geen zin meer in om de IIF functie te gebruiken en moeten er TRUE en FALSE waarden als resultaten gegeven worden, dan is er nog een manier. De code wordt er zelfs korter van, maar het is wel even

wennen. Als u zo'n oplossing nog nooit gezien hebt neem dan de IIF functie erbij en probeer het te vergelijken.

```
B = A = 10
```

Ook dit werkt:

```
B = (GETAL > 100 AND N = 0)     'Haakjes hoeven niet speciaal in BASIC
```

Het voordeel is dat u de variabele in meerdere statements kunt gebruiken zonder telkens de conditie tussen de haakjes te moeten typen. Het is ook niet verkeerd om het even te controleren in een IF ... THEN.

```
IF B THEN      of      IF B = TRUE THEN
```

Hier ook weer zoiets. Twee mogelijkheden: u gebruikt alleen B of u gebruikt het volledig: B = TRUE. Standaard is de waarde altijd TRUE. Vandaar dat de eerste mogelijkheid altijd waar is. Een IF NOT B THEN ... zou dan een FALSE moeten zijn, maar ook die is TRUE ook al weten we dat een NOT TRUE juist FALSE zou zijn. Hoe komt dat? Hieronder staat de regel nogmaals, maar nu voluit:

```
IF NOT B = TRUE THEN
```

Als u niet de appel wilt dan neemt u hem niet, ook al zou u hem in uw handen hebben. Als B de appel is maar u wilt dat BASIC hem niet neemt, moet u NOT gebruiken om ervoor te zorgen dat de code achter THEN uitgevoerd wordt. Het wel aannemen van de appel zou dus in een eventuele ELSE blok staan. Dit in het Nederlands uitgelegd.

Vergeet niet dat het ook andersom kan werken. Zonder NOT zal dus de code achter THEN achter ELSE moeten staan en de code achter ELSE achter THEN moeten staan, anders zal de uitvoer verrassend anders werken. Een uitvoer die zeker niet de goede weg naar Rome zal zijn.

Handigheidjes met lussen.

Niet alleen bovenstaande voorbeelden die de beslissingsmogelijkheden lieten zien kunnen voor meerdere programmeerrichtingen zorgen, ook de lusstructuren kunnen er wat van. In de oude BASICA met regelnummers hadden we wel eens de neiging om een geneste lus kruislings te herhalen. Maar toen al stribbelde BASIC gelijk tegen. We kunnen nou eenmaal geen binnenste NEXT laten herhalen met een lusvariabele van de buitenste NEXT.
Er kunnen echter nog meer knopen ontstaan en als de programmeur niet zo snel de oplossingen kan vinden kan de code slecht uitkomen, ook al zou het normaal werken.

Hoe maken we een slimme lusstructuur? Onderstaand voorbeeld ziet er normaal uit maar heeft niet de goede techniek om handig te noemen:

```
IF N > 10 THEN
    FOR I = 1 TO 10
        ...
    NEXT I
ELSE
    FOR I = 0 TO 10
        ...
    NEXT I
END IF
```

Het voorbeeld laat een keuzestructuur zien met twee FOR lussen. Toch kan het codeblok een stuk kleiner worden geschreven en wel door gebruik te maken van één FOR lus samen met de IIF functie. Bekijk eens de slimme code hieronder. Het werkt zeker!

```
FOR I = IIF(N > 10, 1, 0) TO 10
    ...
NEXT I
```

De IIF functie is ook toegestaan achter TO en STEP. Wordt de FOR lus te ingewikkeld gebruik dan variabelen die de resultaten van de IIF functies krijgen.

Dynamische lussen.

Gokspellen zijn zeer beroemd. Een programma schrijven die zulke slimme opdrachten moet uitvoeren lijkt niet makkelijk. Er ontstaan wel leuke trucjes om tot zoiets te komen, zoals herhalingslussen uitvoeren die tijdens het herhalen pas beslist worden waar de grens ligt of laten bepalen wanneer de stappen gewijzigd mogen worden.
Hebt u al eens een lus gemaakt die zoiets moet doen? Hiermee kunt u de gebruiker voor de gek houden tijdens een leuk spelletje en kunt u de computer vertellen: neem eens een andere stap zodat de gebruiker plotseling een andere gokwaarde voor zich krijgt.

Onderstaand voorbeeld laat zo'n dynamische stappentruc zien. Naast het voorbeeld ziet u het resultaat.

```
Dim n As Integer, i As Integer
n = 12
For i = IIf(n < 10, 1, 0) To 50 Step n
    If i > 25 Then n = 5
    [Debug.]Print i * n
Next i
```

```
0
144
288
180
240
```

Het voorbeeld is geschreven in Visual Basic 6, vandaar het Debug statement die tussen de kromme haken staat. Typ de kromme haken niet in. Ik heb dit expres zo gedaan omdat niet alle BASIC versies een Debug statement hebben, maar alleen het Print statement accepteren.

Het resultaat lijkt misschien geen leuke truc, maar als u de getallen goed volgt kunt u iets effectiefs zien. Variabele i wordt in elke herhaling met n vermenigvuldigd. Het If statement controleert of variabele i groter is dan 25. Zo ja, dan krijgt variabele n de waarde 5. De som expressie bij Print zal daardoor veranderen. We krijgen dan als uitvoer:

- een 3 keer vermenigvuldiging (0 * 12, 12 * 12, 24 * 12);
- variabele n krijgt waarde 5 als i groter is dan 25 en tot en met het 5de getal 2 keer zal worden vermenigvuldigd (36 * 5, 48 * 5).

Daardoor krijgen we na het getal 288 het getal 180 als resultaat.

Zulke effecten kunnen we niet maken met de functie RND(), want we hebben er niks aan om getallen willekeurig te laten kiezen. Het voorbeeld laat een effect zien hoe we midden in een dynamische lus een bepaalde vermenigvuldiging kunnen wijzigen. Leuk om hier eens meer mee te experimenteren. Laat bijvoorbeeld eens de lus niet tot 50, maar tot 80 of 100 herhalen en laat variabele n twee keer van waarde veranderen waardoor we drie verschillende vermenigvuldigingen in één lus kunnen maken die meerdere keren met hetzelfde getal herhaald worden.

Maar... er zit een addertje onder het gras. Het lijkt alsof de aantal stappen 5 worden door de wijziging van variabele n, maar helaas gebeurt dat niet. Verander variabele n achter het Step statement in gewoon 12 en alsnog zullen dezelfde getallen verschijnen.
Dit betekent dat, zodra de lus uitgevoerd wordt, de beginwaarde, de eindwaarde en de stapwaarde niet meer beïnvloedt kunnen worden. Het werkt wel als deze waarden bepaald worden voordat de For lus uitgevoerd wordt. Daardoor werkt de IIf functie wel omdat de waarde van variabele n gecontroleerd wordt. Maar de IIf functie zal dus, jammer genoeg, niet tijdens het herhalen van de lus steeds gecontroleerd worden, evenmin de stapwaarde.

Dit zijn beperkingen in de programmacode. Ook al willen we slim programmeren, te slim programmeren kan ook leiden tot verassende resultaten. Zoals u hebt kunnen zien zegt de compiler ook wel eens: doe het zelf maar!

Hetzelfde programma, maar dan met een While lus.

Maar we laten ons niet op de kop zitten door een eigenwijze compiler, want we kunnen het probleem oplossen door de lus zelf te laten herhalen en zelf de stappen uit te laten voeren. Onderstaand voorbeeld is een While lus die hetzelfde doet. Nu zal de uitvoer heel anders zijn.

```
Dim n As Integer, i As Integer
n = 12
i = IIf(n < 10, 1, 0)
While i <= 50
    If i > 25 Then n = 5
    [Debug.]Print i * n
    i = i + n
Wend
```

```
0
144
288
180
205
230
```

Nu zullen de laatste 3 waarden steeds met 25 worden verhoogd en niet met 60 ondanks we dezelfde code gebruiken voor variabele n. De reden is dat het wijzigen van de stappen nu wel werkt.
Merk op dat er meer getallen zijn. Ook dat komt door de stappenwijziging. We hebben nu één getal meer.

We kunnen ons niet veroorloven dat een FOR lus altijd goed zal werken, in ieder geval niet op de manier zoals men zou willen. Er zit dan niets anders op gebruik te maken van een WHILE lus of in sommige BASIC versies de DO WHILE ... LOOP lus.

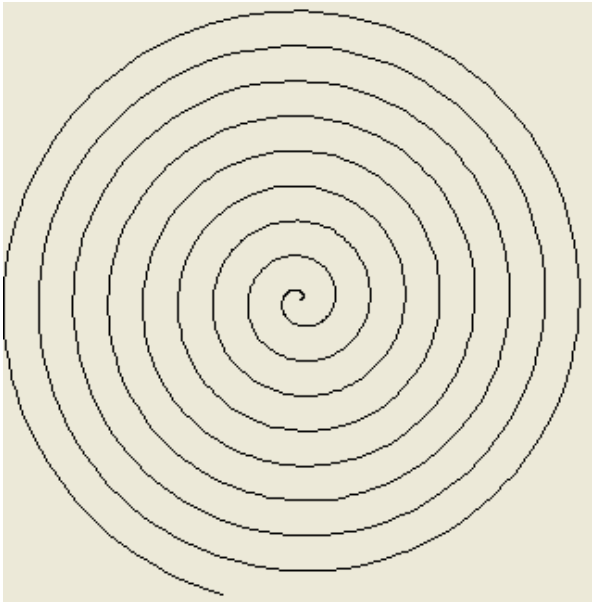
Indien de voorwaarde niet achter het WHILE statement moet staan, maak dan gebruik van de DO ... LOOP UNTIL lus. Kijk in de appendix of uw BASIC versie de DO ... LOOP lus heeft.

Marco Kurvers

Grafisch programmeren in GW-BASIC (6).

Spiralen zijn altijd geliefde figuren. **Programma 16** tekent logaritmische spiralen van Archimedes. Deze laatste soort spiralen hebben als vergelijking

$r = c.\varphi.$



In het programma is voor C de waarde 3 gekozen. Kies gerust andere waarden, maar wel tussen 0.5 en 20.

Ook hier weer kan het voorbeeld uit Visual Basic, zie afbeelding, niet goed overeenkomen met GW-BASIC. Mogelijk kan dat weer liggen aan de functie FNX() die gebruikt wordt om de figuren op schaal te tekenen om platte cirkels te voorkomen. Nogmaals: tegenwoordig is de functie niet meer nodig, omdat nu de schaal automatisch door de methoden berekent wordt zoals de methode DrawLine() ook doet.

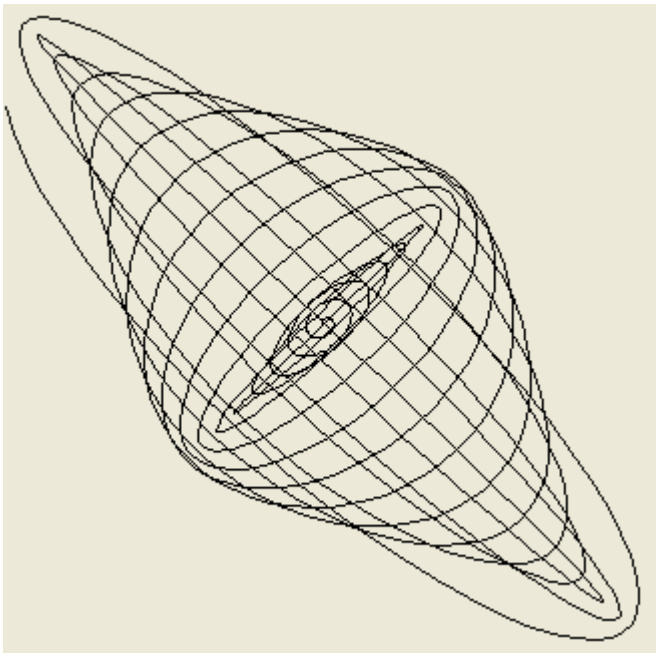
Als het goed is tekent GW-BASIC de spiraal alsof de afbeelding 180 graden is gedraaid.

Tip! Verander in Visual Basic de expressies U + R voor X1 en X2 in U – R en zie dat nu wel plotseling de afbeelding andersom getekend wordt.

```
100 '          programma 16                SPIRALEN
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 U=160 : V=160 : H=.5 : RD=4*ATN(1)/180
150 C=3 : P=0 : GOSUB 1000
160 X1=INT(U+R*COS(P)+H) : Y1=INT(V-R*SIN(P)+H)
170 FOR W=3 TO 10000 STEP 3
180    P=W*RD : GOSUB 1000
190    X2=INT(U+R*COS(P)+H):Y2=INT(V-R*SIN(P)+H)
200    IF X2 < 0 OR X2 > 320 THEN 250
210    IF Y2 < 0 OR Y2 > 320 THEN 250
220    LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
```

```
230    X1=X2 : Y1=Y2
240 NEXT W
250 A$=INKEY$: IF A$="" THEN 250
260 CLS: KEY ON: END
270 '
1000 R=C*P
1010 RETURN
```

```
Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5
Dim RD As Single = 4 * Math.Atan(1) / 180
Dim C As Single = 3, P As Single = 0, R As Single = C * P
Dim X1 As Integer = Int(U + R * Math.Cos(P) + H), _
    Y1 As Integer = Int(V - R * Math.Sin(P) + H)
For W As Integer = 3 To 10000 Step 3
    P = W * RD : R = C * P
    Dim X2 As Integer = Int(U + R * Math.Cos(P) + H), _
        Y2 As Integer = Int(V - R * Math.Sin(P) + H)
    If X2 >= 0 And X2 <= 320 And Y2 >= 0 And Y2 <= 320 Then
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Else
        Exit For
    End If
Next
```



Voor deze afbeelding zou u haast denken dat dit uit een andere listing komt, maar dat is niet het geval. Om dit te kunnen tekenen moet u de COS parameter en SIN parameter wijzigen. Dat is alles.

Verander de cosinus functie bij X2 in: `COS(P * 2.1)` en de sinus functie bij Y2 in: `SIN(P * 2)`.
Indien u de tekening gespiegeld ziet dan hebt u in X1 en X2 geen `U - R`, maar `U + R`.

Door de parameters van de functies COS en SIN te wijzigen, ontstaan er leuke spiraaleffecten.

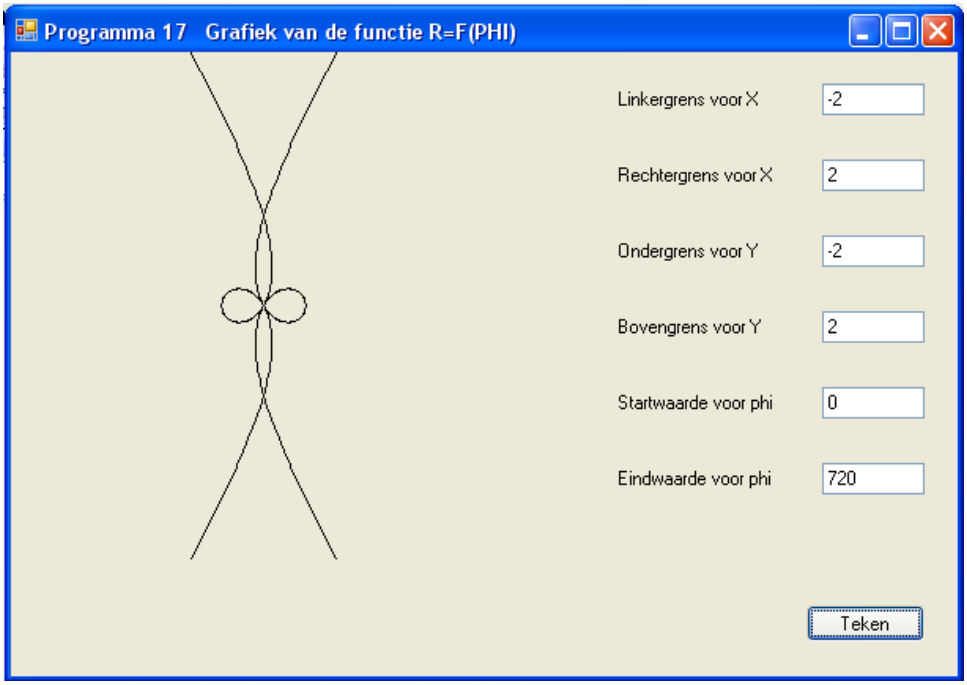
Als laatste programma met poolcoördinaten geven we **programma 17** voor het tekenen van een willekeurige, in poolcoördinaten geformuleerde, continue of niet-continue functie. De programmastructuur komt overeen met de structuur van programma 11 (zie nieuwsbrief nummer 3 jaar 2010). U kunt daar de werking van de vlaggen FZ en FA nog eens bestuderen.
Ook de andere variabelen hebben dezelfde betekenis als hun naamgenoten in programma 11. Als voorbeeld in het programma hebben we een vrij ingewikkelde functie die niet overal continu is gekozen:

$$r = \frac{\sin(1{,}5.\varphi)}{1-2.\cos\varphi}$$.

De figuur is getekend met A = -2, B = 2, LP = -2, HP = 2, WO = 0^0 en WN = 720^0.

```
100 '  programma 17  GRAFIEK VAN DE FUNCTIE R=F(PHI)
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "LINKER  GRENS VOOR X "; A: PRINT
150 INPUT "RECHTER GRENS VOOR X "; B: PRINT
160 INPUT "ONDER   GRENS VOOR Y ";LP: PRINT
```

```
170 INPUT "BOVEN   GRENS VOOR Y ";HP: PRINT
180 INPUT "STARTWAARDE VOOR PHI ";WO: PRINT
190 INPUT "EINDWAARDE  VOOR PHI ";WN
200 KX=320/(B-A):KY=320/(HP-LP):H=.5:RD=4*ATN(1)/180
210 CLS
220 FA=1
230 FOR W=WO TO WN
240     P=W*RD : GOSUB 1000
250     IF FZ=1 THEN FA=1 : GOTO 310
260     IF FA=1 THEN 300
270     X2=INT(KX*(X-A)+H) : Y2=INT(KY*(HP-Y)+H)
280     LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
290     X1=X2 : Y1=Y2 : GOTO 310
300     X1=INT(KX*(X-A)+H):Y1=INT(KY*(HP-Y)+H):FA=0
310 NEXT W
320 A$=INKEY$: IF A$="" THEN 320
330 CLS: KEY ON: END
340 '
1000 N=1-2*COS(P) : IF N=0 THEN FZ=1 : RETURN
1010 R=SIN(3*P/2)/N
1020 IF R<0 THEN FZ=1
1030 X=R*COS(P) : Y=R*SIN(P)
1040 IF X<A OR X>B   THEN FZ=1 : RETURN
1050 IF Y<LP OR Y>HP THEN FZ=1 : RETURN
1060 FZ=0 : RETURN
```



Wilt u een andere functie proberen, dan hoeft u alleen de regels 1000, 1010 en 1020 te veranderen door hierin de nieuwe functie R te berekenen.

Een mooie vlinder zou u bijvoorbeeld kunnen krijgen als u de volgende vergelijking neemt:

$$R = \frac{4.\sin(1{,}5p+2)}{\cos(p)\left(1+\frac{\cos(3p)}{3}\right)}.$$

Graag had ik de uitvoer hiervan aan u willen laten zien, maar helaas; het converteren naar Visual Basic met die functie gaf een raar resultaat. In ieder geval geen vlinder tekening. Lukt het u wel om hiermee een vlinder te tekenen?
Maar ook bovenstaand voorbeeld van programma 17 leek niet helemaal de figuur te zijn die er moest zijn. Er had nog in het midden dezelfde krommingen moeten staan als de verticale krommingen. Ook al heb ik zo goed mogelijk geprobeerd de code te converteren, het hielp niet. Probeer eens het probleem te vinden, want dan vind ik u een hele goede converter.

Om de zeldzame vlinder te krijgen nemen we voor A, B, LP, HP, WO en WN de waarden –4, 4, -4, 4, 0 en 720.

Hieronder ziet u de Visual Basic code van programma 17 met bovenstaand voorbeeld. Onthoud dat u niet de event code overtypt. Kies de events uit en laat Visual Basic het geraamte zelf maken. Vul de subroutines alleen met de blokcode. Maak wel zelf de PoolVorm() functie die subroutine 1000 nabootst.

```
Private Function PoolVorm(ByVal P As Single, ByRef X As Single, ByRef Y As Single) As Boolean
    Dim FZ As Boolean = False
    Dim N As Single = 1 - 2 * Math.Cos(P)
    If N > 0 Then
        Dim R As Single = Math.Sin(3 * P / 2) / N
```

```
        If R < 0 Then FZ = True
        X = R * Math.Cos(P) : Y = R * Math.Sin(P)
        FZ = (X < CSng(txtA.Text()) Or X > CSng(txtB.Text()) Or _
            Y < CSng(txtLP.Text()) Or Y > CSng(txtHP.Text()))
    Else
        FZ = True
    End If
    Return FZ
End Function

Private Sub btnTeken_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ...
    Refresh()
End Sub

Private Sub frmProg17_Paint(ByVal sender As Object, ByVal e As ...PaintEventArgs) ...
    If txtA.Text().Length() > 0 And txtB.Text().Length() > 0 And _
        txtHP.Text().Length() > 0 And txtLP.Text().Length() > 0 And _
        txtWN.Text().Length() > 0 And txtWO.Text().Length() > 0 Then
        Dim KX As Single = 320 / (CSng(txtB.Text()) - CSng(txtA.Text())), _
            KY As Single = 320 / (CSng(txtHP.Text()) - CSng(txtLP.Text())), _
            H As Single = 0.5, RD As Single = 4 * Math.Atan(1) / 180
        Dim FA As Boolean = True, FZ As Boolean = False
        Dim X As Single = 0, Y As Single = 0
        Dim X1 As Integer = 0, X2 As Integer = 0, Y1 As Integer = 0, Y2 As Integer = 0
        For W As Single = CSng(txtWO.Text()) To CSng(txtWN.Text())
            Dim P As Single = W * RD : FZ = PoolVorm(P, X, Y)
            If Not FZ Then
                If Not FA Then
                    X2 = Int(KX * (X - CSng(txtA.Text())) + H)
                    Y2 = Int(KY * (CSng(txtHP.Text()) - Y) + H)
                    e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
                    X1 = X2 : Y1 = Y2
                Else
                    X1 = Int(KX * (X - CSng(txtA.Text())) + H)
                    Y1 = Int(KY * (CSng(txtHP.Text()) - Y) + H)
                    FA = False
                End If
            Else
                FA = True
            End If
        Next
    End If
End Sub
```

De parametervorm
De meest interessante krommen krijgen we bij functies waarvan de vergelijking in parametervorm wordt opgesteld. Dit houdt in dat zowel de X- als de Y-coördinaat als functie van dezelfde, derde, parameter t worden uitgedrukt. In de natuurkunde zien we vaak de tijd als parameter (vandaar de t). In de wiskunde is de parameter t bijna altijd een hoek in radialen.

Zo is de parametervorm van een cirkel met straal r en het middelpunt in de oorsprong:

x = r cost en y = r sint (t loopt van 0 tot 2π)

Uit deze twee vergelijkingen kan gemakkelijk de cartesische vorm $x^2+y^2=r^2$ gedistilleerd worden. De vergelijking van een ellips met het middelpunt in de oorsprong en met een halve lange as a en een halve korte as b is:

x = a cost en y = b sint

Het is eenvoudig programma's te ontwikkelen voor het tekenen van stelsels concentrische of excentrische cirkels. Ook stelsels ellipsen zijn, dankzij de parametervorm, eenvoudig te tekenen. We doen dit echter niet, omdat de figuren niet zo bijzonder zijn en vrij bekend.
In hoofdstuk 1 hebben we trouwens al een ellips met behulp van de parametervorm geprogrammeerd.

De volgende programma's tekenen figuren die u mogelijk nog niet kent en die zich voor computer graphics bijzonder goed lenen.

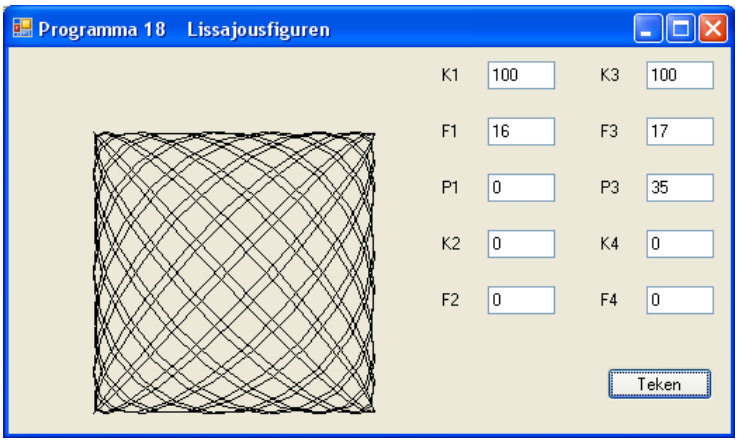
Programma 18 tekent een Lissajousfiguur. In het algemeen is de parametervorm voor een dergelijke figuur:

$x = k_1.\sin(f_1.t+p_1) + k_2.\cos(f_2.t)$
$y = k_3.\sin(f_3.t+p_3) + k_4.\cos(f_4.t)$

Hierin is p de fase, f de frequentie en k de amplitude. Om een mooie figuur te krijgen hebben we de volgende waarden gekozen:

$k_1=k_3=100$, $f_1=16$, $p_1=0$, $f_3=17$, $p_3=35$ en $k_2=k_4=f_2=f_4=0$

```
100 '    programma 18       LISSAJOUSFIGUREN
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 PRINT "TOETS  K1,F1,P1,K2,F2  IN "
150 INPUT K1,F1,P1,K2,F2 : PRINT
160 PRINT "TOETS  K3,F3,P3,K4,F4  IN "
170 INPUT K3,F3,P3,K4,F4
180 U=160 :V=160 : H=.5 : RD=4*ATN(1)/180
190 CLS
200 T=0 : GOSUB 1000
210 X1=INT(U+X+H) : Y1=INT(V-Y+H)
220 FOR W=1 TO 360
230     T=W*RD : GOSUB 1000
240     X2=INT(U+X+H) : Y2=INT(V-Y+H)
250     IF X2 < 0 OR X2 > 320 THEN 300
260     IF Y2 < 0 OR Y2 > 320 THEN 300
270     LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
280     X1=X2 : Y1=Y2
290 NEXT W
300 A$=INKEY$: IF A$="" THEN 300
310 CLS: KEY ON: END
320 '
1000 X=K1*SIN(F1*T+P1)+K2*COS(F2*T)
1010 Y=K3*SIN(F3*T+P3)+K4*COS(F4*T)
1020 RETURN
```




```
Private K1, F1, P1 As Single
Private K2, F2 As Single
Private K3, F3, P3 As Single
Private K4, F4 As Single

Private Sub BerekenLissajous(ByVal T As Single, ByRef X As Single, ByRef Y As Single)
    X = K1 * Math.Sin(F1 * T + P1) + K2 * Math.Cos(F2 * T)
    Y = K3 * Math.Sin(F3 * T + P3) + K4 * Math.Cos(F4 * T)
End Sub

Private Sub btnTeken_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ...
    Refresh()
End Sub

Private Sub frmProg18_Paint(ByVal sender As Object, ByVal e As ...PaintEventArgs) ...
    If txtF1.Text().Length() > 0 And txtF2.Text().Length() > 0 And _
        txtF3.Text().Length() > 0 And txtF4.Text().Length() > 0 And _
        txtK1.Text().Length() > 0 And txtK2.Text().Length() > 0 And _
        txtK3.Text().Length() > 0 And txtK4.Text().Length() > 0 And _
        txtP1.Text().Length() > 0 And txtP3.Text().Length() > 0 Then
        F1 = CSng(txtF1.Text()) : F2 = CSng(txtF2.Text())
        F3 = CSng(txtF3.Text()) : F4 = CSng(txtF4.Text())
        K1 = CSng(txtK1.Text()) : K2 = CSng(txtK2.Text())
        K3 = CSng(txtK3.Text()) : K4 = CSng(txtK4.Text())
        P1 = CSng(txtP1.Text()) : P3 = CSng(txtP3.Text())
        Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5, _
            RD As Single = 4 * Math.Atan(1) / 180
```

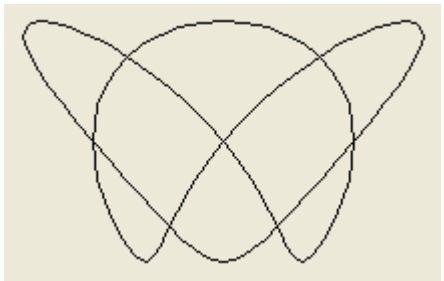
```
        Dim T As Single = 0, X As Single = 0, Y As Single = 0
        BerekenLissajous(T, X, Y)
        Dim X1 As Integer = Int(U + X + H), Y1 As Integer = Int(V - Y + H)
        Dim Eind As Boolean = False
        Dim W As Integer = 1
        Do Until (W > 360 Or Eind)
            T = W * RD : BerekenLissajous(T, X, Y)
            Dim X2 As Integer = Int(U + X + H), Y2 As Integer = Int(V - Y + H)
            If X2 < 0 Or X2 > 320 Then Eind = True
            If Y2 < 0 Or Y2 > 320 Then Eind = True
            e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
            X1 = X2 : Y1 = Y2
            W = W + 1
        Loop
    End If
End Sub
```

De subroutine BerekenLissajous() berekent de X en Y formules, maar geeft geen returnwaarde terug. Wel de parameterwaarden X en Y, vandaar dat die argumenten op ByRef staan.
Wat ik ook anders heb gedaan is de lusstructuur. In plaats van een For lus heb ik een Do Until lus gebruikt. Dat is handig om een boolean variabele te controleren. De boolean variabele Eind wordt op True gezet als de X2 en Y2 waarden buiten de grenzen vallen. Met een For lus is het onmogelijk om met een booleaanse waarde netjes de lus te beëindigen, anders had ik de For lus moeten beëindigen met een Exit For. Zulke Basic statements maakt het makkelijk om code ongestructureerd te maken. Daarom maak ik er geen gebruik van. Waarom moeilijk doen als het makkelijk kan?!

Experimenteer met het programma! Zoek zelf de juiste waarden voor de constanten zodat u mooie figuren krijgt. Natuurkundigen zullen u hierbij graag helpen; zij weten welke waarden u moet nemen!

Programma 19 presenteert een niet alledaagse figuur. Deze figuur wordt dikwijls de 'vliegenkopfiguur' genoemd. Om de vliegenkop horizontaal op het beeldscherm te krijgen moet de parameter t het interval 90^0-450^0 doorlopen.

```
100 '         programma 19         VLIEGENKOP
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 U=160: V=160: H=.5: K=30: RD=4*ATN(1)/180
150 W=90 : T=W*RD : GOSUB 1000
160 X1=INT(U+X+H) : Y1=INT(V-Y+H)
170 FOR W=93 TO 450 STEP 3
180     T=W*RD : GOSUB 1000
190     X2=INT(U+X+H) : Y2=INT(V-Y+H)
200     LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
210     X1=X2 : Y1=Y2
220 NEXT W
230 A$=INKEY$: IF A$="" THEN 230
240 CLS: KEY ON: END
250 '
1000 X=K*SIN(2*T)*(2.5+COS(3*T))
1010 Y=K*2*COS(3*T)
1020 RETURN
```



```
Private Sub BerekenVliegenkop(ByVal K As Single, _
                  ByVal T As Single, ByRef X As Single, _
                  ByRef Y As Single)
    X = K * Math.Sin(2 * T) * (2.5 + Math.Cos(3 * T))
    Y = K * 2 * Math.Cos(3 * T)
End Sub

Private Sub frmProg19_Paint(ByVal sender As Object, ByVal e As ...PaintEventArgs) ...
    Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5, _
        K As Single = 30, RD As Single = 4 * Math.Atan(1) / 180
    Dim W As Integer = 90, T As Single = W * RD
    Dim X As Single = 0, Y As Single = 0
```

```
    BerekenVliegenkop(K, T, X, Y)
    Dim X1 As Integer = Int(U + X + H), Y1 As Integer = Int(V - Y + H)
    For W = 93 To 450 Step 3
        T = W * RD : BerekenVliegenkop(K, T, X, Y)
        Dim X2 As Integer = Int(U + X + H), Y2 As Integer = Int(V - Y + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
End Sub
```

Het tekenen van de x- en y-as is achterwege gelaten om de 'kop' beter te laten uitkomen.

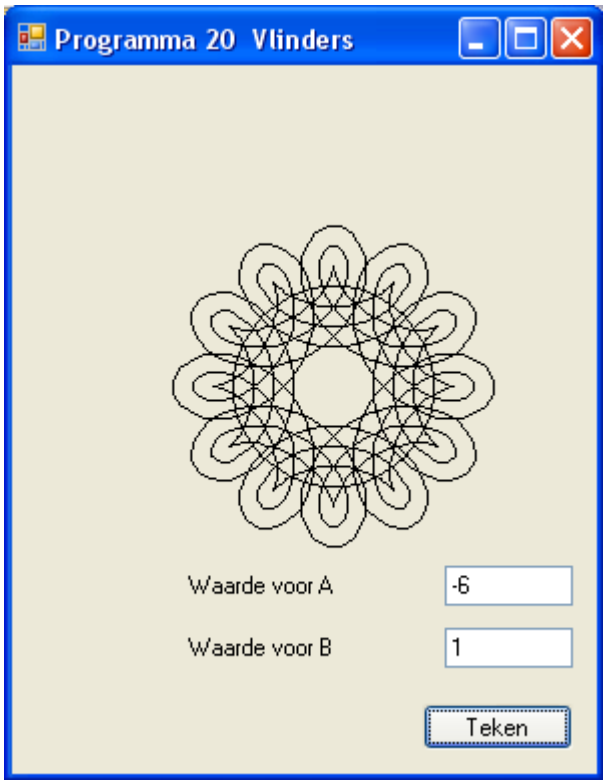
Voor een echte vliegenkop met 'ogen' kunt u kijken in de appendix, later in 'Grafisch programmeren in GW-BASIC'. Eerst komen nog meer mooie voorbeelden met wonderlijke functies.

Hoe geraffineerder u de functies x = f(t) en y = f(t) kiest, des te ongewoner worden de figuren. Probeer hier hoe creatief u kunt zijn.

De 'huiswiskundigen' bij computerfabrikanten hebben vaak hun eigen lievelingsfuncties. Die zie je dan ook vaak in een demonstratieprogramma optreden.

Programma 20 tekent een stelsel krommen. Hewlett-Packard publiceerde deze tekening enige tijd geleden. Onder het programma staat een tabel met waarden voor A en B, die heel mooie figuren opleveren. Probeer zelf andere combinaties. U zult verrukt zijn over wat u ziet!

```
100 '          programma 20           VLINDERS
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 U=160 : V=160 : H=.5 : RD=4*ATN(1)/180
150 KX=10 : KY=10
160 INPUT "TOETS  A EN B  IN "; A,B
170 CLS
180 FOR N=-3 TO 3
190    T=0 : GOSUB 1000
200    X1=INT(U+KX*X+H): Y1=INT(V-KY*Y+H)
210    FOR W=2 TO 360 STEP 2
220       T=W*RD : GOSUB 1000
230       X2=INT(U+KX*X+H):Y2=INT(V-KY*Y+H)
240       LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
250       X1=X2: Y1=Y2
260    NEXT W
270 NEXT N
280 A$=INKEY$: IF A$="" THEN 280
290 CLS: KEY ON: END
300 '
1000 X=(A+B)*COS(T)-N*B*COS((A+B)/B*T)
1010 Y=(A+B)*SIN(T)-N*B*SIN((A+B)/B*T)
1020 RETURN
```




```
Private A, B As Single

Private Sub BerekenVlinders(ByVal N As Integer, _
             ByVal T As Single, _
             ByRef X As Single, ByRef Y As Single)
    X = (A + B) * Math.Cos(T) - N * B * _
        Math.Cos((A + B) / B * T)
    Y = (A + B) * Math.Sin(T) - N * B * _
        Math.Sin((A + B) / B * T)
End Sub

Private Sub btnTeken_Click(...) ...
    Refresh()
End Sub
```

```
Private Sub frmProg20_Paint(ByVal sender As Object, ByVal e As ...PaintEventArgs) ...
    If txtA.Text().Length() > 0 And txtB.Text().Length() > 0 Then
        Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5
        Dim RD As Single = 4 * Math.Atan(1) / 180
        Dim KX As Integer = 10, KY As Integer = 10
        Dim X As Single = 0, Y As Single = 0
        A = CSng(txtA.Text()) : B = CSng(txtB.Text())
        For N As Integer = -3 To 3
            Dim T As Single = 0 : BerekenVlinders(N, T, X, Y)
            Dim X1 As Integer = Int(U + KX * X + H), Y1 As Integer = Int(V - KY * Y + H)
            For W As Integer = 2 To 360 Step 2
                T = W * RD : BerekenVlinders(N, T, X, Y)
                Dim X2 As Integer = Int(U + KX * X + H), Y2 As Integer = Int(V - KY * Y + H)
                e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
                X1 = X2 : Y1 = Y2
            Next
        Next
    End If
End Sub
```

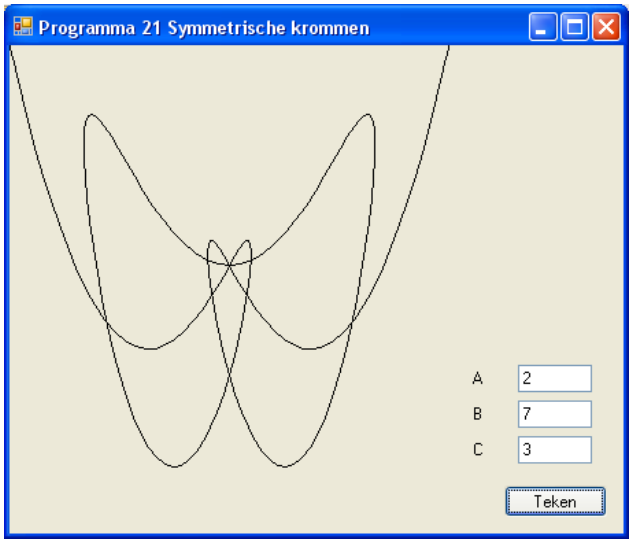
A	-6	-6	-8	4	4	6	4,5	.....
B	1	2	2	1	2	1	1,5	.....

Kies in deze programma's de stapgrootte voor hoek W (FOR W=...) gerust 1; dit duurt wat langer maar geeft mooiere tekeningen.

Het laatste programma uit dit hoofdstuk tekent 'supersymmetrische' figuren. Ook nu geven we een tabel met waarden voor de parameters A, B en C. Het is ongelooflijk hoe de figuur totaal verandert als we andere waarden voor één of meer parameters kiezen. Het idee voor **programma 21** is afkomstig uit het Amerikaanse tijdschrift Creative Computing. Het kan soms even duren voor het tekenen wordt hervat. Druk niet direct op een toets, want dan wordt na het tekenen direct teruggeschakeld naar het gewone scherm en bent u uw tekening kwijt. Druk pas op een toets als u zeker weet dat de tekening af is.

Dit probleem hadden we vroeger heel vaak. Tegenwoordig werken computers zeer snel. In Visual Basic zult u hier geen last van hebben.

```
100 '   programma 21      SYMMETRISCHE KROMMEN
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "TOETS  A,B,C  IN "; A,B,C
150 U=160: V=160: H=.5: RD=4*ATN(1)/180: K=160
160 CLS
170 X1=U : Y1=V
180 FOR W=1 TO 360
190     T=W*RD : R=K*SIN(C*T)
200     X=R*COS(A*T) : X2=INT(U+X+H)
210     Y=R*SIN(B*T) : Y2=INT(V-Y+H)
220     LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
230     X1=X2 : Y1=Y2
240 NEXT W
250 A$=INKEY$: IF A$="" THEN 250
260 CLS: KEY ON: END
```



```
Private Sub btnTeken_Click(...) ...
    Refresh()
End Sub

Private Sub frmProg21_Paint(...) ...
    If txtA.Text().Length() > 0 And txtB.Text().Length() > 0 And _
            txtC.Text().Length() > 0 Then
        Dim A As Single = CSng(txtA.Text()), B As Single = CSng(txtB.Text()), _
            C As Single = CSng(txtC.Text())
        Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5, _
```

```
            RD As Single = 4 * Math.Atan(1) / 180, K As Integer = 160
        Dim X1 As Integer = U, Y1 As Integer = V
        For W As Integer = 1 To 360
            Dim T As Single = W * RD
            Dim R As Single = K * Math.Sin(C * T)
            Dim X As Single = R * Math.Cos(A * T)
            Dim X2 As Integer = Int(U + X + H)
            Dim Y As Single = R * Math.Sin(B * T)
            Dim Y2 As Integer = Int(V - Y + H)
            e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
            X1 = X2: Y1 = Y2
        Next
    End If
End Sub
```

Tabel voor mooie figuren.

A	2	6	4	1	3	2	.....
B	7	6	6	1	3	2	.....
C	3	4	1	4	5	9	.....

In de volgende nieuwsbrief komt er een nieuw hoofdstuk; het tekenen van driedimensionale figuren.

Websites programmeren met Crimson (2).

In de vorige nieuwsbrief heb ik u laten zien dat we eerst het geraamte moeten maken. Het sjabloon voor de index pagina. Maar ook de andere pagina's moeten hetzelfde geraamte hebben. Zorg er dus voor dat u het sjabloon goed bewaard op een plek waar u het niet per ongeluk overschrijft.

De index.html pagina.

Hieronder ziet u nogmaals het geraamte die u in elke nieuwe pagina moet gebruiken. Dit moet als een sjabloon bestand opgeslagen worden zodat u dit niet telkens hoeft over te nemen. Onthoud dat dit alleen voor html bestanden geldt. Niet voor andere bestanden zoals css bestanden.

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>Geef hier uw titel</title>
    </head>
    <body>

    </body>
</html>
```

Tekst en tags

XHTML documenten bestaan uit teksten en tags (codes):

- De tekst is gewoon de inhoud van de webpagina: datgene wat u graag via internet wilt vertellen. Bijvoorbeeld een zin als: Jan is jarig.
- Tags zijn de codes die u als ontwerper aanbrengt om de tekst te structureren of vorm te geven. Ze worden door de browser geïnterpreteerd. Om het bericht dat Jan zijn verjaardag viert extra nadruk te geven kunt u deze bijvoorbeeld classificeren als koptekst: **<h1>Jan is jarig</h1>**. De tag h1 zorgt ervoor dat de tekst in de grootste vorm weergegeven wordt. Hoe hoger het nummer achter de h-tag hoe kleiner de tekst wordt.

U hebt al geleerd dat XHTML documenten bestaan uit tekst en tags. Gewone tekst in een pagina wordt rechtstreeks door de browser weergegeven. Daar hoeft u verder niets aan te doen. Wilt u echter iets speci-

aals met die tekst, bijvoorbeeld als koptekst of als hyperlink weergeven, dan moet u dit aangeven door tags te gebruiken. Alle tags staan tussen < en >.

In de basis zien alle tags in XHTML er hetzelfde uit.
- Er is een openingstag die het begin van de speciale tekstopmaak aangeeft.
- Er is een sluittag die het einde van de speciale tekstopmaak aangeeft.
- De naam van de tag wordt in kleine letters geschreven.

Alle tekst die tussen de open- en sluittag staat, wordt door de tag beïnvloed.
Schematisch gezien zien alle tags er daarom op de volgende manier uit.

```
<tag>Dit is de tekst die beïnvloed wordt</tag>
```

Het woord **tag** wordt in echte XHTML documenten uiteraard vervangen door de gewenste code. Als u bijvoorbeeld een koptekst wilt afbeelden, gebruikt u daarvoor de tag **<h1>…</h1>**:

```
<h1>Dit is een koptekst</h1>
```

Het einde van de tag wordt altijd aangegeven met een code die dezelfde naam heeft als de begintag, maar dan voorafgegaan door een voorwaartse slash (/). Daarom wordt het einde van de koptekst aangegeven met </h1>. Spreek uit: 'slash h1'. Er zijn talloze tags voor allerlei andere effecten, waaronder bijvoorbeeld alinea's, cursief, koppen, paginatitels, lijsten en nog veel meer.

Vereenvoudigd kan worden gesteld dat XHTML staat voor al deze tags samen. En als u alle tags kent, bent u een volleerd webdesigner! Zo simpel is dat. Nou ja, in werkelijkheid is het natuurlijk niet helemáál zo simpel, maar het principe is telkens hetzelfde.

Attributen

XHTML tags vertellen de browser dus hoe de komende tekst of code behandeld moet worden. Bij veel tags kunt u bovendien een aantal extra kenmerken opgeven. Stel dat u bijvoorbeeld een tekst niet alleen als koptekst in het document wilt tonen, maar bovendien wilt aangeven dat hij gecentreerd tussen de linker- en rechtermarge van het scherm moet staan. Dan kunt u dit extra kenmerk aangeven met een attribuut. Een attribuut bestaat altijd uit een naam en een waarde:
- In het geval van centreren is de naam het attribuut **align**.
- De waarde van het attribuut is **center**.

Gecombineerd ziet een gecentreerde koptekst er in de code als volgt uit (het attribuut is vet weergegeven):

```
<h1 align="center">Dit is een gecentreerde koptekst</h1>
```

Voluit geschreven met het sjabloon zou u de koptekst als volgt plaatsen:

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>Een basis webpagina</title>
    </head>
    <body>
        <h1 align="center">Dit is een gecentreerde koptekst</h1>
    </body>
</html>
```

De koptekst moet dus altijd binnenin de body tag staan en u mag meerdere kopteksten gebruiken.
Figuur 1 geeft het voorbeeld als u de html code in Crimson start.

Figuur 1

Ga naar het menu Tools en klik op 'View in Browser' als u de code uit wilt voeren. U hoeft niet speciaal internet op online te zetten. Op offline werkt de browser net zo goed. Zelfs complete website projecten kunt u offline in Crimson uitvoeren.

Zoals u gemerkt heeft worden attributen gebruikt om de functionaliteit van de tags te vergroten. Hieronder staan een aantal punten opgesomd die u vertellen wat de kenmerken van de attributen zijn.

- Attributen worden altijd in de openingscode opgegeven.
- Het attribuut wordt met een spatie gescheiden van de tagnaam.
- Een tag kan meerdere attributen bevatten die elk een bepaald kenmerk beschrijven. De attributen worden met spaties van elkaar gescheiden.
- Tussen de naam en de waarde van het attribuut staan géén spaties. Een goede notatie is dus **align="right"**. Een foute notatie is **align = "right"**.
- De namen van attributen worden in kleine letters geschreven.
- Zodra de browser een tag herkent, probeert hij ook het bijbehorende attribuut (indien aanwezig) te interpreteren. Als het attribuut pas in de sluitcode ter sprake zou komen, zou het immers te laat zijn. Dan staat de tekst al op het scherm.
- De waarde van attributen wordt vaak tussen dubbele aanhalingstekens gezet zoals **align="center"**. Dit is verplicht in XHTML.

Enkele andere voorbeelden van attributen.

Om alvast een indruk te krijgen van het gebruik van tags en attributen kunt u de volgende voorbeelden bekijken.

- ** ... **
  er wordt een alinea gemaakt (** **) waarvan de tekst rechts wordt uitgelijnd met het attribuut **align="right"**.
- **<table id="telefoonnummers">...</table>**
  op de pagina wordt een tabel gedefinieerd (**<table>**) die wordt aangeduid met de naam telefoonnummers (**id="telefoonnummers"**).
- ****
  er wordt een afbeelding ingevoegd (**<img />**) met drie attributen. De bestandsnaam is **logo.gif** (**src="logo.gif"**) en de afbeelding is 400 bij 200 pixels groot (**width="400" height="200"**).

Lege elementen: een andere manier om tags te sluiten.

In het laatste voorbeeld hierboven ziet u een andere manier van het afsluiten van een tag. De tag om een afbeelding in een webpagina te plaatsen heet **img**. Dit is de afkorting van *image*.

Volgens de regels zou u dus geneigd zijn om een afbeelding op de pagina te plaatsen met **<img>...</img>**. De tag **<img>** is echter een voorbeeld van een *leeg element*. Er staat geen tekst tussen de begintag en de sluittag. U wilt immers dat een afbeelding wordt ingevoegd op de pagina. En om lege elementen correct af te sluiten wordt in XHTML vaak een verkorte notatie gebruikt. Het element wordt dan afgesloten door aan het einde van de openingstag direct de slash en de punthaak te plaatsen />. Oftewel:

- **<img></img>** is niet fout, maar wordt weinig gebruikt;
- **<img />** heeft de voorkeur; het is een verkorte schrijfwijze.

Let daarbij weer op dat éérst een spatie wordt geschreven achter de tagnaam en vervolgens de />. Erg oude browsers kunnen in de war raken als deze spatie ontbreekt.

De tag <body>.

Nadat u een titel hebt opgegeven binnen in de header, kunt u verder alle overige tekst, afbeeldingen en codes kwijt in het middendeel, de 'body' van de webpagina. Hier geeft u de eigenlijke pagina vorm. Het is

dan ook niet verwonderlijk dat dit deel omsloten wordt door tags met de naam **<body>** en **</body>**. De standaardstructuur ziet er dan uit zoals de code van Figuur 1. Als u in het sjabloon of in uw eigen index.html pagina de body erbij hebt, is dit voor elke webpagina de complete 'kapstok'. Ik noem het een kapstok, omdat elke pagina aan de indexpagina opgehangen kan worden. Op de plek <body>...</body> wordt vervolgens de inhoud van de pagina geschreven. Hier maakt u verder dan ook uitgebreid kennis mee.

Teksten en alinea's toevoegen.

Eigenlijk is niets zo gemakkelijk als gewone tekst toevoegen aan het webdocument. Alle tekst die u tussen de tags **<body>** en **</body>** typt, wordt door de browser rechtstreeks op het scherm gezet.

Enter, spaties en tabs wordt genegeerd.

Houd er echter rekening mee dat de browser regeleinden die u maakt, door in de editor op Enter te drukken, negeert. Ook extra tabs en spaties worden door de browser genegeerd. Alle tekst wordt dus letterlijk achter elkaar op het scherm geplakt. Doe dus geen moeite om met tabs een mooie tabel te maken in de editor, de browser gooit alles op één hoop.

De enige reden om in uw broncode met tabs en eventueel spaties te werken is om uw HTML code beter leesbaar te maken. Het kan bij het 'teruglezen' van uw pagina helpen wanneer logische blokken code ingesprongen staan ten opzichte van voorgaande code. Later, als het gebruik van het tabel behandeld wordt, zult u zien dat dit handig is.

Alinea's aangeven met .

Om in een webpagina een nieuwe alinea te beginnen, plaatst u de tag ** ** op de plek waar de nieuwe alinea moet beginnen. Het einde van een alinea wordt aangegeven met ** **. De letter **p** is hierbij afkomstig van *paragraph*. Voorafgaand aan elke alinea voegt de browser altijd automatisch een witregel toe.

De volgende HTML code geeft het resultaat zoals u ziet bij Figuur 2:

Figuur 2 Gebruik van de tag om de tekst in alinea's te verdelen.

```
<body>
    <p>Om in een webpagina een nieuwe alinea te beginnen, plaatst u de paragraph tag op
    de plek waar de nieuwe alinea moet beginnen. </p>
    <p>De browser voegt zelf een lege regel toe, dus u hoeft niet zoals in een
```

```
        tekstverwerker twee keer op Enter te drukken om een lege regel in te voegen. Eén
        tag doet het werk.</p>




        <p>Meerdere lege regels in de broncode worden door de browser verder genegeerd. Er
        wordt gewoon één alinea met een standaard witruimte van gemaakt.</p>
    </body>
```

Attributen voor <p>.

Ook de tag **<p>** kent attributen om de tekst uit te lijnen. Net zoals bij de tags voor koppen zijn dit de attributen **align="right"**, **align="left"** en **align="center"**. Het attribuut is van toepassing op de tekst die volgt na de tag.

Centreren met .

In plaats van het attribuut **align="center"** voor de tag **<p>** kunt u ook een andere manier gebruiken om tekst te centreren: ** ** en ** **. Alle objecten binnen deze tag worden vervolgens gecentreerd. De code **<p align="center">** is maar op één alinea van toepassing. In de volgende alinea is automatisch de standaard uitlijning weer van kracht. Door de algemene tag ** ** te gebruiken die een bepaald deel op de pagina afbakent (** ** is de afkorting voor *division*), kunt u alle elementen binnen die div centreren.

Het regeleinde
.

Soms zult u in een tekst doelbewust op een volgende regel willen beginnen, echter zonder een nieuwe alinea te beginnen met bijbehorende witregel. Dit is bijvoorbeeld het geval bij adressen, gedichten of recepten. Het zou lelijk zijn om dan de tag **<p>** te gebruiken, omdat er anders teveel witruimte tussen de regels komt. Hiervoor is de tag **
** (van BReak) ontworpen.

Speciale tekens.

Als u al eens wat html pagina's geschreven heeft dan is het u waarschijnlijk al eens opgevallen dat ook XHTML code zelf afgebeeld wordt, onder andere de tags. Dit zonder dat deze kennelijk door de browser werd geïnterpreteerd! Op het scherm staat dan bijvoorbeeld letterlijk de tag **** in plaats van dat de browser dit als vette tekst opvatte. Hoe kan dat?

Entiteitsnamen

Het antwoord is dat er speciale codes zijn ontworpen voor gebruik van deze symbolen. Als u bijvoorbeeld het groter-dan teken en kleiner-dan teken wilt gebruiken, zou u die niet rechtstreeks kunnen typen, omdat de browser verwacht dat er een tag achter het teken < komt. In plaats van op het toetsenbord het teken < te typen, gebruikt u de code **<** (een afkorting van *less than*, kleiner dan). Voor het teken > gebruikt u de code **>** (*greater than*, groter dan). Deze code is de zogenoemde entiteitsnaam voor het teken < of >.

Elk apart symbool dat niet in het normale alfabet van A-Z, a-z en 0-9 voorkomt, heeft een eigen entiteitsnaam gekregen. Een entiteitsnaam begint altijd met een ampersand (**&**) en eindigt met een puntkomma (**;**). De entiteitsnaam tracht het teken zo goed mogelijk te omschrijven. Zo is de entiteitsnaam van een o-umlaut (**ö**) bijvoorbeeld **ö**.

Tip! Een goed overzicht van speciale tekens (zowel entiteitsnamen als tekens uit de ANSI tekenset) vindt u op **www.w3schools.com/html/html_entitiesref.asp** en ook op **www.handleidinghtml.nl/html/karakters/karakters03.html**.

Een voorbeeld van entiteitsnamen

Het volgende voorbeeld maakt duidelijk hoe de browser speciale tekens behandelt:

De code **De tag
 breekt een regel af** geeft als resultaat:

```
De tag
breekt een regel af
```

De code **De tag
 breekt een regel af** geeft als resultaat:

```
De tag <br /> breekt een regel af
```

Werken met CSS, een inleiding.

Met CSS (Cascading Style Sheets) kunnen we veel doen om tekst op te maken. Ook vervangt CSS een heleboel html tags voor bijvoorbeeld vet en cursief.

Open het sjabloon of uw index pagina en voeg een lege regel in na de titel.

Er zijn verschillende manieren om stijlbladen te koppelen aan een webdocument. Voor het gemak laat ik u zien hoe we een stijlenblok definiëren in de header van het document. Hiervoor kent XHTML de tag **<style>...</style>**. Met het attribuut **type="..."** wordt aangegeven dat het stijlenblok CSS regels bevat.

Typ de openingstag van het stijlblok **<style type="text/css">**.
Druk een paar keer op Enter om enkele lege regels te creëren.
Typ dan de sluittag **</style>**. Tussen deze tags wordt zodadelijk het stijlenbestand geschreven. Het document ziet er nu uit zoals onderstaand voorbeeld.

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>Een basis webpagina</title>
        <style type="text/css">


        </style>
    </head>
    <body>
        <h1 align="center">Dit is een gecentreerde koptekst</h1>
    </body>
</html>
```

Verander in de body de tag **<h1>** door het attribuut te verwijderen.

Plaats tussen **<style...>** en **</style>** de volgende regels:

```
h1
{
    align: center;
    color: blue;
}
p
{
    color: red;
}
```

Typ in de body onder de regel <h1> ... </h1> een alinea met de tag ... .

Start de pagina. Als het goed is moet de koptekst gecentreerd en blauw zijn met een rode alineatekst eronder.

Meer over CSS komt in de volgende nieuwsbrief.

Heeft u nog niet de editor Crimson? U kunt op internet deze gratis downloaden door het programma via google te vinden, maar u kunt het programma ook op de BASIC website vinden.

Marco Kurvers

De listings in de kwartaalbestanden.

Om uit de listings geen fouten te maken, staat hieronder nogmaals hoe u de listings in de kwartaalbestanden kunt gebruiken.
Als u de programmavoorbeelden, de listings genoemd, wilt gebruiken dan kunt u de tekst kopiëren die in de kwartaalbestanden staan.
Alles meteen kopiëren en in een codevenster plakken kan werken, maar houd er rekening mee dat delen van de code soms in de voorbeelden worden afgekort. De code die niet nodig is schrijf ik met drie punten '...' om de regels af te korten, zodat toch de listing netjes uitgelijnd is en goed leesbaar blijft.
Voor de Visual Basic gebruikers heb ik daarom de volgende tips:

- maak niet zelf de private event subs aan maar kies de events die voor de code nodig zijn door in de object properties op de event te dubbelklikken;
- kopieer de code tussen de regels **Private Sub** en **End Sub** en plak het in het codevenster in de juiste Private Sub event;
- er kunnen in de programmavoorbeelden ook normale subroutines staan (geen events), die kunt u helemaal kopiëren en in het codevenster plakken;
- staat er geen **Private Sub** en **End Sub** dan is het een één listing die niet uit meerdere subroutines bestaat en kunt u de code helemaal kopiëren en plakken in de juiste Private Sub event.

Let op! Maakt de code gebruik van controls, plaats dan eerst op het formulier de controls zoals u op de voorbeelden ziet. Pas dan kunt u de code kopiëren, plakken en uitvoeren. Anders zullen er fouten ontstaan, omdat de controls nog niet herkend worden.

Heeft u niet de Visual Basic .NET versie, gebruik dan uw BASIC versie. Pas de code aan voor de BASIC versie als het niet werkt. Komt u er nog niet uit, raadpleeg dan eventueel de converteerlijsten die u op de BASIC IG website kunt vinden of stuur een bericht naar mij. Geef voldoende informatie over het probleem zodat ik de juiste oplossing naar u terug kan sturen. Heeft de oplossing een voordeel voor de converteerlijsten dan pas ik die gelijk aan.

Mededeling converteerlijsten.

Graag had ik nu de converteerlijsten willen weergeven, maar helaas heb ik ze nog niet klaar. Ik hoop dat ik ze in de volgende nieuwsbrief wel klaar heb. Mijn excuses daarvoor.

Cursussen

Qbasic: Cursus, lesmateriaal en voorbeelden op CD-ROM € 6,00 voor leden. Niet leden € 10,00.

QuickBasic: Cursusboek en het lesvoorbeeld op diskette, € 11,00 voor leden. Niet leden € 13,50

Visual Basic 6.0: Cursus, lesmateriaal en voorbeelden op CD-ROM, € 6,00 voor leden. Niet leden € 10,00

Basiscursus voor senioren, Windows 95/98,
Word 97 en internet voor senioren, (geen diskette). € 11,00 voor leden. Niet leden € 13,50

Computercursus voor iedereen: tekstverwerking met Office en eventueel met VBA, Internet en program-meertalen, waaronder ook Basic, die u zou willen leren.
Elke dinsdag in Buurthuis Bronveld in Barneveld van 19:00 uur tot 21:00 uur. Kosten € 5,00 per week.
Meer informatie? Kijk op 'http://www.i-t-s.nl/rdkcomputerservice/index.php' of neem contact op met mij.

Software

Catalogusdiskette,	€ 1,40 voor leden. Niet leden € 2,50
Overige diskettes,	€ 3,40 voor leden. Niet leden € 4,50
CD-ROM's,	€ 9,50 voor leden. Niet leden € 12,50

Hoe te bestellen

De cursussen, diskettes of CD-ROM kunnen worden besteld door het sturen van een e-mail naar penm@basic-gg.hcc.nl en storting van het verschuldigde bedrag op:

ABN-AMRO nummer 49.57.40.314
HCC BASIC ig
Haarlem

onder vermelding van het gewenste artikel. Vermeld in elk geval in uw e-mail ook uw adres aangezien dit bij elektronisch bankieren niet wordt meegezonden. Houd rekening met een leveringstijd van ca. 2 weken. Teksten en broncodes van de nieuwsbrieven zijn te downloaden vanaf onze website (http://www.basic.hccnet.nl). De diskettes worden bij tijd en wijlen aangevuld met bruikbare hulp- en voorbeeldprogramma's.
Op de catalogusdiskette staat een korte maar duidelijke beschrijving van elk programma.

Alle prijzen zijn inclusief verzendkosten voor Nederland en België.

Vraagbaken

De volgende personen zijn op de aangegeven tijden beschikbaar voor vragen over programmeerproblemen. Respecteer hun privé-leven en bel alstublieft alleen op de aangegeven tijden.

Waarover	Wie	Wanneer	Tijd	Telefoon	Email
Liberty Basic	Gordon Rahman	ma. t/m zo.	19-23	(023) 5334881	grahman@planet.nl
MSX-Basic	Erwin Nicolai	vr. t/m zo.	18-22	(0516) 541680	basic@lordthanatos.com
PowerBasic CC	Fred Luchsinger	ma. t/m vr.	19-21		f.luchsinger@kader.hcc.nl
QBasic	Jan v.d. Linden				j.vd.linden@kader.hcc.nl
QuickBasic					
Visual Basic voor Windows	Jeroen v. Hezik	ma. t/m zo.	19-21	(0346) 214131	j.a.van.hezik@kader.hcc.nl
Visual Basic .NET	Marco Kurvers	do. t/m zo.	19-22	(0342) 424452	m.a.kurvers@hccnet.nl
Basic algemeen, zoals VBA	Marco Kurvers	do. t/m zo.	19-22	(0342) 424452	m.a.kurvers@hccnet.nl
Office					
Web Design, met XHTML en CSS