Nieuwsbrief

17de jaargang september 2010 Nummer 3

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Het Bestuur	Penningmeester gezocht: 39

Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Willem Gobel	0118-850837	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

Het werkboek is de map in Excel waar we mee werken. We kunnen meerdere werkboeken openen, maar helaas werken ze niet samen, of toch wel? We kunnen zien wat we nog meer met het Range object kunnen doen en waar het Range type voor nodig is.

Er bestaan verschillende soorten gegevenstypen. Hoeveel kent BASIC er?

De Intermezzo heeft u alles laten zien hoe u een Werknemers project kunt maken. De conclusie geeft een lijst van informatie en het programmavoorbeeld in code weer.

Wat voor sprites bestaan er? Er zijn een aantal soorten waar ik wat over vertel en afbeeldingen van Game Maker laat zien.

Marco Kurvers

BASIC cursus: VBA met Excel (3).

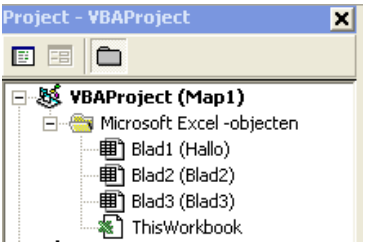
De werkbladen zitten in één werkboek, een conclusie die we moeten onthouden. Elk nieuw werkboek begint met drie nieuwe werkbladen, Blad1, Blad2 en Blad3.

Het werkboek.

Als we iets in Blad1 willen typen, zoals het wijzigen van de tekst *Blad1*, dan zal men haast denken dat gelijk de naam van het blad wordt gewijzigd. Dit is echter niet het geval, zie de Figuren 1 en 2.

Figuur 1.
Hier is *Blad1* gewijzigd in *Hallo*.

Dit kunt u ook doen door simpel op de tab *Blad1* te klikken. Zorg ervoor dat de cursor erin verschijnt. Zo ja, dan kunt u de naam wijzigen. Is er geen cursor maar wel de naam geselecteerd dan kunt u ook typen. De oude naam zal direct verdwijnen.



Figuur 2.
Hier ziet u het project van Map1. U kunt zien dat inderdaad de naam van object Blad1 niet de ingetoetste naam is op het werkblad, maar wel de tekst tussen haakjes staat vermeldt.

De werkbladnamen zelf kunnen niet worden gewijzigd. Het zijn objectnamen waar we mee werken. We kunnen ook geen bladinstanties maken. Onderstaande code bevat een fout.

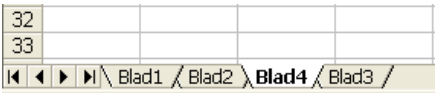
```
Private Sub Test()
    Dim objBlad As Blad1
    objBlad.Cells(1, 1) = 10  'hier is de fout
End Sub

Private Sub Worksheet_Activate()
    Test
End Sub
```

VBA zal een foutmelding geven dat de instantie van Blad1 niet geset is, of anders genoemd niet is gecreëerd. Een objectinstantie moet gecreëerd worden met een `New` sleutelwoord, maar als we dat doen dan zal object `Blad1` niet meer in de lijst staan van objecttypen.
We kunnen ons alleen bezig houden met de materialen van VBA. Alleen Excel materialen en eigen gemaakte materialen zijn mogelijk. We kunnen ook materialen importeren, maar

daar ga ik het voorlopig nog niet over hebben.

Nu we gezien hebben dat we niet in code bladinstanties kunnen creëren, moeten we gebruik maken van de mogelijkheden van VBA, zoals de methode `Add()` waarmee we nieuwe werkbladen kunnen invoegen. Eerst gaan we kijken hoe we dat in het werkboek zelf doen.

- Klik met de rechter muisknop op Blad3, eventueel hebt u een andere naam als laatste blad. Er verschijnt een menu.
- Klik op de menukeuze 'Invoegen'. Figuur 3 zal het resultaat laten zien.



Figuur 3.
Resultaat na het invoegen van een nieuw blad.

Wilt u Blad4 achter Blad3 hebben, sleep dan de naam naar rechts. De tab zal automatisch meegaan. Helaas is er geen manier om direct werkbladen aan de rechterkant toe te voegen. We kunnen het wel doen in de code van VBA, maar dan moeten we echter het werk zelf doen. Bovenstaande methode gebruiken door middel van een macro zal voor onderstaande code zorgen die door de macro is opgenomen:

```
Sub NieuwBlad()
'
' NieuwBlad Macro
' De macro is opgenomen op 26-5-2010 door TRONICA.
'

'
    Sheets("Blad3").Select
    Sheets.Add
End Sub
```

Omdat we weer een selectie hebben gemaakt, zal weer de methode `Select` in de macro `NieuwBlad` worden opgenomen. Pas daarna zal de `Add` methode worden uitgevoerd die eigenlijk het nieuwe blad toevoegt, maar door de selectie helaas het blad invoegt.

De `Add()` methode heeft zelf parameters om aan te geven hoe we een nieuw werkblad willen plaatsen. U ziet de syntaxis en de werking van de parameters hieronder omschreven:

[objectvariabele =]`Sheets.Add([Before],[After],[Count],[Type]) As Object`

Before	Zorgt ervoor dat de bladen voor het opgegeven blad ingevoegd worden.
After	Zorgt ervoor dat de bladen achter het opgegeven blad ingevoegd worden. Is het opgegeven blad het laatste blad dan zullen de bladen toegevoegd worden.
Count	Hier kunt u opgeven hoeveel werkbladen u wilt invoegen of wilt toevoegen.
Type	Hier kunt u opgeven om wat voor object het gaat. Voor een werkblad is dat type nr. 1. Raadpleeg de help van Excel voor meer informatie over verschillende types. Voor werkbladen kunt u deze parameter weglaten.

Onderstaande voorbeeld voegt 3 nieuwe bladen toe achter Blad7:

```
Sheets.Add After:=Blad7, Count:=3, Type:=1
```

Hier kunt u dus het type nr. weglaten.
Wilt u maar één werkblad toevoegen dan kunt u parameter `Count` weglaten.

De besturing in de mapobjecten.

Wat u hierboven gelezen hebt is ook besturing. Er bestaan methoden die u kunt gebruiken die niet als opties in Excel te vinden zijn, zoals de extra parameters in de `Add()` methode. De mappen zelf zijn geen objecten zoals eerder vernomen. Bekijk eens onderstaande macro:

```
Sub Map2ToMap1()
'
' Map2ToMap1 Macro
' De macro is opgenomen op 26-5-2010 door TRONICA.
'

'
    Application.CutCopyMode = False
    ActiveCell.FormulaR1C1 = "bbb"
    Range("A3").Select
    Selection.Copy
    Windows("Map1").Activate
    Range("A6").Select
    ActiveSheet.Paste
End Sub
```

We kunnen ons gelijk richten op een nieuw object, en wel het `Windows` object. Wat doet die hier in VBA?
Dankzij het object kunnen we elke map die geopend is activeren. Het clipboard zal na het activeren van de andere map niet leeg worden gemaakt, zodat we de gegevens uit de vorige map in de nieuwe map kunnen plakken. Maar zoals we van het opnemen van macro's weten, missen we weer wat in de code. Het grote nadeel van een macro is altijd dat actieve en geselecteerde code wordt aangemaakt voordat het eigenlijke werk wordt uitgevoerd. U kunt in het voorbeeld zien dat de cellen geselecteerd worden, maar we kunnen niet zien in welke bladobjecten die uitgevoerd worden. Evenzo weten we niet in welk blad het geplakt wordt.

Dit werk hoeft u niet op te nemen in macro's. Het is aan te raden om zelf subroutines te schrijven met de aangegeven bladen. Gebruik ook niet `ActiveCell` en `ActiveSheet`, maar geef zelf de bladen en cellen op van waar het gekopieerd en geplakt moet worden.

Tip! Heeft er iemand een document met een map die bestaat uit meerdere werkbladen die meerdere malen toegevoegd of bewerkt worden, dan is het niet verkeerd om code te

gebruiken die eruit ziet als bovenstaande voorbeeld. Vaste bladobjecten gebruiken kan ook wel eens problemen opleveren als plotseling het blad niet meer bestaat. Dan is het wel beter om vanuit het actieve blad de methoden te programmeren om fouten te voorkomen. Wie kan zeggen of Blad7 er wel is!? Dit noemen we *dynamisch* programmeren en dat is altijd de standaardstructuur van de macro.

De besturing in het ThisWorkbook object.

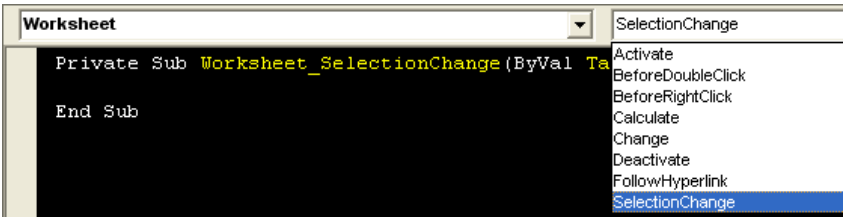
Nog altijd hebben we een object dat de meester is van alle Excel VBA objecten, want hier kunnen we het project dat we kunnen maken netjes initialiseren en opmaken. Wat kunnen we allemaal in het ThisWorkbook object doen?

- Variabelen declareren die we in de map nodig hebben. Nog altijd wil ik zeggen: declareer alleen variabelen die in de hele map belangrijk zijn. Het is echt zeer belangrijk om daar op te letten, want alles wat we in één object plaatsen zou iemand anders op een gegeven moment er geen touw meer aan vast kunnen knopen hoe het project in elkaar zit. Declareer dus ook lokale variabelen in de objecten waar ze horen.
- De werkbladen initialiseren in de `Workbook_Open()` event. De event die niet in de bladobjecten bestaat.
- Een eventuele database inladen en de gegevens uitlezen naar de bladobjecten. Voor een database hebben we een variabele nodig zodat we de database kunnen beheren. Die variabele moet overal bekend zijn.
- Code programmeren in de events die nodig zijn. U kunt bijvoorbeeld een bericht tonen als de gebruiker een nieuw werkblad aanmaakt. Het bericht plaatst u in de `Workbook_NewSheet()` event. Wilt u het ook weergeven als de gebruiker op een ander geopend werkblad klikt, gebruik dan de `Workbook_SheetActivate()` event.

Het is teveel om op te noemen. De namen van de events geven aan waar ze voor dienen. Zo kunt u snel een event vinden bij berekeningen uitvoeren door in de eventlijst te zoeken naar `SheetCalculate()`.

De bladobjecten en het werkboekobject (ThisWorkbook) zijn verschillend in de aantal events die ze hebben. Figuur 4 laat een deel van de editor zien met een geopende eventlijst van een blad.

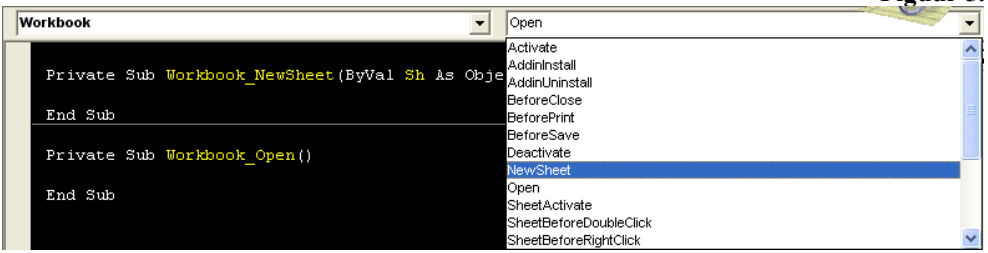
Figuur 4.



Figuur 5 laat een deel van de editor zien met een geopende eventlijst van ThisWorkbook met nu een schuifbalk erbij. De bladobjecten hebben minder events dan het werkboek heeft,

maar daar tegenover kunnen we genoeg code programmeren om goede bestuurbare bladen te maken .

Figuur 5.



Het Range type.

Het Range object kunnen we gebruiken om een gebied van een werkblad te bewerken. Als we de parameters van het object intoetsen dan zien we dat het Range object ook weer een Range teruggeeft, dat wil zeggen; we krijgen een lokaal gebied terug en mag worden toegekend aan een variabele die gedeclareerd is van het type Range.

We kunnen echter geen instanties van objecten maken. Hoe kunnen we dan de variabele gebruiken? Onderstaande twee voorbeelden zijn verschillend. Het eerste voorbeeld is echter fout, zoals ik eerder heb laten zien over het sleutelwoord `New` die we helaas in VBA niet voor dit doel kunnen gebruiken.

```
Dim objRange As Range
objRange.Cells(1, 1) = 50          'Hier is de fout

Dim objRange As Range
Set objRange = Range("A1:A5")      'Nu is de regel goed
```

Uit het tweede voorbeeld zien we dat we met elk opgegeven gebied een lokale Range variabele kunnen toekennen. Het gebied die nu is toegekend aan `objRange` kan een lokale variabele zijn in bijvoorbeeld een subroutine of in een event. Dit werkt zeer makkelijk, vooral voor grote gebieden. U hoeft niet meer steeds het gebied op te geven, maar door lokale objectvariabelen te declareren van het `Range` type verkort u de code een heel stuk. Zie onderstaande event eens:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

End Sub
```

Zodra een selectie wordt gewijzigd roept VBA deze event aan. De programmeur kan het target argument gebruiken om de wijziging te beheren. Target heeft als parameter het gebied meegekregen dat in de event bewerkt kan worden.

Een gebied teruggeven uit een vorige Range.

In VBA kunnen we gebieden nesten of overlappen. Met die techniek kunnen zeer krachtige gebieden met elkaar samenwerken. Het nesten van gebieden is mogelijk, omdat een Range object ook weer een Range object heeft. Maar die heeft ook weer een Range object en zo kunnen we wel doorgaan.

We kunnen rijen en kolommen in lusstructuren opvragen door de waarde van de lus in het Range gebied te gebruiken, zie onderstaande voorbeeld die 5 keer herhaalt:

```
For R = 1 To 5
    Set objRij = Range("A" & CStr(R), ":E" & CStr(R))
Next R
```

Onderstaande voorbeeld laat een truc zien waarmee we makkelijk 5 keer kunnen herhalen en met de luswaarde de juiste kolomletter kunnen gebruiken:

```
For K = 65 to 70
   Set objKolom = Range(Chr$(K) & "1:", Chr$(K) & "10")
Next K
```

Met het laatste voorbeeld krijgen we dus 10 rijen per kolom die begint met karakter 65. Die karaktercode is gelijk aan hoofdletter A. Controleer het maar eens. Voer onderstaande regel eens in venster Direct:

```
Print Chr$(65) & "1" = "A1"
```

Is uw versie Nederlandstalig, dan zal venster Direct het resultaat `Waar` geven, anders zal er `True` komen te staan.

Op die manier kunnen we meerdere rijen en kolommen uit de werkbladen halen. Dit werkt veel sneller en efficiënter dan telkens de rijen te moeten selecteren en ook nog eens het op te moeten nemen in een macro. Bovendien kunnen we de gebruiker laten bepalen hoeveel rijen en kolommen het moeten zijn door deze aantal eerst in te laten voeren. Nu nog door die in te laten voeren in de cellen, maar later zal ik laten zien hoe we een invoer dialoog-formulier kunnen weergeven. Door de gegevens aan de gebruiker over te laten kunnen we de code makkelijk dynamisch houden.

Marco Kurvers

Ik kan programmeren.

In de vorige nieuwsbrief heb ik u laten weten dat BASIC leren niet voldoende is om gelijk projecten te kunnen schrijven. In de praktijk komt er veel meer bij kijken dan alleen maar de programmeertaal kennen.
Wat komen we in de praktijk allemaal tegen? Dat is wat u in de komende nieuwsbrieven

zult vinden.

Subprocedures en functies: geen verschil?

We kennen BASIC met het statement `GOTO` en de structuurstatements `GOSUB...RETURN`. Het `GOTO` statement springt naar een regelnummer of label zodat de code vanaf daar verder uitgevoerd kan worden en de statements `GOSUB...RETURN` geeft een structuur als een codeblok beginnend vanaf een regelnummer of label, maar teruggaand naar de volgende regel van waar het `GOSUB` statement werd aangeroepen.

Een `GOTO` lijkt daarom veel op een subprocedure die geen returnwaarde heeft en een `GOSUB...RETURN` op een functie lijkt die wel een returnwaarde heeft. Maar schijn bedriegt!

Tegenwoordig zal na de subprocedure uitgevoerd is, de code verder uitgevoerd worden na de aanroep van de subprocedure. Dit geldt ook voor de functie. Het enige verschil is: een functie zal altijd een resultaat teruggeven.

Conclusie: Alle methoden in klassen zijn zowel functies als subroutines. Buiten de klassen zijn het geen methoden.
Soms worden subroutines ook wel procedures genoemd. In Pascal en Delphi moeten ze ook met het sleutelwoord *procedure* geschreven worden.

Er zijn programmeertalen die een andere manier hebben om methoden aan te roepen, bijvoorbeeld in C++. Die methoden hebben een returntype. Een sub zoals we in BASIC kennen bestaat in C++ niet, maar als we een soortgelijke methode willen maken in C++ die geen resultaat terug mag geven, moeten we het type `void` gebruiken.

Het is uw keus wanneer een subprocedure of een functie gemaakt moet worden. Codeblokken kunt u in beide plaatsen, maar bepaal zelf in welke het resultaat teruggegeven moet worden. Vergeet ook niet aan het einde van de functienaam het returntype op te geven. Geeft u toch geen returntype op dan zal de returnwaarde (resultaat) automatisch een variant zijn.

Let op! Na Visual Basic 6.0 bestaat het type `Variant` niet meer. In alle Visual Basic .NET versies hebben de functies het type `Object` als standaardtype.

Zelf gegevenstypen maken.

Tegenwoordig is het in Basic mogelijk zelf gegevenstypen te maken. Daarbij spreken de arrays ook een grote rol. Wat we ook voor typen kunnen definiëren zijn verbindingstypen, zodat een variabele uit verscheidene afzonderlijke variabelen kan worden samengesteld.

Tip! **De verbindingstypen, ook wel records genoemd, zijn voor het begrip van objecten buitengewoon handig.**

Zelf gegevenstypen maken? Hebben we niet voldoende aan Doubles, Longs, Integers,

Strings en ook nog een booleaans gegevenstype? Inderdaad kunnen we met deze gegevenstypen alles beschrijven wat in het geheugen van onze computers kan worden ondergebracht. Maar waarom hebben we dan al deze verschillende gegevenstypen? Tenslotte wordt in het geheugen toch alles in bytes en zelfs in bits opgeslagen? Met de gegevenstypen die we al kennen, worden gegevens *geabstraheerd*. Daardoor wordt programmeren handiger. Maar deze abstractie kan nog abstracter worden gemaakt (waardoor het programmeercomfort groter wordt) – juist met zelf gedefinieerde gegevenstypen.

Stel je voor dat je een variabele als volgt kunt definiëren:

```
DIM vrucht AS typevrucht
```

en je het statement zo kunt schrijven:

```
vrucht = appel
```

Zoals we dadelijk zullen zien, is dit inderdaad zonder meer mogelijk. Begrijpelijker kan het beslist niet. De definitie van eigen gegevenstypen verhoogt dus de leesbaarheid van het programma. Maar de definitie van eigen gegevenstypen heeft nog een groot voordeel: Basic kan foute toewijzingen ontdekken, zodat moeilijk op te sporen fouten ons bespaard blijven:

```
vrucht = fiets
```

Dit zou Basic ontdekken, want 'fiets' is geen vrucht (vooropgesteld natuurlijk dat we 'fiets' als constante van een bepaald type hebben gedefinieerd, anders zou Basic een syntaxisfout melden!). Het tweede grote voordeel van zelf gedefinieerde typen is dus de domeincontrole.

Hoe hadden we dergelijke voorbeelden tot nu toe gedefinieerd? Vermoedelijk had je constanten gebruikt om de leesbaarheid van het programmeren te verbeteren, bijvoorbeeld zo:

```
CONST appel = 1
CONST peer = 2
CONST sinaasappel = 3
CONST fiets = 1
CONST auto = 2
DIM vrucht AS BYTE, transportmiddel AS BYTE

SUB Test()
    vrucht = appel
    transportmiddel = fiets
END SUB
```

Dit is ook wel leesbaar, maar Basic kan onlogische toewijzingen als:

```
vrucht = auto
transportmiddel = appel
```

niet herkennen.

Typen zelf definiëren.

Hoe definiëren we dan zelf typen? Daarvoor dient (niet zo verrassend) het sleutelwoord TYPE. Maar Basic kent het sleutelwoord TYPE echter niet voor het probleem die we hebben.

Enumeraties

Dergelijke talen, onder andere Pascal, kennen alleen maar het sleutelwoord TYPE om allerlei soorten gegevenstypen te maken en het probleem op te kunnen lossen. Basic heeft daar een ander sleutelwoord voor: `ENUM`.

We kunnen ons probleem bijvoorbeeld zo definiëren:

```
ENUM TypeVrucht
    appel
    peer
    sinaasappel
END ENUM
```

We hebben de vruchten opgesomd die voor ons vruchttype TypeVrucht mogelijk zijn. Het is niet erg verrassend dat een type dat zo wordt beschreven *enumeratietype* wordt genoemd.

Merk op dat we voor de begrippen geen aanhalingstekens hebben gebruikt. Het gaat niet om Strings, maar om *constanten*. 'Appel' is dus nu een van de mogelijke waarden voor variabelen van het type `TypeVrucht`.

Om een enumeratietype te definiëren, worden de afzonderlijke waarden onder elkaar gescheiden. Gescheiden door komma's is in Basic niet toegestaan.
Is het type gedefinieerd dan kan men variabelen van dit type definiëren en ze een van de opgesomde waarden toewijzen.

Door de toepassing van een enumeratietype voeren we een nieuw abstractieniveau in en maken we het programma zo leesbaarder. Maar hoe realiseert Basic variabelen van het nieuw gedefinieerde enumeratietype intern? Heel eenvoudig, als een ordinaal type, dat wil zeggen als Integer-getal van overeenkomende grootte! Elke mogelijke waarde krijgt intern een Integer-waarde toegewezen, waarbij Basic aan de eerste waarde 1 toewijst, als de optie `OPTION BASE` niet op nul staat ingesteld. Eigenlijk is het wel aanbevolen om ordinale waarden en array-elementen met waarde nul te laten beginnen. Ik neem verder ook aan dat het ook zo staat ingesteld.

Bij bovenstaande `ENUM` type ontstaat dit:

`appel` krijgt de ordinale waarde 0
`peer` krijgt de ordinale waarde 1
`sinaasappel` krijgt de ordinale waarde 2

Eigenlijk zou het voor ons niet mogen uitmaken hoe Basic de waarden intern realiseert. Maar het is wel handig, omdat we daarmee inzicht krijgen in enkele interessante eigenschappen van enumeratietypen.

Een enumeratietype kan overal worden gebruikt waar een ordinaal type wordt verwacht, bijvoorbeeld in expressies en condities. Dit betekent dat een conditie als: `appel < peer` is toegestaan, en levert in dit geval `True` op.

Misschien ziet u het nut daarvan niet in, maar kijk eens naar het volgende voorbeeld:

```
ENUM MaandType
     januari
     februari
     maart
     april
     mei
     juni
     juli
     augustus
     september
     oktober
     november
     december
END ENUM
DIM Maand AS MaandType
```

Nu kun je heel handig vergelijkingen van de soort:

```
IF Maand = januari THEN ...
```

of

```
IF Maand < maart THEN ...
```

maken.

Vergelijkingen van het soort `januari < februari` zijn dus gemakkelijk uit te voeren als we met enumeratietypen werken. Maar hoe zit het met de weergave? Kort gezegd ziet het er somber uit. Want een praktische toewijzing als bijvoorbeeld in Visual Basic:

```
Label1.Caption = Maand
```

is **niet** mogelijk. Want bij het maandtype gaat het om een zelf gedefinieerd gegevenstype, bij `Caption` om een String type. Ook al geeft Basic voor dit geen foutmelding, dan nog zal een waarde als 2 voor sinaasappel nietszeggend zijn.

Als oplossing daarvoor schrijven we een functie die ons zelf gedefinieerde type als argument accepteert en het bijbehorende resultaat teruggeeft (waarmee al duidelijk is dat we ook in functieaanroepen zelf gedefinieerde typen kunnen toepassen). Zo moet bijvoorbeeld een functie eruitzien die ons maandtype in een String omzet:

```
FUNCTION maandToStr(BYVAL maand AS MaandType) AS STRING
     SELECT CASE maand
          CASE januari
               maandToStr = "januari"
          CASE februari
               maandToStr = "februari"
          CASE maart
```

```
            maandToStr = "maart"
        CASE april
            maandToStr = "april"
        CASE mei
            maandToStr = "mei"
        CASE juni
            maandToStr = "juni"
        CASE juli
            maandToStr = "juli"
        CASE augustus
            maandToStr = "augustus"
        CASE september
            maandToStr = "september"
        CASE oktober
            maandToStr = "oktober"
        CASE november
            maandToStr = "november"
        CASE december
            maandToStr = "december"
    END SELECT
END FUNCTION
```

Tot nu toe hebben we de waarden voor de begrippen aan Basic overgelaten die ervoor zorgt dat aan de instelling van `OPTION BASE` de eerste constante met 0 of 1 begint. U kunt ook zelf een beginwaarde opgeven en het maakt zelfs niet uit bij welke begrip u de waarde opgeeft, bijvoorbeeld:

```
ENUM TKleur
    kleur_rood
    kleur_blauw = 10
    kleur_geel
END ENUM
```

Hier heeft de waarde 'kleur_blauw' expliciet een nieuwe ordinale waarde gekregen, namelijk 10. Dit betekent:

```
kleur_rood = 0
kleur_blauw = 10
kleur_geel = 11
```

Merk op dat vanaf de laatst aangegeven ordinale waarde verder wordt geteld. Je kunt ook een aantal elementen dezelfde ordinale waarde toewijzen. In dat geval ontstaat een element dat met verschillende namen kan worden aangesproken.

! Het is niet toegestaan enumeraties te nesten, dat wil zeggen: u kunt geen enumeratietype definiëren in een enumeratietype.

Deelbereiken

Behalve de enumeratietypen zijn er nog andere manieren om een type te declareren. Een methode is de opgave van een deelbereik van een (al bestaand) ordinaal type dat hier als basistype wordt aangeduid. Het deelbereik wordt gedefinieerd door de onderste en bovenste waarden van het basistype op te geven.

Maar de deelbereiken die we in Pascal kennen kunnen we helaas niet in Basic gebruiken. De deelbereiken zijn alleen nuttig om de aantal elementen in een arraytype op te geven. Het

is niet toegestaan om onderstaande deelbereik te definiëren:

```
DIM getal AS INTEGER
getal = 10 TO 20      'werkt helaas niet
```

Deze mogelijkheden, waarmee we zelfs kapitaalbereiken zouden kunnen maken, is gewoonweg niet in Basic aanwezig. In Pascal kunnen we onderstaande deelbereiken maken:

```
type     Kapitalen = 'A'..'Z';
         Kleine_letters = 'a'..'z';
         Char_Cijfers = '0'..'9';
         Cijfers = 0..9;
```

Het enige wat we in Basic mogen doen is onderstaande array met een deelbereik declareren: `DIM arraydeel(10 TO 20) AS INTEGER 'deze kan wel`

Ook onderstaande voorbeeld kan van nut zijn:

```
TYPE TDeelbereik
    bereik(-15 TO 20) AS INTEGER
END TYPE
DIM Deelbereik AS TDeelbereik
Deelbereik.bereik(-5) = 100
Debug.Print Deelbereik.bereik(-5)  'hier wordt de waarde 100 geprint
```

Bovendien heeft een `FOR` lus ook een deelbereik, die echter altijd verplicht is en in een array niet verplicht is. Een `FOR` lus heeft ook als uitzondering dat dit deelbereik wel van hoog naar laag mag lopen. Vergeet dan niet het `STEP` statement te gebruiken, want die wordt dan verplicht voor deelbereiken die van hoog naar laag lopen, anders zal de lus niets doen.

Records

Stel je de volgende situatie voor: je werkt aan een programma dat de arbeidstijd voor de werknemers van je onderneming moet opslaan en analyseren. Je moet dus voor elke werknemer een reeks gegevens opslaan:

- personeelsnummer
- voornaam van de werknemer
- achternaam van de werknemer
- begintijd – uur
- begintijd – minuut
- eindtijd – uur
- eindtijd – minuut

Elke dag moet dus voor elke werknemer een hele reeks variabelen worden verwerkt. Hoe zou je besluiten welke variabele welk type moet krijgen? Misschien door elke betekenis een aparte variabele van een bepaald type te declareren.
We moeten dus voor een werknemer (en per dag) variabelen van het type Integer, String en Byte gebruiken.

Hoewel de verschillende variabelen van verschillende typen zijn, horen ze toch bij elkaar omdat ze informatie beschrijven. We kunnen deze informatie dus op een gegevenskaart (Engels: recordcard) schrijven, en dit is in Basic ook mogelijk. Ook al kent Basic geen sleutelwoord `Record` zoals die wel in Pascal bestaat, we kunnen gelukkig wel met `TYPE`

`...  END TYPE` hetzelfde doel bereiken. Onderstaande voorbeeld laat een Pascal record zien met dezelfde structuur in Basic. U kunt zien dat er weinig verschil in zit.

```
type TWerknemer = record
        Pnummer      : Integer;
        Vnaam        : String;
        Anaam        : String;
        Btijduur,
        Btijdmin,
        Etijduur,
        Etijdmin     : Byte;
    end;
```

```
TYPE TWerknemer
        Pnummer      AS INTEGER
        Vnaam        AS STRING
        Anaam        AS STRING
        Btijduur, _
        Btijdmin, _
        Etijduur, _
        Etijdmin     AS BYTE
END TYPE
```

We kunnen nu een variabele voor het recordtype definiëren:

```
DIM werknemer AS TWerknemer
```

Met dit statement hebben we een nieuwe *verbindingsvariabele werknemer* gedefinieerd. Deze bestaat uit de afzonderlijke variabelen `Pnaam`, `Vnaam` enzovoort.

! **Onthoud dat Basic geen record kent, dus ook geen recordtype. Wat we hier met de gegevenstypen in Basic doen hoeven niet altijd te dienen als records. U kunt ook andere soorten gegevenstypen maken.**

Marco Kurvers

Intermezzo: een Werknemers project (3).

In deel 1 en deel 2 hebt u kennis gemaakt met de onderdelen van het Werknemers project. In deel 1 ging het over het databasemanagement en in deel 2 over de besturing van het hoofdmenu en de subprocedures die voor de database, de dialoogschermen en voor het afdrukken zorgen.

Nu we alles hebben gehad en we nu weten hoe we een project moeten opbouwen, wordt het tijd voor de praktijk. Voordat de listing er staat komt er eerst nog een conclusie. Een lijst met informatie over het project en wat tips.

Conclusie en tips: het Werknemers project.

De QuickBASIC random-access filestatements ondersteunen verfijnde technieken om een databasefile op te zetten en te beheren. U heeft nu voldoende kennis vergaard om een random-access file te openen, de recordstructuur te definiëren, records uit de file te lezen,

nieuwe records naar het einde van de file te schrijven en de records te wijzigen die momenteel in de file zijn opgeslagen. In combinatie met een kleine bibliotheek met essentiële subprogramma's zoals *Sorteren* en *Zoek* vormen deze statements de noodzakelijke gereedschappen om complete databasetoepassingen te creëren.
Mocht u zin hebben om verder te experimenteren met het *Werknemer* programma, dan zijn de volgende suggesties wellicht interessant:

1. Schrijf een foutafhandelingsroutine die de besturing overneemt indien het programma de indexfile WERKNMER.NDX niet kan vinden. (Soms wissen computergebruikers per ongeluk belangrijke files. Deze foutafhandelingsroutine dient het functioneren van de database te waarborgen wanneer de index verloren is gegaan.) De routine moet de databasefile openen (WERKNMER.DAT) en alle records uit de file lezen. Terwijl de routine de database doorwerkt, slaat deze de namen en nummers in de `index()` array op. Na het sorteren van de nieuwe index kan het *Werknemer* programma de normale gang van zaken hervatten.
2. Ontwerp een subprogramma dat vertrokken werknemers uit de database verwijdert. Hiertoe kan bijvoorbeeld de WERKNMER.DAT file in twee files worden gesplitst: WERKNWEG.DAT om de records van vertrokken werknemers in op te slaan en WERKNHDG.DAT voor de records van de huidige werknemers. Wanneer dit proces is voltooid, kan de originele WERKNMER.DAT worden gewist (of worden hernoemd tot backupfile WERKNMER.BAK) en dient WERKNHDG.DAT worden hernoemd tot WERKNMER.DAT. Tevens kunt u een aparte serie subprogramma's schrijven, die de gebruiker toegang verlenen tot de database met de vertrokken werknemers.
3. Creëer andere indices om de records in de file te adresseren. Bijvoorbeeld, een index die de records op afdeling en op werknemernaam binnen een afdeling sorteert. Elk element in de index stringarray zal dan een samenvoeging van de afdelingnaam (voorop) en een werknemernaam moeten bevatten. Aan de hand van een dergelijke index kan het programma een gedrukt overzicht voortbrengen, waarin de werknemers per afdeling zijn gesorteerd.
4. Schrijf een utility die de veldstructuur van de database kan uitbreiden. Een dergelijke utility kan bijvoorbeeld velden voor de geboortedatum van de werknemer en de burgerlijke staat toevoegen. Teneinde deze nieuwe databasestructuur door te voeren, moet de utility een geheel nieuwe database met de benodigde recordlengte en veldstructuur creëren. Wanneer vervolgens de nieuwe database en WERKNMER.DAT gelijktijdig worden geopend, zou het programma de reeds beschikbare veldgegevens in de WERKNMER.DAT records naar de nieuwe database moeten kopiëren. Uiteraard moet een dergelijk proces tevens een invoerdialoog omvatten om de nieuwe veldwaarden aan de werknemerrecords toe te wijzen.

De praktijk: het Werknemer programma.

Onderstaande listing vormt het hele project als één programma. Om de onderdelen, die besproken zijn, makkelijk in het programma te kunnen vinden, staan aan de linkerkant de namen van de onderdelen.

Let op! De listing is geschreven voor QuickBASIC 4.0 en 4.5. Symbolen, bijvoorbeeld de umlaut, zijn in de teksten niet aanwezig, ook al zou zo'n symbool er wel moeten staan. Voorbeeld: het woord 'geïndexeerd' zal gewoon worden getoond als 'geindexeerd'.

```
'   Het Werknemer programma beheert een database met informatie over
```

```
'   de werknemers van een bedrijf. Het programma onderhoudt twee files
'   op disk of diskette: WERKNMER.DAT, de database zelf (een random-
'   access file), en WERKNMER.NDX, een index van de database (een
'   sequentiële file).
'   (Het Werknemer programma is geschreven voor QuickBASIC 4.0 en 4.5.)
```

Declaraties

```
'   ---- Definities en declaraties.

CONST false% = 0, true% = NOT false%
CONST tabPos% = 25

TYPE werknType
     achterNaam AS STRING * 16
     voorNaam AS STRING * 10
     zkfNummer AS STRING * 11
     beginDatum AS STRING * 10
     afdeling AS STRING * 15
     functie AS STRING * 15
     salaris AS SINGLE
     salType AS STRING * 1
     vertrekDatum AS STRING * 10
END TYPE

TYPE indexType
     werknNummer AS INTEGER
     werknNaam AS STRING * 26
END TYPE

DECLARE FUNCTION LeesSalaris! ()
DECLARE FUNCTION LeesSalType$ ()
DECLARE FUNCTION Menu% (opties$())
DECLARE FUNCTION WisSpatie$ (invoer$)
DECLARE FUNCTION Zoek% (welkeTekst$)
DECLARE FUNCTION JaNee% (prompt$)
DECLARE SUB Toevoegen ()
DECLARE SUB Wijzigen ()
DECLARE SUB CloseFiles ()
DECLARE SUB Vertrek ()
DECLARE SUB DisplayMenu (optieLijst$(), coordLinks%, prompt$, ok$)
DECLARE SUB DisplayRec (toonRecord AS werknType, nummer%)
DECLARE SUB Frame (links%, rechts%, boven%, onder%)
DECLARE SUB LeesNaam (werknRecord AS werknType, RecNr%, doel$)
DECLARE SUB OpenFile ()
DECLARE SUB OpenIndex ()
DECLARE SUB Pauze ()
DECLARE SUB PrintRecords ()
DECLARE SUB NieuweIndex (werknNaam$)
DECLARE SUB Sorteren ()

DIM werknRec AS werknType, werknMenu$(6)

COMMON SHARED totWerkns%, index() AS indexType

DATA Display record
DATA Toevoegen record
DATA Wijzigen record
DATA Registratie van vertrek
DATA Printen van records
DATA Einde
```

Hoofdprogramma

```
'   ---- Hoofdprogramma.

CLS
LOCATE , , 1
FOR i% = 1 TO 6
      READ werknMenu$(i%)
NEXT i%

OpenFile

'   ---- Geef het hoofdmenu op het scherm weer en roep het subprogramma
'        aan dat de menukeuze uitvoert.

finito% = false%
```

```
                DO
                      LOCATE 2, 32: PRINT "Werknemer-records"

                      SELECT CASE Menu%(werknMenu$())

                      CASE 1
                            IF totWerkns% <> 0 THEN
                                  DisplayRec werknRec, dummy%
                                  Pauze
                            END IF

                      CASE 2
                            Toevoegen

                      CASE 3
                            IF totWerkns% <> 0 THEN Wijzigen

                      CASE 4
                            IF totWerkns% <> 0 THEN Vertrek

                      CASE 5
                            IF totWerkns% <> 0 THEN PrintRecords

                      CASE ELSE
                            CloseFiles
                            finito% = true%

                      END SELECT

                LOOP UNTIL finito%

                END

CloseFiles      SUB CloseFiles STATIC

                '   Het CloseFiles subprogramma sluit de database en
                '   slaat de bijgewerkte index in de WERKNMER.NDX file op.

                      CLOSE #1

                      IF totWerkns% > 0 THEN
                            OPEN "WERKNMER.NDX" FOR OUTPUT AS #1

                            FOR i% = 1 TO totWerkns%
                                  WRITE #1, index(i%).werknNummer, RTRIM$(index(i%).werknNaam)
                            NEXT i%

                            CLOSE #1
                      END IF

                END SUB

DisplayMenu     SUB DisplayMenu (optieLijst$(), coordLinks%, prompt$, ok$)

                '   Het DisplayMenu subprogramma geeft de menuopties op het scherm
                '   weer en zet de prompt- en de controlestring op.
                '   Deze routine wordt vanuit de Menu% functie aangeroepen.

                '   ---- Bepaal de aantal opties (aantalOpties%);
                '         initialiseer de variabelen.

                      aantalOpties% = UBOUND(optieLijst$)
                      prompt$ = " "
                      ok$ = ""
                      langsteOptie% = 0

                '   ---- Zet de promptstring (prompt$) en de string met de geldige
                '         invoerkarakters (ok$) op. Bereken tevens de lengte van
                '        de langste optiestring (langsteOptie%).

                      FOR i% = 1 TO aantalOpties%
                            eerste$ = UCASE$(LEFT$(optieLijst$(i%), 1))
                            ok$ = ok$ + eerste$
                            prompt$ = prompt$ + eerste$ + "  "
```

```
                tijdelijkLangste% = LEN(optieLijst$(i%))
                IF tijdelijkLangste% > langsteOptie% THEN
                     langsteOptie% = tijdelijkLangste%
                END IF
           NEXT i%

           langsteOptie% = langsteOptie% + 1
           prompt$ = prompt$ + "-> "

'     ---- Controleer of de promptstring langer is dan langsteOptie%.

           IF LEN(prompt$) >= langsteOptie% THEN langsteOptie% = LEN(prompt$) + 1

'     ---- Bepaal de afmetingen van het menukader aan de hand van
'          langsteOptie% en aantalOpties%.
'          Roep het Frame subprogramma aan om het kader te tekenen.

           coordLinks% = 37 - langsteOptie% \ 2
           coordRechts% = 80 - coordLinks%
           coordBoven% = 3
           coordOnder% = 10 + aantalOpties%
           Frame coordLinks%, coordRechts%, coordBoven%, coordOnder%

'     ---- Geef de menuopties weer. Elke optie begint
'          met een hoofdletter gevolgd door een haakje.

           FOR i% = 1 TO aantalOpties%
                LOCATE 6 + i%, coordLinks% + 3
                PRINT UCASE$(LEFT$(optieLijst$(i%), 1)) + ")" + MID$(optieLijst$(i%), 2)
           NEXT i%

           LOCATE 4, 38: PRINT "Menu"
           regel$ = STRING$(langsteOptie%, 196)
           LOCATE 5, coordLinks% + 3: PRINT regel$
           LOCATE 7 + aantalOpties%, coordLinks% + 3: PRINT regel$

'     ---- Genereer de inputprompt.

           LOCATE 9 + aantalOpties%, coordLinks% + 3: PRINT prompt$;

     END SUB

DisplayRec SUB DisplayRec (toonRecord AS werknType, nummer%) STATIC

'    Het DisplayRec subprogramma geeft een specifiek werknemerrecord
'    op het scherm weer.

           PRINT : PRINT : PRINT : PRINT
           PRINT TAB(tabPos%); "Voer de naam van de werknemer in:"
           PRINT TAB(tabPos%); "---- -- ---- --- -- --------- --"
           LeesNaam toonRecord, nummer%, doelNaam$
           IF nummer% = 0 THEN
                PRINT
                PRINT TAB(tabPos%); "Deze naam komt niet in de werknemer-file voor."
           ELSE
                CLS
                GET #1, nummer%, toonRecord
                PRINT : PRINT : PRINT
                PRINT : PRINT : PRINT
                PRINT TAB(tabPos%); "Naam      :   ";
                PRINT RTRIM$(toonRecord.voorNaam); " "; toonRecord.achterNaam
                PRINT TAB(tabPos%); "Zkfnummer :   "; toonRecord.zkfNummer
                PRINT TAB(tabPos%); "Begindatum:   "; toonRecord.beginDatum

                IF LEFT$(toonRecord.afdeling, 10) <> "VERTROKKEN" THEN
                     PRINT TAB(tabPos%); "Afdeling  :   "; toonRecord.afdeling
                END IF

                PRINT TAB(tabPos%); "Functie   :   "; toonRecord.functie
                PRINT TAB(tabPos%); "Salaris   :   ";

                SELECT CASE toonRecord.salType
                CASE "U"
                     PRINT USING "**###.## per uur"; toonRecord.salaris
```

```
                  CASE "M"
                        PRINT USING "**#,#### per maand"; toonRecord.salaris
                  CASE ELSE
                        PRINT USING "**#,##### per jaar"; toonRecord.salaris
                  END SELECT

'   ---- Indien de werknemer niet meer bij de firma werkt,
'        laat dit dan uit de recordweergave blijken.

                  IF LEFT$(toonRecord.afdeling, 10) = "VERTROKKEN" THEN
                        PRINT
                        PRINT TAB(tabPos%); "*** Niet meer in dienst ***"
                        PRINT
                        PRINT TAB(tabPos%); "Vertrekdatum:   "; toonRecord.vertrekDatum
                  END IF
            END IF

END SUB
```

Frame

```
SUB Frame (links%, rechts%, boven%, onder%) STATIC

'   Het Frame subprogramma tekent een rechthoekig kader met dubbele
'   lijnen op het scherm door middel van grafische karakters
'   uit de IBM Extended Character Set.

'   ---- Teken de vier hoeken.

      LOCATE boven%, links%: PRINT CHR$(201)
      LOCATE boven%, rechts%: PRINT CHR$(187)
      LOCATE onder%, links%: PRINT CHR$(200);
      LOCATE onder%, rechts%: PRINT CHR$(188);

'   ---- Teken de verticale lijnen.

      FOR vert% = boven% + 1 TO onder% - 1
            LOCATE vert%, links%: PRINT CHR$(186);
            LOCATE vert%, rechts%: PRINT CHR$(186);
      NEXT vert%

'   ---- Teken de horizontale lijnen.

      horiz% = rechts% - links% - 1
      hlijn$ = STRING$(horiz%, 205)
      LOCATE boven%, links% + 1: PRINT hlijn$
      LOCATE onder%, links% + 1: PRINT hlijn$;

END SUB
```

JaNee%

```
FUNCTION JaNee% (prompt$)

'   De JaNee% functie stelt een ja-of-nee vraag en retourneert
'   een logische waarde die het antwoord van de gebruiker vertegenwoordigt:
'   true betekent ja en false is nee.

'   ---- Geef de vraag weer en initialiseer de variabelen.

      PRINT prompt$; " (J of N) -> ";
      ant$ = ""
      karPos% = 0

'   ---- Lees het antwoord.

      DO
            LOCATE , , 1
            ant$ = UCASE$(INKEY$)
            IF ant$ <> "" THEN
                  karPos% = INSTR("JN", ant$)
                  IF karPos% = 0 THEN BEEP
            END IF
      LOOP UNTIL karPos% > 0
      PRINT ant$

'   ---- Zet het antwoord in een logische waarde om.
```

```
      JaNee% = (ant$ = "J")

END FUNCTION
```

LeesNaam

```
SUB LeesNaam (werknRecord AS werknType, RecNr%, doel$) STATIC

'   Het LeesNaam subprogramma accepteert de werknemernaam die
'   het programma vervolgens in de database opzoekt.
'   LeesNaam retourneert twee scalaire waarden aan de aanroepende
'   procedure: RecNr% is het recordnummer (of 0 indien het
'   record niet aanwezig is), en doel$ is de geindexeerde
'   werknemernaam. Bovendien retourneert LeesNaam een werknRecord structuur
'   met de voornaam- en achternaamelementen die de gebruiker heeft ingevoerd.

      PRINT : PRINT
      PRINT TAB(tabPos%); : INPUT "Achternaam    --> ", tijdelijkAchternm$
      PRINT TAB(tabPos%); : INPUT "Voornaam      --> ", tijdelijkVoornm$
      doel$ = UCASE$(WisSpatie$(tijdelijkAchternm$ + tijdelijkVoornm$))

      IF totWerkns% <> 0 THEN
            RecNr% = Zoek$(doel$)
      END IF

      werknRecord.achterNaam = tijdelijkAchternm$
      werknRecord.voorNaam = tijdelijkVoornm$

END SUB
```

LeesSalaris!

```
FUNCTION LeesSalaris!

'   De LeesSalaris! functie leest een numerieke invoerwaarde voor
'   het salaris van een werknemer. LeesSalaris! slaat deze waarde
'   in eerste instantie in een stringvariabele op en accepteert
'   de waarde alleen indien deze in een nonzero getal kan worden geconverteerd.

      posX% = POS(0)
      posY% = CSRLIN

      DO
            INPUT "Salaris       --> ", sal$
            sal = VAL(sal$)
            IF sal = 0 THEN
                  LOCATE posY%, posX%
                  PRINT SPACE$(LEN(sal$) + 16)
                  LOCATE posY%, posX%
            END IF
      LOOP UNTIL sal > 0

      LeesSalaris! = sal

END FUNCTION
```

LeesSalType$

```
FUNCTION LeesSalType$

'   De LeesSalType$ functie accepteert een enkel karakter
'   dat aangeeft of het salaris van een werknemer op uur- (U),
'   maand- (M), of op jaarbasis (J) wordt berekend. De functie
'   negeert ongeldige invoerkarakters.

      DO
            LOCATE , , 1
            salarisType$ = UCASE$(INKEY$)
            IF INSTR("UMJ", salarisType$) = 0 THEN
                  BEEP
                  salarisType$ = ""
            END IF
      LOOP UNTIL LEN(salarisType$) > 0

      PRINT salarisType$

      LeesSalType$ = salarisType$

END FUNCTION
```

Menu%

```
FUNCTION Menu% (opties$()) STATIC

'   De Menu% functie geeft een menu op het scherm weer en
'   verwerkt de menukeuze van de gebruiker. Menu% ontvangt
'   stringarray opties$ met de menuopties en
'   retourneert een integer getal dat de keuze van
'   de gebruiker vertegenwoordigt.

      menuLengte% = UBOUND(opties$)
      DisplayMenu opties$(), margeLlinks%, promptStr$, okStr$

'   ---- Lees de menukeuze en controleer deze.

      besturingsToetsen$ = CHR$(13) + CHR$(27)
      DO
            LOCATE , , 1
            karPos% = 0
            DO
                  antwoord$ = UCASE$(INKEY$)
                  IF antwoord$ <> "" THEN
                        karPos% = INSTR(okStr$, antwoord$)
                        IF karPos% = 0 THEN BEEP
                  END IF
            LOOP UNTIL karPos% > 0

            PRINT antwoord$
            LOCATE 11 + menuLengte%, 23, 0
            PRINT "<Enter> voor bevestiging; <Esc> voor opnieuw."
            keuze% = karPos%

            karPos% = 0
            DO
                  antwoord$ = INKEY$
                  IF antwoord$ <> "" THEN
                        karPos% = INSTR(besturingsToetsen$, antwoord$)
                        IF karPos% = 0 THEN BEEP
                  END IF
            LOOP UNTIL karPos% > 0

            IF karPos% = 1 THEN
                  finito% = true%
                  CLS
            ELSE
                  finito% = false%
                  LOCATE 11 + menuLengte%, 23: PRINT SPACE$(35)
                  LOCATE 9 + menuLengte%, margeLlinks% + 3 + LEN(promptStr$)
                  PRINT " ";
                  LOCATE , POS(0) - 1:
            END IF
      LOOP UNTIL finito%

      Menu% = keuze%

END FUNCTION
```

NieuweIndex

```
SUB NieuweIndex (werknNaam$) STATIC

'   Het NieuweIndex subprogramma zet een nieuwe index op zodra
'   een nieuwe werknemer aan de database is toegevoegd.

'   ---- Kopieer de actieve index eerst naar een tijdelijke index-array.

      IF totWerkns% > 1 THEN
            oudTot% = totWerkns% - 1
            REDIM tijdelijkeIndex(oudTot%) AS indexType

            FOR i% = 1 TO oudTot%
                  tijdelijkeIndex(i%) = index(i%)
            NEXT i%
      END IF

'   ---- Dimensioneer de index() array tot het vereiste formaat.
'        (Dit proces wist tevens alle data uit de array, hetgeen de reden
'        is dat het programma eerst een tijdelijke kopie van de index moet maken.)
```

```
      REDIM index(totWerkns%) AS indexType

'   ---- Kopieer de indexinhoud tenslotte weer naar de index() array
'        en sla de nieuwe invoer aan het einde van de index op.

      IF totWerkns% > 1 THEN
            FOR i% = 1 TO oudTot%
                  index(i%) = tijdelijkeIndex(i%)
            NEXT i%
      END IF

      index(totWerkns%).werknNummer = totWerkns%
      index(totWerkns%).werknNaam = werknNaam$

'   ---- Sorteer de nieuwe index.

      Sorteren

END SUB
```

OpenFile

```
SUB OpenFile STATIC

'   Het OpenFile subprogramma opent de feitelijke databasefile
'   en definieert de veldvariabelen. Deze variabelen zijn reeds aan
'   het begin van het programma als globaal gedeclareerd in
'   het COMMON SHARED statement.

      DIM openRecord AS werknType

      OPEN "WERKNMER.DAT" FOR RANDOM AS #1 LEN = LEN(openRecord)

'   ---- Bereken de aantal records in de database door middel van de LOF functie.

      totWerkns% = LOF(1) / LEN(openRecord)

'   ---- Open vervolgens de indexfile op voorwaarde dat de database niet leeg is.

      IF totWerkns% <> 0 THEN OpenIndex

END SUB
```

OpenIndex

```
SUB OpenIndex STATIC

'   Het OpenIndex subprogramma opent de indexfile, WERKNMER.NDX,
'   en leest de inhoud van de file in de index() array.
'   Recordvariabelen werknNummer en werknNaam bevatten respectievelijk
'   de werknemer recordnummers en de werknemernamen.

'   ---- Aangezien het formaat van de index verandert zodra
'        een nieuwe werknemer aan de file wordt toegevoegd,
'        wordt de dimensie van de index() array
'        door middel van het REDIM statement bepaald.

      REDIM index(totWerkns%) AS indexType

      OPEN "WERKNMER.NDX" FOR INPUT AS #2

      FOR i% = 1 TO totWerkns%
            INPUT #2, index(i%).werknNummer, index(i%).werknNaam
      NEXT i%

'   ---- De index blijft tot het einde van de programma-executie
'        in het geheugen aanwezig. Sluit de indexfile dus voorlopig.

      CLOSE #2

END SUB
```

Pauze

```
SUB Pauze STATIC

'   Het Pauze subprogramma onderbreekt de programma-executie tijdelijk,
'   zodat de gebruiker een volgeschreven scherm rustig kan bestuderen.
'   De gebruiker drukt gewoon de spatiebalk in om door te gaan.
```

```
        LOCATE 25, 45, 1
        PRINT "Druk op spatiebalk voor doorgaan.";

        DO
              ka$ = INKEY$
        LOOP UNTIL ka$ = " "

        CLS

END SUB
```

PrintRecords

```
SUB PrintRecords STATIC

'   Het PrintRecords subprogramma drukt een lijst met werknemers af.
'   Het subprogramma biedt de gebruiker de mogelijkheid om vertrokken
'   werknemers al dan niet in de printout op te nemen.

        DIM printRecord AS werknType

        PRINT : PRINT : PRINT
        PRINT TAB(tabPos%); "Worden vertrokken werknemers"
        PRINT TAB(tabPos%); "in de printout "
        prompt$ = "opgenomen?"
        PRINT TAB(tabPos%);
        vertrokken% = JaNee%(prompt$)

        PRINT
        PRINT TAB(tabPos%); "Druk de spatiebalk in "
        PRINT TAB(tabPos%); "wanneer de printer operationeel is. ";

        DO
              pr$ = INKEY$
        LOOP UNTIL pr$ = " "

        LPRINT "   Achternaam        Voornaam      ZkfNr.  Functie";
        LPRINT "         Salaris"
        LPRINT "   ----------        --------      ------  -------";
        LPRINT "         -------"
        LPRINT

        FOR i% = 1 TO totWerkns%
              GET #1, index(i%).werknNummer, printRecord
              weg% = (LEFT$(printRecord.afdeling, 10) = "VERTROKKEN")

              IF weg% IMP vertrokken% THEN
                    IF weg% THEN
                          LPRINT "** ";
                    ELSE
                          LPRINT "   ";
                    END IF

                    LPRINT printRecord.achterNaam; " "; printRecord.voorNaam; " ";
                    LPRINT printRecord.zkfNummer; " "; printRecord.functie;

                    IF printRecord.salType = "U" THEN
                          LPRINT USING " **###.## /uur"; printRecord.salaris
                    ELSEIF printRecord.salType = "M" THEN
                          LPRINT USING " **#,#### /ma"; printRecord.salaris
                    ELSE
                          LPRINT USING " **#,##### /ja"; printRecord.salaris
                    END IF
              END IF
        NEXT i%

        IF vertrokken% THEN
              LPRINT : LPRINT : LPRINT
              LPRINT "** Vertrokken werknemer."
        END IF

        Pauze

END SUB
```

Sorteren

```
SUB Sorteren STATIC

'   Het Sorteren subprogramma sorteert de index() array met
'   het werknNaam element van de index() recordstructuur als
'   de sorteersleutel.
'   Dit subprogramma is gebaseerd op het Shell sorteeralgoritme.

      lengte% = totWerkns%
      stap% = 1
      DO WHILE stap% <= lengte%
            stap% = stap% * 2
      LOOP

      DO WHILE stap% > 1
            stap% = (stap% - 1) \ 2
            DO
                  voltooid% = true%
                  FOR hoger% = 1 TO lengte% - stap%
                        lager% = hoger% + stap%
                        IF index(hoger%).werknNaam > index(lager%).werknNaam THEN
                              SWAP index(hoger%), index(lager%)
                              voltooid% = false%
                        END IF
                  NEXT hoger%
            LOOP UNTIL voltooid%
      LOOP

END SUB
```

Toevoegen

```
SUB Toevoegen STATIC

'   Het Toevoegen subprogramma assisteert de gebruiker bij
'   het toevoegen van een nieuwe werknemer aan de database.

      DIM toevoegenRecord AS werknType

      PRINT : PRINT : PRINT : PRINT
      PRINT TAB(tabPos%); "Voer de gegevens voor de nieuwe werknemer in"
      PRINT TAB(tabPos%); "---- -- -------- ---- -- ------ --------- --"

'   ---- LeesNaam verwerkt de naam van de werknemer.

      LeesNaam toevoegenRecord, inFile%, nieuweNaam$

'   ---- Duplicaten van namen zijn niet toegestaan.

      IF inFile% THEN
            PRINT
            PRINT TAB(tabPos%); "Deze naam komt reeds in de file voor."
      ELSE

'   ---- Verwerk de overige gegevens voor de nieuwe werknemer.

            PRINT TAB(tabPos%);
            INPUT "ZkfNummer    --> ", toevoegenRecord.zkfNummer
            PRINT TAB(tabPos%);
            INPUT "Afdeling     --> ", toevoegenRecord.afdeling
            PRINT TAB(tabPos%);
            INPUT "Functie      --> ", toevoegenRecord.functie
            PRINT TAB(tabPos%);
            toevoegenRecord.salaris = LeesSalaris!
            PRINT TAB(tabPos%); "U)ur,"
            PRINT TAB(tabPos%); "M)aand, of"
            PRINT TAB(tabPos%); "J)aarbasis   --> ";
            toevoegenRecord.salType = LeesSalType$
            toevoegenRecord.beginDatum = DATE$

            PRINT
            PRINT TAB(tabPos%);

'   ---- De gebruiker dient de keuze te hebben om een
'        nieuw record ongedaan te maken (bijvoorbeeld indien een
'        fout is ontdekt) of naar de database te schrijven.
```

```
                IF JaNee%("Record opslaan?") THEN
                    totWerkns% = totWerkns% + 1
                    PUT #1, totWerkns%, toevoegenRecord
                    NieuweIndex nieuweNaam$
                END IF
            END IF

            Pauze

        END SUB
```

Vertrek

```
        SUB Vertrek STATIC

        '   Het Vertrek subprogramma herziet een record om aan te geven dat
        '   de werknemer in kwestie het bedrijf heeft verlaten, en wel in
        '   die zin dat het woord 'VERTROKKEN' in het vertrekRecord.afdeling veld
        '   wordt opgeslagen en het vertrekDatum veld de vertrekdatum
        '   (de systeemdatum) krijgt toegewezen.

            DIM vertrekRecord AS werknType

            DisplayRec vertrekRecord, welk%
            PRINT : PRINT : PRINT TAB(tabPos%);

            IF welk% > 0 THEN
                IF LEFT$(vertrekRecord.afdeling, 10) <> "VERTROKKEN" THEN
                    IF JaNee%("Is dit de vertrokken werknemer?") THEN
                        vertrekRecord.afdeling = "VERTROKKEN"
                        vertrekRecord.vertrekDatum = DATE$
                        PRINT
                        PRINT TAB(tabPos%); "*** Het vertrek is geregistreerd. ***"
                        PUT #1, welk%, vertrekRecord
                    ELSE
                        PRINT
                        PRINT TAB(tabPos%); "Geen verandering in status van werknemer."
                    END IF
                ELSE
                    PRINT
                    PRINT TAB(tabPos%); "Deze werknemer was al vertrokken."
                END IF
            END IF

            Pauze

        END SUB
```

Wijzigen

```
        SUB Wijzigen

        '   Het Wijzigen subprogramma verwerkt modificaties voor een werknemerrecord.

            DIM wijzigRecord AS werknType, wijzigMenu$(4)

            afdelingStr$ = "Afdeling ...> "
            functieStr$ = "Functie ....> "
            salStr$ = "Salaris ....> "
            wijzigMenu$(4) = "Record OK."
            PRINT : PRINT : PRINT : PRINT
            PRINT TAB(tabPos%); "Voer de naam van de werknemer in:"
            PRINT TAB(tabPos%); "---- -- ---- --- -- --------- ---"
            LeesNaam wijzigRecord, num%, dummy$
            IF num% = 0 THEN
                PRINT
                PRINT TAB(tabPos%); "Deze naam komt niet in de database voor."
                Pauze
            ELSE
                CLS
                GET #1, num%, wijzigRecord
                wijzigOptie% = 0
                DO
                    PRINT TAB(tabPos%); "Wijzigen van een werknemerrecord."
                    PRINT TAB(tabPos%); "Naam:  ";
                    PRINT RTRIM$(wijzigRecord.voorNaam); " "; wijzigRecord.achterNaam
                    wijzigMenu$(1) = afdelingStr$ + RTRIM$(wijzigRecord.afdeling)
                    wijzigMenu$(2) = functieStr$ + RTRIM$(wijzigRecord.functie)
```

```
                        SELECT CASE wijzigRecord.salType

                        CASE "U"
                              sal$ = " per uur"

                        CASE "M"
                              sal$ = " per maand"

                        CASE ELSE
                              sal$ = " per jaar"

                        END SELECT
                        sal$ = "f" + LTRIM$(STR$(wijzigRecord.salaris)) + sal$
                        wijzigMenu$(3) = salStr$ + sal$
                        finito% = false%
                        SELECT CASE Menu%(wijzigMenu$())

                        CASE 1
                              PRINT : PRINT : PRINT : PRINT : PRINT TAB(tabPos%);
                              INPUT "Nieuwe afdeling --> ", wijzigRecord.afdeling

                        CASE 2
                              PRINT : PRINT : PRINT : PRINT : PRINT TAB(tabPos%);
                              INPUT "Nieuwe functie --> ", wijzigRecord.functie

                        CASE 3
                              PRINT : PRINT : PRINT : PRINT : PRINT TAB(tabPos%); "Nieuw ";
                              wijzigRecord.salaris = LeesSalaris!
                              PRINT TAB(tabPos%);
                              PRINT "U)ur, M)aand, J)aarbasis --> ";
                              wijzigRecord.salType = LeesSalType$
                              Pauze

                        CASE ELSE
                              finito% = true%

                        END SELECT
                        CLS
                  LOOP UNTIL finito%
                  PUT #1, num%, wijzigRecord
            END IF

      END SUB

WisSpatie$  FUNCTION WisSpatie$ (invoer$)

      '   De WisSpatie$ functie verwijdert alle spaties uit het ontvangen
      '   stringargument. WisSpatie$ wordt gebruikt om de naam entries
      '   voor de index op te zetten en om de velditems op het scherm weer te geven.

            invoer$ = LTRIM$(RTRIM$(invoer$))
            tijdelijk$ = ""
            FOR i% = 1 TO LEN(invoer$)
                  enkelKar$ = MID$(invoer$, i%, 1)
                  IF enkelKar$ <> " " THEN
                        tijdelijk$ = tijdelijk$ + enkelKar$
                  END IF
            NEXT i%

            WisSpatie$ = tijdelijk$

      END FUNCTION

Zoek%       FUNCTION Zoek% (welkeTekst$) STATIC

      '   De Zoek% functie voert een binaire zoekactie uit.
      '   De routine zoekt naar een doelstring (welkeTekst$) in het
      '   werknNaam element van de index() array. Indien deze is gevonden
      '   retourneert Zoek% de overeenkomstige integer in het
      '   werknNummer element van de index() array.
      '   Indien de doelstring niet is gevonden retourneert Zoek% een waarde van nul.

            begin% = 1
            einde% = totWerkns%
```

```
    gevonden% = false%
    leesNr% = 0

    DO WHILE begin% <= einde% AND NOT gevonden%
          midden% = (begin% + einde%) \ 2
          indexNaam$ = RTRIM$(index(midden%).werknNaam)
          IF welkeTekst$ = indexNaam$ THEN
                gevonden% = true%
                leesNr% = index(midden%).werknNummer
          ELSEIF welkeTekst$ > indexNaam$ THEN
                begin% = midden% + 1
          ELSE
                einde% = midden% - 1
          END IF
    LOOP

    Zoek% = leesNr%

END FUNCTION
```

Voor het kopiëren en gebruiken van de listing hebt u het kwartaalbestand nodig. Meer over listings in kwartaalbestanden, zie het hoofdstuk 'De listings in de kwartaalbestanden'.

Marco Kurvers

Grafisch programmeren in GW-BASIC (4).

De volgende drie programma's, programma 8, 9 en 10, zullen nog meer grafieken van functies in cartesische vorm laten zien. In de vorige nieuwsbrief heeft u daarvan programma 7 kunnen zien die als voorbeeld een trilling tekende.

Denk er weer aan dat u voor Visual Basic .NET de grafische `e` parameter in de `Paint` event gebruikt. Straks ga ik het ook even over de parameter hebben die voor kleur zorgt: het `Pen` object.

De lijst die de vaste variabelen laat zien staat nu in een apart bestand. Normaal stond die hieronder. Open het kwartaalbestand 'Variabelen.txt' als u de beschrijvingen wilt weten.

De voorbeelden

Programma 8 tekent tien in fase verschoven sinuskrommen, allemaal in hetzelfde coördinatenstelsel. De vergelijking van het stelsel is:

$$y = \sin(x+np), \qquad \text{met als fase} \qquad p = \frac{\pi}{9} \qquad (20^0).$$

We vinden de tien vergelijkingen door voor n de waarden 0, 1, 2 t/m 9 te kiezen. De lusvariabele `J` (regel 170) komt overeen met de HRG beeldschermcoördinaten X2. Omdat we in dit programma de beeldschermcoördinaat X2, dat wil zeggen `J`, in stappen van 5 naar 320 laten lopen (regel 170), moeten we eerst de bij X2 behorende waarde voor x berekenen. Dan pas kunnen we de functiewaarde **y = f(x)** berekenen. We kunnen de evenredigheid

$$(x-a) : (b-a) = X2 : 320$$

natuurlijk ook gebruiken om x uit X2 te berekenen; hieruit volgt:

$$x = \frac{(b-a)}{320} . X2 + a.$$

In het onderstaande programma zijn de gekozen intervalgrenzen **a = 0** en **b = 2**π, zodat de bovenstaande vergelijking wordt:

$$x = \frac{2\pi}{320} . X2.$$

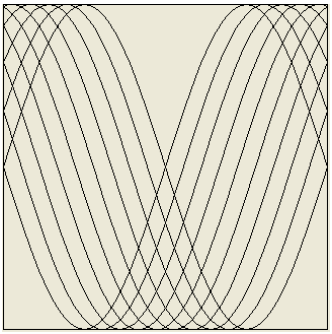
Als we $C = \frac{2\pi}{320}$ nemen, wordt in het programma het argument van de sinusfunctie dus C*X2+N*P, ofwel C*J+N*P.

In het programma zien we C*J+N*P in regel 180 als argument van de sinusfunctie. Natuurlijk moeten we de functiewaarde y = SIN(C*J+N*P) nog omzetten naar de beeldschermcoordinaat Y zodat we krijgen

```
Y2 = INT(V-K*SIN(C*J+N*P)+H).
```

Door voor de schaalconstante `K` de waarde 160 te kiezen (regel 140) krijgen we een mooie 'grote' tekening.

Wie met kleuren wil werken kan dit programma uitbreiden door de diverse krommen andere kleuren te geven. Iets moeilijker zal het zijn de 'banen' tussen de sinuskrommen in te kleuren. Probeer het eens, als u tenminste over kleuren beschikt. Straks daar meer over.



```
100 '           programma 8          10 SINUSCURVEN
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 V=160:K=160:H=.5:P=4*ATN(1)/9:C=2*4*ATN(1)/320
150 FOR N=0 TO 9
160     X1=0: Y1=INT(V-K*SIN(N*P)+H)
170     FOR J=5 TO 320 STEP 5
180         X2=J: Y2=INT(V-K*SIN(C*J+N*P)+H)
190         LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
200         X1=X2 : Y1=Y2
210     NEXT J
220 NEXT N
230 LINE (FNX(0),0)-(FNX(320),320),1,B
240 A$=INKEY$: IF A$="" THEN 240
250 CLS: KEY ON: END
```

```
Dim V As Single = 160, K As Single = 160
```

```
Dim H As Single = 0.5
Dim P As Single = 4 * Math.Atan(1) / 9
Dim C As Single = 2 * 4 * Math.Atan(1) / 320
For N As Single = 0 To 9
    Dim X1 As Single = 0
    Dim Y1 As Single = Int(V - K * Math.Sin(N * P) + H)
    For J As Single = 5 To 320 Step 5
        Dim X2 As Single = J
        Dim Y2 As Single = Int(V - K * Math.Sin(C * J + N * P) + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
Next
e.Graphics.DrawRectangle(Pens.Black, 0, 0, 320, 320)
```

Hieronder ziet u de kleurcodes die u kunt gebruiken.

Code	Kleur
0	zwart
1	blauw
2	groen
3	cyaanblauw
4	rood
5	paars
6	bruin
7	wit
8	grijs
9	lichtblauw
10	lichtgroen
11	licht cyaanblauw
12	lichtrood
13	lichtpaars
14	geel
15	superwit

Hebt u een kleuren/graphics adapter, kies dan, als u met kleuren wilt werken, in plaats van `SCREEN 105,,3,3` de opdrachten `SCREEN 1,0,, : COLOR 9,1`. Dit stelt de middenresolutie (320 bij 200) in met een lichtblauwe achtergrond (kleur 9) en een voorgrondkleurenpalet (1) met de kleuren cyaanblauw (nr. 1), paars (nr. 2) en wit (nr. 3). Met `COLOR 9,0` kiest u een ander palet met de drie mogelijke afdrukkleuren groen (nr. 1), rood (nr. 2) en bruin (nr. 3).

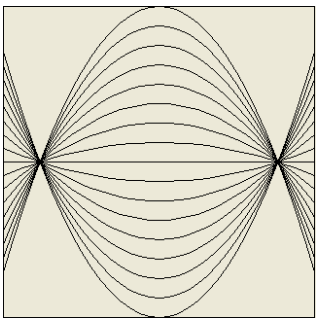
Werkt u met Visual Basic .NET, zorg er dan voor dat u een variabele declareert van het `Pen` object die u dan als eerste parameter in de methode `DrawLine()` kunt meegeven. De volgende programma's geven meer voorbeelden voor het werken met kleuren.

Programma 9 tekent een stelsel parabolen met de vergelijking:

$$y = -tx^2 + t$$

Alle parabolen (bepaalde t-waarden) hebben dezelfde snijpunten met de x-as (x = 1 en x = -1).

```
100 '           programma 9     PARABOOLSTELSEL
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
150 FOR K= -160 TO 160 STEP 20
160     X=-160: Y=-K*X*X/15000+K
170     X1=INT(U+X+H) : Y1=INT(V-Y+H)
```



```
180     FOR X=-150 TO 160 STEP 10
190         Y=-K*X*X/15000+K
200         X2=INT(U+X+H) : Y2=INT(V-Y+H)
210         LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
220         X1=X2 : Y1=Y2
230     NEXT X
240 NEXT K
250 LINE (FNX(0),0)-(FNX(320),320),1,B
260 A$=INKEY$: IF A$="" THEN 260
270 CLS: KEY ON: END
```

```
Dim U As Single = 160, V As Single = 160
Dim H As Single = 0.5
For K As Single = -160 To 160 Step 20
    Dim X As Single = -160
    Dim Y As Single = -K * X * X / 15000 + K
    Dim X1 As Single = Int(U + X + H), Y1 As Single = Int(V - Y + H)
    For X = -150 To 160 Step 10
        Y = -K * X * X / 15000 + K
        Dim X2 As Single = Int(U + X + H), Y2 As Single = Int(V - Y + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
Next
e.Graphics.DrawRectangle(Pens.Black, 0, 0, 320, 320)
```

Kunt u met kleuren werken, verander dan `SCREEN 105,,3,3` in:

`SCREEN 1,0,, : COLOR 9,1` en voeg toe

```
    185 IF X<=0 THEN KLEUR=2 ELSE KLEUR=3
```

en verander de `LINE` opdracht in regel 210 in:

```
    LINE (FNX(X1),Y1)-(FNX(X2),Y2),KLEUR
```

Ook heel mooi wordt het met

```
    185 IF INT((K/10)/2)*2=K/10 THEN KLEUR=2 ELSE KLEUR=3
```

Vergeet ook niet de `DEF FNX` in regel 120 te veranderen, zie nieuwsbrief nr. 4 vorig jaar, en neem in regel 140 voor `V` de waarde `100`. Ook in regel 250 moet de y-coördinaat `320` veranderd worden in `200`.

Nu is de vraag: hoe doen we dit met het `Pen` object als we een eigen `Pens` argument willen meegeven? Om bovenstaande GW-BASIC kleurvoorbeelden te gebruiken, moeten we eerst bovenaan de `Dim` statements een objectinstantie declareren van het objecttype `Pen`.
De declaratie gaat als volgt:

```
    Dim EigenPen As System.Drawing.Pen
```

We kunnen dan een regel als regel 185 toevoegen

```
    If X <= 0 Then EigenPen = Pens.Green Else EigenPen = Pens.DarkCyan
```

en net zo als in GW-BASIC het tekenen wijzigen in:

```
    e.Graphics.DrawLine(EigenPen, X1, Y1, X2, Y2)
```

We kunnen regel 185 nabootsen door het in Visual Basic .NET net zo mooi te krijgen:

```
    If Int((K / 10) / 2) * 2 = K / 10 Then EigenPen = Pens.Green Else
                                                    EigenPen = Pens.DarkCyan
```

Wat regel 250 betreft, die hoeft voor Visual Basic niet gewijzigd te worden.

Soms willen we de oppervlakte tussen de grafiek van een functie en de x-as berekenen (de integraal bepalen) of laten tekenen.

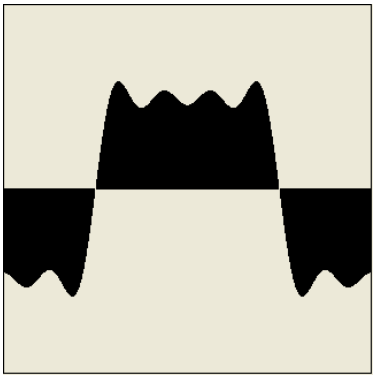
Programma 10 kleurt zo'n oppervlak tussen de x-as en de grafiek van de functie.

$$y = \cos(x) - \frac{\cos(3x)}{3} + \frac{\cos(5x)}{5} - \frac{\cos(7x)}{7}.$$

Ook nu is de lusvariabele J de schermcoördinaat X2. De waarde voor a is $-\pi$ en die voor b is $+\pi$, waaruit volgt dat

$$x = \frac{2\pi}{320}. J - \pi.$$

Ook nu nemen we in het programma $C = \frac{2\pi}{320}$.



```
100 '   programma 10   OPPERVLAK ONDER EEN KROMME
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 V=160: H=.5: K=100: PI=4*ATN(1): C=2*PI/320
150 FOR J=0 TO 320
160     X=C*J-PI : GOSUB 1000
170     Y=INT(V-K*Y+H)
180     LINE (FNX(J),V) - (FNX(J),Y),1
190 NEXT J
200 LINE (FNX(0),0) - (FNX(320),320),1,B
210 A$=INKEY$: IF A$="" THEN 210
220 CLS: KEY ON: END
230 '
1000 Y=COS(X)-COS(3*X)/3+COS(5*X)/5-COS(7*X)/7
1010 RETURN
```

```
Function Cartvorm(ByVal pX As Single) As Single
     Return Math.Cos(pX) - Math.Cos(3 * pX) / 3 + Math.Cos(5 * pX) / 5 -
                                                          Math.Cos(7 * pX) / 7
End Function

'Onderstaande code moet in de Paint event

Dim V As Single = 160, H As Single = 0.5, K As Single = 100
Dim C As Single = 2 * Math.PI / 320
For J As Single = 0 To 320
     Dim X As Single = C * J - Math.PI, Y As Single = Cartvorm(X)
     Y = Int(V - K * Y + H)
     e.Graphics.DrawLine(Pens.Black, J, V, J, Y)
Next
e.Graphics.DrawRectangle(Pens.Black, 0, 0, 320, 320)
```

Merk op dat in Visual Basic .NET niet de `PI` variabele nodig is. Die is al aanwezig in het `Math` object.
Omdat de cartesische functie maar één keer aangeroepen wordt, kunt u de inhoud ook rechtstreeks in de `Paint` event schrijven. U hoeft dan geen aparte functie te maken. U moet dan in de lus de aanroep van de functie vervangen door de inhoud die in regel 1000 staat.

Probeer dit programma zo te maken dat de drie oppervlakken in de drie beschikbare paletkleuren gekleurd worden. Dit kan alleen voor de systemen met een kleuren/graphics adapter in middenresolutie (320 bij 200). Met de `COLOR` opdracht kan dan een achtergrond (16 kleuren) gekozen worden en voor de afdrukkleur kan gekozen worden uit twee groepen van drie kleuren (zie programma 8).

Niet overal continue functies

De functie $y = \frac{x^2+3}{x^2-x-6}$ kan niet met behulp van programma 7 getekend worden. Zouden we bijvoorbeeld voor x het interval –5 <= x <= 5 kiezen, dan liggen hierin de punten x = 3 en x = -2. Dit zijn de twee punten waarvoor de noemer van bovenstaande functie nul is. Als de computer bij het tekenen bij één van deze twee punten belandt, zal op het scherm een foutmelding (DIVISION BY ZERO) 'delen door nul' verschijnen en zal het programma afgebroken worden.

Dit zal ook gebeuren bij logaritmische functies met een niet-positief argument of bij wortelfuncties met een negatief argument. Zelfs bij het tekenen van continue (nette) functies kunnen problemen optreden. We komen hierbij in de problemen als de functiewaarden heel groot of heel klein worden. Transformatie van dergelijke waarden naar onze beeldschermcoördinaten (0-320 en 0-320) heeft dan geen enkele zin meer, omdat de aard van het verloop van de functie in het geheel niet meer tot uitdrukking komt. Het is daarom beter om de functiewaarden van een continue functie naar boven en beneden te begrenzen. Dit heeft tot gevolg dat de grafiek van een continue functie er uit zou kunnen zien als de grafiek van een niet-continue functie.

Het zal duidelijk zijn dat je een algemeen programma voor het tekenen van een willekeurige continue of niet-continue functie niet zomaar even opschrijft. Dergelijke programma's kom je in de vakliteratuur dan ook niet of nauwelijks tegen, en mocht je er wel een tegenkomen, dan betreft het òf een heel groot programma òf een programma dat voor een bepaalde functie, waarvan men van te voren weet waar de discontinuïteiten zitten, geschreven is.

Ik geef nu een verbazend kort programma voor het tekenen van de grafiek van een willekeurige functie **y = f(x)**. Veel wiskundeleraren, scholieren en studenten zullen hun 'oren' spitsen. U begrijpt dat, gezien het bovenstaande, hierbij wel enkele beperkingen gelden:

1. Degene die het programma gaat gebruiken moet een beetje kunnen programmeren. De functie moet namelijk in gedeelten in een subroutine (vanaf regel 1000) beschreven worden.

2. De programmagebruiker moet voldoende wiskundige kennis bezitten om te kunnen bepalen voor welke x-waarden functies moeilijkheden kunnen opleveren.

We zullen zien dat, normaal gesproken, slechts een paar BASIC regels nodig zijn om de functiebeschrijving te programmeren. Het berekenen van nulpunten met behulp van een of ander numeriek wiskundig algoritme is in het geheel niet nodig.

In het onderstaande programma 11 gebruiken we twee variabelen `FZ` en `FA` als zogeheten vlaggen (Flags). Een vlag is een variabele die slechts twee waarden (vaak 0 en 1) kan aannemen. (Ook wel booleaanse waarden onwaar en waar genoemd, voor de kenners.)

Om de werking van de vlag FZ duidelijk te maken geven we hieronder de subroutine 1000 met de functiebeschrijving van de functie.

$$y = \frac{x^2+3}{x^2-x-6}$$

```
1000 N=X*X-X-6: IF N=0 THEN FZ=1: RETURN
1010 Y=(X*X+3)/N
1020 IF Y<LP OR Y>HP   THEN FZ=1: RETURN
1030 FZ=0: RETURN
```

De vlag FZ wordt alleen op 1 gezet als het punt (x,y) niet op het scherm getekend kan worden. Dit is het geval als de functie niet-continu is in het punt (x,y) (N = 0; x = 3 en x = -2) of als de functiewaarde buiten het opgegeven functiebereik (LP <= y <= HP) valt.

Waarvoor dient nu de tweede vlag FA? Bekijk het volgende stukje programma eens:

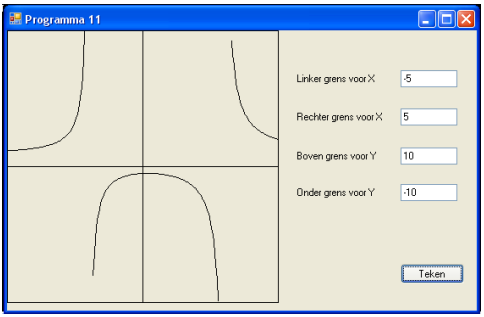
```
220 FA=1
230 FOR X=A TO B STEP DX
240     X2=INT(KX*(X-A)+H) : GOSUB 1000
250     IF FZ=1 THEN FA=1 : GOTO 310
260     IF FA=1 THEN 300
270     Y2=INT(KY*(HP-Y)+H)
280     LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
290     X1=X2 : Y1=Y2 : GOTO 310
300     X1=X2 : Y1=INT(KY*(HP-Y)+H) : FA=0
310 NEXT X
```

Als FZ = 1 dan wordt FA ook gelijk aan 1 gemaakt en wordt de volgende X-waarde bekeken (FOR lus) zonder dat het nieuw berekende punt met het laatstgetekende punt verbonden wordt. Is FZ echter 0 (berekende punt ligt in het beeldvlak) dan wordt onderzocht of FA daarvoor soms op 1 gezet is. Is dit zo (tweede THEN) dan moet het nieuw berekende punt als nieuw beginpunt (X1,Y1) gekozen worden en dit mag dan ook niet met het vorige getekende punt verbonden worden. FA wordt in dit geval nul gemaakt.

Dus als zowel FZ als FA nul zijn, wordt het nieuw berekende punt verbonden met het laatstgetekende punt.

Om te zorgen dat het programma probleemloos werkt moet voor het berekenen van het eerste punt van de grafiek, dus voor de FOR lus, de vlag FA op 1 gezet worden, waardoor we er zeker van zijn dat de waarden X1 en Y1 berekend worden. Hier volgt het volledige programma:

```
100 ' progr.11  GRAFIEK VAN EEN WILLEKEURIGE FUNCTIE
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 "BOVEN   GRENS VOOR Y " ; HP: PRINT
170 INPUT "ONDER   GRENS VOOR Y " ; LP
180 IF A > B THEN C=A : A=B : B=C
190 KX=320/(B-A) : KY=320/(HP-LP) : H=.5
200 DX=(B-A)/256
210 CLS
220 FA=1
230 FOR X=A TO B STEP DX
240     X2=INT(KX*(X-A)+H) : GOSUB 1000
250     IF FZ=1 THEN FA=1 : GOTO 310
260     IF FA=1 THEN 300
270     Y2=INT(KY*(HP-Y)+H)
280     LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
290     X1=X2 : Y1=Y2 : GOTO 310
300     X1=X2 : Y1=INT(KY*(HP-Y)+H) : FA=0
310 NEXT X
320 X=INT(KX*(-A)+H): Y=INT(KY*HP+H)
330 IF Y<0  OR  Y>320  THEN GOTO 350
340     LINE (FNX(0),Y) - (FNX(320),Y),1
350 IF X<0  OR  X>320  THEN GOTO 370
360     LINE (FNX(X),0) - (FNX(X),320),1
370 LINE (FNX(0),0) - (FNX(320),320),1,B
380 A$=INKEY$: IF A$="" THEN 380
390 CLS: KEY ON: END
1000 N=X*X-X-6: IF N=0 THEN FZ=1: RETURN
1010 Y=(X*X+3)/N
1020 IF Y<LP OR Y>HP   THEN FZ=1: RETURN
1030 FZ=0: RETURN
```



Dit programma tekent deze afbeelding uit de volgende waarden:
Kies voor A, B, HP en LP respectievelijk –5, 5, 10 en –10

Onderstaande listing is voor Visual Basic .NET. Omdat er veel berekend en gesprongen wordt en ook weer gebruik wordt gemaakt van invoer, zal de code er weer heel anders uit zien. Probeer de regels te volgen en hoe ik vooral de functie Cartvorm heb geschreven die regel 1000 heeft vervangen en hoe ik de GOTO statements heb vermeden.

```
Private FZ, FA As Boolean     'declareer deze variabelen buiten de subprocedures
Private A, B, HP, LP As Single

Private Function Cartvorm(ByVal X As Single, ByRef Y As Single) As Boolean
     Dim N As Single = X * X - X - 6
     If N = 0 Then Return True
     Y = (X * X + 3) / N
     If Y < LP Or Y > HP Then Return True
```

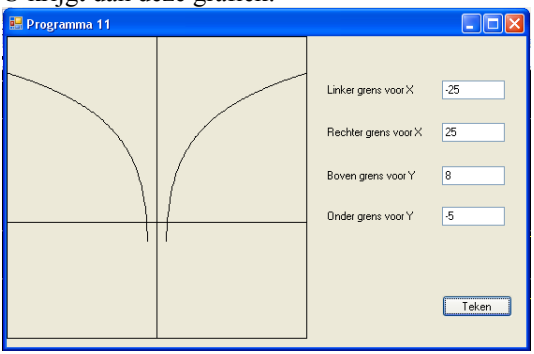
```
     Return False
End Function

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

Private Sub frmProg11_Paint(ByVal sender As Object, ByVal e As ...PaintEventArgs) ...
     If Not (Val(txtA.Text()) = 0 Or Val(txtB.Text() = 0 Or _
               Val(txtHP.Text()) = 0 Or Val(txtLP.Text()) = 0 Then
          A = Val(txtA.Text()) : B = Val(txtB.Text())
          HP = Val(txtHP.Text()) : LP = Val(txtLP.Text())
          If A > B Then
               Dim C As Single = A
               A = B
               B = C
          End If
          Dim KX As Single = 320 / (B - A), KY As Single = 320 / (HP - LP)
          Dim H As Single = 0.5, DX As Single = (B - A) / 256
          Dim X As Single = 0, Y As Single = 0
          Dim X1 As Single = 0, Y1 As Single = 0, X2 As Single = 0, Y2 As Single = 0
          FA = True
          For X = A To B Step DX
               X2 = Int(KX * (X - A) + H) : FZ = Cartvorm(X, Y)
               If Not FZ Then
                    If Not FA Then
                         Y2 = Int(KY * (HP - Y) + H)
                         e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
                         X1 = X2 : Y1 = Y2
                    Else
                         X1 = X2 : Y1 = Int(KY * (HP - Y) + H) : FA = False
                    End If
               Else
                    FA = True
               End If
          Next
          X = Int(KX * (-A) + H) : Y = Int(KY * HP + H)
          If Not (Y < 0 Or Y > 320) Then
               e.Graphics.DrawLine(Pens.Black, 0, Y, 320, Y)
          End If
          If Not (X < 0 Or X > 320) Then
               e.Graphics.DrawLine(Pens.Black, X, 0, X, 320)
          End If
          e.Graphics.DrawRectangle(Pens.Black, 0, 0, 320, 320)
     End If
End Sub
```

Wilt u een andere functie tekenen, bijvoorbeeld **$y = \ln(x^2 - 2)$**, herschrijf dan subroutine 1000, en uiteraard de functie Cartvorm, als volgt:

```
'GW-BASIC
1000 U=X*X-2 : IF U<=0 THEN FZ=1:RETURN
1010 Y=LOG(U)
1020 IF Y < LP OR Y > HP THEN FZ=1:RETURN
1030 FZ=0: RETURN

'Visual Basic.NET
Dim U As Single = X * X - 2 : If U <= 0 Then Return True
Y = Math.Log(U)
If Y < LP Or Y > HP Then Return True
```

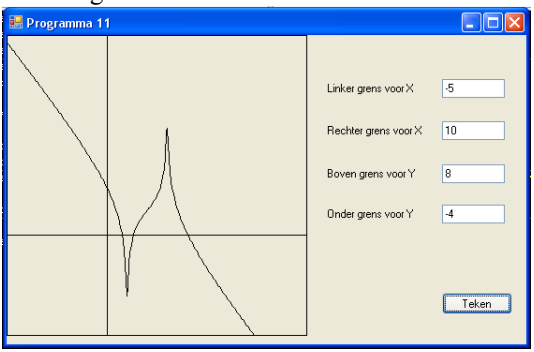
```
Return False
```

U krijgt dan deze grafiek:



Kies voor A, B, HP en LP de waarden -25, 25, 8 en –5.

Hier volgt een hele fraaie:



$$y = 3 - x + \ln\left|\frac{x-1}{x-3}\right|$$

Subroutine 1000 en de functie Cartvorm worden nu:

```
'GW-BASIC
1000 N=X-3 : IF N=0 THEN FZ=1:RETURN
1010 Y=3-X+LOG(ABS((X-1)/N))
1020 IF Y < LP OR Y > HP THEN FZ=1:RETURN
1030 FZ=0: RETURN

'Visual Basic.NET
Dim N As Single = X - 3 : If N = 0 Then Return True
Y = 3 - X + Math.Log(Math.Abs((X - 1) / N))
If Y < LP Or Y > HP Then Return True
Return False
```

Kies voor A, B, HP en LP de waarden –5, 10, 8 en –4.

Dit was het hoofdstuk over continue en niet-continue functies in cartesische vorm. In de volgende nieuwsbrief komt het volgende hoofdstuk met uitleg en voorbeelden over krommen in poolcoördinaten en in parametervorm.

Bron: IBM- en GW-BASIC graphics van Academic Service
Tekst overname, tips en veranderingen: Marco Kurvers

BASIC nieuws en tips

Penningmeester gezocht.

Onze huidige penningmeester, Piet Boere, heeft te kennen gegeven dat hij op de Algemene Ledenvergadering van 2010 zijn functie ter beschikking stelt.
Het bestuur roept daarom de leden op zich voor deze functie beschikbaar te stellen.
Aanmeldingen graag per e-mail naar een van de volgende adressen :

voorz@basic-gg.hcc.nl
secr@basic-gg.hcc.nl
penm@basic-gg.hcc.nl

Puzzel: het juiste BASIC onderwerp.

Onderstaande puzzel bestaat uit omschrijvingen waarvan de juiste betekenissen gevonden moeten worden. Probeer het juiste BASIC onderwerp te vinden met de eerste letters van de betekenissen. Dus de eerste letter van de eerste betekenis is de eerste letter van het onderwerp, de eerste letter van de tweede betekenis is de tweede letter van het onderwerp enzovoort.

Aan deze variabele(n) kunnen alfanumerieke waarden toegekend worden.	Aantal letters: 6 (of 7)
Met deze functie verwijderen we de spaties in de tekst.	Aantal letters: 4
Deze functie zorgt ervoor dat negatieve waarden positief worden.	Aantal letters: 3
We moeten een beginwaarde en een eindwaarde opgeven met een FOR lus. Wat hebben we daarvoor nodig?	Aantal letters: 2
De meeste BASIC versies gebruiken dit commando om een programma te beëindigen.	Aantal letters: 3
Sommige BASIC versies ondersteunen een commando om twee teksten samen te voegen.	Aantal letters: 5
In QuickBASIC en latere versies kunnen we foutafhandeling routines schrijven met een commando na ON.	Aantal letters: 5
Met dit BASIC sleutelwoord kunnen we het resultaat van een booleaanse conditie omdraaien.	Aantal letters: 3
Met deze BASIC systeemvariabele kunnen we de tijd instel-	Aantal letters: 4

len.	

Het BASIC onderwerp is: . . . . . .

Marco Kurvers

De verschillende soorten sprites in Game Maker.

Sprites kunnen we opbergen in verschillende soorten bestanden, te weten:

- in één image bestand;
- in een GIF bestand met meerdere images;
- in een strip bestand.

In één image bestand.

Een image kan een JPEG of een BMP zijn, maar Game Maker kent nog meer soorten imagetypen. Wanneer maar één image voor een sprite gebruikt wordt, spreken we van een één image sprite.
Onthoud dat de sprite zelf niet het bestand is. De sprite heeft alleen de inhoud van de image(s). Dat geldt ook als we bestanden van verschillende soorten typen in Basic in willen laden. Visual Basic 6 heeft een image control die u kunt plaatsen en het imagebestand kunt opgeven. Helaas is dat het enige in Visual Basic 6. De Visual Basic .NET versies bieden nog veel meer mogelijkheden door geen gebruik te maken van een image control, maar direct de images te plaatsen (te tekenen) in de Paint event. Ook hier hebt u parameter e voor nodig met het Graphics object, zie het onderwerp 'Grafisch programmeren in GW-BASIC'.
In Visual Basic 6 zou u gebruik moeten maken van API subprocedures, mocht u gebruik willen maken van de code en niet van de control.

In een GIF bestand met meerdere images.

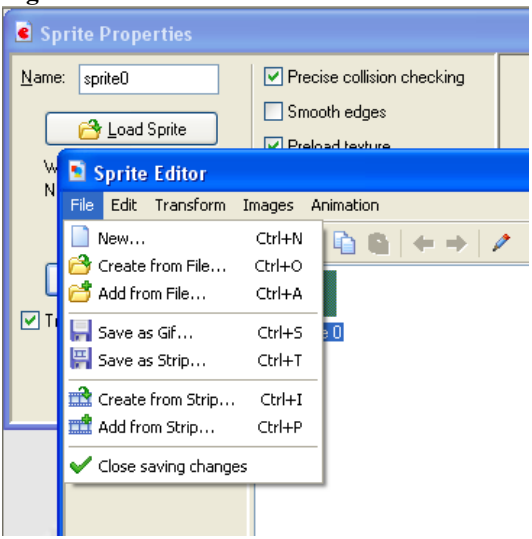
Een GIF (Graphics Images Format) kan meerdere images hebben. Als voorbeeld zorgen 10 images ervoor dat er een sprite-animatie ontstaat met 10 images. Hoe meer images, hoe gladder de animatie zal zijn.
Onthoud dat elke GIF maar één sprite is, ook al kan de inhoud uit meerdere images bestaan.

In een strip bestand.

Het lijkt erop, een stripblad, maar eigenlijk moet u het zien als een vel met allemaal stickers die naast elkaar en onder elkaar geplakt zitten.
Voordat we een strip kunnen bewerken om de juiste sprites te kiezen, moeten we aangeven dat het bestand niet gaat om een image en ook niet gaat om een GIF. Met andere woorden, we moeten Game Maker zelf een grid op de plaatjes laten leggen, zodat we elk plaatje kun-

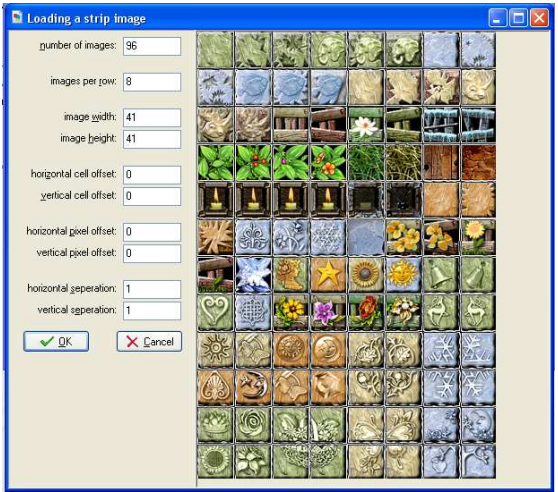
nen kiezen en niet zelf alles eruit hoeven te knippen. Figuur 1 laat het menu zien van de sprite editor.

Figuur 1.



Om sprite0 te kunnen creëren, moet u eerst naar het menu Resources van het hoofdmenu. U kunt ook de rechter muisknop gebruiken door aan de linker-kant op Sprite te klikken en Create Sprite te kiezen. U ziet dan het formulier Sprite Properties verschijnen.
Klik op de knop Edit Sprite en de Sprite Editor verschijnt. Als u op menu File klikt, ziet u het popup menu met de mogelijkheden die hierboven besproken zijn.
Een enkele file maken of laden, dus één image, een GIF file maken of laden, dus één of meerdere images in een bestand en een strip file maken of laden die dus eerst uit meerdere plaatjes bestaan en daarna pas laten bepalen welke plaatjes in een sprite toegevoegd moeten worden.

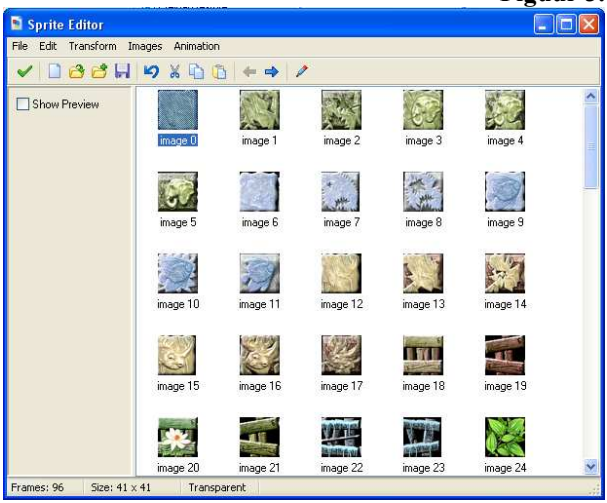
Kies het menu-item 'Create from Strip...'. Eerst zullen de bestanden verschijnen waaruit u het juiste strip bestand kunt openen. Als voorbeeld heb ik een prachtig stripbestand dat bewerkt kan worden, zie Figuur 2. Hier kunt u zien dat het gaat om een vel vol met sticker-plaatjes. Door de juiste grootte van de image en het aantal op te geven, zullen er randen om de plaatjes worden gemaakt, zodat elk plaatje goed in elk vakje past.



Figuur 2.
Hier ziet u dat het gaat om 96 images met een grootte van 41x41 punten per

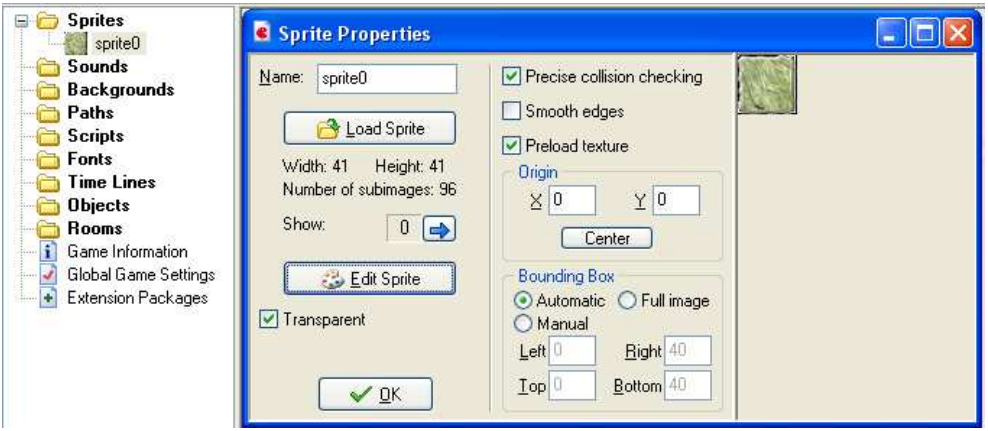
image. Toch kan het gebeuren dat sommige images van de andere images afwijken van grootte. Om ervoor te zorgen ze in alle vakjes te krijgen, kunnen er offsets en separaties opgegeven worden. Figuur 2 laat zien dat zowel een horizontale separatie als een verticale separatie nodig is. Hiermee kunt u de vakken op afstand van elkaar houden. Wilt u de vakken verschuiven (alleen de randen), geef dan een andere pixel offset op. Ook kunt u een deel van de strip kiezen door een cell offset op te geven.
Als u klaar bent en op OK klikt, ziet u dat de images uit de vakken worden geknipt, zie Figuur 3.

Figuur 3.



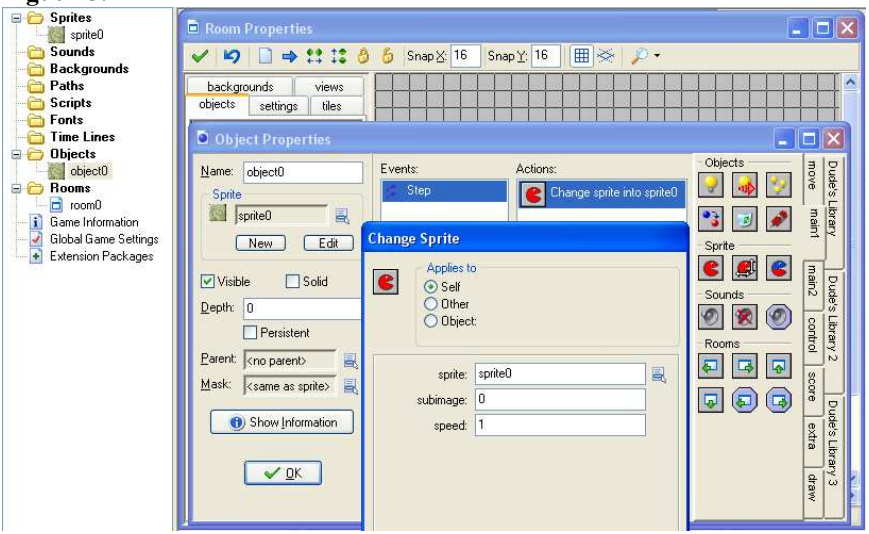
Selecteert u 'Show Preview' dan zullen alle images achter elkaar getoond worden. Al deze images zullen dan ook in één sprite opgeslagen worden. Dit is misschien niet wat u wilt. Gelukkig is het mogelijk om ook per image te werken door de index van de image op te geven, zoals we ook in BASIC kunnen werken met een array.
Zoals u Figuur 4 zult zien, lijkt er niets aan de hand te zijn terwijl de sprite wel uit 96 images bestaat. We hoeven dit dus niet als een GIF te beschouwen.

Figuur 4.



U ziet hier dat index 0 de eerste image is. Maken we er een object van en plaatsen we dan een instantie van het object op een room dan zullen we nog steeds de eerste image zien. Starten we echter de room dan zullen alle images getoond worden op één plaats. Figuur 5 laat zien hoe we dit kunnen vermijden.

Figuur 5.



Door in de Object Properties de Step event te gebruiken en daar uit sprite0 subimage: 0 op te geven, zal tijdens het draaien van de game niet meer alle images op de room getoond worden. Dat komt doordat de Step event bij elke stap die uitgevoerd wordt de actie 'Change Sprite' uitvoert. We kunnen daarom de Step event vergelijken met de Basic event Paint. Maar de objecten in Game Maker hebben ook een Draw event die we ook als de Paint event kunnen beschouwen. Willen we bijvoorbeeld de score weergeven, dan moeten we ervoor zorgen dat het tekenen van de score in de Draw event staat. De Step event kunnen we daar-

voor niet gebruiken, die zorgt voor het bepalen van de spelonderdelen, de objectinstanties en voor de controle (bijvoorbeeld of het scherm wel of niet leeg is, het lopende mannetje buiten de room komt enzovoort). Dit is een goede voorbeeld om ook in Basic zulke acties in de Paint event aan te roepen. Helaas missen we in Basic een soortgelijke event als de Step event in Game Maker. Het maken van games is daarom in een programmeertaal een stuk moeilijker. In Basic is het wel mogelijk om games te maken.

Marco Kurvers

De listings in de kwartaalbestanden.

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.

Cursussen

Liberty Basic Cursus, lesmateriaal en voorbeelden op CD-ROM € 6,- voor leden. Niet leden € 10,-

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

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

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

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

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 QuickBasic	Jan v.d. Linden				j.vd.linden@kader.hcc.nl
Visual Basic voor Windows Visual Basic .NET	Jeroen v. Hezik Marco Kurvers	ma. t/m zo. do. t/m zo.	19-21 19-22	(0346) 214131 (0342) 424452	j.a.van.hezik@kader.hcc.nl m.a.kurvers@hccnet.nl
Basic algemeen, zoals VBA Office Web Design, met XHTML en CSS	Marco Kurvers	do. t/m zo.	19-22	(0342) 424452	m.a.kurvers@hccnet.nl