Nieuwsbrief

17de jaargang juni 2010 Nummer 2

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Het Bestuur	Penningmeester gezocht: 38
Henk van Weers	Oplossing puzzel: 38

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

Wie kan er programmeren, iedereen toch die een programmeertaal heeft geleerd? Helaas is het ontwerpen en schrijven van een applicatie nog altijd een klus en vergt het veel tijd. 'Ik kan programmeren' vertelt u wat u tegen kunt komen als u na het leren de praktijk in gaat.

De Intermezzo gaat verder met het Werknemers project. Deel 2 laat u zien hoe het menu ingedeeld wordt en hoe de dialoogschermen van de menuopties eruit komen te zien.

In nieuwsbrief nr 4 van vorig jaar heb ik u wat laten proeven met Game Maker. Een goed en makkelijk programma om spellen te maken, zonder eventueel code te hoeven schrijven. Voor beginners heb ik daarom wat pagina's geschreven over hoe we sprites, objecten en acties op kunnen bouwen en kunnen besturen.

Marco Kurvers

BASIC cursus: VBA met Excel (2).

De werkbladen van Excel hebben iets wat elke programmeur zou willen; zowel tekstverwerken als notabonnen kunnen schrijven. Om bonnen te kunnen schrijven moeten we wel berekeningen uit kunnen voeren. Daar hebben we helemaal geen programmeertaal voor nodig, ook al heeft Excel de ingebouwde VBA macrotaal. Maar VBA geeft een Excel gebruiker de mogelijkheid om de werkbladen zelfstandig te laten werken door bijvoorbeeld een gegevensinvoer dialoogformulier te laten openen wanneer op een rij wordt geklikt, of wanneer op een nieuw geprogrammeerde werkbalk knop wordt geklikt. Kortom, u hoeft Visual Basic of een ander Basic dialect niet te gebruiken om een werkblad te maken die aan uw eisen moet voldoen.

In de eerste les gaf ik u een introductie over Excel met uitleg waar u de macro's kunt vinden, hoe u een macro opneemt en hoe u die kunt bewerken.
Er is één hoofdonderdeel in VBA: het ThisWorkbook object. Met dit object kunt u de werkbladen beheren door acties in de gebeurtenissen uit te voeren. Het object is echter optioneel, want de werkbladen zijn ook objecten en hebben zelf ook gebeurtenissen om ze te beheren. Het enige verschil is dat het ThisWorkbook object een subroutine heeft die we in het werkblad object niet kunnen vinden: `Workbook_Open()`

Tekst en getallen in de cellen.

De cellen van een werkblad reageren op de invoer van de gebruiker. Dat merken we als we in de cellen wat invoeren.

- Start Excel en gebruik **Blad1**.
- Voer in cel **A1** het getal 4 in. Wat gebeurt er? Excel plaatst de waarde rechts in de cel.
- Voer in cel **A2** de waarde '4 (komma 4, toets met de aanhalingstekens) in. Wat gebeurt er? Excel plaatst de waarde links in de cel. De waarde wordt gezien als tekst.
- Voer in cel **A3** de waarde =f in. Wat gebeurt er? Excel geeft een foutmelding: #NAAM? Kennelijk mag een onbekende waarde niet beginnen met het = teken. Wilt u die waarde toch in de cel weergeven, toets dan eerst weer een komma, dus: '=f
- Voer in cel **A5** de waarde =A1+4 in. Wat gebeurt er? Excel laat een resultaat zien van een berekening en wel van de waarde in cel A1 plus 4. U kunt dus rekenen met de namen van de cellen. In cel A5 zal dus de waarde 8 staan.
- Voer in cel **A6** de waarde =A2+4 in. Wat gebeurt er? Excel rekent de som op dezelfde manier uit als met de cel A1, ook al hebben we in cel A2 het getal 4 als tekst opgegeven. Kennelijk mogen we van Excel alfanumerieke getallen in de sommen gebruiken.

De ingevoerde waarden bekijken in VBA.

- Open de Visual Basic Editor (menu Extra, keuze Macro's).
- Zorg ervoor dat het projectvenster geopend is.
- Dubbelklik op het object **Blad1** die op het projectvenster staat aangegeven.
- U ziet het codevenster van blad 1 verschijnen.

- Zorg ervoor dat het venster **Direct** geopend is. Indien het venster niet is geopend, klik dan op het menu Beeld en klik op de menukeuze Venster Direct.
- Voer de volgende code in het venster in: `Print Blad1.Cells("A1")`
  VBA geeft een foutmelding. De foutmelding betekent dat u geen cellen kunt opgeven.
- Voer onder de vorige regel in: `Print Blad1.Cells(1,1)`
  VBA geeft de waarde 4 als resultaat.
  De twee opgegeven parameters hebben de volgorde (rij,kolom). Voer nogmaals dezelfde code in, maar nu met de parameters (2,1). VBA zal nu als resultaat de alfanumerieke waarde 4 geven.
- Voer nu eens de code in met de celparameters (4,1). Nu geeft VBA de foutmelding: `Fout 2029` vanwege de onjuiste waarde.
- Voer nu de volgende code in: `Blad1.Cells(2,2)=10`
  Er lijkt niets te gebeuren, maar ga eens terug naar het werkblad en zie eens wat er in cel B2 staat: het getal 10.

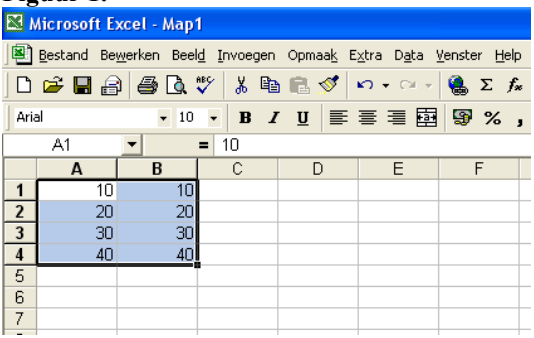
Door hiermee veel te oefenen, zult u snel in de gaten hebben dat er drie nadelen zijn:

- `Cells()` is geen object maar een eigenschap. U kunt alleen waarden lezen en toekennen.
- De eigenschap werkt omgekeerd. Een cel als A1 leest men als kolom en rij, maar de parameters moeten als rij en kolom worden opgegeven.
- En zoals eerder is vernomen kunnen we als parameter niet de naam van de cel opgeven. Dat is natuurlijk niet fijn en moeten we telkens het rijgetal en het kolomgetal opgeven.

Selecties en sommen.

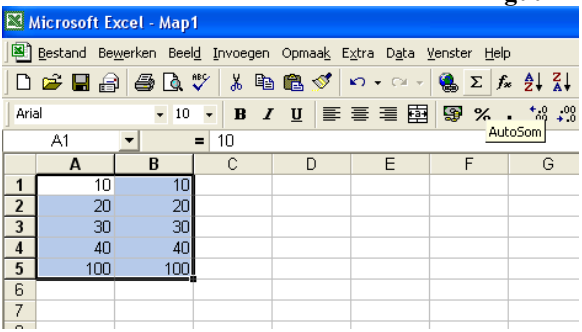
Toch is er een oplossing voor het derde probleem. Elk blad object heeft een eigen gebied object waarmee we meerdere cellen kunnen bewerken. Dat is handig om bijvoorbeeld een heel gebied als een som uit te laten rekenen. Laten we eerst eens zien hoe dat gaat op het werkblad zelf.
Voer getallen in van 10 t/m 40 in de kolommen A en B en selecteer ze, zie figuur 1.

Figuur 1.



Wanneer u klaar bent zoals Figuur 1 laat zien, klikt u met de linker muisknop op de knop **Autosom**, die u aan de rechterkant ziet staan. Figuur 2 zal een helptekst tonen zodat u de knop snel kunt vinden.

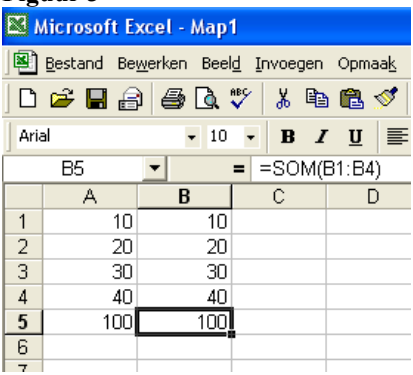
Figuur 2



Hier rechts op de afbeelding van Figuur 2 kunt u de helptekst vinden. Op de eerste rij met werkbalken kunt u een knop zien als een actief vakje. Als u op de knop klikt, zal op de rij onderaan de geselecteerde cellen nieuwe getallen verschijnen en wel de uitgerekende sommen van de twee kolommen. Gelijk zal Excel deze nieuwe rij ook selecteren.
Als u nu nogmaals probeert op de Autosom knop te klikken, zult u ontdekken dat het maar één keer werkt. Dat komt doordat de berekende som in de selectie staat.

Klik nu eens op cel B5 en zie eens wat er op de invoerbalk staat, zoals Figuur 3 ook laat zien.

Figuur 3



Ook al heb ik het hele 'twee kolommen' gebied geselecteerd, de knop Autosom heeft de twee kolommen toch apart berekent. Als we dat niet willen dan kunnen we in de som ook typen: `A1:B4`. Zoals Figuur 3 laat zien, kunt u al raden wat dan het resultaat is van de som: juist, de waarde 200.
Maar wie niet sterk is kan ook slim zijn door het nog makkelijker te maken: `=SOM(A5:B5)` en dat bijvoorbeeld in cel C5 in te toetsen. Ook dan zal het antwoord 200 zijn. Door andere sommen in sommen te berekenen is het niet nodig om telkens weer een heel gebied op te geven.

Geselecteerde gebieden in VBA beheren: het Range object.

Elk werkblad object heeft een Range object waarmee we gebieden op kunnen geven en kunnen beheren. Het voordeel is dat we niet zelf op het werkblad zo'n gebied hoeven te selecteren. De vraag is alleen: hoe kunnen we een gebied in VBA gebruiken?
Het Range object heeft de volgende syntaxis:
[variabele =] *[object.]*`Range(`*linkerbovenhoek*`,`*rechteronderhoek*`)`*.eigenschap/functie*

We kunnen bijvoorbeeld uit het gebied `A1:B4` de waarde uit de cel A1 halen, door onderstaande code in venster Direct in te toetsen:

```
Print Blad1.Range(Blad1.Cells(1,1),Blad1.Cells(4,2)).Cells(1,1)
```

Maar omdat het blad object optioneel is, mag de code ook worden verkort:

```
Print Range(Cells(1,1),Cells(4,2)).Cells(2,1)
```

Beide coderegels zullen de resultaten 10 en 20 geven. Dankzij het Range object kunnen we elke waarde uit het opgegeven gebied halen. De volgende code zal ook de waarde 20 geven, maar niet uit de cel A1! Bovendien kunt u zien dat het Range object wel celnamen accepteert.

```
Print Range("A2","B3").Cells(1,1)
```

Zoals u ziet werkt dit veel makkelijker dan telkens de celparameters op te moeten geven. Hoe u met de code wilt werken is helemaal uw eigen keuze. Houd er ook rekening mee dat de linkerbovenhoek van een gebied altijd cel 1,1 is en dus niet altijd cel A1 kan zijn.

Tip! U kunt de code nog meer verkorten. Probeer maar eens: `Range("A2:B3")` en u zult zien dat die manier ook werkt.

Het ThisWorkbook object.

In nieuwsbrief nr 1 heb ik verteld wat het ThisWorkbook object inhoudt en dat er een subroutine aanwezig is waarmee we het werkboek kunnen initialiseren. ThisWorkbook heeft een heleboel werkboek events die reageren op acties zoals het activeren van een werkblad en het creëren van een nieuw werkblad. Deze events zijn zeer krachtig en zeer functioneel.

De gebruiker hoeft niet aan het VBA project te komen, tenzij u de deur open laat staan. Wanneer u van plan bent VBA te gebruiken om acties uit te laten voeren die voor de gebruiker makkelijk te bedienen zijn, moet u één ding goed onthouden: u kunt ook de gebruiker zijn! Hier bedoel ik mee dat u het beste de deur open kunt laten staan als u zeker weet dat het VBA project in Excel nog niet klaar is. Zorg er voor dat het wiel niet meer uitgevonden hoeft te worden en dat is ook het voordeel van het ThisWorkbook object waarmee u een heel document project kunt bewerken. In het object hebt u toegang tot alle werkbladen van het document. Onthoud wel dat u niet meerdere mappen kunt gebruiken in één ThisWorkbook object, zie uitleg nieuwsbrief nr 1, want elke map dat een document is heeft een eigen ThisWorkbook object.

In de volgende nieuwsbrief zal ik codevoorbeelden laten zien die gegevens uit een werkblad halen vanuit het ThisWorkbook object, en gaan we eens kijken hoe we kunnen werken met het Range object in een lusstructuur. Ook zal ik u laten zien dat VBA een Range type heeft, die door het Range object teruggegeven wordt.

Marco Kurvers

Ik kan programmeren.

Iedereen die wil programmeren moet een programmeertaal kennen. Zonder een programmeertaal is een computer programmeren niet mogelijk. Gelukkig zijn er programma's waarmee we kunnen programmeren zonder gebruik te maken van een programmeertaal. Maar wat we ook maken, bouwen en ontwerpen, er zal altijd geprogrammeerd moeten worden.

Verschillende generaties.

Het is de keus van de gebruiker welke programmeertaal nodig is. BASIC, Pascal en C++ worden hogere programmeertalen genoemd en vallen onder de 4de generatietalen. Er bestaan echter ook 5de generatietalen die besturingstalen worden genoemd. Die programmeertalen worden gebruikt in supercomputersystemen en robots. Wanneer een robot een actie ontvangt, zal die nog niet direct uitgevoerd worden. Binnen in het systeem werkt het programma dat eerst met een hogere programmeertaal is geschreven. De code die later uitgevoerd wordt, de 5de generatiecode, kunnen we zien als parameters in instructiestructuren. Ter vergelijking leeft ons lichaam in een 4de generatie besturing, maar we krijgen telkens nieuwe instructies binnen als 5de generatiecode. Wij denken: oh, we moeten linksaf en automatisch doet ons lichaam de rest.

Om de computer net zo te laten 'denken', moeten we goed om kunnen gaan met de programmacode. Hebben we BASIC geleerd? Hoera, we kunnen aan de slag! Helaas, dit zal echter tegenvallen, want ook al kennen we de BASIC taal; de praktijk ligt toch heel anders dan alleen maar de programmeertaal te hebben geleerd.

Macrotalen en scripttalen.

VBA in de Office programma's en GML in Game Maker lijken op 5de generatietalen, maar dat zijn ze echter niet. VBA is een applicatieprogrammeertaal die met macro's werkt. Een macro is een subroutine of codeblok met een naam die alleen bepaalde opgenomen code uitvoert. Macro's krijgen geen instructies van de gebruikers en voeren ook geen acties uit. Hoewel GML sterk op VBA lijkt, werkt die echter op een andere manier dan macro's. Ook een script heeft een codeblok met een naam, maar een script kan wel instructies binnen krijgen. Deze moeten in de script worden gelezen als argumenten. Meer over GML kunt u zien bij Game Maker voor beginners, en ook in de toekomstige nieuwsbrieven.

En als ik zeker weet dat ik het kan?

Als u echt zeker weet dat u genoeg ervaring hebt, goed om kunt gaan met de structuren en weet hoe een programma opgebouwd moet worden, begin dan eerst met de grove structuur en maak een schets van uw project. Als u daarmee bezig bent, is het gebruik van een programmeertaal nog niet nodig. Welke formulieren hebt u nodig, welke tabellen moet u gebruiken voor een eventuele database en hebt u klassen nodig om elke record uit een databasetabel te kunnen halen? Allemaal vragen die u dus niet tijdens het leren van een program-

meertaal tegenkomt, maar wel zodra u aan een project wilt beginnen. U zult wel leren hoe u structuren programmeert, maar uw docent kan natuurlijk niet weten welke structuren u later gaat gebruiken als u projecten maakt. Besef goed dat wat u leert op een standaardmanier zal gaan. De echte ervaring zult u helemaal zelf op moeten bouwen.

Marco Kurvers

Intermezzo: een Werknemers project (2).

In de vorige nieuwsbrief heb ik u uitgelegd wat allemaal nodig is in een Werknemers project. In deel 1 kon u leren welke gestructureerde recordtypes gebruikt moeten worden om de database te beheren. Daar is QuickBASIC voor gekozen, omdat die BASIC versie de meeste databasemanagement taken moeiteloos uit kan voeren.
U hebt ook kunnen zien hoe de lees- en schrijfmogelijkheden werken met `GET#` en `PUT#` en hoe u dit binnen in een geopend bestand bewerkt met `OPEN` en `CLOSE`.
Nu we weten hoe de recordstructuur werkt en hoe we gebruik kunnen maken van de gegevensbestanden, gaan we verder met het Werknemer programma en laat ik u zien hoe u een menubesturing maakt.

Het Werknmer programma, zoals die in de DOS tijd met maximaal 8 letters zou moeten heten.

Wijzelf noemen het programma nu gewoon het Werknemer programma. Gelukkig hoeven we ons niet meer te bekommeren om hoeveel letters een bestandsnaam mag hebben.

Figuur 1 demonstreert het terugkerende menu van *Werknemer*. De eerste vier menuopties hebben betrekking op de afzonderlijke records in de werknemerdatabase en één optie geldt voor de gehele database:

- **Display record**
  Deze optie geeft een record op het scherm weer.
- **Toevoegen record**
  Deze optie verzoekt de gebruiker om de veldgegevens voor een nieuw werknemerrecord in te voeren. Wanneer deze invoerdialoog voltooid is, schrijft het programma het record naar de file.
- **Wijzigen record**
  Deze optie maakt het mogelijk om de informatie in bepaalde velden van een werknemerrecord te wijzigen. Deze optie is vooral bestemd om nieuwe gegevens over de afdeling, de functie en het salaris van de werknemer in te voeren.
- **Registratie van vertrek**
  Deze optie wijzigt het record van een werknemer die het bedrijf verlaat. Dit vertrek wordt op een speciale manier geregistreerd, waarbij het record niet fysiek uit de file wordt verwijderd. Dit onderwerp komt later nog ter sprake.
- **Printen van records**

Deze optie drukt een alfabetisch geordend overzicht van de werknemerrecords af. Al naar gelang de instructies betrekt het programma hierbij ook records van werknemers die de firma hebben verlaten of slaat deze juist over.

Figuur 1.
Het hoofdmenu van het programma.

Aangezien het menu na elke bewerking weer op het scherm verschijnt, kunt u tijdens een sessie een onbeperkt aantal werknemerrecords bewerken. De laatste optie in het menu is *Einde*. Deze optie beëindigt het programma. Het programma voert alle benodigde fileoperaties uit voordat de uitvoering wordt beëindigd.

De Werknemer datafiles.

Werknemer creëert en onderhoudt twee verschillende files op schijf. De database wordt opgeslagen in random-access file *WERKNMER.DAT*. Wanneer u een nieuw record aan de database toevoegt of de inhoud van een bestaand record wijzigt, schrijft *Werknemer* de informatie naar deze databasefile. De records worden per definitie in chronologische volgorde in deze file opgeslagen; dat wil zeggen dat elk nieuw record aan het einde van de file wordt geplaatst.
Bovendien creëert *Werknemer* een indexfile die in de sequentiële file WERKNMER.NDX wordt opgeslagen. Tijdens de programma-uitvoering zult u weinig van de aanwezigheid van deze file merken, maar het programma is niet in staat om de records te lokaliseren zonder het bestaan ervan. De file bevat twee informatie-items per werknemer:

1. De naam van de werknemer.
2. De locatie van het werknemer recordnummer in de database.

Aan het begin van de uitvoering leest het programma de index in het computergeheugen en tijdens de uitvoering van *Werknemer* wordt de index voortdurend in alfabetische volgorde gesorteerd. Telkens wanneer u een nieuw record aan de database toevoegt, voegt het programma de naam van de nieuwe werknemer en het recordnummer aan de index toe en sorteert de index opnieuw. Wanneer de uitvoering wordt beëindigd, schrijft het programma de index naar diskette of harde schijf. De index wordt bij een volgende programma-uitvoering automatisch in het computergeheugen geladen.
Wanneer u een optie selecteert die betrekking heeft op een record in de database, vraagt het programma allereerst om de naam van de werknemer. Vervolgens spoort het programma deze naam in de alfabetische index op. Vooropgesteld dat de naam zich in de index bevindt, leest het programma het recordnummer van de werknemer uit de index en leest dit record vervolgens uit de databasefile. Dit gebeurt zó snel dat u niets van de zoekactie merkt. Kortom, een werknemernaam vormt de sleutel tot de locatie van een record in de database.

Daarom verzoekt het programma allereerst om een naam indien één van de eerste vier menuopties is geselecteerd.

Display record

Wanneer u de *Display record* optie heeft geselecteerd, vindt de volgende dialoog plaats:

```
Voer de naam van de werknemer in:
---- -- ---- --- -- --------- --


Achternaam --> Drost
Voornaam   --> Louise

            Naam      :  Louise Drost
            Zkfnummer :  12345
            Begindatum:  11-22-1988
            Afdeling  :  Administratie
            Functie   :  Secretaresse
            Salaris   :  ***2,000 per maand
<aantal lege regels>
                                    Druk op spatiebalk voor doorgaan.
```

Figuur 2.
Hypothetische inhoud van een werknemerrecord.

Tengevolge van deze invoer zoekt het programma Louise Drost en het werknemer recordnummer in de index van de database op, leest het desbetreffende record uit de database en geeft dit op het scherm weer. Figuur 2 geeft een indruk van de inhoud van het record.

Indien u een naam invoert, die niet in de index voorkomt, genereert het programma een desbetreffende foutmelding:

```
Voer de naam van de werknemer in:
---- -- ---- --- -- --------- --


Achternaam --> Drost
Voornaam   --> Karel

Deze naam komt niet in de werknemerfile voor.
```

Na deze foutmelding verschijnt het menu weer op het scherm en kunt u de *Display record* optie opnieuw selecteren om een andere naam in te voeren.

Toevoegen record

Het selecteren van de *Toevoegen record* optie houdt in dat het programma alle benodigde veldgegevens vraagt voor een nieuwe werknemerrecord. De gang van zaken is als volgt:

```
Voer de gegevens voor de nieuwe werknemer in:
---- -- -------- ---- -- ------ --------- --

Achternaam --> Pronk
Voornaam   --> Jeroen
Zkfnummer  --> 003-16-4420
Afdeling   --> Verkoop
Functie    --> Verkoopleider
Salaris    --> 5000
U)ur,
M)aand of
J)aarbasis --> M
```

Vreemd genoeg vraagt het programma om zeven informatie-items, terwijl de recordstructuur uit negen velden bestaat. De twee overige velden zijn echter datumvelden: de datum van indiensttreding en de datum van vertrek. Het programma slaat de actuele systeemdatum als de datum van indiensttreding op als u geen datum opgeeft. De datum van vertrek blijft uiteraard open totdat een werknemer de firma verlaat.
Let ook op de manier waarop het programma om het salaris van de werknemer verzoekt. Eerst voert u het salarisbedrag in, waarna het programma om de periode verzoekt waarover dit salaris van toepassing is. Dit verzoek beantwoordt u door de U-, M- of J-toets in te drukken voor respectievelijk *uur*-, *maand*- of *jaarbasis*. Wanneer het programma later een record op het scherm weergeeft, wordt dit veld gecombineerd met het salaris:

```
Salaris:       ***31,00 per uur
Salaris:       ***5000 per maand
Salaris:       ***60000 per jaar
```

Nadat alle veldgegevens voor de *Toevoegen record* optie zijn ingevoerd, verschijnt de volgende vraag op het scherm:

```
Record opslaan? (J of N) -->
```

Deze faciliteit geeft u de gelegenheid om de veldgegevens te controleren op eventuele tikfouten. Vervolgens drukt u J in om te bevestigen dat het record moet worden opgeslagen of, indien u een fout heeft gemaakt, N om het record te annuleren. In dat geval schrijft het programma het record niet naar de database.
De *Toevoegen record* optie staat niet toe dat een nieuw record onder een werknemernaam wordt ingevoerd die reeds in de file voorkomt. Het indexsysteem van het programma is gebaseerd op een unieke naam per werknemer. Om deze reden doorzoekt het programma de index zodra u de werknemernaam aan de *Toevoegen record* optie opgeeft. Wanneer de naam reeds in de index voorkomt, genereert het programma de volgende mededeling:

```
Deze naam komt reeds in de file voor.
```

Indien twee werknemers toevallig dezelfde voor- en achternaam hebben, dient de gebruiker een onderscheid aan te brengen door bijvoorbeeld één van de voornamen af te korten of een extra voorletter op te nemen.

Tip! Tegenwoordig is zo'n onderscheid niet meer nodig. U kunt namelijk ook een unieke sleutelnaam definiëren, zodat u alsnog dezelfde werknemernamen kunt gebruiken. Houd wel de sleutelnaam apart van de index. In deze intermezzo wordt geen unieke sleutelnaam gebruikt, maar alleen de index.

Wijzigen record

De *Wijzigen record* optie vraagt eerst om de naam van de werknemer, wiens record moet worden gewijzigd. Wanneer het desbetreffende record is gevonden, geeft het programma een submenu op het scherm weer, zie Figuur 3. Dit menu heeft twee doeleinden: het geeft actuele informatie uit de werknemerfile weer en geeft u de gelegenheid het veld of de velden te selecteren die u wilt veranderen. U kunt nieuwe gegevens over de afdeling, functie en/of salaris van de werknemer invoeren. In alle gevallen voert het programma een korte dialoog met u om de nieuwe veldgegevens in te voeren.

```
Wijzigen van een werknemer-record.
Naam:  Louise Drost
                Menu
 ____________________________________

A)fdeling ...> Administratie
F)unctie ....> Secretaresse
S)alaris ....> €2000 per maand
R)ecord OK.
 ____________________________________

  A  F  S  R  ->
```

Figuur 3.
Het Wijzigen-record menu.

Indien u bijvoorbeeld de Salaris optie kiest, ontstaat de volgende dialoog:

```
Nieuw salaris --> 35.00
U)ur-, M)aand-, J)aarbasis --> U
```

Na deze dialoog verschijnt het *Wijzigen record* menu weer op het scherm en geeft de informatie weer die u zojuist heeft ingevoerd. Wanneer alle gewenste wijzigingen in het record zijn aangebracht, worden de veranderingen naar de database geschreven door middel van de laatste optie van dit submenu:

```
R)ecord OK.
```

Hierna schrijft het *Werknemer* programma het record inclusief de veranderingen naar de database.

Vertrek optie.
De *Vertrek* optie wijzigt het record van een vertrokken werknemer op twee manieren:

1. Het woord "VERTROKKEN" wordt in het veld 'afdeling' opgeslagen.
2. De actuele systeemdatum wordt in het veld 'vertrekdatum' opgeslagen.

Zoals gebruikelijk vraagt de procedure eerst om de naam van de werknemer. Vervolgens geeft het programma het volledige record op het scherm weer, zodat u kunt controleren of u het juiste record gaat herzien.
De schermweergave voor een vertrokken werknemer ziet er als volgt uit:

```
Naam      : Arthur Huisman
Zkfnummer : 901-23-4567
Begindatum: 04-09-1984
Afdeling  : Inkoop
Functie   : Manager
Salaris   : ***4,000 per maand

Is dit de vertrokken werknemer (J of N) --> J

*** Het vertrek is geregistreerd ***
```

Het programma verzoekt u om te bevestigen of dit het werknemerrecord is, waarop de vertrekregistratie van toepassing is. Indien u bevestigend antwoordt, wijzigt het programma het record, schrijft de veranderingen weg en informeert u over deze activiteiten. Antwoordt u echter ontkennend door N in te drukken, dan onderneemt het programma geen actie met betrekking tot het record en verschijnt de volgende mededeling op het scherm:

```
Geen verandering in status van werknemer.
```

De recordweergave van een vertrokken werknemer wijkt af van de gebruikelijke weergave. Bijvoorbeeld, de *Display record* optie genereert de volgende weergave voor een vertrokken werknemer:

```
Naam      : Arthur Huisman
Zkfnummer : 901-23-4567
Begindatum: 04-09-1984
Afdeling  : Inkoop
Functie   : Manager
Salaris   : ***4,000 per maand

*** Niet meer in dienst ***

Vertrekdatum: 12-05-1988
```

Zoals u ziet wordt de vertrekdatum onderaan de recordweergave vermeld.

Printen van records

De laatste optie *Printen van records* stuurt een alfabetisch geordend overzicht van werknemerrecords naar de printer. Eerst stelt het programma u echter de volgende vraag:

```
Worden vertrokken werknemers
in de printout opgenomen (J of N) -->
```

Afhankelijk van uw antwoord drukt het programma de volledige database af of alleen de records van de werknemers die nog bij de firma in dienst zijn.
Voordat het overzicht wordt afgedrukt geeft het programma u de gelegenheid de printer operationeel te maken en te controleren of het papier correct is ingevoerd:

```
Druk de spatiebalk in
wanneer de printer operationeel is.

   Achternaam     Voornaam              ZkfNr.  Functie         Salaris
   ----------     --------              ------  -------         -------

** Arends         Kees             345-543-654  Monteur         ***75.00 /uur
   Barend         Fred             435-654-675  Vertegenw.      ***4,100 /ma
   Bos            Johan            456-432-567  Manager         ***72,000 /ja
 ... enz. ...     ... enz. ...   ... enz. ...    ... enz. ...    ... enz. ...

** Vertrokken werknemer.
```

Figuur 4.
Printout van een werknemersoverzicht. Het programma zelf zal meer rijen per overzicht naar de printer sturen dan Figuur 4 weergeeft, vandaar ook de rij '... enz. ...'.

Zodra de spatiebalk wordt ingedrukt, begint het printen. Figuur 4 toont een printout van een werknemersoverzicht dat zowel huidige werknemers als vertrokken werknemers omvat. Zoals u ziet, plaatst het programma een dubbele asterisk aan de linkerkant van een record van een vertrokken werknemer.

U zult zo langzamerhand wel benieuwd zijn naar de listing van het *Werknemer* programma en naar de technieken om de database te beheren. Hier ga ik dan ook meteen iets aan doen. Hoewel het programma een eenvoudig voorbeeld van een database applicatie is, lenen de toegepaste technieken zich ook goed voor programma's, die meer geavanceerde databasebewerkingen uitvoeren en meer complexe recordstructuren ondersteunen.

Wat gebeurt er in het Werknemer programma?

In de programmalisting zijn de diverse definities en declaraties aan het begin geplaatst. Daarna volgen het hoofdprogramma en de functies en subprogramma's in alfabetische volgorde. Tijdens de toelichting op de diverse programmataken wordt veelvuldig naar functies en subprogramma's verwezen die aan de linkerkant van de listing worden aangeduid. De toelichting richt zich in het bijzonder op twee aspecten, die bij het programmeren van databases essentieel zijn:

1. De toepassing van de QuickBASIC random-access filestatements.
2. Het ontwikkelen van een techniek om de database te indexeren.

Definities en declaraties

Het eerste gedeelte van de programmalisting definieert diverse typen, waarden en variabelen. Een overzicht van de definities en declaraties is dus geen overbodige zaak:

- De **CONST** statements definiëren drie symbolische constanten: de logische waarden **true%** en **false%** en de integer **tabPos%** die de positie van de linkerkantlijn van de schermweergaven vertegenwoordigt.
- Twee **TYPE** statements zetten de gebruikergedefinieerde typen voor het programma op. Zoals we reeds hebben gezien, is het eerste gebruikergedefinieerde type de recordstructuur voor de werknemer databasefile **werknType**. Het tweede type is **indexType**, een structuur voor de indexfile. Deze structuur bevat twee elementen die de werknemer recordnummers en –namen vertegenwoordigen, respectievelijk:

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

- De **DECLARE** statements definiëren de namen en parameters van de programma procedures.
- Een **DIM** statement definieert twee structuren: de recordvariabele **werknRec** ten behoeve van het hoofdprogramma en stringarray **werknMenu$()** om de menuopties op te slaan.
- Het **COMMON SHARED** statement declareert twee globale variabelen: **totWerkns%** vertegenwoordigt de totaal aantal records dat zich op een bepaald moment in de file bevindt en **index()** is een array met **indexType** records die de structuur van de indexfile bepalen.
- Dit alles wordt gevolgd door een aantal **DATA** statements, die de feitelijke tekst van de menuopties bevatten.

Het hoofdprogramma

Het hoofdprogramma begint met een `FOR...NEXT` lus, die de menuopties aan de `werknMenu$()` array toevoert. Vervolgens roept het programma het `OpenFile` subprogramma aan dat zowel de werknemerdatabase als de indexfile opent.

Openen van de werknemer- en de indexfile.

De `OpenFile` routine creëert ten eerste recordvariabele `openRecord` door middel van een `DIM` statement. Deze lokale variabele vertegenwoordigt de werknemer recordstructuur tijdens het openen van de file. Elke belangrijke routine in het *Werknemer* programma creëert een eigen lokale recordvariabele om de werknemerrecords te adresseren.
Een `OPEN` statement opent aansluitend WERKNMER.DAT als een random-access file met filenummer 1 en een bepaalde recordlengte:

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

Zoals reeds opgemerkt, bepaalt de interne `LOF` functie de lengte van de geopende file. De aantal records die op dat moment in de file is opgeslagen, wordt berekend door de lengte

van de file in kwestie door de recordlengte te delen:

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

Dit is de beginwaarde die aan globale variabele `totWerkns%` wordt toegekend. Aangenomen dat de database niet leeg is, roept `OpenFile` vervolgens het `OpenIndex` subprogramma aan om de WERKNMER.NDX file (indexfile) te openen:

```
IF totWerkns% <> 0 THEN OpenIndex
```

De eerste taak van de `OpenIndex` routine bestaat uit het definiëren van de voorlopige dimensie van de `index()` array. De array krijgt één element per werknemerrecord in de file toegewezen:

```
REDIM index(totWerkns%) AS indexType
```

Hierna opent de routine de indexfile. Het volgende `OPEN` statement geeft aan dat WERKNMER.NDX als een sequentiële file moet worden gelezen:

```
OPEN "WERKNMER.NDX" FOR INPUT AS #2
```

U ziet dat de indexfile nummer 2 krijgt toegewezen, omdat er nu twee geopende files zijn. Een `INPUT#` statement leest vanuit een `FOR...NEXT` lus elk data-item van de indexfile in de recordelementen van de `index()` array:

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

Aangezien de index tot het einde van de programma-uitvoering in het geheugen aanwezig blijft, kan de file op schijf worden gesloten:

```
CLOSE #2
```

Eenmaal terug in het hoofdprogramma is alles gereed voor de feitelijke actie. Een goede gelegenheid om de centrale programmabesturing te onderzoeken.

Programmabesturing.
Het programma roept vanuit een `DO...UNTIL` lus herhaaldelijk de `Menu%` functie aan om het hoofdmenu weer te geven en de menukeuze van de gebruiker te verwerken:

```
DO
   LOCATE 2, 32: PRINT "Werknemerrecords"
   SELECT CASE Menu(werknMenu$())
```

Een `SELECT CASE` structuur bepaalt na elke menukeuze van de gebruiker welk subpro-

gramma moet worden aangeroepen: `DisplayRec` om een individueel werknemerrecord weer te geven; `Toevoegen` om een nieuw record aan de file toe te voegen; `Wijzigen` om de inhoud van een record te wijzigen; `Vertrek` om het vertrek van een werknemer te registreren en `PrintRecords` om een overzicht van de records af te drukken. In alle gevallen, uitgezonderd `Toevoegen`, voorkomt een genest `IF` statement een aanroep naar een subprogramma indien `totWerkns%` nul is, dat wil zeggen dat de database nog leeg is. Het volgende voorbeeld betreft een aanroep naar de `PrintRecords` routine:

```
IF totWerkns% <> 0 THEN PrintRecords
```

In feite is dus alleen de *Toevoegen* optie beschikbaar totdat de database minstens één record bevat.
Wanneer de gebruiker de *Einde* optie selecteert om de uitvoering te beëindigen, roept de `CASE ELSE` constructie de `CloseFiles` routine aan om de databasefiles te kunnen sluiten. Tenslotte beëindigt het programma de iteraties van de `DO...UNTIL` lus door de waarde van logische variabele `finito%` op `true%` te zetten:

```
CASE ELSE
   CloseFiles
   finito% = true%

END SELECT

LOOP UNTIL finito%
```

De toelichting op voornoemde subprogramma's begint logischerwijs met `Toevoegen`.

Het Toevoegen subprogramma

Terwijl we ons een weg banen door de activiteiten van het `Toevoegen` subprogramma en van de diverse andere routines die het aanroept, zullen we geleidelijk een beeld krijgen van het indexsysteem van het *Werknemer* programma. Hierbij gaat het vooral om de manier waarop het programma de database-index opbouwt en hoe de individuele gegevens in de index worden geformatteerd. Tevens komt de manier waarop het programma de index door middel van de sorteer- en zoekroutines beheert aan bod.
De routine zet eerst de lokale recordvariabele `toevoegenRecord` op, waaraan de velden van het nieuwe werknemerrecord worden toegewezen. Vervolgens geeft de procedure een titel op het scherm weer en roept de `LeesNaam` procedure aan. Alle hoofdroutines van *Werknemer* roepen `LeesNaam` aan. Deze routine verzoekt de gebruiker om gegevens in te voeren voor de naamvelden van het record die aan de database moet worden toegevoegd, of in een ander geval moet worden weergegeven of moet worden gewijzigd. `LeesNaam` controleert daarna in de database-index of de naam in de file voorkomt.

De LeesNaam procedure.
Een aanroep naar `LeesNaam` voert drie argumenten die altijd als referentie worden overgedragen. `LeesNaam` retourneert de informatie aan het aanroepende subprogramma via de

drie variabelen werknRecord%, recNr% en doel$:

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

De recordvariabele werknRecord bevat het werknemerrecord in kwestie. De integer variabele recNr% geeft aan of de naam al dan niet in de file voorkomt:

- Een waarde 0 betekent dat de naam niet in de file aanwezig is.
- Een integer waarde groter dan 0 vertegenwoordigt het werkelijke recordnummer van de werknemernaam.

De stringwaarde doel$ bevat de werknemernaam, die ten behoeve van de indexfile op een speciale wijze wordt geformatteerd. Hiertoe verzoekt LeesNaam eerst om de achternaam en voornaam van de werknemer en slaat de ingevoerde waarden in de variabelen tijdelijkeAchternm$ respectievelijk tijdelijkeVoornm$ op:

```
PRINT TAB(tabPos%); : INPUT "Achternaam -->", tijdelijkeAchternm$
PRINT TAB(tabPos%); : INPUT "Voornaam -->", tijdelijkeVoornm$
```

De uiteindelijke indexinvoer is een samenvoeging van die twee stringwaarden waaruit alle spaties zijn verwijderd en alle letters in hoofdletters zijn omgezet. LeesNaam realiseert dit gestandaardiseerde indexformaat door middel van de programmafunctie WisSpatie$ en de ingebouwde QuickBASIC functie UCASE$:

```
doel$ = UCASE$(WisSpatie$(tijdelijkeAchternm$ + tijdelijkeVoornm$))
```

De WisSpatie$ functie verwijdert alle spaties uit de toegewezen string. We kijken even naar het stringformaat dat uit dit proces voortkomt. Stel dat de gebruiker twee naamvelden heeft ingevoerd in antwoord op de inputprompts:

```
Achternaam --> La Monte
Voornaam   --> Joost
```

Door deze twee strings samen te voegen, de spatie te verwijderen en de letters in hoofdletters om te zetten, ontstaat de volgende string: LAMONTEJOOST
Dit is dus de gestandaardiseerde string voor werknemer Joost La Monte, waarnaar het LeesNaam subprogramma in de database-index zal zoeken. De volgende aanroep naar de Zoek% functie met deze string in variabele doel$ activeert de zoekactie:

```
recNr% = Zoek%(doel$)
```

De Zoek% functie zoekt in de index() array naar de string in doel$. Wanneer de string is gevonden, retourneert Zoek% het overeenkomstige recordnummer aan de hand van het indexelement werknNummer. Wordt de string niet gevonden, dan retourneert de functie een waarde van nul, zoals uit de volgende paragraaf zal blijken.

De Zoek% functie.
`Zoek%` krijgt de te zoeken string (doelstring) toegewezen in de vorm van variabele `welkeTekst$`:

```
FUNCTION Zoek%(welkeTekst$) STATIC
```

Vergeet niet dat het programma de `index()` array altijd volgens de werknemernamen sorteert. Dit is de sleutel tot het welslagen van de zoekfunctie.
`Zoek%` voert een zogenoemde binaire zoekactie uit. De functie initialiseert eerst de variabelen `begin%` en `einde%`, die de `index()` array markeren, en de logische variabele `gevonden%`, die een waarde van true aanneemt wanneer het doelrecord is gevonden:

```
begin% = 1
einde% = totWerkns%
gevonden% = false%
```

De zoekactie begint met een vergelijking tussen de doelstring en de naamstring die in het midden van de index is gelokaliseerd. Deze vergelijking bepaalt of de doelnaam uiteindelijk in de eerste helft of tweede helft van de lijst zal worden gevonden. Vervolgens vergelijkt `Zoek%` de doelnaam met de string in het midden van de bewuste helft. Deze operatie wordt vanuit een `DO...WHILE` lus op een vierde van de lijst uitgevoerd, vervolgens op een achtste, een zestiende van de lijst enzovoorts, totdat de naam is gevonden of totdat het programma bepaalt dat de naam niet in de index voorkomt.
Binnen de `DO...WHILE` lus worden de waarden van `begin%` en `einde%` constant aangepast, zodat de routine de zoekactie op steeds kleinere gedeelten van de lijst kan uitvoeren. De vergelijkingen gaan door totdat de doelstring is gevonden, en de variabele `gevonden%` een waarde van true aanneemt, of totdat de waarde van `einde%` kleiner is dan de waarde van `begin%`, hetgeen betekent dat de doelstring niet aanwezig is in de lijst:

```
DO WHILE begin% <= einde% AND NOT gevonden%
```

Wanneer de doelstring is gevonden, wijst de routine het overeenkomstige recordnummer aan variabele `leesNr%` toe:

```
indexNaam$ = RTRIM$(index(midden%).werknNaam)
IF welkeTekst$ = indexNaam$ THEN
    gevonden% = true%
    leesNr% = index(midden%).werknNummer
```

Deze waarde wordt geretourneerd als het resultaat van de zoekfunctie:

```
Zoek% = leesNr%
```

Zoals reeds opgemerkt, slaat de `LeesNaam` procedure deze retourwaarde in de variabele

recNr% op.

Lezen van velden.
Eenmaal terug in het Toevoegen subprogramma wordt de waarde van recNr% opgeslagen in variabele inFile%. Toevoegen gebruikt inFile% als een logische variabele om de volgende actie te bepalen:

```
IF inFile% THEN
    PRINT
    PRINT TAB(tabPos%) "Deze naam komt reeds in de file voor."
ELSE
{Invoeren van de overige gegevens voor de nieuwe werknemer}
```

Ter verduidelijking, indien inFile% een waarde ongelijk aan 0 heeft en de ingevoerde naam zich dus in de database-index bevindt, gaat het Toevoegen subprogramma niet verder, omdat er geen dubbele namen in de werknemerdatabase zijn toegestaan.
Indien inFile% echter een waarde van 0 heeft, verzoekt Toevoegen om de werknemergegevens voor de overige datavelden van het record in te voeren. De routine stuurt een aantal invoerprompts naar het scherm en slaat de ingevoerde waarden in de desbetreffende elementen van de variabele toevoegenRecord op. Het volgende voorbeeld presenteert de INPUT statements voor het ziekenfondsnummer, de afdeling en de functie van de werknemer:

```
PRINT TAB(tabPos%);
INPUT "ZkfNummer --> ", toevoegenRecord.zkfNummer
PRINT TAB(tabPos%);
INPUT "Afdeling  --> ", toevoegenRecord.afdeling
PRINT TAB(tabPos%);
INPUT "Functie   --> ", toevoegenRecord.functie
```

Het salaris en de salarisbasis worden door middel van twee speciale invoerfuncties verwerkt. De LeesSalaris! functie accepteert een geldige numerieke waarde voor het salaris en de LeesSalType$ functie accepteert één van de drie mogelijke karakters: U voor een uurloon; M voor een maandsalaris of J voor een jaarsalaris. De Toevoegen routine slaat deze twee waarden in de desbetreffende recordelementen op:

```
toevoegenRecord.salaris = LeesSalaris!
toevoegenRecord.salType = LeesSalType$
```

Het programma leest de actuele systeemdatum door middel van de interne DATE$ functie. Deze datum wordt toegewezen aan het beginDatum element:

```
toevoegenRecord.beginDatum = DATE$
```

Wanneer alle veldwaarden hun plaats in het record hebben ingenomen, is het Toevoegen subprogramma zo ver dat het record naar de databasefile kan worden geschreven. Eerst

roept het subprogramma echter de `JaNee%` functie aan om de operatie te laten bevestigen:

```
IF JaNee%("Record opslaan?") THEN
```

De functie verzoekt om een ja of nee antwoord op de vraagstring, die de functie als argument voert. Indien de gebruiker bevestigend antwoordt, verhoogt `Toevoegen` het aantal records in de database met 1:

```
totWerkns% = totWerkns% + 1
```

Hierna wordt het record in kwestie naar het einde van de databasefile geschreven:

```
PUT#1, totWerkns%, toevoegenRecord
```

Hieruit blijkt dat de huidige waarde van `totWerkns%` het recordnummer van het nieuwe record wordt.
Hiermee zijn we er nog niet, want het programma dient voor deze nieuwe record een ingang aan de index toe te voegen en vervolgens de index opnieuw te sorteren. Deze operaties worden uitgevoerd door het `NieuweIndex` subprogramma. `Toevoegen` verstuurt in de aanroep naar deze routine de geformatteerde stringwaarde die oorspronkelijk was gegenereerd door het `LeesNaam` subprogramma, bijvoorbeeld 'LAMONTEJOOST'. Deze waarde wordt overgedragen in de vorm van de variabele `nieuweNaam$`:

```
NieuweIndex nieuweNaam$
```

In de volgende paragraaf wordt uitgelegd hoe `NieuweIndex` deze bewerkingen uitvoert.

Het NieuweIndex subprogramma

Het `NieuweIndex` subprogramma voert de volgende stappen uit:

1. Het programma maakt een kopie van de **`index()`** array die tijdelijk in een array met **`indexType`** records, **`tijdelijkeIndex()`**, wordt opgeslagen.
2. De dimensie van de **`index()`** array wordt door middel van het **`REDIM`** statement opnieuw gedefinieerd. De array wordt met één element uitgebreid.
3. De tijdelijke kopie van de index wordt weer naar de opnieuw gedimensioneerde **`index()`** gekopieerd.
4. De indexingang voor het nieuwe record wordt aan het einde van de **`index()`** array opgeslagen.
5. De index wordt gesorteerd.

De `NieuweIndex` routine moet aan het begin van dit proces wel een tijdelijke kopie van de index maken, omdat de inhoud van de originele `index()` array verloren gaat wanneer deze opnieuw wordt gedimensioneerd met het `REDIM` statement.

Tip! In sommige Basic versies is het maken van een kopie van de `index()` array niet nodig, omdat die versies het `PRESERVE` statement kennen. Die zorgt er namelijk voor

dat de inhoud na het definiëren niet verloren gaat, maar juist behouden blijft.

De lengte van de tijdelijke array wordt gedefinieerd aan het begin van de `NieuweIndex` routine. De globale variabele `totWerkns%` is reeds in het `Toevoegen` subprogramma verhoogd om de nieuwe lengte van de database aan te geven. De lengte van de tijdelijke array wordt op één element minder dan deze waarde bepaald:

```
oudTot% = totWerkns% - 1
REDIM tijdelijkeIndex(oudTot%) AS indexType
```

Een `FOR...NEXT` lus kopieert de index naar de tijdelijke array:

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

Zoals u ziet, kopieert de toekenningsopdracht in deze lus complete records van `index()` naar `tijdelijkeIndex()`.
Vervolgens wordt de werkelijke `index()` array opnieuw gedimensioneerd tot een formaat, dat één element langer is dan de oorspronkelijke lengte teneinde de nieuwe recordingang onder te brengen:

```
REDIM index(totWerkns%) AS indexType
```

Na dit statement bevatten alle `werknNummer` elementen van `index()` een waarde 0 en zijn alle `werknNaam` elementen nul oftewel leeg. Een tweede `FOR...NEXT` lus kopieert de index weer vanuit de tijdelijke array naar deze array:

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

Het laatste element in de `index()` array is de indexingang voor het nieuwe record. Het `werknNummer` element krijgt het recordnummer in `totWerkns%` toegewezen:

```
index(totWerkns%).werknNummer = totWerkns%
```

Evenzo geldt dat ook voor `werknNaam$` die de geformatteerde werknemernaam bevat:

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

Tenslotte worden de werknemernamen in de index door middel van een aanroep naar het `Sorteren` subprogramma opnieuw gesorteerd in alfabetische volgorde. Deze `Sorteren` routine is gebaseerd op het Shell sorteeralgoritme. Dankzij deze routine vertraagt het herindexeringsproces de programma-uitvoering niet, zelfs wanneer de werknemerdatabase tot enige duizenden records uitgroeit. Op deze sorteermethode zal ik dieper in gaan.

De Shell sorteer methode.
Dit sorteeralgoritme is naar zijn ontwerper Donald Shell genoemd. De routine begint met het vergelijken van twee records die betrekkelijk ver van elkaar zijn verwijderd in de array en verwisselt twee records die niet in volgorde staan, met elkaar. Wanneer alle paren bij een bepaalde stapgrootte in volgorde staan, wordt het interval verkleind voor de volgende vergelijkingsronde. Tijdens dit proces wordt het interval dus bij elke vergelijkingsronde geleidelijk verkleind. Tegen de tijd dat de routine opeenvolgende records in de lijst gaat vergelijken, is de array bijna gesorteerd.
De `Sorteren` routine werkt met de indexrecords in de `index()` array. De lengte van deze array wordt bepaald door de globale variabele `totWerkns%`:

```
lengte% = totWerkns%
```

`Sorteren` gebruikt de variabele `stap%` om het interval tussen twee records aan te duiden. Het sorteren wordt door drie geneste lussen uitgevoerd. De binnenste `FOR...NEXT` lus vergelijkt elk paar records met een `stap%` interval en verwisselt deze door middel van het `SWAP` statement met elkaar indien de volgorde niet klopt:

```
IF index(hoger%).werknNaam > index(lager%).werknNaam THEN
   SWAP index(hoger%),index(lager%)
```

U ziet dat het `werknNaam` element de sorteersleutel is.

De middelste `DO...UNTIL` lus herhaalt deze vergelijkingen totdat alle paren records bij een bepaalde stapgrootte in volgorde staan. De buitenste `DO...WHILE` lus voert dit proces uit via geleidelijk kleiner wordende `stap%` intervallen totdat de laatste ronde opeenvolgende records in de array vergelijkt.
Dit was tevens de laatste ronde met betrekking tot het indexsysteem van het Werknemerprogramma, maar het programma bevat nog meer procedures!

Het DisplayRec subprogramma

`DisplayRec` is een betrekkelijk eenvoudig subprogramma dat een record op het scherm weergeeft. Om te beginnen, wordt `LeesNaam` aangeroepen om de werknemernaam van het gewenste record aan het toetsenbord te onttrekken:

```
LeesNaam toonRecord, nummer%, doelNaam$
```

Zoals reeds opgemerkt, retourneert `LeesNaam` een waarde 0 aan `nummer%` indien het record niet in de index wordt aangetroffen. In dat geval genereert `DisplayRec` een desbetreffende foutmelding:

```
IF nummer% = 0 THEN
   PRINT
   PRINT TAB(tabPos%); "Deze naam komt niet in de werknemer-file voor."
```

Indien het record wordt gevonden, leest DisplayRec het gehele record uit de databasefile en slaat deze op in de recordvariabele toonRecord:

```
ELSE
   CLS
   GET #1, nummer%, toonRecord
```

Na het GET# statement heeft het programma toegang tot de veldgegevens via de elementen van de variabele toonRecord. Het volgende voorbeeld demonstreert hoe de werknemernaam wordt weergegeven:

```
PRINT TAB(tabPos%) "Naam        :        ";
PRINT RTRIM$(toonRecord.Voornaam); " ";
toonRecord.achterNaam
```

De RTRIM$ functie verwijdert hierbij de spaties aan het einde van het voorNaam element. Dit is noodzakelijk omdat QuickBASIC de strings met vaste lengte automatisch opvult met spaties, om de gedefinieerde lengte van de string te handhaven.
DisplayRec geeft het salaris en het salaristype via een SELECT CASE constructie weer. Deze constructie bevat drie verschillende PRINT USING statements:

```
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
```

DisplayRec eindigt met een speciale passage die betrekking heeft op vertrokken werknemers. Zoals u weet, slaat de Vertrek optie het woord 'VERTROKKEN' in het record van een vertrokken werknemer op. Wanneer DisplayRec deze waarde detecteert, geeft de routine een desbetreffende mededeling weer tezamen met de vertrekdatum:

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

Het Wijzigen subprogramma

Het Wijzigen subprogramma genereert een submenu dat veranderingen in drie velden van een werknemerrecord accepteert: de afdeling-, functie- en salarisvelden.
Evenals Toevoegen en DisplayRec roept het Wijzigen subprogramma LeesNaam aan om de naam van het doelrecord te lezen. Wanneer de naam is gevonden zet Wijzigen de menuoptiestrings in de array wijzigMenu$() op. De eerste drie elementen van deze array

krijgen de promptstrings toegewezen die de huidige waarden van de drie velden bevatten. Wanneer de menu array is opgezet, wordt de `Menu%` functie vanuit een `SELECT CASE` constructie aangeroepen om het menu op het scherm weer te geven en de menukeuze van de gebruiker te verwerken:

```
SELECT CASE Menu%(wijzigMenu$())
```

De daaruit volgende actie hangt af van de keuze van de gebruiker. De `SELECT CASE` constructie voert een afgestemde invoerdialoog om de nieuwe veldwaarden binnen te halen. Dankzij een `DO...UNTIL` lus verschijnt het menu na elke invoer weer op het scherm, totdat de gebruiker de vierde optie '*Record OK*' kiest. Wanneer de wijzigingen zijn aangebracht schrijft het `Wijzigen` subprogramma het gemodificeerde record weer naar de oorspronkelijke locatie in de databasefile:

```
PUT #1, num%, wijzigRecord
```

Het Vertrek subprogramma

De `Vertrek` routine is verantwoordelijk voor het wijzigen van het record van een vertrokken werknemer. De routine zet eerst de lokale recordvariabele `vertrekRecord` op, waarin het doelrecord wordt opgeslagen. Vervolgens geeft de procedure het doelrecord op het scherm weer en verzoekt de gebruiker om te bevestigen dat het record in kwestie bij de vertrokken werknemer hoort. Het `Vertrek` subprogramma laat het record weergeven via een aanroep naar de `DisplayRec` routine:

```
DisplayRec vertrekRecord, welk%
```

`DisplayRec` retourneert het doelrecord aan variabele `welk%`.
Wanneer de gebruiker bevestigend heeft geantwoord, slaat de routine het woord 'VERTROKKEN' in het afdelingveld op en de actuele systeemdatum in het vertrekveld:

```
IF JaNee%("Is dit de vertrokken werknemer?") THEN
   vertrekRecord.afdeling = "VERTROKKEN"
   vertrekRecord.vertrekDatum = DATE$
```

`Vertrek` is de enige routine in het programma die een waarde aan het `vertrekDatum` veld toewijst. Andere routines laten dit veld blanco.
De routine schrijft vervolgens het gewijzigde record weer naar de originele locatie in de databasefile:

```
PUT #1, welk%, vertrekRecord
```

Het PrintRecords subprogramma

`PrintRecords` drukt een overzicht van de records uit de werknemerdatabase af. Het subprogramma informeert eerst bij de gebruiker of vertrokken werknemers in het overzicht moeten worden opgenomen. De `JaNee%` functie verwerkt het antwoord op deze vraag en de

routine slaat het logische resultaat van de functie in variabele `vertrokken%` op:

```
vertrokken% = JaNee%(prompt$)
```

Vervolgens herhaalt een `DO...UNTIL` lus de toetsdruk actie totdat de gebruiker de spatiebalk indrukt:

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

Dit stelt de gebruiker in de gelegenheid om de printer operationeel te maken.
Vervolgens voert `PrintRecords` een aantal `LPRINT` statements uit om de tekstregels naar de printer te sturen, te beginnen met de kopregels voor de tabel. De records worden via een `FOR...NEXT` lus geprint. Aan het begin van elke lus leest een `GET#` statement een record uit de databasefile in lokale recordvariabele `printRecord`:

```
FOR i% = 1 TO totWerkns%
   GET #1, index(i%).werknNummer, printRecord
```

Deze lus leest de recordnummers één voor één uit het `werknNummer` element van de `index()` array. Aangezien de records in alfabetische volgorde in de index zijn opgeslagen, wordt het overzicht ook in alfabetische volgorde afgedrukt.
Om te bepalen of een record wel of niet moet worden geprint, controleert het programma of het record een vertrokken werknemer vertegenwoordigt. De status van de werknemer is opgeslagen in de Boolese variabele `weg%`:

```
weg% = (LEFT$(printRecord.afdeling, 10) = "VERTROKKEN")
```

De variabelen `weg%` en `vertrokken%` bepalen of een record zal worden geprint:

```
IF weg% IMP vertrokken% THEN
```

Dit `IF` statement bepaalt de uitkomst aan de hand van de logische operator `IMP`. Indien `weg%` waar is, dat wil zeggen dat het om een vertrokken werknemer gaat, en `vertrokken%` onwaar is, dus de gebruiker wenst geen vertrokken werknemers in het overzicht, dan zal de IMP uitdrukking in een onwaar resulteren. In dat geval wordt het record niet afgedrukt.
Alle andere logische combinaties van `weg%` en `vertrokken%` zullen tot een waarde leiden die waar zijn, hetgeen betekent dat het record wordt afgedrukt.
Aan het einde van de programma-uitvoering sluit het `CloseFiles` subprogramma de databasefile en slaat de actieve index op.

Het CloseFiles subprogramma

`CloseFiles` sluit eerst de databasefile die filenummer 1 heeft:

```
CLOSE #1
```

Aangenomen dat de databasefile tenminste één record bevat, opent de routine vervolgens de sequentiële file WERKNMER.NDX teneinde de nieuwe versie van de index weg te kunnen schrijven:

```
OPEN "WERKNMER.NDX" FOR OUTPUT AS #1
```

Het gevolg van dit statement is dat een eventuele bestaande versie van de index op diskette wordt overschreven. In een `FOR...NEXT` lus wordt de actieve index naar de file via de recordelementen in de `index()` array geschreven:

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

Tenslotte sluit de routine de indexfile:

```
CLOSE #1
```

Een volgende programma-uitvoering zal dan kunnen terugvallen op een indexfile, die de records in de database correct specificeert.

Samenvatting

De intermezzo 'een Werknemers project' liet u in deze twee delen zien hoe een database, een menu en de subprogramma's opgebouwd worden. U heeft veel theorie kunnen leren met deze delen, maar de praktijk zal u duidelijkheid kunnen geven over hoe we het project opbouwen. In de volgende nieuwsbrief zal eerst een conclusie komen over het project. Daarna zal de hele listing worden getoond met elke getoonde naam van een programmadeel zoals een subprogramma. In de conclusie zal ik ook wat uitbreiding vertellen, bijvoorbeeld over hoe u een foutafhandeling routine kunt schrijven om ervoor te zorgen dat het programma niet onderbroken wordt mocht een bepaalde file niet bestaan.

Marco Kurvers

Grafisch programmeren in GW-BASIC (3).

In de vorige nieuwsbrief liet ik u de drie belangrijke onderwerpen zien die in GW-BASIC nodig zijn om goed te kunnen tekenen, de juiste schaal, de correctiefactor en de formule die daarbij hoort.

Denk eraan dat u, mocht u Visual Basic .NET willen gebruiken, om de grafische `e` parameter in de `Paint` event te gebruiken. Dus, om te tekenen: `e.Graphics.`<tekenmethode>

Nogmaals zal ik u de variabelen laten zien. Omdat onderstaande gegevens nog niet in een aparte bestand staan, worden ze in deze nieuwsbrief weer getoond.

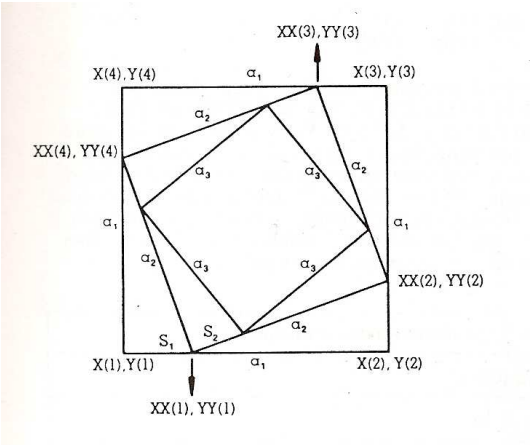
X1, Y1	coördinaten van het beginpunt
X2, Y2	coördinaten van het eindpunt
U, V	oorsprong van het wiskundige coördinatensysteem; dikwijls is dit het midden van het beeldscherm (160, 160)
H	de waarde 0.5 voor het afronden op hele getalwaarden
K	de vermenigvuldigingsfactor voor functiewaarden
W,W1	hoeken bij trigonometrische functies
RD	het getal $\pi/180$ voor het omrekenen van graden in radialen

De voorbeelden

In de vorige nieuwsbrief werd de wiskunde wat ingewikkelder en moest ik goed uitleggen wat de volgende programmalistings deden. Ook de afbeeldingen waren zeer technisch.

Programma 6 is er weer zo één. Deze is wat veeleisender in die zin dat het wat voorbereiding nodig heeft, net zo als programma 5 voorbereiding nodig had.

We willen een reeks ingeschreven vierkanten zo tekenen dat de hoekpunten van een ingeschreven vierkant uit de reeks op de zijden liggen van zijn voorganger. We zien deze situatie voor het eerste, tweede en derde vierkant van de reeks bij Figuur 1. De afstand tussen een hoekpunt van een vierkant en een hoekpunt van het volgende vierkant (S_1 en S_2) is steeds een vast deel van de zijde waarop de hoekpunten liggen. Zo is $S_1 = a_1 / k$ en $S_2 = a_2 / k$. In het algemeen is $S_n = a_n / k$, waarbij a_n de zijde is van het n^{de} vierkant in de reeks en k de verkleiningsfactor. De waarde voor k kunnen we zelf kiezen. Als k = 16 dan geeft dat een fraaie tekening.



Figuur 1.
(X(1),Y(1));
(X(2),Y(2));
(X(3),Y(3)) en
(X(4),Y(4)) zijn de coördinaten van de hoekpunten van een bepaald vierkant uit de reeks. Hoe berekenen we nu de coördinaten van de hoekpunten van het volgende ingeschreven vierkant ((XX(1),YY(1)); (XX(2),...)?

Voor het eerste hoekpunt van het eerste ingeschreven vierkant geldt:

$$XX(1) = X(1) + \frac{X(2) - X(1)}{K} \quad \text{èn}$$

$$YY(1) = Y(1) + \frac{Y(2) - Y(1)}{K}$$

Probeer eens na te gaan dat dit klopt in de tekening van Figuur 1.

Voor het tweede hoekpunt van het ingeschreven vierkant geldt:

$$XX(2) = X(2) + \frac{X(3) - X(2)}{K} \quad \text{èn}$$

$$YY(2) = Y(2) + \frac{Y(3) - Y(2)}{K}$$

In het algemeen geldt:

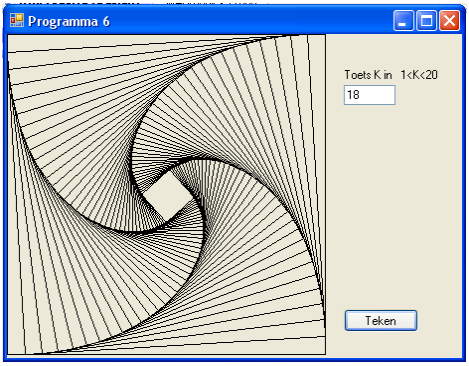
voor J = 1, 2, 3, 4, waarbij X(5) = X(1) en Y(5) = Y(1).

$$XX(J) = X(J) + \frac{X(J+1) - X(J)}{K}$$ èn

$$YY(J) = Y(J) + \frac{Y(J+1) - Y(J)}{K}$$

Als we het tweede vierkant getekend hebben, veranderen we XX(1) in X(1), YY(1) in Y(1), XX(2) in X(2), YY(2) in Y(2), enzovoorts, en tenslotte X(5) in X(1) en Y(5) in Y(1) en we gaan met dezelfde formules opnieuw XX(1), YY(1).... enz. berekenen, maar dan voor de hoekpunten van het volgende vierkant. We zien dit gebeuren in de regels 280 t/m 310 van het volgende programma. De regels 210 t/m 230 tekenen het vierkant, terwijl de regels 240 t/m 270 de hoekpunten van het volgende vierkant berekenen. Een structuurdiagram voor het programma ziet er zo uit:

- begin programma ingeschreven vierkanten
- superresolutie instellen
- lees waarde voor K in
- bepaal coördinaten voor het eerste vierkant
- voor het eerste t/m 40ste vierkant doe
    - teken dit vierkant
    - bepaal coördinaten voor het volgende vierkant
- einde programma

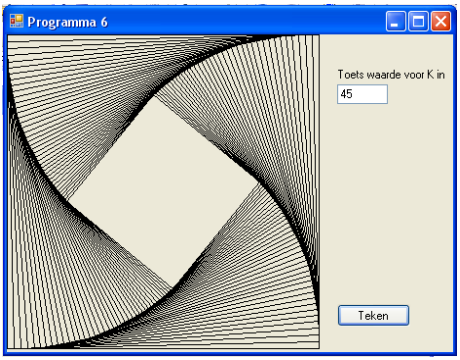


Hier komt het programma:

```
100 ' programma 6     INGESCHREVEN VIERKANTEN
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "TOETS K IN   1<K<20  "; K
150 DIM X(5), Y(5), XX(5), YY(5)
160 X(1)=0  : X(2)=320: X(3)=320: X(4)=0  : X(5)=0
170 Y(1)=320: Y(2)=320: Y(3)=0  : Y(4)=0  : Y(5)=320
180 H=.5
190 CLS
200  FOR N=1 TO 40
210     FOR J=1 TO 4
220        LINE (FNX(X(J)),Y(J))-(FNX(X(J+1)),Y(J+1)),1
230     NEXT J
240     FOR J=1 TO 4
250        XX(J)=X(J)+INT((X(J+1)-X(J))/K+H)
260        YY(J)=Y(J)+INT((Y(J+1)-Y(J))/K+H)
270     NEXT J
280     FOR J=1 TO 4
290        X(J)=XX(J)  : Y(J)=YY(J)
300     NEXT J
310     X(5)=X(1) : Y(5)=Y(1)
320 NEXT N
330 A$=INKEY$: IF A$="" THEN 330
340 CLS: KEY ON: END
```

```
Dim K As Single = CType(Val(txtK.Text()), Single)
Dim X() As Single = {0, 0, 320, 320, 0, 0}
Dim Y() As Single = {0, 320, 320, 0, 0, 320}
Dim XX(5), YY(5) As Single
Dim H As Single = 0.5
For N As Single = 1 To 40
    For J As Single = 1 To 4
        e.Graphics.DrawLine(Pens.Black, X(J), Y(J), X(J + 1), Y(J + 1))
    Next
    For J As Single = 1 To 4
        XX(J) = X(J) + Int((X(J + 1) - X(J)) / K + H)
        YY(J) = Y(J) + Int((Y(J + 1) - Y(J)) / K + H)
    Next
    For J As Single = 1 To 4
        X(J) = XX(J) : Y(J) = YY(J)
    Next
    X(5) = X(1) : Y(5) = Y(1)
Next
```

Ook al zouden we in GW-BASIC maar een K waarde mogen gebruiken t/m 20 vanwege de resolutie, een hogere waarde is in Visual Basic .NET ook toegestaan zodat we een heel ander afbeelding krijgen:



U kunt dit programma als uitgangspunt voor een uitgebreider programma gebruiken. Verdeel het beeldscherm in, bijvoorbeeld negen vierkanten (elk vierkant 100x100 punten). In elk van deze negen vierkanten tekent u, met bovenstaand programma, een reeks van ingeschreven vierkanten. Teken negen identieke reeksen, of kies steeds een andere waarde voor K of probeer ze ten opzichte van elkaar te laten draaien.

Grafieken van functies in cartesische vorm

Tot nu toe hebben we alleen rechtlijnige patronen getekend, bijvoorbeeld het programma voor het tekenen van alle diagonalen in een regelmatige n-hoek.

In dit en het volgende hoofdstuk zullen we ons uitvoerig bezig houden met het tekenen van de grafiek van een aantal continue en niet-continue functies. In de wiskunde noemen we de grafiek van een niet-lineaire functie een kromme. De grafiek van een lineaire functie (bijvoorbeeld de functie y = 4x + 3) noemen we een rechte. We kunnen de vergelijking van zo'n kromme op drie manieren formuleren:

A : met cartesische coördinaten : y = f(x)
B : met poolcoördinaten : r = f(φ)
C : of in parametervorm : x = f(t), y = g(t)

Niet-wiskundigen kennen vaak alleen de gewone (cartesische) vorm `y = f(x)`. Een voorbeeld van een niet-lineaire functie in cartesische vorm is de functie $y = 2x^2 - 3x + 4$. De grafiek van deze functie is een parabool.

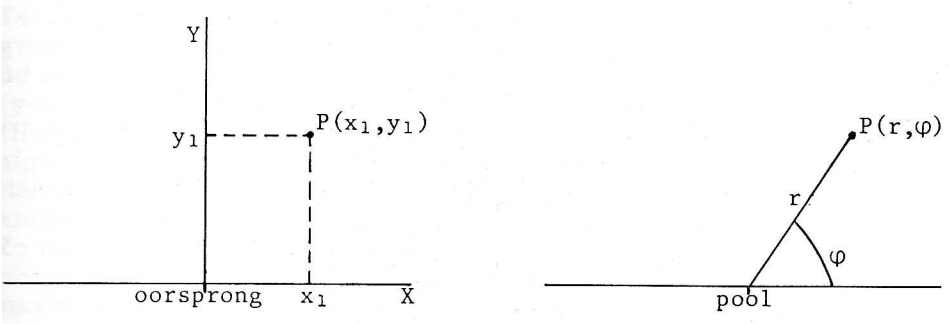
De functie r = 110.cos(4φ) stelt een kromme in poolcoördinaten voor. Als we de hoek φ laten lopen van 1 tot 360 graden en we tekenen bij elke hoek φ onder die hoek een punt op afstand r van de oorsprong, dan krijgen we de bloemfiguur die later te zien is in programma 12.
Deze vorm r = 110.cos(4φ) heet de poolcoördinatenvorm. Een punt in het platte vlak wordt nu niet gekenmerkt door de afstand van dat punt tot de x- en y-as (cartesisch), maar door de afstand tot de oorsprong (r) en de hoek φ die de x-as maakt met de lijn die dat punt met de oorsprong verbindt. Poolcoördinaten komen in de volgende hoofdstuk aan de orde vanaf programma 12, zie Figuur 2.

Er is nog een manier om de vergelijking van een kromme weer te geven en dat is de parametervorm. Hierbij worden de x- en y-coördinaat van een punt op de kromme afhankelijk gemaakt van een bepaalde parameter. Een voorbeeld zijn de vergelijkingen:

$$x = a.\cos(t) \quad \text{en} \quad y = a.\sin(t)$$

Als we t laten lopen van 0 tot 360^0, beschrijven de daarbij horende punten (x, y) precies de omtrek van een cirkel met straal a. De cartesische vorm van deze cirkelvergelijking is $x^2 + y^2 = a^2$, die velen direct zullen herkennen als de 'cirkelvergelijking'.

Figuur 2. **Continue functies**



Hetzelfde punt P in een cartesische coördinatenstelsel en een poolcoördinatenstelsel

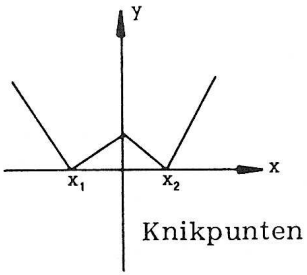
In dit hoofdstuk, tot en met programma 11, houden we ons dus alleen bezig met het tekenen van grafieken van functies en relaties die door middel van cartesische coördinaten beschreven worden. De andere twee vormen komen in het volgende hoofdstuk na programma 11 aan de orde.

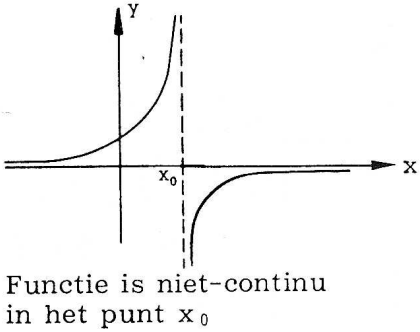
Continue functies

Een continue functie is een functie waarvan de grafiek in één vloeiende beweging van ons 'potlood', dat wil zeggen zonder het potlood van het papier te hoeven halen, getekend kan worden. Er mogen 'knikken' in voorkomen maar geen onderbrekingen, zie Figuur 3.

De linkergrafiek kunnen we in één beweging, zonder het potlood van het papier te halen, tekenen; bij de rechter grafiek kan dat niet. De linker grafiek is de grafiek van een continue functie; de rechter van een niet-continue functie. Als we alleen links of alleen rechts van het punt x_0 kijken dan is de functie op die intervallen natuurlijk wel continu! De functie is alleen niet-continu in het punt x_0. Aan de linkerkant van x_0 is de functiewaarde heel groot positief (plus oneindig!), aan de rechterkant van x_0 is de functiewaarde heel groot negatief (min oneindig!).

Figuur 3. **Grafieken van functies in cartesische vorm**





Programma 7 zal in het algemeen, gegeven de grenzen a en b van een bepaald interval ($a \leqq x \leqq b$), de grafiek tekenen van een willekeurige continue functie $y = f(x)$. Mochten de x-as en de y-as in het gebied liggen waarin de grafiek van de functie wordt getekend, dan willen we dat ook beide assen door het programma getekend worden. Om het programma kort, maar toch algemeen, te houden brengen we de volgende vereenvoudigingen aan.

1. Het programma zal steeds de functie, die getekend moet worden, als een aparte subroutine aanroepen. Programma 7 in versie Visual Basic .NET zal een functienaam aanroepen met een meegegeven parameter x.
2. Er zal geen schaalverdeling op de coördinaat assen worden aangebracht en zal er ook geen tekst worden afgedrukt. Later, als ik de appendix 'GW-BASIC voorbeelden' laat zien, zullen er twee voorbeelden staan met wel het afdrukken van tekst. In de versie Visual Basic 6 en .NET is dit helemaal geen probleem en zouden makkelijk label controls geplaatst kunnen worden die de tekenwaarden weergeven.

Terug naar het programmaontwerp. In het eerste deel van het programma berekent de computer, na het inlezen van de waarden van de intervalgrenzen a en b, de grootste en de kleinste functiewaarde (y-waarde) in het opgegeven interval [a,b]. De waarde van HP (Hoogste Punt) en LP (Laagste Punt) worden op het beeldscherm afgedrukt. We weten zo in welk gebied de computer gaat tekenen. Hierna vraagt het programma of we HP nog groter en of

we LP nog kleiner willen maken om daarmee een fraaiere tekening te krijgen. Zo niet, dan toetsen we voor HP en LP dezelfde waarden in als het programma heeft berekend; anders toetsen we de door ons gewenste maximale en minimale functiewaarden in.

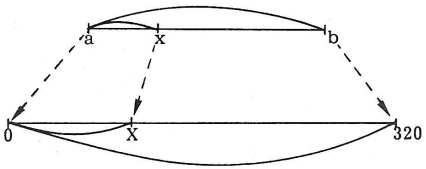
Het volgende programmadeel (regels 310-360) bevat de lus

```
FOR X=A + DX TO B STEP DX
```

voor het tekenen van de grafiek. De coördinaten x,y van een punt op de grafiek moeten met de juiste transformatieformules omgezet worden in beeldschermcoördinaten X,Y.

Om het interval [a,b] voor te kiezen x-waarden ($a \leqq x \leqq b$) om te zetten in ons scherminterval [0,320] ($0 \leqq X \leqq 320$) gebruiken we de volgende evenredigheid:

$$(x-a) : (b-a) = X : 320 \quad \text{ofwel} \quad \frac{x-a}{x-b} = \frac{X}{320}$$



in BASIC:

```
X2 = INT(KX*(X-A)+0.5),
```

met `KX = 320/(B-A).`

Omdat BASIC in hoofdletters 'werkt' hebben we in deze formule voor x de variabele `X` genomen, voor X de variabele `X2`, voor a de variabele `A` en voor b de variabele `B`. Nu we ook Basic versies hebben, is het niet meer nodig om in hoofdletters te werken. Toch zal ik in Programma 7 voor .NET dezelfde variabelen gebruiken om verwarring te voorkomen.

Dit geldt ook voor het afbeelden van het functiebereik $LP \leqq y \leqq HP$ op het HRG-beeldbereik $0 \leqq Y \leqq 320$:

$$(HP-y) : (HP-LP) = Y : 320 \quad \text{ofwel} \quad \frac{HP-y}{HP-LP} = \frac{Y}{320}$$

Dit geeft in BASIC:

```
Y2 = INT(KY*(HP-Y)+0.5)    met KY = 320/(HP-LP)
```

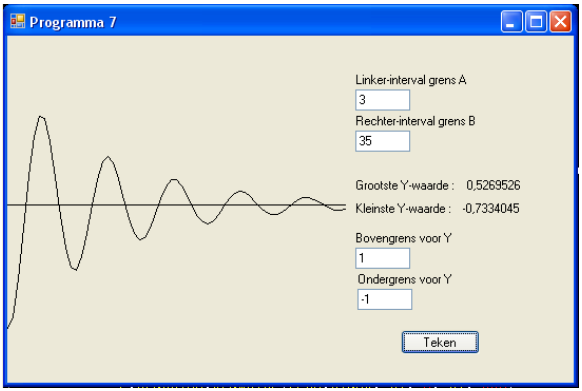
We zien deze berekeningen van `X2` en `Y2` in regel 330. In de volgende regel worden twee, op deze wijze berekende, punten (X1,Y1) en (X2,Y2) door een recht lijntje met elkaar verbonden. Vanaf regel 370 wordt de ligging van de x- en y-as bepaald. Kiezen we in bovenstaande transformatievergelijkingen voor X en Y de waarde nul, dan vinden we respectievelijk de ligging van de y-as en van de x-as (regel 370).

Met deze uitleg is de werking van het programma hopelijk te volgen.

Test u het Programma 7 met willekeurige, maar continue, functies. In dit voorbeeldprogramma is de functie $y = e^{-0,1x}.\cos(x)$ gekozen, hetgeen de grafiek van een gedempte trilling oplevert.
Hieronder zijn enkele ideeën voor minder moeilijke functies.

functie	**regel 1000**	**waarde voor a**	**waarde voor b**
$y = \sin(x)$	`Y = SIN(X)`	0	2π (= 6,2832)
$y = x^2$	`Y = X^2`	-4	+4
$y = e^x$	`Y = EXP(X)`	-3	+3
$y = x^3-2x^2-x$	`Y = X^3-2*X^2-X`	-1	+3

```
100 '  programma 7  GRAFIEK VAN EEN CONTINUE FUNKTIE
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "LINKER-INTERVAL  GRENS A "; A : PRINT
150 INPUT "RECHTER-INTERVAL GRENS B "; B : PRINT
160 IF A > B THEN C=A : A=B : B=C
170 HP=-100000! : LP=100000! : DX=(B-A)/64
180 FOR X=A TO B STEP DX
190    GOSUB 1000
200    IF Y > HP THEN HP=Y
210    IF Y < LP THEN LP=Y
220 NEXT X
230 PRINT "GROOTSTE Y-WAARDE : " ; HP : PRINT
240 PRINT "KLEINSTE Y-WAARDE : " ; LP : PRINT
250 INPUT "BOVENGRENS VOOR Y    "; HP : PRINT
260 INPUT "ONDERGRENS VOOR Y    "; LP
270 CLS
280 KX=320/(B-A) : KY=320/(HP-LP) : H=.5
290 X=A : GOSUB 1000
300 X1=0 : Y1=INT(KY*(HP-Y)+H)
310 FOR X=(A+DX) TO B STEP DX
320    GOSUB 1000
330    X2=INT(KX*(X-A)+H): Y2=INT(KY*(HP-Y)+H)
340    LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
350    X1=X2 : Y1=Y2
360 NEXT X
370 Y1=INT(KY*HP+H): X1=INT(KX*(-A)+H)
380 IF Y1 < 0   OR   Y1 > 320   THEN GOTO 400
390      LINE (FNX(0),Y1) - (FNX(320),Y1),1
400 IF X1 < 0   OR   Y1 > 320   THEN GOTO  430
410      LINE (FNX(X1),0) - (FNX(X1),320),1
420 LINE (FNX(0),0) - (FNX(320),320),1,B
430 A$=INKEY$: IF A$="" THEN 430
440 CLS: KEY ON: END
450 '
1000 Y=COS(X)*EXP(-.1*X)
1010 RETURN
```



De code voor Visual Basic .NET lijkt er heel anders uit te zien, vooral de eerste `If ... End If` die in GW-BASIC code niet aanwezig is. De eerste controle op de invoervakken is nodig om het vastlopen van de Paint event te voorkomen.
Bekijk ook de functie die niet meer als een regel 1000 aangeroepen wordt, maar als y = Cartvorm(x).

```
Private Function Cartvorm(ByVal X As Single) As Single
    Return Math.Cos(X) * Math.Exp(-0.1 * X)
End Function

Private Sub frmProg7_Paint(ByVal sender As Object, ByVal e As ...PaintEventArgs) ...
    If txtA.Text() = "" OrElse txtB.Text() = "" _
          OrElse txtHP.Text() = "" OrElse txtLP.Text() = "" Then
        Exit Sub
    End If
    Dim A As Single = Val(txtA.Text())
    Dim B As Single = Val(txtB.Text())
    If A > B Then
        Dim C As Single = A
        A = B
        B = C
    End If
    Dim HP As Single = -100000.0!, LP As Single = 100000.0!, DX As Single = (B - A) / 64
    Dim X, Y As Single
    For X = A To B Step DX
        Y = Cartvorm(X)
        If Y > HP Then HP = Y
        If Y < LP Then LP = Y
    Next
    lblHP.Text() = HP.ToString()
    lblLP.Text() = LP.ToString()
    HP = Val(txtHP.Text())
    LP = Val(txtLP.Text())
    Dim KX As Single = 320 / (B - A), KY As Single = 320 / (HP - LP), H As Single = 0.5
    X = A : Y = Cartvorm(X)
    Dim X1 As Single = 0, Y1 As Single = Int(KY * (HP - Y) + H)
    For X = (A + DX) To B Step DX
        Y = Cartvorm(X)
        Dim X2 As Single = Int(KX * (X - A) + H), Y2 As Single = Int(KY * (HP - Y) + H)
        e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
        X1 = X2 : Y1 = Y2
    Next
    Y1 = Int(KY * HP + H) : X1 = Int(KX * (-A) + H)
    If Y1 >= 0 And Y1 <= 320 Then
        e.Graphics.DrawLine(Pens.Black, 0, Y1, 320, Y1)
    End If
    If X1 >= 0 And X1 <= 320 Then
        e.Graphics.DrawLine(Pens.Black, X1, 0, X1, 320)
    Else
        Exit Sub
    End If
    e.Graphics.DrawRectangle(Pens.Black, 0, 0, 320, 320)
End Sub

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

Voor de zekerheid heb ik de Click event van de Tekenknop ook in Programma 7 gezet. Elke keer, wanneer u een Tekenknop op het formulier plaatst, moet u in de Click event de code `Me.Refresh()` plaatsen om ervoor te zorgen dat de Paint event aangeroepen wordt en nogmaals de grafiek getekend zal worden met de nieuwe ingevoerde waarden.

In de volgende nieuwsbrief gaan de programma's 8, 9, 10 en 11 ook over hetzelfde hoofdstuk 'Grafieken van functies in cartesische vorm'. Deze programmavoorbeelden zullen al-

lemaal in nieuwsbrief nr. 3 met de theorie aanwezig zijn. In nieuwsbrief nr. 4 zal het nieuwe hoofdstuk beginnen over het poolcoördinatenstelsel.

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

Oplossing puzzel: de spion en de wachter.

Het was niet zozeer een puzzel als wel een raadsel, maar omdat er meer alternatieve antwoorden inzitten kunnen we het toch een raadsel noemen die we in BASIC kunnen uitproberen. Laten we nogmaals kijken wat de personen zeggen en waarom de antwoorden goed zijn. De spion geeft op precies hetzelfde manier het antwoord, maar toch wordt hij gearresteerd. Wat zit er achter het wachtwoord en wat zijn de verschillen met de andere wachtwoorden?

Een spion luistert verborgen achter een struik het wachtwoord af, dat bij de wachter moet worden uitgesproken.
Eerste persoon: mag ik naar binnen?
wachter: wachtwoord is "twaalf"
persoon: "zes" wachter: "kom maar".

Tweede persoon: mag ik naar binnen?
wachter: wachtwoord is "acht"
persoon: "vier" wachter: "kom maar".

Derde persoon: mag ik naar binnen?
wachter: wachtwoord is "zes"
persoon: "drie" wachter: "kom maar".

Nu durft de spion het ook. Mag ik naar binnen?
Wachter: wachtwoord is "tien" spion: "vijf" Wachter: "Alarm, arresteer deze spion!!"

Vraag: "Wat had de spion moeten zeggen om ongehinderd naar binnen te mogen?"

Een alternatief antwoord zou zijn dat elke persoon het antwoord moet delen door de helft:

1. Twaalf gedeeld door twee is zes
2. Acht gedeeld door twee is vier
3. Zes gedeeld door twee is drie
4. Tien gedeeld door twee is vijf, maar helaas werd de spion toch gearresteerd. Dit alternatieve antwoord is dus niet de juiste oplossing terwijl dat wel had gekund.

De oplossing van punt 4 heeft te maken met het aantal letters in het wachtwoord dat de wachter voor elke persoon zegt. Elke persoon zegt het juiste aantal letters van het wachtwoord, maar de spion niet. Het wachtwoord "tien" is vier letters en niet vijf letters.

Henk van Weers

Game Maker voor beginners.

Het programma Game Maker is zeer makkelijk voor (beginnende) programmeurs die graag sneller games willen maken dan te moeten werken met alle gereedschappen die in de code nodig zijn. Met Game Maker kunnen de spellen zonder code gemaakt worden, maar het is niet verkeerd om toch wat scriptcode te programmeren voor bijvoorbeeld muisbesturing.

In de nieuwsbrief van vorig jaar, nummer 4, heeft u al kennis gemaakt met Game Maker en heb ik u een script laten zien hoe een batje met een muis bestuurd kan worden. Zonder die script is het haast onmogelijk om een batje te laten bewegen met de muis en niet te vergeten om ook het balletje mee te sturen als het vast zit op het batje.

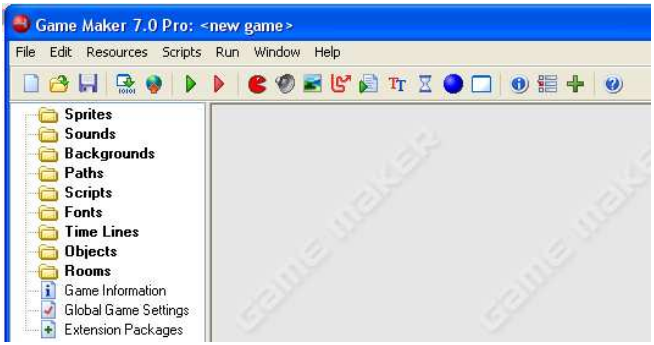
GML (Game Maker Language)
GML is geen programmeertaal maar een scripttaal, te vergelijken met VBScript en eigenlijk ook met VBA in de Office toepassingen. De scripttaal ziet er echter niet uit als Basic, maar tussen C en Basic in. Er kunnen dus wat Basic statements in voorkomen, maar tegelijkertijd ziet het geraamte van GML eruit als C code.

In Basic: `If A = 0 Then A = 1`
In GML: `if (A == 0) A = 1` of met Basic statement: `if (A == 0) then A = 1`

In GML moeten we gebruik maken van haakjes en moeten we de C operator == gebruiken, maar we zien wel dat GML het Basic statement `then` herkent.

Game Maker IDE
Het is beter om eerst Game Maker te verkennen dan gelijk GML te gebruiken. Op de volgende pagina's leg ik u uit wat sprites, objecten van sprites en de instanties daarvan zijn.

Figuur 1. De Game Maker IDE



Figuur 1 laat een volledige versie van Game Maker zien. Alleen de niet-volledige versie is gratis. De volledige professionele versie heeft meer mogelijkheden zoals tekeneffecten (explosies, fade up en fade down, enzovoorts).
Aan de linkerkant ziet u een boomlijst met mappen en daaronder wat instellingen. De laatste map, Extension Packages, is een wat geavanceerdere bezigheid. Hiermee kan code van andere programmeertalen in GML worden gepacked, zodat de scriptcode meer functionaliteit krijgt. Ook u kunt Basic package DLL's maken en in Game Maker gebruiken. Meer over het schrijven van eigen DLL's kunt u vinden in de help van MSDN (Basic).

Sprites
Met de eerste map kunt u een sprite inladen of een hele nieuwe maken. Game Maker heeft een eigen sprite tekenprogramma zodat u Paint of een ander tekenprogramma niet hoeft te gebruiken.
Sounds
Game Maker heeft helaas geen sound editor. Als u een nieuwe maakt, zal Game Maker om een sound editor vragen die u voor nieuwe geluiden kunt gebruiken. U moet daarvoor het pad en de programmanaam opgeven.
Backgrounds
De achtergronden kunnen in Game Maker helemaal zelf worden gemaakt. Het werkt net zo als u een sprite maakt. De background editor heeft nog een mogelijkheid. U kunt de achtergrond verdelen in vakken, ook wel *tiles* genoemd. In de volgende nieuwsbrieven kom ik daar op terug.
Paths
Een path is een bewegingsrichting die in de path editor gemaakt moet worden. De objecten van de sprites kunnen hun eigen path krijgen om in verschillende richtingen op het scherm te kunnen bewegen.

Scripts
De script editor is een aparte codevenster waarmee functies geschreven kunnen worden voor extra functionaliteit. Elke script kan argumenten meekrijgen, maar er moet wel eerst een lokale variabele worden gedeclareerd en aan een argument worden toegekend, voordat het argument gebruikt mag worden.
De naam van de script wordt in een event van een object aangeroepen, maar het is ook mogelijk om een naam van een script in een andere script aan te roepen. Denk maar aan Private subs in Basic die bijvoorbeeld in de Click event aangeroepen worden.
Fonts
Game Maker heeft een eigen lettertype instellingsvenster, maar geen editor.
De font, met het juiste lettertype en lettergrootte, wordt bewaard in een fontnaam die in de scriptcode of als actie in de draw event van een object gebruikt kan worden. Een eigen lettertype maken is helaas niet mogelijk.
Time Lines
Op een bepaalde tijd telkens een actie uit kunnen voeren zonder dat het spel gaat hangen of gaat pauzeren. Dankzij de Time Lines kunnen meerdere acties tegelijkertijd worden uitgevoerd of kunnen we acties die in meerdere events hetzelfde doen in één tijdmethode plaatsen. Toch is het gebruik hiervan minimaal, omdat de Step event het meeste werk doet en tijdens het spel alles controleert.

Tip! Denk maar aan de Tekenknop bij hoofdstuk 'Grafisch programmeren in GW-BASIC', want daar zien we de methode `Refresh()` die dus ook als een soort Time Line te vergelijken is. Zonder die knop kan er niet opnieuw getekend worden.

Wat dus het verschil is tussen een Time Line en de Step event zal ook later ter sprake komen.
Objects
Het object is het belangrijkste materiaal in Game Maker. Zonder de objecten kunnen we geen spel maken, ook al zouden we al de sprites klaar hebben. Dit is ook de reden waarom we geen Basic programma kunnen schrijven terwijl we al de klassen library's hebben. We hebben objectvariabelen nodig om instanties te kunnen creëren die we in Basic verder moeten gebruiken. Hetzelfde geldt dat voor de objectinstanties die we pas kunnen gebruiken als we de objecten hebben, en een spriteinstantie bestaat niet.
Rooms
Een speelveld is een scherm waar de camera op is gericht. Voordat we een omgeving met de grafische materialen kunnen bouwen, moeten we de camera instellen. Waarom eigenlijk?
Een kamer kan heel groot zijn, wel groter dan het beeldscherm zelf. Een room (kamer) is verdeeld onder views (beelden) en de camera moet dan op één van die beelden zijn ingesteld. In gewone programmeertalen is dat een grote klus voordat we het beeld met materialen kunnen vullen. Creëren we in Game Maker een nieuwe room, dan is er eerst heel wat in te stellen voordat we aan het speelscherm mogen beginnen.

Omdat een room uit meerdere views bestaat, kan het gebeuren dat objectinstanties 'per ongeluk' buiten de cameraview liggen. Game Maker let daar automatisch op en zal dan ook vragen of de buitenliggende materialen verwijderd moeten worden, mochten ze erbuiten liggen.

Game Information

Game Maker biedt voor u een automatische helpscherm voor uw spel. U hoeft alleen maar de informatie over het spel te schrijven. U moet wel de helpactie uitvoeren in een eventuele toets, die dan door de speler wordt ingedrukt.

Global Game Settings

Veel globale variabelen gebruiken is een doorn in het oog. Game Maker heeft een tabel waarmee de veelgebruikte variabelen ingesteld kunnen worden, zodat ze niet in de code globaal gedeclareerd hoeven te worden. Globale variabelen is een ramp, gebruik ze zo min mogelijk.

Volgende keer

In de volgende nieuwsbrief laat ik u de sprite editor zien en kunt u zien hoe de object event lijst met de acties werkt en hoe we een room in kunnen stellen en die met objectinstanties kunnen opbouwen.

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;
- 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.

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

Qbasic: Cursus, lesmateriaal en voorbeelden op CD-ROM € 6,00 voor leden. Niet leden € 10,00.
QuickBasic: Cursusboek en het lesvoorbeeld op diskette,
€ 11,00 voor leden. Niet leden € 13,50
Visual Basic 6.0: Cursus, lesmateriaal en voorbeelden op CD-ROM,
€ 6,00 voor leden. Niet leden € 10,00
Basiscursus voor senioren, Windows 95/98,
Word 97 en internet voor senioren, (geen diskette). € 11,00 voor leden. Niet leden € 13,50

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