Nieuwsbrief

17de jaargang maart 2010 Nummer 1

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Gordon Rahman	30
Het Bestuur	Penningmeester gezocht: 24
Henk van Weers	Puzzel: 24

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

In de nieuwsbrieven zal er weer van alles in komen te staan: voorbeelden, ideeën, informatie, noem maar op, maar als u ook hele leuke voorbeelden, ideeën en informatie heeft, stuur het dan naar mij toe. Het is toch leuk als leden elkaar informatie kunnen geven en elkaar kunnen helpen.

Ook dit jaar ga ik verder met onderwerpen waar ik vorig jaar in nummer 4 gebleven was. Er zullen ook onderwerpen volgen op de volgende nieuwsbrieven, bijvoorbeeld cursussen en intermezzo's. De eerste intermezzo zal gelijk interessant worden, omdat de informatie over database ontwikkeling gaat en hoe we een project opbouwen die uit eigen functies en subroutines zal bestaan.

Heel veel lees- en programmeerplezier.

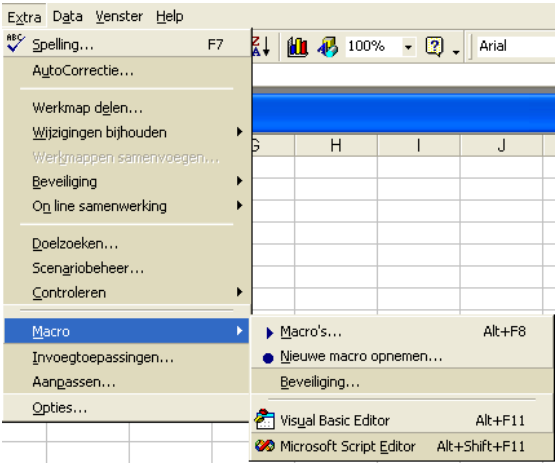
Marco Kurvers

BASIC cursus: VBA met Excel (1).

In vorige nieuwsbrieven staan ook onderwerpen over VBA in Excel, maar toen had ik het ook echt over VBA in Excel en niet met Excel. Ik wil nu beide samen brengen, want samen ermee werken is anders dan alleen maar direct naar de Visual Basic editor gaan en methoden schrijven voor de werkbladen.

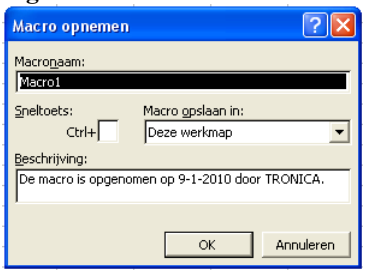
In deze eerste les wil ik u uitleggen hoe we in Excel de werkbladen en VBA samen kunnen gebruiken. Het is zelfs mogelijk om onderdelen die u in VBA maakt naar voren te halen. Als voorbeeld kunt u er voor zorgen dat een bepaalde cel in een blad een dialoogformulier zal aanroepen en de gegevens zal doorsturen. Dat is pas voor latere zorg. Eerst gaan we kijken wat het Excel menu Extra voor keuzes heeft. Kijk eens naar Figuur 1.

Figuur 1.



Het menu Extra heeft een keuze Macro die zelf ook weer een menu heeft. Er is één keuze die al makkelijk te begrijpen is. De 'Visual Basic Editor'. De andere editor 'Microsoft Script Editor' wordt gebruikt om webpagina's te programmeren. Daar zijn echter nieuwe functies voor nodig en het kan zijn dat ze ontbreken en geïnstalleerd moeten worden.
De andere keuzes hebben te maken met macro's. Wat u er mee kunt doen is alsof u een recorder met een bandje gebruikt: het opnemen en aanroepen van methoden.

Figuur 2.



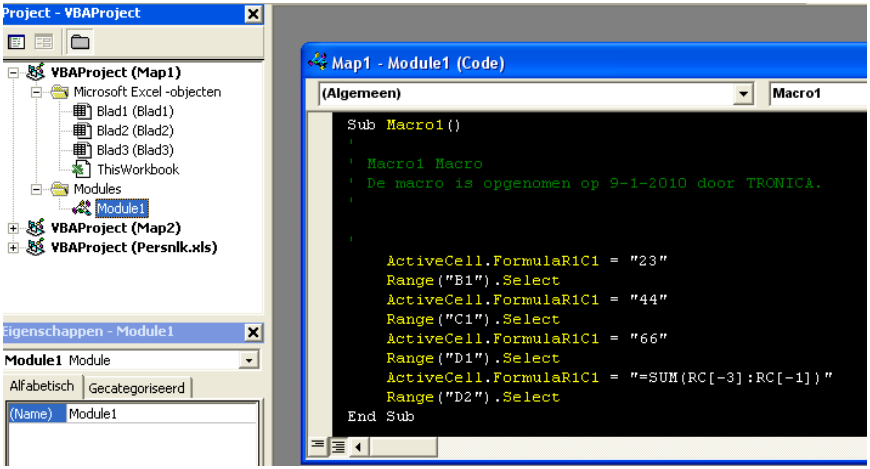
Als u kiest om een macro op te nemen, zal er een formulier verschijnen zoals Figuur 2 laat zien.

In het vak van 'Macronaam' kunt u een andere naam opgeven dan de standaard naam. Een macro kan ook een sneltoets bevatten. Dat is handig wanneer u de macro wilt starten zonder steeds naar Macro te moeten, zie Figuur 1. Wanneer de opname van een macro klaar is, zal deze automatisch een subroutine worden met als subroutinenaam degene die u hier in het vak hebt ingetoetst.
U kunt de macro opslaan op een willekeurige plek die u kunt kiezen door op het pijltje te klikken, zie Figuur 3.

Figuur 3.

Er zijn drie soorten werkmappen. Elke werkmap heeft zijn eigen werkbladen, modules en een ThisWorkbook. Het ThisWorkbook is geen module maar een Excel object. Vanuit het ThisWorkbook object kan ook code worden aangeroepen, zoals initialisatie om de werkbladen voor te bereiden, zie Figuur 4.
Onthoud wel: elke map werkt lokaal. U kunt dus niet in Map1 een module gebruiken die geplaatst is in Map3.

Figuur 4



Zie ook dat elke map in feite een heel project is met elk één werkboek en tenminste één module. Net als in de andere Visual Basic versies kunt u ook klassen en formulieren maken.

De vrijheid voor het werkblad en het gebruiken van cellen.

Elke cel op een werkblad is te vergelijken met een textbox. Het zijn dus allemaal vakken die eigenschappen, methoden en gebeurtenissen hebben, ook een beetje te zien aan de code van Figuur 4. Aan elke actieve cel werd een formule toegewezen, maar ik heb eigenlijk in de cellen de getallen ingetoetst. Kennelijk wordt tijdens de opname van de macro gebruik gemaakt van de eigenschap `FormulaR1C1` en om telkens de actieve cel te gebruiken moet er weer een ander geselecteerd worden. Dat is een groot nadeel van macro's opnemen, want alles wat u doet tijdens de opname, met fouten erbij, wordt meegenomen en in de subroutine geplaatst. Elke keer als u in een cel klikt om een getal te typen, zal de cel geselecteerd worden. Die functie wordt ook opgenomen nadat u de waarde typt en vaak is dat niet de bedoeling. Ook al kan het opnemen van macro's nuttig zijn; u hebt niet uw eigen vrijheid om code te gebruiken die u wilt gebruiken.

Maar wat is de vrijheid als we in een cel niet kunnen dubbelklikken om in de code editor te komen? Hoe kunnen we de cellen van een werkblad besturen?
Als ik u vertel dat we via het object `ThisWorkbook` een nieuw werkblad kunnen openen en zelfs ingevulde cellen kunnen berekenen dan bent u misschien wel nieuwsgierig. U kunt naar de Visual Basic editor gaan en dubbelklikken op het workbook, zie Figuur 4. Er zal een zwarte scherm, of een andere kleur scherm, verschijnen. Het ligt eraan hoe de kleurinstellingen bij u zijn. Ik heb het zwart gemaakt. Voor programmeren is het beter voor uw ogen en zult u weinig of geen last hebben van hoofdpijn dan wanneer u de standaard kleurinstellingen gebruikt. Dat is namelijk een wit scherm.

De indeling van het werkblad.

Een werkblad in Excel ziet eruit als een tabel. Vakken die ingedeeld zijn in kolommen en rijen. Per kolom staan de letters aangegeven en per rij staan de getallen aangegeven. Die waarden kunnen niet gewijzigd worden. Het zijn vaste cellen die niet te bewerken zijn.
Een klik op zo'n kolom of rij zorgt ervoor dat een hele kolom of rij geselecteerd wordt. Dat gebeurt niet als we op een cel klikken die te bewerken is, maar we kunnen ze wel selecteren zonder gebruik te maken van de vaste cellen.

Onder het werkblad zien we de tabbladen, want we kunnen meerdere bladen gebruiken per map. We kunnen ze aanklikken of schuiven door op de pijltjes te klikken die links naast de tabbladen staan.
Rechts van de tabbladen is de schuifbalk. Hiermee kunnen we het blad naar links en rechts verschuiven. De andere schuifbalk, helemaal aan de rechterkant, is om de rijen te verschuiven.

Hoe we het werkblad in VBA kunnen indelen kan wat meer tijd kosten als u nog niet zo goed met VBA overweg kunt.

De volgende les.

Het indelen van het werkblad in VBA en de besturing via het `ThisWorkbook` object zal in de volgende les ter sprake komen, maar het is ook een heel belangrijk object. Daarom zal het object in de komende lessen steeds terugkomen om de werkmap projecten te initialiseren en in te delen voor de praktijkoefeningen. In elke les zal er een oefening komen die u dan stap voor stap moet volgen, dus niet op de manier zoals nu. Dit was alleen een introles.

Marco Kurvers

Meneer van Dalen wacht op antwoord.

Al vanaf de basisschool leren we rekenen met +, -, x en :, de bekende symbolen met basisrekenen. De symbolen worden in de programmeertalen operatoren genoemd. Op school leerden we de tafels t/m 10 en we vonden het knap van onszelf als we even een tafeltje hoger konden. De tafels kunnen we zelf ook maken in Basic door een lus uit te voeren die van 1 t/m 10 loopt. Door er een binnenlus te nesten, die ook van 1 t/m 10 loopt, hebben we de 10 tafels compleet.

Maar het rekenen bleef daar niet bij. Het werd wiskunde. Wiskundig rekenen is soms niet makkelijk en zeker niet als we ons ook nog eens aan een wiskundige wet moeten houden: de wet van het berekenen in een wiskundige volgorde. Als we ons daar aan houden, is er niets aan de hand en zal de som van links naar rechts berekend worden. Breken we echter de wet, dan moeten we aangeven dat het niet van links naar rechts berekend kan worden. Ook Basic weet dat niet. Basic zal zich altijd aan de wiskundige wet houden, maar soms moeten we wel Basic vertellen dat we een andere volgorde willen hebben in een berekening.

Variabele 'x' vrijmaken voordat het in Basic uitgevoerd kan worden.

Misschien kunt u zich nog de formules herinneren met 'x' die er uitzien als:

```
5x = 10
3x + 4 = 16
```

We kunnen dit echter niet op die manier invoeren. Basic zal dit niet als formules, zelfs niet als condities, zien en dat komt doordat er een variabele aan de verkeerde kant van het getal staat. Basic ziet wel `x5` maar niet `5x`. Onderstaande antwoorden zullen verschijnen als we de eerste formule proberen:

```
Print 5x = 10
 5 False
```

Andere BASIC versies/dialecten geven geen False maar een nulwaarde.

`y = 5x = 10` Er zal nu een foutmelding verschijnen: 'Expected: end of statement.'

De enige oplossing om zulke formules ook in de programmeertalen te kunnen gebruiken is, door eerst de variabele vrij te maken. Echter moeten we ook letten op Meneer van Dalen, want als we de variabele vrij gaan maken dan zal ook het een en ander veranderd worden. Hieronder een tabel met wat we moeten doen en wat er gaat gebeuren.

Wat moeten we doen?	**Wat gebeurt er?**
Getallen die aan de linkerkant van de '=' operator staan moeten naar de rechterkant.	Positieve getallen worden negatief en negatieve getallen worden positief.
Variabelen die aan de rechterkant van de '='	Positieve variabelen (unsigned) worden

operator staan moeten naar de linkerkant. Let op! Eventuele getallen die niet vrijstaan gaan mee!	negatief (signed) en negatieve variabelen worden positief.
Meer variabelen van dezelfde naam en meer getallen samen uitrekenen.	Alles zal gelijknamig worden gemaakt, totdat we één variabele links, met eventueel een getal, van de '=' en rechts één getal van de '=' overhouden.
Heeft de variabele een getal bij zich? Zo ja, dan moet dat getal ook naar de rechterkant van de '='.	Het getal wordt samen met het andere getal aan de rechterkant een deling. Let op! Deel altijd het rechter getal met de linker, die u dus wilt verplaatsen. Oplossing is klaar, de variabele staat vrij.

Theoretisch lijkt het een beetje lastig, maar hieronder kunt u het stap voor stap bekijken wat er allemaal gebeurt zoals in het tabel uitgelegd wordt.

- `5x + 3 = 2x + 15`
- `5x – 2x + 3 = 15`
- `5x – 2x = 15 – 3`
- `5x – 2x = 12`
- `3x = 12`
- `x = 12 / 3`

De laatste stap kan nu wel in BASIC worden gebruikt. Tja, bestond er maar een programmeertaal die zoiets voor ons zou kunnen doen. Dat zou heel wat werk schelen. Ook Excel kent het gelijknamig maken en vrij maken van variabelen niet. We moeten dus ook BASIC een hulpsteuntje geven.

Haakjes gebruiken, wanneer en hoe.

Meneer van Dalen heeft een Wet waar we ons ook in code aan moeten houden om goede software te kunnen schrijven. BASIC kent ook die wet. Gelukkig maar, zult u denken, maar het is niet altijd nodig om ons daar ook aan te houden. Soms willen we liever dat er eerst opgeteld wordt voordat het vermenigvuldigt wordt en dus hebben we hulp nodig – de haakjes.
De haakjes zorgen ervoor dat wat er tussen staat eerst uitgevoerd zal worden, maar dat betekent niet dat Meneer van Dalen daar niet op let. Ook binnenin de haakjes zal weer de volgorde worden genomen. We moeten dus eerst zelf bepalen hoe we een formule willen schrijven, voordat we die gaan gebruiken in ons programma.
Kunnen we dan niet gewoon de delen van de formule die we als eerste willen uitvoeren naar voren halen, zal men denken, zodat we de juiste volgorde hebben? Echter kan dat niet direct worden gedaan, want als we dat zouden doen dan moeten we de theorie van bovenstaand tabel gebruiken. Doen we dat niet, dan zal het resultaat van de formule niet meer juist zijn. Zie eens onderstaande formule.
`n = 10 * 2 + 8` of in de wiskunde ook wel gezegd: `n = 10x + 8` (10 maal x)
Als we een tweede variabele inderdaad erbij zouden gebruiken, dan wordt het:

`n = 10 * x + 8` die voor BASIC een normale formule is.
We nemen dan ook aan als voorbeeld dat het getal 2 is toegekend aan variabele `x`.
Als antwoord zal dan op de juiste volgorde `n = 28` zijn. Maar stel dat we dat niet willen en juist willen dat eerst variabele `x` met getal 8 samen opgeteld wordt. Wat we ook verschuiven, het zal niet helpen. Telkens zal weer hetzelfde antwoord als resultaat komen.
De enige oplossing om de wet te dwarsbomen is gebruik te maken van haakjes, zodat de formule er plots heel anders uit gaat zien:

```
n = 10 * (x + 8)
```

Nu eerst de deelformule, of in de computertaal genoemd: deelexpressie, uitgevoerd wordt voordat de vermenigvuldiging plaatsvindt, zal variabele `n` ook een heel ander resultaat krijgen. Bereken ook eens de formule met de haakjes en u zult zien dat het resultaat nu `100` zal zijn.

Meneer van Dalen in condities.

Niet alleen wiskundige formules en programmeerbare expressies hebben met Meneer van Dalen te maken, ook condities hebben een bepaalde verwerkingsvolgorde voordat het resultaat juist of onjuist kan zijn. Condities kunnen zelfs moeilijker zijn dan expressies en dat komt doordat er meerdere condities aan elkaar kunnen staan met alleen operatoren ertussen die met controle te maken hebben. In expressies is dat niet het geval en zal elke expressie samen met een andere expressie als één expressie worden gezien. Hieronder kunt u het verschil zien tussen een meervoudige expressie en een meervoudige conditie.
Expressie: `n = (x + 9) - (6 * k)`
Conditie: `b = (x = 10) Or (p > 0)`
Voor de conditie operatoren is er een volgorde die op een heel andere manier werkt. Zouden we in BASIC de haakjes weghalen uit de conditie, dan zal alsnog hetzelfde resultaat worden gegeven. Kijk eens naar onderstaande codefragment. Daar zult u zien dat geen enkele conditie zijn eigen recht heeft!

```
x = 10
p = 0
Print Not x = 5 And p = 5
```

De uitvoer zal niet waar zijn, maar onwaar! Ondanks dat variabele `x` inderdaad niet gelijk is aan `5`. Vreemd, zou u misschien denken. Als ik tegen de groenteman zeg: 'Gets, niet de appel en banaan zijn goed meer.' dan zou de groenteman heus wel weten dat ik zowel de appel als de banaan bedoel. Zulke gevallen weet BASIC echter niet en we moeten BASIC veel duidelijker maken wat we bedoelen. We kunnen twee oplossingen gebruiken:
`Print Not x = 5 And Not p = 5` of de oplossing met haakjes:
`Print Not (x = 5 And p = 5)` die nog meer duidelijkheid geeft.
De laatste met haakjes zal weer dezelfde uitvoer hebben als de vorige formules met haakjes. In BASIC zal eerst de conditie tussen haakjes uitgevoerd worden voordat bepaald wordt of het resultaat onwaar is. Zoja, dan zal het hele resultaat 'waar' zijn.

Programmeertaaltip!
Let op! In andere programmeertalen, zoals Pascal en Delphi, is de uitvoerwet in condities anders. In die talen moet er juist voor gezorgd worden dat altijd de `And` en `Or` operatoren buiten de haakjes staan, maar de condities zonder die operatoren binnenin de haakjes staan. In Pascal moeten we bovenstaand voorbeeld invoeren als:

```
Writeln (not ((x = 5) and (p = 5)))
```

De operatoren in die programmeertalen staan bovenaan de Wet van Dalen en dankzij de haakjes zal de `and` operator worden omzeild.

Marco Kurvers

Intermezzo: een Werknemers project (1).

Met deze intermezzo laat ik u zien hoe een project kan worden geschreven voor het beheren van werknemers. Een intermezzo houdt in, dat een onderwerp zo algemeen mogelijk moet blijven. Daarom zal de informatie, en het op het laatst getoonde voorbeeld, behandeld worden voor QuickBASIC gebruikers. Gebruikers van andere BASIC dialecten zullen dan weinig moeite hebben met de informatie en het overnemen van de code. Mocht er een functie of statement zijn die niet overeenkomt met het BASIC dialect dat u gebruikt, bekijk dan eens de appendix die de conversietabellen laat zien. U kunt de appendix vinden in de lijst van onderwerpen.

Database manipulaties.

QuickBASIC was na de voorloper QBASIC de eerste BASIC versie die databases kan beheren dankzij goed gestructureerde recordtypes. Dankzij extra 'tools', zoals sorteer- en zoekroutines, is QuickBASIC in staat om talloze database-managementtaken moeiteloos uit te voeren.

Menubesturingen.

De informatie van dit project en het voorbeeld laat u zien dat QuickBASIC een handige BASIC versie is waarmee makkelijk menubesturing geprogrammeerd kan worden, zonder gebruik hoeven te maken van de IDE van Visual Basic waarmee we een menu niet meer in code hoeven te ontwerpen. Toch is het altijd nog interessant en leerzaam om te weten hoe we een menu in code maken en besturen. Gebruikers die geen Windows formulieren willen gebruiken, kunnen gebruik maken van PowerBASIC en de eigenlijke versie QuickBASIC waar de intermezzo over gaat. Wie Liberty Basic wil gebruiken kan ook de intermezzo gebruiken, maar controleer dan wel de conversies die nodig zijn voor de code en eventueel kunt u in Liberty Basic de menubesturingen programmeren met formulieren zonder gebruik

te maken van de IDE, want die is er niet bij in die versie.

De mogelijkheden van het project.

- Invoeren van nieuwe werknemer records in de database.
- Selecteren van een bepaald werknemer record volgens de naam en een schermweergave van dit record.
- Wijzigen van specifieke gegevens in een record.
- Records van een speciale vermelding voorzien wanneer werknemers het bedrijf verlaten.
- Printen van records die volgens een bepaalde volgorde zijn gerangschikt.

Zoals u uit deze punten kunt opmaken, is de opslagstructuur van een database gebaseerd op records. Een record bevat alle informatie over een bepaald onderwerp en in het geval van de werknemerdatabase bevat elk record alle belangrijke gegevens over één werknemer. Een record wordt weer verdeeld in velden die een bepaald soort gegevens bevatten.
Bijvoorbeeld, het Werknemer programma verzoekt om diverse informatie-items voor elk nieuw werknemerrecord: naam, ziekenfondsnummer, de datum van indienstregeling, afdeling, functie en salaris. Deze items vormen de velden van een werknemerrecord.
Een random-access file heeft enige belangrijke kenmerken, die deze structuur uitermate geschikt maken voor database applicaties. Ten eerste hebben de records in een random-access file een vaste lengte. Ten tweede verschaft een database programma aan de hand van de specificaties voor een bepaalde toepassing een duidelijke structuur om de file-records in velden met informatie te verdelen. Bovendien kunnen de records in elke willekeurige volgorde naar de file worden geschreven of worden opgevraagd dankzij de vaste lengte van de records in een random-access file.
Aangezien deze aspecten in de intermezzo nog menigmaal ter sprake zullen komen, verdiep ik me eerst in de statements en functies die random-access files kunnen creëren, en te manipuleren. Hiertoe behoren:

- Het openen van een random-access file en het definiëren van een recordstructuur. Statements **`OPEN`** en **`TYPE`**.
- Het vaststellen van de totale lengte van een geopende file. Functie **`LOF`**.
- Het lezen van een record uit een random-access file. Statement **`GET#`**.
- Het wegschrijven van een record naar het einde van de file of een gewijzigd record naar een bepaalde locatie in de file wegschrijven. Statement **`PUT#`**.
- Het sluiten van een file wanneer het programma een bepaalde bewerking heeft voltooid. Statement **`CLOSE`**.

Het gebruikergedefinieerde datatype moet het aangewezen medium zijn om de recordstructuur van een random-access file te definiëren. U zult zien hoe het `TYPE` statement met deze structuur in samenhang met de verbeterde QuickBASIC random-access filesubroutines wordt ondersteund.
In de intermezzo zult u ook enige routines tegenkomen die een index voor de werknemerdatabase realiseren. Deze routines omvatten een sorteersubroutine die de index op alfabet rangschikt en een zoekfunctie die bepaalde elementen in de gesorteerde index opzoekt. Het programma kan dan via deze index de werknemerrecords lokaliseren ongeacht de volgorde waarin de records zijn opgeslagen.

Waarom een random-access file?

In tegenstelling tot sequentiële files kunnen random-access files tegelijkertijd voor lezen en schrijven worden geopend. Het is niet nodig om deze files apart voor lezen of schrijven te openen – of u nu records uit een file gaat lezen of records naar de file wegschrijft of beide bewerkingen in hetzelfde programma wilt uitvoeren. Ik begin met de syntax van het `OPEN` statement voor random-access files.

Het `OPEN` statement.

Onderstaande structuur van het statement laat zien hoe het een random-access file opent en hoe de recordlengte van de file gespecificeerd wordt:

```
OPEN "filenaam" FOR RANDOM AS #filenummer LEN = recordlengte
```

De filenaam mag een string zijn die correspondeert met de MS-DOS regels voor filenamen: een basisnaam van maximaal 8 karakters en een optionele extensie van maximaal 3 karakters. In de MS-DOS tijd moesten we ons aan die lengte van 8 karakters houden. Tegenwoordig is dat niet meer nodig. Tevens mag een filenaam voorafgaan door drive- en padspecificaties.
Aangezien QuickBASIC toestaat om tijdens de programma-uitvoering meer files tegelijkertijd te openen, dient een geopende file door een filenummer te worden vertegenwoordigd. Na `OPEN` zullen daaropvolgende statements via het filenummer naar de desbetreffende file verwijzen. De statements `GET#` en `PUT#` zullen een bepaalde file herkennen aan het filenummer.
Tenslotte duidt `LEN` in het `OPEN` statement de recordlengte van de file in kwestie aan. De lengte wordt in karakters of bytes opgegeven. Dit getal moet gelijk zijn aan de totale lengte van alle velden in een record. Indien `LEN` wordt weggelaten, is de standaardwaarde van de recordlengte 128 karakters. De standaardlengte was er toen in die tijd van QuickBASIC; of dat nu nog zo is kunt u zelf uitproberen door de lengte niet op te geven.
Het volgende `OPEN` statement is afkomstig uit het Werknemer programma:

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

Nogmaals, tegenwoordig is de maximumlengte van de filenaam niet meer 8 karakters. U kunt de DAT file dus ook gewoon Werknemer noemen.
Dit statement opent de random-access file WERKNMER.DAT en wijst deze filenummer 1 toe. In dit geval is *openRecord* een recordvariabele die de structuur van de file definieert. De rol van recordvariabelen komt later nog ter sprake. Tevens wijs ik erop dat de QuickBASIC functie `LEN` de totale vaste lengte van alle elementen van deze gebruikergedefinieerde structuur levert. Deze nieuwe toepassing van de `LEN` functie is vanaf QuickBASIC 4.0 beschikbaar.
Om dit `OPEN` statement uit te voeren is het geen voorwaarde dat WERKNMER.DAT reeds bestaat. Indien de file bestaat, bereidt QuickBASIC zich voor op een lees- of schrijfopera-

tie. In het tegenovergestelde geval wordt de file gecreëerd, hoewel deze leeg blijft totdat er records naartoe worden geschreven. Tengevolge van het OPEN statement reserveert QuickBASIC een geheugengebied waarin de records op weg naar of afkomstig van de file worden opgeslagen. Dit geheugengebied wordt een buffer genoemd.

Recordstructuren: het TYPE statement.

Met dit statement kunnen we eigen type structuren maken en op die manier BASIC uitbreiden met zelfgemaakte structuren. Het is het TYPE statement dat we nodig hebben om een recordstructuur te maken. De syntax van het TYPE statement is als volgt:

```
TYPE typeNaam
   elementNaam1 AS type
   elementNaam2 AS type
   elementNaam3 AS type
   ...
END TYPE
```

Het type voor een element kan met één van de volgende gereserveerde woorden worden gespecificeerd: INTEGER, SINGLE, DOUBLE of STRING. Alleen strings met een vaste lengte zijn toegestaan als element.
Al naar gelang het gebruikergedefinieerde type beschikt QuickBASIC over diverse statements voor de definitie van de bijbehorende recordvariabele. Deze statements omvatten DIM, REDIM, COMMON, STATIC en SHARED. In het volgende voorbeeld wordt een recordvariabele door middel van het DIM statement gedefinieerd:

```
DIM recordNaam AS typeNaam
```

Een recordvariabele slaat per element één datawaarde op. Een recordelement wordt geadresseerd door in de notatie zowel de naam van de recordvariabele als de naam van het element, zoals dit in het originele TYPE statement is gedefinieerd, op te geven. De twee namen worden door een punt van elkaar gescheiden, bijvoorbeeld: recordNaam.elementNaam

Het doel van een gebruikergedefinieerde type in een database applicatie is om de vaste lengte van een record in een random-access file in benoemde velden te verdelen. Het volgende voorbeeld is het recordtype dat in het Werknemer programma voor records wordt gebruikt:

```
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
```

Deze definitie specificeert dus negen verschillende velden, waaronder acht strings met een vaste lengte en één numerieke enkele precisiewaarde. Het volgende `DIM` statement wijst de recordvariabele `werknRec` aan dit type toe:

```
DIM werknRec AS werknType
```

Vanaf QuickBASIC 4.0 kunnen de random-access statements de structuur van een file via een dergelijke variabele adresseren. Oudere versies van QuickBASIC kennen alleen het `FIELD` statement om de recordstructuur van een random-access file te definiëren. Uit een oogpunt van comptabiliteit ondersteunt QuickBASIC het `FIELD` statement nog wel, maar nu de gebruikergedefinieerde typen beschikbaar zijn, wordt het gebruik van `FIELD` niet langer aanbevolen. Het `FIELD` statement definieert speciale veldvariabelen die overeenkomen met een bepaalde structuur. Indien u in één van de oudere QuickBASIC versies met random-access files heeft gewerkt, zult u zich wel de omslachtige operaties en conversies herinneren om met deze veldvariabelen te kunnen werken. De variabelen in een `FIELD` statement moeten altijd strings zijn. Voor numerieke waarden vereist dat een aantal speciale bewerkingen door middel van de `MKI$`, `MKL$`, `MKS$` en `MKD$` functies om strings met een vaste lengte te produceren, en vervolgens de `CVI`, `CVL`, `CVS` en `CVD` functies om deze formaten weer in getallen om te zetten. Bovendien moeten de waarden via één of twee bepaalde statements, `LSET` of `RSET` aan de FIELD variabelen worden toegewezen om de structuur van de file te handhaven.
Deze bewerkingen behoren vanaf QuickBASIC 4.0 tot het verleden. De gebruikergedefinieerde typen en recordvariabelen verschaffen een aantal duidelijke voordelen boven de oude FIELD benadering. Vooral het feit dat de elementen van een recordvariabele tot de elementaire datatypen kunnen behoren die in QuickBASIC beschikbaar zijn, is een grote stap voorwaarts. Dankzij een recordvariabele kunt u zowel string gegevens als numerieke gegevens rechtstreeks naar een random-access file schrijven. In het geval van een numeriek veld berekent QuickBASIC per definitie een formaat met vaste lengte om de waarde in de file op te slaan, zonder speciale conversiefuncties. Tenslotte kunt u de waarden door middel van standaardtoekenningsopdrachten in een recordvariabele opslaan; `LSET` en `RSET` zijn hierbij overbodig.
De QuickBASIC statements `GET#` en `PUT#` kunnen rechtstreeks op recordvariabelen worden uitgevoerd om records in een random-access file te lezen en weg te schrijven. De volgende informatie met voorbeeldstukken zal gaan over hoe het leesproces werkt.

Lezen van records uit een random-access file.

Het formaat van het `GET#` statement luidt als volgt:

```
GET #fileNummer, recordNummer, variabeleNaam
```

Het filenummer moet overeenkomen met het nummer dat de file is toegewezen in het bijbehorende `OPEN` statement. Het recordnummer, een integer, specificeert de positie van het record dat uit de file moet worden gelezen. De variabelenaam is een vooraf gedefinieerde recordvariabele waarin de recordvelden worden opgeslagen die het resultaat zijn van de leesoperatie.
Na een `GET#` statement zijn de veldgegevens rechtstreeks toegankelijk via de elementen van de gespecificeerde recordvariabele. Bijvoorbeeld, de volgende statements gebruiken een recordvariabele die overeenkomt met de `werknType` structuur die eerder werd gedefinieerd:

```
DIM werknRec AS werknType
GET #1, 12, werknRec
PRINT werknRec.achterNaam, werknRec.voorNaam
```

Hier wordt record 12 gelezen uit de file die geopend is via buffer 1. De data worden opgeslagen in recordvariabele `werknRec`. Het `PRINT` statement geeft twee velden van de record op het scherm weer. Tijdens dit proces ondergaat de recordvariabele de volgende bewerkingen:

1. Het `DIM` statement creëert de recordvariabele conform de structuur van het gebruikergedefinieerde type.
2. Het `GET#` statement leest een record uit de file en wijst deze aan de recordvariabele toe.
3. Het programma heeft directe toegang tot de veldgegevens via de elementnamen van de recordvariabele.

Indien een `GET#` statement een recordnummer tracht te lezen dat groter is dan het totale aantal records in de file, is het resultaat onvoorspelbaar. Om deze situatie te vermijden moet het programma op een of andere manier het aantal records in een random-access file kunnen bepalen. Hierin voorziet QuickBASIC in de vorm van de ingebouwde `LOF` functie (Length Of File). De functie retourneert de totale lengte in bytes van een geopende file. Door deze waarde door de recordlengte te delen, wordt het aantal records in de file verkregen. Er wordt één argument gebruikt, het filenummer:

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

De uitdrukking aan de rechterkant van de = operator bepaalt het aantal records in file 1. Zoals u ziet levert de `LEN` functie de recordlengte. De expressie kent het resultaat aan variabele `totWerkns%` toe (totaal werknemers).
Wanneer het aantal records in een random-access file bekend is, kunnen de records door middel van eenvoudige lussen worden bewerkt. Onderstaande lus leest één record per iteratie vanaf het begin tot het einde van de geopende file, waarbij de waarde van `totWerkns%` de eindwaarde van de lus vertegenwoordigt:

```
FOR i% = 1 TO totWerkns%
   GET #1, i%, werknRec
```

```
   DisplayRec werknRec
NEXT i%
```

Wanneer een record in de variabele werknRec is ingelezen, stuurt de lus de record ter verdere bewerking naar de subroutine `DisplayRec`.
Kort samengevat bestaat het lezen van records uit een random-access file uit de volgende operaties:

1. Creëren van een recordvariabele die de filestructuur vertegenwoordigt, `TYPE` en `DIM`.
2. Openen van de file en specificeren van een recordlengte die gelijk is aan de lengte van de gedefinieerde recordvariabele, `OPEN` en `LEN`.
3. Bepalen van het aantal records in de file, `LOF`, echter is die functie optioneel.
4. Een specifiek record uit de file in de recordvariabele inlezen, `GET#`.
5. Adresseren van de veldgegevens via de elementnamen van de recordvariabele.

QuickBASIC accepteert meer dan één veldstructuur voor een random-access file, zodat recordgegevens op verschillende manieren kunnen worden gelezen. Hierbij kunt u denken aan een subprogramma dat alleen de eerste twee velden van de werknemerdatabase leest, `achterNaam` en `voorNaam` en een tabel met deze twee velden afdrukt. Het volgende gebruikergedefinieerde type komt tegemoet aan de eisen van dit bepaalde subprogramma:

```
TYPE altType
   geheleNaam AS STRING * 26
   garbage AS STRING * 66
END TYPE
```

De lengte van het `geheleNaam` element is gelijk aan de som van de lengten van de eerste twee velden in de werknemersdatabase. Derhalve bevat het element twee informatie-items: de achternaam en de voornaam van de werknemer. Het `garbage` element (garbage = afval) krijgt de resterende karakters van een record toegewezen, die in dit bepaalde subprogramma niet worden gebruikt.
Aan de hand van deze tweede recordstructuur kan het programma een recordvariabele creëren om de individuele records beurtelings uit de file te lezen. De volgende regels creëren een dergelijke variabele en gebruiken deze vervolgens om de werknemernamen naar het scherm te sturen:

```
DIM altRecord AS altType
OPEN "WERKNMER.DAT" FOR RANDOM AS #1 LEN = LEN(altRecord)
aantalRecords% = LOF(1) / LEN(altRecord)
FOR i% = 1 TO aantalRecords%
   GET #1, i%, altRecord
   PRINT altRecord.geheleNaam
NEXT i%
CLOSE#1
```

Bedenk wel dat deze tweede recordstructuur naast de eerste kan bestaan. De verschillende

programma onderdelen kunnen derhalve terugvallen op de recordstructuur die voor een bepaalde situatie is vereist.
Records worden naar een random-access file geschreven door middel van het `PUT#` statement, waarbij een recordvariabele fungeert als het medium om de uitgaande records in op te slaan.

Records naar een random-access file schrijven.

Het `PUT#` statement schrijft een record via een variabele naar een geopende random-access file. De structuur ziet er als volgt uit:

```
PUT #filenummer, recordnummer, variabelenaam
```

Onderstaande regels wijzen gegevens aan vier elementen van recordvariabele `werknRecord` toe, waarna de gegevens als het tiende record naar de file worden geschreven:

```
DIM werknRecord AS werknType
OPEN "WERKNMER.DAT" FOR RANDOM AS #1 LEN = LEN(werknRecord)
werknRecord.achterNaam = "Smit"
werknRecord.voorNaam = "Paul"
werknRecord.beginDatum = DATE$
werknRecord.functie = "Verkoopleider"
PUT #1, 10, werknRecord
```

Kort samengevat komt het schrijven van records naar een random-access file op de volgende bewerkingen neer:

1. Creëren van een recordvariabele, waarvan de elementen de velden van de file vertegenwoordigen, `TYPE` en `DIM`.
2. Waarden toewijzen aan de desbetreffende elementen van de recordvariabele.
3. De record naar een bepaalde locatie in de file schrijven, `PUT#`.

Volgens deze procedure kan zowel een nieuw record naar een file worden geschreven als de inhoud van een reeds bestaand record worden gewijzigd. Het Werknemer programma bevat voorbeelden van beide operaties.
Het `CLOSE` statement sluit een geopende file, zoals we eerder zagen. Dit statement kent verschillende formaten, bijvoorbeeld het sluiten van file #1: `CLOSE #1`
Wanneer deze file is gesloten, kan het programma filenummer 1 gebruiken om een andere file te openen. Het filenummer behoort namelijk alleen bij de file zolang deze is geopend.
Het volgende `CLOSE` statement sluit twee files, #1 en #3: `CLOSE #1, #3`. Het statement gebruiken zonder een filenummer op te geven zorgt ervoor dat alle geopende files worden gesloten.

Nu we weten hoe de statements en functies voor de manipulatie van random-access files in QuickBASIC werken, wordt het tijd voor de praktijk en wel in de vorm van het Werknemer

programma. In de volgende nieuwsbrief komt het werken met records ter sprake en kunt u zien hoe in QuickBASIC een menu op het scherm wordt weergegeven en hoe de besturing werkt.

Marco Kurvers

Grafisch programmeren in GW-BASIC (2).

In nieuwsbrief nr 4 vorig jaar liet ik u zien dat ondanks de grafische mogelijkheden die GW-BASIC heeft, drie belangrijke onderwerpen nodig zijn om goed te kunnen tekenen,

- de juiste schaal;
- de correctiefactor;
- de formule die bij de juiste schaal en correctiefactor nodig is.

Nu kunnen we nog steeds GW-BASIC code gebruiken, alleen in Visual Basic .NET is het onmogelijk en moeten we gebruik maken van de grafische `e` parameter in de `Paint` event.

Programmastructuur en structuurdiagram.

Hier nogmaals de aangegeven variabelen die in de programma's worden gebruikt. Door steeds weer dezelfde te nemen, hoeven we ook niet in de war te raken. Dit zal overigens de laatste keer zijn dat ik onderstaande tabellen nogmaals laat zien. Later kunt u het terugvinden op deze nieuwsbrieven (nr 4, jaar 2009; nr 1, jaar 2010) mocht u de omschrijvingen niet meer weten. U kunt ook mij vragen om een kopie van onderstaande tabellen.

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 helen
K	de vermenigvuldigingsfactor voor functiewaarden
W,W1	hoeken bij trigonometrische functies
RD	het getal $\pi/180$ voor het omrekenen van graden in radialen

Een grof structuurdiagram voor de programma's ziet er zo uit:

begin programma	
graphic-stand kiezen	
variabelen U, V, H, K, RD, enz. vastleggen	

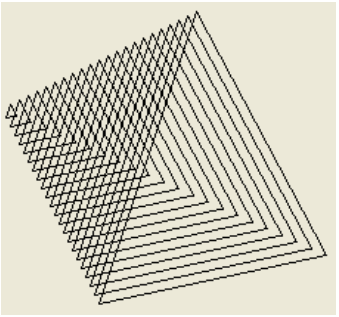
punt P1(X1, Y1) vastleggen	
	doe zolang nodig nieuw punt P2(X2, Y2) berekenen verbind P1 met P2 punt P2 wordt punt P1 (X1=X2 Y1=Y2)
einde programma	

De voorbeelden

In nieuwsbrief nr 4 vorig jaar kwam ik gelijk al met twee mooie voorbeelden: het diagonaalweb en het moiree-effect.

Programma 3 tekent een reeks driehoeken in perspectief. Het 'centrum van vermenigvuldiging' heeft de coördinaten S(0, 72). De drie hoekpunten van de driehoek die vermenigvuldigd wordt zijn P1(6, 84), P2(20, 81) en P3(12, 66). Alle coördinaten zijn beeldschermcoordinaten (oorsprong links bovenaan). De drie richtingen van vermenigvuldiging (richtingsvectoren) zijn SP_1, SP_2 en SP_3. Deze worden in het programma als SX(1);SY(1), SX(2);SY(2) en SX(3);SY(3) vastgelegd. De vermenigvuldigingsfactor K is de lusvariabele van de buitenste FOR-NEXT lus. K loopt van 0 tot 11 in stappen van 0,5. Hieronder zien we het programma met daarnaast het resultaat van het programma.

```
100 ' programma 3     DRIEHOEKEN IN PERSPECTIEF
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 DIM X(3), Y(3), SX(3), SY(3)
150 X(0)=0: Y(0)=72
160 SX(1)= 6: SX(2)=20: SX(3)=12
170 SY(1)=12: SY(2)= 9: SY(3)=-6
180 FOR K=0 TO 11 STEP .5
190    FOR J=1 TO 3
200       X(J)=X(0)+K*SX(J)
210       Y(J)=Y(0)+K*SY(J)
220    NEXT J
230    LINE (FNX(X(1)),Y(1))-(FNX(X(2)),Y(2)),1
240    LINE (FNX(X(2)),Y(2))-(FNX(X(3)),Y(3)),1
250    LINE (FNX(X(3)),Y(3))-(FNX(X(1)),Y(1)),1
260 NEXT K
270 A$=INKEY$: IF A$="" THEN 270
280 CLS: KEY ON: END
```

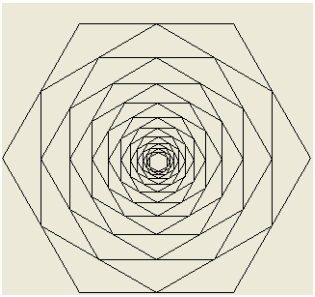


```
Dim X(3), Y(3), SX(3), SY(3) As Single
X(0) = 0 : Y(0) = 72
SX(1) = 6 : SX(2) = 20 : SX(3) = 12
SY(1) = 12 : SY(2) = 9 : SY(3) = -6
For K As Single = 0 To 11 Step 0.5
   For J As Integer = 1 To 3
      X(J) = X(0) + K * SX(J)
      Y(J) = Y(0) + K * SY(J)
   Next
   With e.Graphics
      .DrawLine(Pens.Black, X(1), Y(1), X(2), Y(2))
      .DrawLine(Pens.Black, X(2), Y(2), X(3), Y(3))
      .DrawLine(Pens.Black, X(3), Y(3), X(1), Y(1))
   End With
```

```
Next
```

Programma 4 tekent een reeks ingeschreven regelmatige zeshoeken. Behalve bij de eerste zeshoek, liggen alle zes de hoeken van elke zeshoek precies in het midden van een zijde van de omschreven zeshoek. De coördinaten van de hoekpunten van de eerste (buitenste) zeshoek worden met trigoniometrische functies (sinus- en cosinusfunctie) berekend. De hoek die hierbij als argument van de functies gebruikt wordt, doorloopt de waarden 0°, 60°, 120°, 180°, 240°, 300° en 360°. De middelpunten van de zijden van een zeshoek worden aangegeven met MX(0);MY(0) t/m MX(5);MY(5). Dit worden de hoekpunten van de volgende zeshoek.

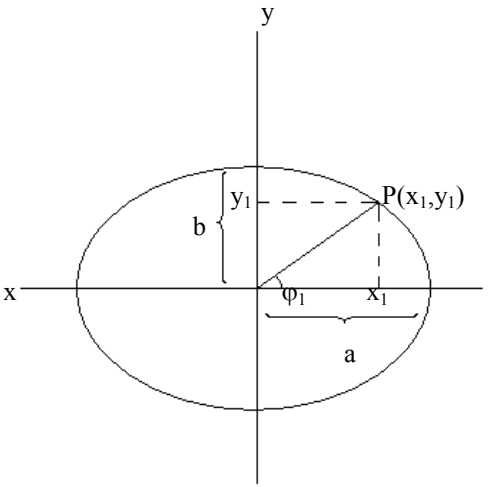
```
100 '       programma 4       INGESCHREVEN ZESHOEKEN
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 DIM X(6),Y(6),MX(6),MY(6)
150 R=160: U=160: V=160: H=.5: W=60*4*ATN(1)/180
160 FOR J=0 TO 6
170    W1=J*W
180    X(J)=INT(U+R*COS(W1)+H)
190    Y(J)=INT(V-R*SIN(W1)+H)
200 NEXT J
210 FOR N=1 TO 20
220    FOR J=0 TO 5
230       LINE (FNX(X(J)),Y(J))-(FNX(X(J+1)),Y(J+1)),1
240    NEXT J
250    FOR K=0 TO 5
260       MX(K)=INT((X(K)+X(K+1))/2+H)
270       MY(K)=INT((Y(K)+Y(K+1))/2+H)
280    NEXT K
290    MX(6)=MX(0) : MY(6)=MY(0)
300    FOR J=0 TO 6
310       X(J)=MX(J) : Y(J)=MY(J)
320    NEXT J
330 NEXT N
340 A$=INKEY$: IF A$="" THEN 340
350 CLS: KEY ON: END
```



```
Dim X(6), Y(6), MX(6), MY(6) As Single
Dim R As Single = 160, U As Single = 160, V As Single = 160
Dim H As Single = 0.5, W As Single = 60 * 4 * Math.Atan(1) / 180
For J As Integer = 0 To 6
   Dim W1 As Single = J * W
   X(J) = Int(U + R * Math.Cos(W1) + H)
   Y(J) = Int(V - R * Math.Sin(W1) + H)
Next
For N As Integer = 1 To 20
   For J As Integer = 0 To 5
      e.Graphics.DrawLine(Pens.Black, X(J), Y(J), X(J + 1), Y(J + 1))
   Next
   For K As Integer = 0 To 5
      MX(K) = Int((X(K) + X(K + 1)) / 2 + H)
      MY(K) = Int((Y(K) + Y(K + 1)) / 2 + H)
   Next
   MX(6) = MX(0) : MY(6) = MY(0)
   For J As Integer = 0 To 6
      X(J) = MX(J) : Y(J) = MY(J)
   Next
Next
```

Programma 5 tekent alle diagonalen in een n-hoek. De n hoekpunten liggen op de omtrek van een ellips of van een cirkel afhankelijk van de waarden die we in het begin van het programma voor de halve lange as a en halve kleine as b kiezen. Voor de berekening van de coördinaten van de punten op de omtrek van de ellips gebruiken we niet de (cartesische) vergelijking $(x^2/a^2) + (y^2/b^2) = 1$, maar de parametervorm $x = a.\cos\varphi$ en $y = b.\sin\varphi$.

Als de parameter π loopt van 0^0 tot 360^0, dan beschrijft het daaraan toegevoegde punt P(x,y), met $x = a\cos\varphi$ en $y = b\sin\varphi$, precies de omtrek van een ellips. We zien dat een bepaald punt $P(x_1,y_1)$ op de omtrek van de ellips met de daarbij horende waarde van de parameter φ_1. Voor dat punt geldt: $x_1 = a\cos\varphi_1$ en $y_1 = b\sin\varphi_1$.



Voor het berekenen van de coördinaten van het j^{de} hoekpunt (X(J),Y(J)) verdelen we de maximale middelpuntshoek van 360^0 in n gelijke hoeken van elk 360/n graden.

Als A de lengte van de halve lange as is èn
B de lengte van de halve korte as èn
N het aantal hoekpunten van de n-hoek èn
W gelijk is aan 360/N èn
W1 de hoek die hoort bij het j^{de} hoekpunt èn
X(J) de x-coördinaat van het j^{de} hoekpunt t.o.v. het middelpunt van de ellips èn
Y(J) de y-coördinaat van het j^{de} hoekpunt t.o.v. het middelpunt van de ellips

dan zou je in een BASIC programma de coördinaten X(J) en Y(J) als volgt kunnen berekenen:

W = 360/N : W1 = J*W
X(J) = INT(A*COS(W1))
Y(J) = INT(B*SIN(W1))

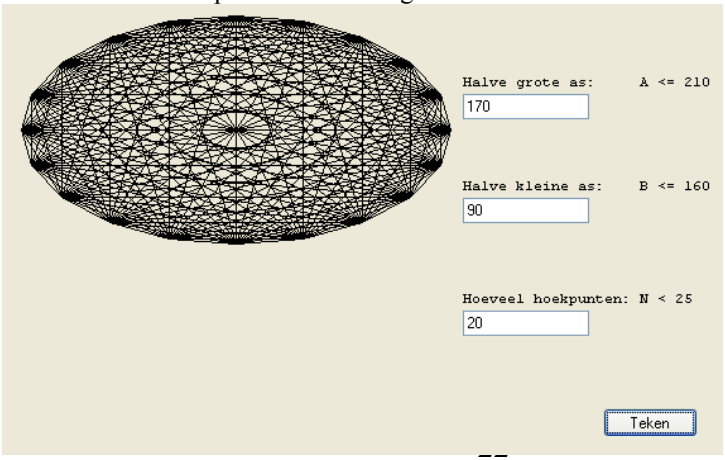
Als we dit doen maken we twee fouten. De eerste fout is dat de hoek W1, als argument van de COSinus- en de SINusfunctie, in graden is uitgedrukt, terwijl BASIC vereist dat het argument van deze functies in radialen wordt uitgedrukt. De tweede fout is dat het graphic-systeem van de computer de oorsprong van het coördinatenstelsel voor het scherm in de linkerbovenhoek van het scherm verwacht en niet in het middelpunt van de ellips. Wat we dus nog moeten doen is:

a. bereken W1 in radialen èn
b. transformeer X(J) en Y(J) naar beeldschermcoördinaten.

We weten dat een hoek van 360^0 overeenkomt met 2π radialen, waarin $\pi = 3{,}14159.....$. Dit betekent dat een hoek van 1 graad overeenkomt met een hoek van $2\pi/360$ of $\pi/180$ radialen. De transformatieformules voor de transformatie van 'onze' coördinaten naar beeld-schermcoördinaten ziet u in het structuurdiagram op pagina 15 van deze nieuwsbrief. De BASIC code voor het berekenen van de beeldschermcoördinaten van het j^{de} hoekpunt van de n-hoek worden nu:

W = (360/N)*π/180 : W1 = J*W
X(J) = INT(U + A*COS(W1) + H)
Y(J) = INT(V – B*SIN(W1) + H)

Voor π gebruiken we in de programma's dikwijls de variabele PI die als waarde krijgt 4*ATN(1). ATN is de inverse tangensfunctie. Er geldt dat $\tan(45^0)=1$, dus $\tan(\pi/4)=1$. Dit betekent dat de inverse tangens van 1 gelijk is aan $\pi/4$, dus $\text{atan}(1) = \pi/4$, hetgeen tot gevolg heeft dat $\pi = 4*\text{atan}(1)$; in BASIC: PI=4*ATN(1). We hadden ook PI=3.14159 kunnen nemen.
In het navolgende programma komen we bovenstaande opdrachten tegen. Kiezen we voor A en B dezelfde waarden, dan krijgen we een regelmatige n-hoek met al zijn diagonalen. Bovendien is de ellips dan een cirkel geworden.



Het programma maakt gebruik van het `INPUT` statement om de waarden A, B en N in te kunnen voeren. Voor Visual Basic zijn echter invoervak-ken nodig, omdat het `INPUT` statement alleen nog maar voor bestanden gebruikt wordt. De onder-staande knop 'Teken'

heb ik geplaatst om het formulier te kunnen vernieuwen voordat een nieuwe tekening gemaakt wordt. Wilt u echter niet een formulier hebben zoals het voorbeeld, maak dan gebruik van de functie `InputBox`. Met die functie kunt u het `INPUT` statement nabootsen en hoeft u niet uitgebreid een formulier te maken. Zorg er voor dat u de functie gebruikt in de `Paint` event.
Het voordeel van een formulier zoals het voorbeeld is dat u niet steeds het programma hoeft te starten als u nieuwe waarden in wilt geven. Dankzij de knop kunt u telkens weer een nieuwe tekenen.

```
100 '       programma 5       DIAGONALEN IN EEN N-HOEK
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "HALVE GROTE AS      A <= 210 "; A : PRINT
150 INPUT "HALVE KLEINE AS     B <= 160 "; B : PRINT
160 INPUT "HOEVEEL HOEKPUNTEN  N <  25  "; N
170 CLS
180 DIM X(N), Y(N)
190 U=210: V=160: H=.5: W=(360/N)*4*ATN(1)/180
200 FOR J=0 TO N-1
210    W1=J*W
220    X(J)=INT(U+A*COS(W1)+H)
230    Y(J)=INT(V-B*SIN(W1)+H)
240 NEXT J
250 FOR I=0 TO N-2
260    FOR J=I+1 TO N-1
270       LINE (FNX(X(I)),Y(I))-(FNX(X(J)),Y(J)),1
280    NEXT J
290 NEXT I
300 A$=INKEY$: IF A$="" THEN 300
310 CLS: KEY ON: END
```

```
Dim X(Val(N.Text())), Y(Val(N.Text())) As Single
Dim U As Single = 210, V As Single = 160, H As Single = 0.5
Dim W As Single = (360 / Val(N.Text())) * 4 * Math.Atan(1) / 180
For J As Integer = 0 To Val(N.Text()) - 1
   Dim W1 As Single = J * W
   X(J) = Int(U + Val(A.Text()) * Math.Cos(W1) + H)
   Y(J) = Int(V - Val(B.Text()) * Math.Sin(W1) + H)
Next
For I As Integer = 0 To Val(N.Text()) - 2
   For J As Integer = I + 1 To Val(N.Text()) - 1
      e.Graphics.DrawLine(Pens.Black, X(I), Y(I), X(J), Y(J))
   Next
Next
'
'Onderstaande code niet vergeten als u gebruik maakt van bovenstaand formulier-
'voorbeeld.
'
Private Sub btnTeken_Click(...)
   Me.Refresh()
End Sub
```

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

Appendix conversietabellen.

Deze appendix laat drie conversietabellen zien van verschillende BASIC types:

- string functies;
- rekenkundige en logische functies;
- statements.

De tabellen kunnen nuttig zijn om programma's in meerdere soorten BASIC types te kunnen programmeren. Ik heb geprobeerd zoveel mogelijke functies op te nemen en de overeenkomstige functies daarvan aan te geven.

Tabel 1: string functies

string-functies	BBC-Computer	Commodore 64 BASIC 2.0	APPLE II	DAI	EXIDY SORC.	PET/CBM	P2000	TRS80	ZX81	Commodore VIC 20	ACORN ATOM	Commodore 128 BASIC 7.0	Commodore 64 Simon's BASIC	QuickBASIC	GW-BASIC	PowerBASIC	Visual Basic 6.0	Liberty Basic	overeenkomstige string-functies
$ (n)											X								CHR$(n)
ASC(str)	X	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	CH
CH											X								ASC(str)
CHR$(n)	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	$ (n)
CODE									X										
CODEPTR n																X			VARPTR n
CODESEG																X			VARSEG
FRE("")		X					X												FRE(n$)
FRE(n$)					X			X											FRE("")
GET	X	X	X	X		X				X		X	X						INKEY$
INKEY$	X							X	X					X	X	X			GET
INPUT$(n)																X			
INSTR(n,kar)												X		X	X	X	X	X	
LEFT$(str,n)	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	

LEN(str)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
MID$(str,m,n)	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	
RIGHT$(str,n)	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	
slicing									X									geeft LEFT$, MID$ en RIGHT$
STR$(n)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
STRING$(n,kar)	X						X	X							X	X	X	
VAL(str)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
VARPTR n				X			X	X						X	X	X	X	CODEPTR
VARSEG				X			X	X								X		CODESEG

Tabel 2: Rekenkundige en logische functies.

functies	BBC-Computer	Commodore 64 BASIC 2.0	APPLE II	DAI	EXIDY SORC.	PET/CBM	P2000	TRS80	ZX81	Commodore VIC 20	ACORN ATOM	Commodore 128 BASIC 7.0	Commodore 64 Simon's BASIC	QuickBASIC	GW-BASIC	PowerBASIC	Visual Basic 6.0	Liberty Basic	overeenkomstige functies
& n	X										X								HEX$(n)
%											X								CINT, FIX
?FRE(0)			X																FRE
ABS	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ACS	X								X										
AND	X		X	X	X	X	X	X	X		X								
ASN	X								X										
ATN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
CALLM...,(x)				X															LINK n, USR(x)
CDBL							X	X									X		
CINT							X	X									X		%, FIX
COS(n)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
CSGN							X	X											
CSRLIN														X	X	X			
CUR				X															POS
DEG	X										X								
EL												X							ERL, ERRLN
ER												X							ERR, ERRN, ERRTEST
ERL	X						X	X								X	X	X	EL, ERRLN
ERR	X						X	X								X	X	X	ER, ERRN, ERRTEST
ERRLN													X						EL, ERL
ERRN													X						ER, ERR, ERRTEST
ERRTEST																X			ER, ERR, ERRN
EXP n	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
FIX							X	X											%, CINT
FLT											X								
FRAC(n)				X									X			X			
FRE		X		X	X	X	X	X		X		X	X	X	X	X	X	X	?FRE(0)
HEX$(n)				X			X					X				X	X		& n
HTN											X								
INT	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	
LINK n											X								CALLM...,(x), USR(x)
LN	X								X										LOG

LOG	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	LN
LOGT				X															
MOD	X		X	X			X						X	X	X	X	X	X	x%y, x↑y
NOT			X	X	X	X	X	X	X										
OCT$(n)							X									X			
OR	X		X	X	X	X	X	X	X		X								
POS	X	X	X		X	X	X	X		X		X	X	X	X	X			CUR
RAD	X										X								
RND	X			X							X			X	X	X	X	X	RND(0), RND(neg)
RND%n											X								
RND(0)					X	X	X	X	X	X									RND(neg), RND
RND(n)		X					X	X				X	X						RND(pos)
RND(neg)			X																RND(0), RND
RND(pos)			X																RND(n)
SCRN(x,y)			X	X															
SGN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
SIN(n)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
SPC(m)	X	X	X	X	X		X					X	X	X	X	X	X	X	
SPC(n)						X				X		X	X	X	X	X	X	X	
SQR(n)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
SWAP							X												
TAN(n)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
USR(x)	X	X	X		X	X	X	X	X	X		X	X	X	X	X	X	X	CALLM...,(x), LINK n
x%y											X								MOD, x↑y
x↑y		X								X		X	X						MOD, x%y

Tabel 3: Statements.

statements	BBC-Computer	Commodore 64 BASIC 2.0	APPLE II	DAI	EXIDY SORC.	PET/CBM	P2000	TRS80	ZX81	Commodore VIC 20	ACORN ATOM	Commodore 128 BASIC 7.0	Commodore 64 Simon´s BASIC	QuickBASIC	GW-BASIC	PowerBASIC	Visual Basic 6.0	Liberty Basic	overeenkomstige statements
$											X								DEFSTR, IMPSTR
%											X								DEFDBL
'														X	X	X	X	X	REM
?m,n	X										X								POKE m,n
?n	X										X								IN(poort), IN#n, INP, INP(n)
AT									X				X			X			CURSOR m,n
BEEP x,y									X										SOUND
CALL	X		X										X	X		X	X	X	CALLM, SYS, LINK
CALLM				X															CALL, SYS, LINK
CASE														X		X	X	X	zie SELECT CASE
CIRCLE												X	X	X	X	X	X	X	
CLEAR n	X										X				X	X			MODE n
CLEAR 0	X										X								MODE 0, TEXT, GR
CLOAD*							X												INPUT#n,m, RECALL, LOADA, LOAD, GET A
CLOSE	X	X				X				X		X	X	X	X	X	X	X	SHUT
CLS	X							X	X					X	X	X			HOME, PRINT $12, SCNCLR

COLOR n											X	X		X	X	X			HCOLOR=n
COLOR=n			X																COLOUR n
COLOUR n	X												X						COLOR=n
COUNT	X																		INCR
COUNT n	X												X						TAB
CSAVE*							X												SAVE A, STORE
CURSOR m,n				X															AT
DATA	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	
DEF FNnaam(...)	X	X	X			X	X			X		X	X	X	X	X	X	X	FUNCTION naam(...) ... END FUNCTION
DEFDBL							X	X						X	X	X	X	X	%
DEFINT							X	X						X	X	X	X	X	IMPINT
DEFSNG							X	X						X	X	X	X	X	IMPFPT
DEFSTR								X						X	X	X	X	X	IMPSTR, $
DIM var(n)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
DO...UNTIL											X	X		X	X	X	X	X	REPEAT...UNTIL
DO WHILE...LOOP												X		X		X	X	X	WHILE...WEND
DOT				X															SET (x,y), PLOT, HPLOT, PLOT k,x,y, PSET (x,y)
DRAW	X			X								X	X			X			LINE
DRAWTO												X							LINE -
DSP n			X																
END	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	
ERROR	X						X	X								X			
EXT	X										X								
FILL				X									X						PAINT
FIN											X								OPEN
FOR...TO...STEP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
FUNCTION naam(...) ... END FUNCTION														X		X	X	X	DEF FNnaam(...)
GET	X	X								X	X	X	X			X		X	
GET A	X										X								INPUT#n,m, RECALL, LOADA, CLOAD*, LOAD
GOSUB n	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
GOTO n	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
GR			X																CLEAR 0
HCOLOR=n			X																COLOR n
HOME			X																CLS, PRINT $12, SCNCLR
HPLOT			X																DOT, PLOT, PLOT k,x,y, PSET (x,y)
HTAB(n)			X																
IF...GOSUB n	X		X	X	X	X	X	X			X								
IF...GOTO n	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	
IF...THEN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
IF...THEN...ELSE	X						X	X			X	X	X	X	X	X	X	X	
IF...THEN...END IF														X	X	X	X	X	IF...THEN BEGIN...BEND
IF...THEN BEGIN...BEND												X							IF...THEN...END IF
IMPFPT				X															DEFSNG
IMPINT				X															DEFINT
IMPSTR				X															DEFSTR, $
IN(poort)					X		X												IN#n, INP, INP(n), ?n
IN#n			X																IN(poort), INP, INP(n), ?n
INP				X												X			IN(poort), IN#n, INP(n), ?n
INP(n)								X											IN(poort), IN#n, INP, ?n
INPUT n	X	X							X	X		X	X	X	X	X			INPUT "...";n

INPUT "...";n	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X			INPUT n
INPUT#n,m		X			X	X		X		X		X	X	X	X	X	X	X	RECALL, LOADA, CLOAD*, LOAD, GET A
LET	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
LINE	X													X	X	X	X	X	DRAW
LINE -	X													X	X	X	X	X	DRAWTO
LINK											X								CALL, CALLM, SYS
LOAD	X								X		X								INPUT#n,m, RECALL, LOADA, CLOAD*, GET A
LOAD "...",n		X								X		X	X						
LOADA				X															INPUT#n,m, RECALL, CLOAD*, LOAD, GET A
LOCATE m,n												X		X	X	X	X	X	
LPRINT							X	X	X					X	X	X			PRINT $2, PRINT =
MID$(...)=n												X		X	X	X	X	X	
MODE n	X			X															CLEAR n
MODE 0				X															TEXT, CLEAR 0
MOVE	X										X		X						
NEXT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
ON ERROR GOTO n														X		X	X	X	ON...ERROR GOTO n, ONERR, TRAP n
ON...ERROR GOTO n	X										X								ON ERROR GOTO n, ONERR, TRAP n
ON...GOSUB n	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	
ON...GOTO n	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	
ONERR			X																ON ERROR GOTO n, ON...ERROR GOTO n, TRAP n
OPEN		X				X				X		X	X	X	X	X	X	X	FIN
OUT(poort,n)				X			X	X					X			X			PR#n, PRINT#n
PAUSE									X				X						SLEEP, WAIT
PDL			X	X															
PEEK(n)		X	X	X	X	X	X	X	X	X		X	X	X	X	X			PRINT ?n
PLOT			X						X										SET (x,y), HPLOT, DOT, PLOT k,x,y, PSET (x,y)
PLOT k,x,y	X										X		X						RESET (x,y), UNPLOT, SET (x,y), HPLOT, DOT
POINT	X							X								X			
POKE m,n		X	X	X	X	X	X	X	X	X		X	X			X			?m,n
PR#n			X				X	X											OUT(poort,n), PRINT#n
PRINT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
PRINT $12											X								CLS, HOME, SCNCLR
PRINT $2											X								PRINT =, LPRINT
PRINT =						X													SAVEA, STORE, LPRINT
PRINT ?n	X										X								PEEK(n)
PRINT @n								X											
PRINT USING							X	X				X	X	X		X			
PRINT#n	X	X				X				X		X	X	X	X	X	X	X	OUT(poort,n), PR#n
PROC ... END PROC	X												X						SUB ... END SUB
PSET (x,y)														X	X	X	X	X	SET (x,y), HPLOT, DOT, PLOT, PLOT k,x,y
PTR (n)	X										X								
PUT											X					X			
READ	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X			
RECALL			X																INPUT#n,m, LOADA, CLOAD*, LOAD, GET A
REM	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	'
RESET (x,y)								X											UNPLOT, PLOT k,x,y
RESTORE	X	X	X	X	X	X		X		X	X	X	X	X	X	X			RSTORE n

RESUME			X				X	X				X						
RETURN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
RSTORE n							X											RESTORE
SAVEA				X														STORE, PRINT =
SET (x,y)								X										PLOT, HPLOT, DOT, PLOT k,x,y, PSET (x,y)
SGET											X							
SLEEP												X				X		PAUSE, WAIT
SHUT											X							CLOSE
SOUND	X			X								X	X			X		BEEP x,y
SPEED			X															
STOP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
STORE			X															SAVEA, PRINT =
SYS		X								X		X	X		X			CALL, CALLM, LINK
TAB	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	COUNT n
TEXT			X										X					MODE 0, CLEAR 0
TOP											X							
UNPLOT									X									RESET (x,y), PLOT k,x,y
VLIN ... AT			X															
VTAB (n)			X															
WAIT		X	X	X	X	X	X			X	X	X	X	X	X	X		X	PAUSE, SLEEP

Hopelijk komen de meeste functies en statements met elkaar overeen en dat u er wat mee kunt doen. Mocht dat niet zo zijn, stuur dan een emailtje naar mij en geef mij de gegevens zodat ik de tabellen kan verbeteren. Bellen kan ook.

Marco Kurvers

Puzzel: de spion en de wachter.

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!!"

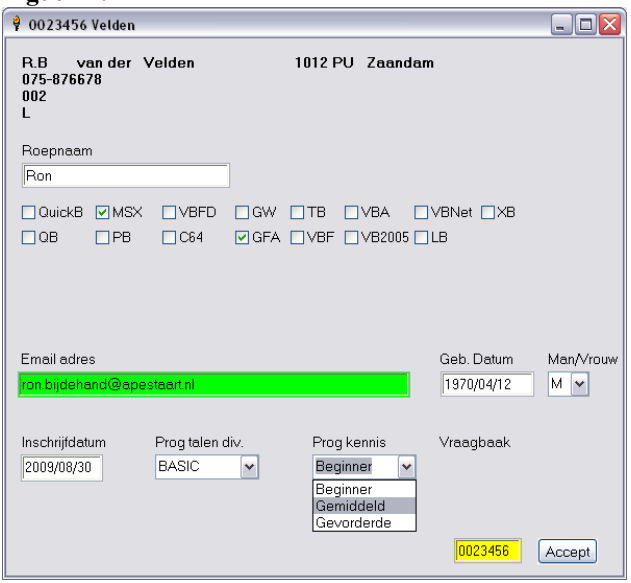
Vraag: "Wat had de spion moeten zeggen om ongehinderd naar binnen te mogen?"

Henk van Weers

BASIC cursus: Liberty BASIC (3)

Dit is het derde deel uit de serie. In dit deel zullen we een nieuw bestand maken met aanvullingen voor het reeds bestaande ledenbestand. Daartoe openen we een venster met gegevens invoermogelijkheden. In dit invoerscherm zijn ook duidelijk gegevens uit het ledenbestand zichtbaar. In het nieuw te maken bestand komen alleen de nieuwe extra gegevens en het **lidnummer** te staan. Het invoervenster ziet eruit als in figuur 1.

Figuur 1.



In de figuur valt op dat de titelomschrijving van het venster gevormd wordt door het **lidnummer** en de **naam** van het record dat aangevuld zal worden. De recordgegevens uit het ledenbestand zijn vetgedrukt weergegeven. De meeste invulvelden zijn opgezet om met de muis bediend te kunnen worden.

```
[list]
#main3.list1 "selectionindex? index"
    get #r,index
    goto [wijzigen1]
    Wait
```

Met bovenstaand stukje listing halen we het record met volgnummer index op. Dat is een "event" uit de listbox. Oké, we kunnen nu de opgehaalde gegevens, **naam$** enz. uit de velden, op het scherm plaatsen. Dat doen we in het volgende stuk.

```
[wijzigen1]
    statictext #main.st1, "Inschrijfdatum", 15, 380, 109, 20
    statictext #main.st2, "Vraagbaak", 416, 380, 109, 20
    textbox #main.tbx1, 14, 400, 79, 26
    TextboxColor$ = "green"
    textbox #main.tbx2, 11, 320, 377, 25
    statictext #main.st3, "Email adres", 14, 300, 109, 20
    TextboxColor$ = "white"
```

```
    textbox #main.tbx3, 417, 320, 90, 24
    statictext #main.st3a, "Roepnaam", 15, 100, 72, 20
    textbox #main.tbx3a, 15, 120, 200, 24
    statictext #main.st4, "Geb. Datum", 416, 300, 72, 20
    statictext #main.st4a, "Man/Vrouw", 520, 300, 72, 20
    combobox #main.cmb0, MV$(, [cmbox0], 520, 320, 40, 30
    combobox #main.cmb1, progT$(, [cmbox1], 143, 400, 100, 100
    statictext #main.st5, "", 15, 15, 550, 80
    combobox #main.cmb2, progK$(, [cmbox2], 294, 400, 100, 100
    statictext #main.st6, "Prog kennis", 294, 380, 72, 20
    statictext #main.st7, "Prog talen div.", 143, 380, 90, 20
    checkbox #main.cb1, "QuickB", [c1Set], [c1Reset], 15, 155, 68, 25
    checkbox #main.cb2, "QB", [c2Set], [c2Reset], 15, 179, 44, 25
    checkbox #main.cb3, "MSX", [c3Set], [c3Reset], 86, 155, 53, 25
    checkbox #main.cb4, "PB", [c4Set], [c4Reset], 86, 179, 43, 25
    checkbox #main.cb5, "VBFD", [c5Set], [c5Reset], 150, 155, 61, 25
    checkbox #main.cb6, "C64", [c6Set], [c6Reset], 150, 179, 48, 25
    checkbox #main.cb7, "GW", [c7Set], [c7Reset], 220, 155, 48, 25
    checkbox #main.cb8, "GFA", [c8Set], [c8Reset], 220, 179, 52, 25
    checkbox #main.cb9, "TB", [c9Set], [c9Reset], 273, 155, 43, 25
    checkbox #main.cb10, "VBF", [c10Set], [c10Reset], 273, 179, 51, 25
    checkbox #main.cb11, "VBA", [c11Set], [c11Reset], 325, 155, 52, 25
    checkbox #main.cb12, "VB2005", [c12Set], [c12Reset], 325, 179, 65, 25
    checkbox #main.cb13, "VBNet", [c13Set], [c13Reset], 392, 155, 64, 25
    checkbox #main.cb14, "LB", [c14Set], [c14Reset], 392, 179, 41, 25
    checkbox #main.cb15, "XB", [c15Set], [c15Reset], 456, 155, 42, 25
    TextboxColor$ = "yellow"
    textbox #main.tbx8, 430, 480, 65, 25
    button #main.button1, "Accept", [button1Click], UL, 510, 480
    TextboxColor$ = "white"

    progK$(1) = "Beginner"
    progK$(2) = "Gemiddeld"
    progK$(3) = "Gevorderde"

    progT$(1) = "Pascal"
    progT$(2) = "C++ C#"
    progT$(3) = "BASIC"

    MV$(1) = "M" : MV$(2) = "V"

Open LIDNUMMER$ + " " + NAAM$ For Window As #main
    #main "trapclose [quit]"
    #main "font ms_sans_serif 10"

    #main.cmb0 "select M"
    #main.cmb1 "select BASIC"
    #main.cmb2 "select Gemiddeld"

    #main.tbx1 date$("yyyy/mm/dd")

    display1$ = VOORLETT$ + VOORVOEG$ + NAAM$ + POSTC$ + "  " + PLAATS$ + chr$(13) + _
       TEL$ + chr$(13) + AFD$ + chr$(13) + LIDCODE$

    #main.st5 "!font ms_sans_serif 10 bold"
    #main.st5 display1$
    Wait

[quit]
    close #main
    Wait
```

De listing spreekt voor zichzelf, maar als je nog geen ervaring met het opzetten van vensters hebt kun je de vertaalde helpfile raadplegen: http://www.libertybasic.nl/listbox2.htm of je gaat via: http://www.libertybasic.nl/index1.html naar de helpbestand onderdelen. Natuurlijk kun je ook het FREEFORM programma gebruiken om het venster op te zetten.

Nu zijn de gegevens op het scherm ingevoerd. We gaan nu deze gegevens in een nieuwe file plaatsen. Zo halen we de invoer uit de schermvelden.
De checkboxen kunnen we op twee manieren bekijken. We kunnen de events van elke box (set en reset) apart bekijken, maar we kunnen ook gewoon de status van elke box opvragen. In het eerste geval staat in de listing zo iets dergelijks als:

```
[c4Set]
    Antw$ = "PB is geset"
    Wait
```

Ik kies voor de elegantere tweede methode, waarbij we alle checkboxen naar dezelfde labels laten verwijzen en alleen de status van de checkboxen opvragen. Bijvoorbeeld met:

```
checkbox #main.cb7, "GW", [Set], [Reset], 220, 155, 48, 25
checkbox #main.cb8, "GFA", [Set], [Reset], 220, 179, 52, 25
...
[Set]
    Wait
[Reset]
    Wait
```

Een goed moment om de status van alle checkboxen en de overige invoervelden op te nemen is na de event die gegenereerd wordt als er op de knop met opschrift "accept" gedrukt wordt. Nu lezen we de status van deze checkboxen af na de label van die knop.

```
[button1Click]
    #main.cb7 "value? a1$"
    #main.cb8 "value? a2$"
    ...
```

In `a1$` en `a2$` staan nu de woordjes "set" of "reset", afhankelijk van de status (vinkje of geen vinkje) van de checkboxen cb7 en cb8.

Om de ingevoerde data uit de comboboxen te lezen doen we iets in de trant van:

```
    #main.cb5 "contents? cb5invoer$"
    wait
```

Om de ingevoerde data uit de tekstboxen te lezen, bijvoorbeeld het opgegeven email adres, doen we iets in de trant van:

```
    #main.tbx3, "!contents? EMAIL$"
    wait
```

Let erop dat in het voorbeeld een uitroepteken extra staat, anders zou in de tekstbox gewoon de tekst `"contents? EMAIL$"` komen te staan. We willen hier echter geen tekst in de tekstbox plaatsen, maar een commando laten uitvoeren.

We gaan nu de data in het nieuwe bestand plaatsen, maar natuurlijk is het beter dit nieuwe bestand eerst na te lopen en daarin te zoeken naar een reeds bestaand record met het overeenkomstige lidnummer. We openen daarom dus eerst ons nieuwe bestand dat bestaat uit records met de volgende velden:

```
Field #r,
    7 As MENGNUMMER$, _
    60 As EMAIL$, _
    7 As BGGLID$, _
    2 As BAMAAND$, _
    1 As NIVEAU, _
    5 As VRAAGBAAK$, _
    5 As QUICK$, _
    5 As QB$, _
    5 As MSX$, _
    5 As PB$, _
    5 As VBFD$, _
    5 As C64$, _
    5 As GW$, _
    5 As GFA$, _
    5 As TB$, _
    5 As VBFW$, _
    5 As VBA$, _
    5 As VB2005$, _
    5 As VBNET$, _
    5 As LIBBAS$, _
    5 As XBASIC$, _
    40 As TALENDIV$, _
    40 As TIJDDAG$, _
    65 As FUNCT$, _
    141 As OPMERKING$, _
    20 As ROEPNAAM$, _
    1 As GESLACHT$, _
    10 As GEBDATUM$
```

Het veld waarin het "lidnummer" in onze nieuwe bestand staat heette oorspronkelijk MENGNUMMER en het ging om een numeriek veld van 7 tekens. Om allerlei problemen van het vergelijken van een string (reeks tekens) en een numerieke waarde van zeven tekens te voorkomen heb ik hier de numerieke variabele `mengnummer` gewijzigd in de stringvariabele `MENGNUMMER$`. We kunnen nu `LIDNUMMER$` uit het ene bestand vergelijken met alle `MENGNUMMER$` uit het tweede bestand, totdat we eventueel een match vinden. Bij een match kunnen we dan dat record overschrijven met de nieuwe aanvullende gegevens. Zoniet, dan moeten we het nieuwe record aan het einde van het nieuwe bestand toevoegen.

Ik schrijf toch nog een aflevering.
In het volgende deel doe ik de losse eindjes. Een werkend voorbeeld van de listing – onderdelen uit de voorgaande delen kun je nu al downloaden van de Liberty BASIC forumsite www.libertybasic.nl. Je moet daarvoor naar het subforum "Programmeren met Liberty BASIC 2009".

Gordon Rahman

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