Nieuwsbrief

16de jaargang maart 2009 Nummer 1

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Fred Luchsinger kwam met een	
bijdrage.	17

Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Willem Gobel	0118-850837	voorz@basic-gg.hcc.nl
Secretaris a.i.	Willem Gobel Heemskerckplein 27 4384 BD Vlissingen	0118-850837	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.hccnet.nl

Redactioneel

Al die jaren hebben we interessante nieuwsbrieven op papier gehad. U hebt er wat van kunnen leren, door nieuwe, mooie, onderwerpen over Basic te kunnen lezen. Helaas werd het steeds moeilijker om via papier door te gaan. We hebben geprobeerd om er wat tegen te doen om het makkelijker te maken, maar uiteindelijk moesten we wel overgaan in digitaal. Toch zal het voor de nieuwsbrief hetzelfde blijven. Steeds zal ik proberen duidelijk leesbare nieuwsbrieven voor elkaar te krijgen.

Marco Kurvers

VBA en de Excel werkbladen.

VBA is een BASIC dialect waarmee we de Office applicaties uit kunnen breiden, als doel documenten te ontwerpen met onderdelen die standaard niet in de applicaties aanwezig zijn. Ook VBA is object georiënteerd en dat betekent dat we moeten weten hoe we het document gaan opbouwen. Bovendien is er niet voor alles code nodig. Het is de bedoeling om zorgvuldig met VBA om te gaan en de rest aan de applicatie over te laten.

Een voorbeeld daarvan zijn de Excel werkbladen. De werkbladen kunnen in VBA ook als objecten dienen. Het nadeel is dat de bladen vrij toegankelijk in VBA aanwezig zijn en de methoden direct zijn aan te roepen. Dat probleem heeft Visual Basic 6.0 ook met de formulieren. Eigenlijk is VBA een deel van versie 6, want de syntaxis is precies hetzelfde. Het enige verschil is dat VBA alleen maar uit objecten van de applicatie, die u gestart hebt, bestaat.

Figuur 1

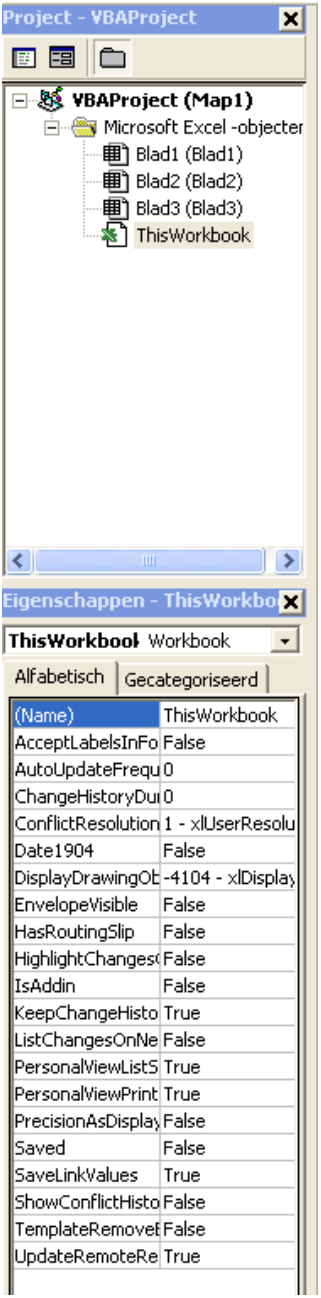
Links bij figuur 1 ziet u een deel van een object inspector. De aangegeven eigenschappen zijn van het eerste werkblad van Excel, Blad1. Het object is van het type `Worksheet`. De naam van het blad kan gewijzigd worden. Houd er dan wel rekening mee dat u de juiste identifiers in de code gebruikt en niet alles door elkaar haalt. Naarmate u steeds meer bladobjecten in code gaat gebruiken, kan dat leiden tot grote functionele gevolgen. U bent bijvoorbeeld van plan het derde blad te verwijderen, terwijl dat het tweede blad blijkt te zijn.

Door de werkbladen lokaal te gebruiken voorkomt u die problemen. In een sub of functie die u schrijft, declareert u bijvoorbeeld een lokaal blad van het type `Worksheet`. Ken dan het juiste blad toe aan de lokale instantie. Voer wat acties uit en vernietig de instantie zodra u met de acties klaar bent. Onderstaande code geeft een voorbeeld.

```
Private Sub Workbook_Open()
    Dim objBlad1 As Worksheet   'Instantie declaratie

    Set objBlad1 = Blad1
    objBlad1.Cells(1, 1) = 25 * 2 'Actie die lokaal uitgevoerd wordt
    Set objBlad1 = Nothing   'Instantie wordt weer opgeruimd.
End Sub
```

Figuur 2



Bij figuur 2 ziet u nogmaals een deel van een object inspector, maar nu met het project waarmee gewerkt wordt. Deze keer is er geen werkblad actief, maar het hoofdobject van alle werkbladen. In `ThisWorkbook` zal ook bovenstaande code in terecht komen. Nergens anders zal de gebeurtenis `Workbook_Open()` worden uitgevoerd. De geraamte van de subroutine zal ook automatisch worden aangemaakt zodra u het hoofdobject opent.

Door de code overal net zo te programmeren als bovenstaand voorbeeld laat zien, zal het functioneel beter werken. De kans op verkeerd gebruik van de werkbladen is stukken kleiner, doordat u lokaal in een subroutine of functie acties uitvoert. Het hergebruik van de code kan ook in VB6.0 en in VBA, ook al kennen ze overerving niet. U mag ook het werkblad direct gebruiken zonder eerst lokaal een instantie aan te maken. Dus zo:

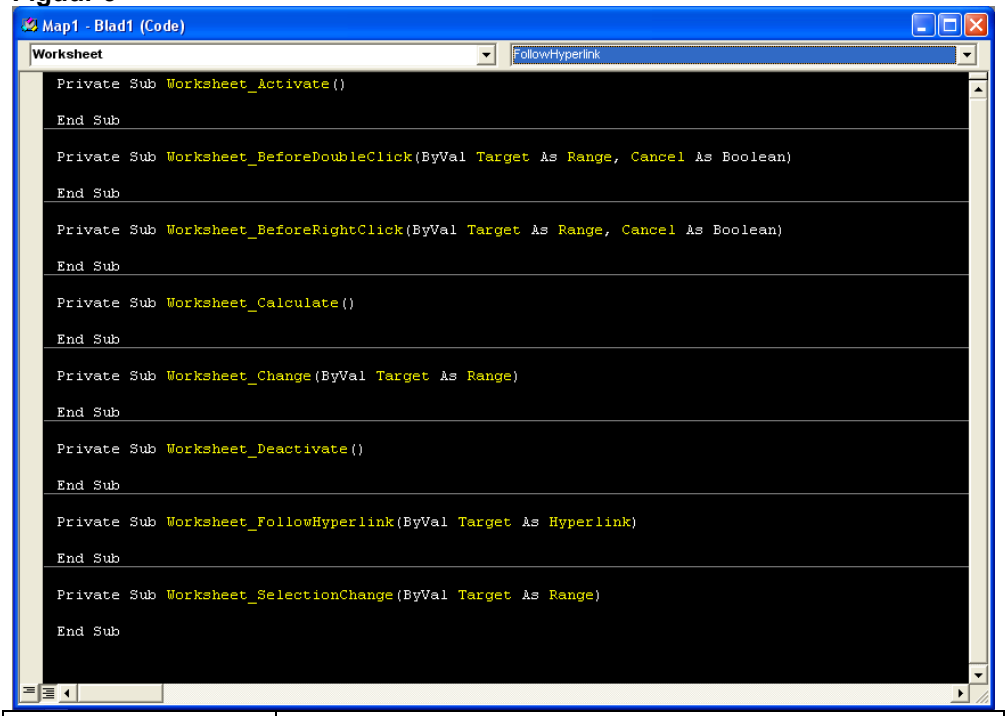
```
Blad1.Cells(1, 1) = 25 * 2
```

Maar het globaal gebruiken van de bladen kan gevaar opleveren, functionele gevolgen hebben. Bovendien kan, zoals die regel laat zien, alleen een waarde op het blad worden geschreven of worden gelezen. Er kan niet worden gecontroleerd of `Blad1` is verwijderd, en dan?! De code die met een lokale instantie werkt kan worden gecontroleerd voordat de actie wordt uitgevoerd.

Figuur 2 laat zien dat ook de naam van het hoofdwerkboek gewijzigd mag worden, zie hier rechts. Bijvoorbeeld de naam `twbMijnboek`. De afkorting `twb` komt van **T**his**W**ork**b**ook, maar u mag ook een eigen afkorting geven. Als u later maar weet dat dit object het hoofdwerkboek is en u niet zelf in de war komt mocht de naam niet duidelijk zijn.

De events van de werkbladen.
Als u op een werkblad dubbelklikt, wordt de code editor geopend. Dezelfde editor als bij `ThisWorkBook`. Maar als u de eventlijst opent, zal er een hele andere lijst te zien zijn. Zo is bijvoorbeeld de `Open` event niet aanwezig.
Figuur 3 geeft een voorbeeld van alle geopende events.

Figuur 3



Event procedure	Doel
Activate	Dit event wordt uitgevoerd zodra het werkblad geactiveerd wordt.
BeforeDoubleClick BeforeRightClick - Target As Range - Cancel As Boolean	Voordat er op de dubbelklik of de rechtsklik van de muis gereageerd wordt, zullen eerst de eventuele acties in dit event uitgevoerd worden. De Target parameter heeft een gebied van het werkblad en de Cancel parameter bepaald of de dubbelklik of de rechtsklik uitgevoerd mag worden.
Calculate	Dit event wordt uitgevoerd als op het werkblad een berekening wordt gemaakt, bijvoorbeeld een som die uitgevoerd wordt.
Change - Target As Range	Dit event wordt uitgevoerd zodra op het aangegeven Target – gebied van het werkblad – een wijziging plaatsvindt.
Deactivate	Dit event wordt eerst uitgevoerd voordat het werkblad niet meer actief wordt gemaakt, bijvoorbeeld: de gebruiker klikt naast het werkblad.
FollowHyperlink - Target As Hyperlink	Dit event wordt uitgevoerd zodra er op een hyperlink wordt geklikt. Hier kunnen wat acties worden uitgevoerd, zoals het bepalen wat de variabele Target voor hyperlink heeft.
SelectionChange - Target As Range	Dit event wordt uitgevoerd zodra de selectie op het werkblad gewijzigd wordt. De parameter Target bepaald de nieuwe selectie.

Macro's

Met VBA kunt u veel doen en, zoals u hebt gezien, is er veel te besturen met de events. Dankzij de voorgeprogrammeerde events kunt u acties uitvoeren wat Excel zelf niet kan doen, zoals het controleren op fouten die de gebruiker maakt. U kunt in de Calculate event als voorbeeld onderstaande code maken.

```
Private Sub Workbook_Calculate()
   Dim Werkblad As Worksheet
   Dim X As Double, Y As Double, A As Double, B As Double

   Set Werkblad = Blad1
   A = Werkblad.Rows.Row
   B = Werkblad.Columns.Column
   X = Werkblad.Cells(A, B)
   'U wilt een deling uitvoeren, maar eerst controleren of X ge-
   'lijk is aan 0.
   If X = 0 Then
      MsgBox "Delen door nul kan niet!", vbInformation, _
      "Rekenfout"
   Else
      Y = 100 / X
      Werkblad.Cells(10, 10) = Y
   End If
   Set Werkblad = Nothing
End Sub
```

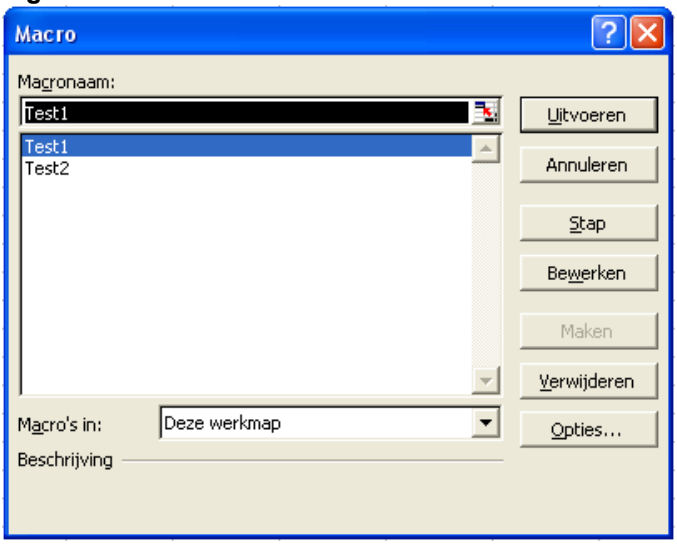
Wanneer telkens een 0 ingevoerd zal worden, wordt dit event uitgevoerd. Maar er zijn situaties waarbij waarde 0 juist wel nodig is. U wilt misschien niet op het hele werkblad dat er gecontroleerd wordt of er gedeeld wordt door nul. Er is dan één oplossing: een macro op slaan in een sneltoets, zie figuur 4.

Figuur 4

Hier kunt u zien dat u uw eigen handelingen in een macro kunt opnemen. De acties komen terecht in een methode met een opgegeven macronaam en een sneltoets. Als dat gedaan is kiest u een van de drie mogelijkheden waar de macro in opgeslagen moet worden.

Een macro opnemen kan ook op een andere manier. Gewoon een methode maken zonder handelingen uit te moeten voeren. U kunt daarmee gewoon een module gebruiken. Zie onderstaande figuren.

Figuur 5



Kies het menu-item Macro's in het menu Extra. Hier ziet u een macrolijst met namen en knoppen. De knop Maken is uitgeschakeld en is alleen ingeschakeld als er geen macro's zijn. Als dat zo is dan zullen de andere knoppen daaraan aangepast worden. Het is bijvoorbeeld niet mogelijk op de knop Uitvoeren te klikken als de macrolijst nog leeg is. De knop Bewerken zal figuur 6 openen.

Figuur 6

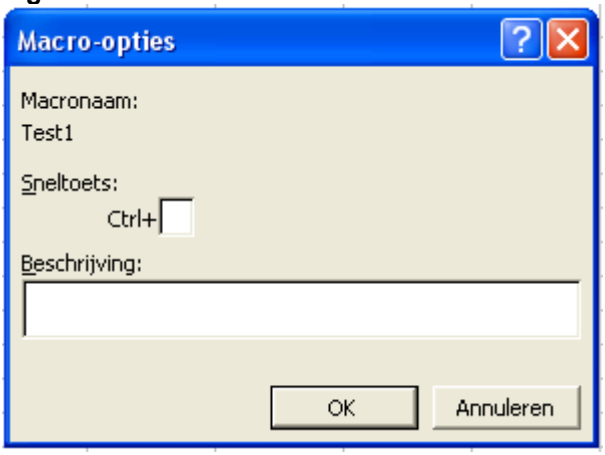
Wat u hier ziet wil niet zeggen dat het speciale macro subroutines zijn. U kunt uw eigen code maken, bewaren in een module, en dan figuur 5 gebruiken om de juiste macro te kiezen. Zoals figuur 6 laat zien zou de code die in de Calculate event staat hier in een subroutine gemaakt kunnen worden. Figuur 5 plaatst uw subroutines automatisch in de lijst.

Ook kunt u een subroutine verwijderen door op de knop Verwijderen te klikken. U hoeft daarvoor niet eens naar de module te gaan.

Wanneer u denkt dat u de belangrijkste macro's klaar hebt en sneltoetsen hebt gedefinieerd, zal het werkblad nog functioneler gaan werken. Dan zijn tenminste niet alle events in gebruik en kunt u zelf bepalen welke macro uitgevoerd mag worden. Tja, gebeurtenissen kunnen voordelen hebben; nadelen hebben ze ook en dat is het automatisch uitvoeren van de events.

Om een sneltoets te definiëren klikt u op de knop Opties..., zie figuur 5. U ziet dan figuur 7 verschijnen.

Figuur 7



Hier kunt u een sneltoets definiëren. Zorg er wel voor dat de sneltoets geldig is.

U kunt ook een beschrijving noteren over wat de macro uitvoert.

Meer mogelijkheden.
Naast bovenstaande mogelijkheden zijn er in het algemeen nog veel meer ontwerpmogelijkheden. Je zou bijna denken dat het niet meer met BASIC te maken heeft en het buiten VBA ligt!
In het algemeen wil zeggen dat die ontwerpmogelijkheden in elke Office applicatie te vinden zijn, maar dat wil niet zeggen dat wat u in Excel ontwerpt daadwerkelijk ook in Word te gebruiken is. De vorige keer heb ik verteld dat VBA geen globale scripttaal is van Office. Elke applicatie heeft zijn eigen VBA boomstructuur, waarvan Excel de grootste heeft.

In de volgende nieuwsbrief wil ik laten zien hoe u de werkbalken, opdrachten en knoppen ontwerpt en zelf programmeert.

Marco Kurvers

Webpagina's ontwerpen.

Internet pagina's maken is een behoorlijke klus. Er bestaat een scripttaal die daarvoor gebruikt wordt. De scripttaal HTML (HyperText Markup Language) was eerder de hoofdtaal om een pagina te kunnen maken. Ook al bestaan er nu veel meer soorten scripttalen; nu nog moeten we beginnen met HTML om een hele website te kunnen maken.
Er is één pagina nodig om de webpagina's te kunnen publiceren. Dat wil zeggen: om in de browser van Internet Explorer de hele paginagroep – de site – te kunnen tonen. Deze ene pagina wordt de index pagina genoemd. Van daaruit wordt de rest van het ontwerp gedaan. Alleen maar de wereldbekende scripttaal HTML gebruiken kost heel wat tijd. Vooral, omdat HTML maar een beperkte scripttaal is. Het heeft maar weinig opmaakcodes om de pagina vorm te geven en zelf codes maken; dat kan helaas niet.

XHTML en CSS
De opvolger van HTML, XHTML (eXtensible HyperText Markup Language), is geen programmeertaal. Het slaat een brug over de ouderwetse opmaakcodes en het moderne XML, die tegenwoordig ook gebruikt wordt om databases te structureren. XHTML is de taal voor de toekomst als het gaat om het vormgeven van webpagina's.
Met XHTML geeft u bijvoorbeeld aan welke tekst een koptekst is, welke tekst als een tabel getoond moet worden, welke tekst een hyperlink is, enzovoort. Vervolgens kunt u een beroep doen op de aanvullende opmaaktaal CSS (Cascading Style Sheets). Het is nu zelfs mogelijk om via de opmaakcode `Language` een andere opmaaktaal te kiezen. Dat betekent niet dat we dan geen CSS zouden kunnen gebruiken.

VBScript
Visual Basic 6.0 heeft een eigen scripttaal om pagina's te maken. Om goede, mooie, pagina's te maken is VBScript echter af te raden. Vooral de structuur van de code is zeer slecht. U kunt dan nog beter XHTML en CSS leren gebruiken dan lastig met een ongestructureerde scripttaal Basic te werken.

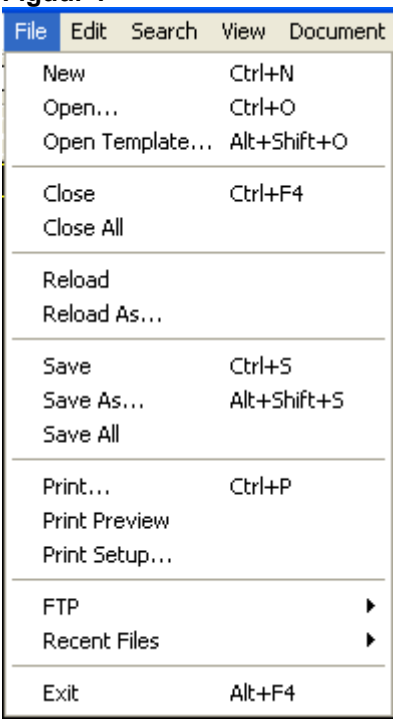
Basic kan ook samen met XHTML worden gebruikt. Er bestaat namelijk een Basic opmaaktaal die met de opmaakcode `Language` gekozen kan worden. Waar u dat kunt doen ligt eraan welk web ontwerpprogramma u gebruikt. Hebt u er echter geen een en zou u XHTML en CSS samen met andere opmaaktalen best willen proberen; hieronder staat welk programma die mogelijkheden heeft.

Crimson Editor
Dit programma is niet te vergelijken met Microsoft Frontpage. Hier kan alleen code worden ingevoerd met verschillende scripttalen. Net als in Visual Basic kan in Crimson projecten geprogrammeerd worden waar alle pagina's en taalmodules bij elkaar gehouden worden.

Het programma is makkelijk te vinden op de site www.crimsoneditor.com en is bovendien freeware. U hoeft niks te betalen. Als u ook overweg kunt met de programmeertaal C++ dan is het zelfs mogelijk de source code van Crimson Editor te downloaden.

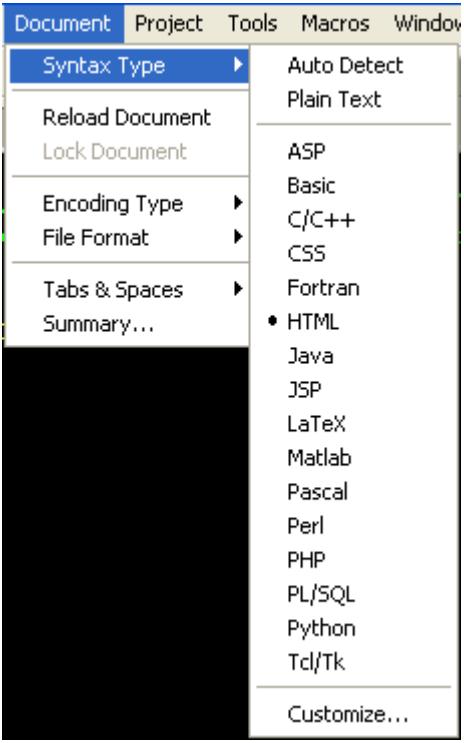
De editor verkennen.
Voordat we gaan kijken hoe XHTML eruit ziet en hoe we Basic kunnen gebruiken, gaan we eerst de editor bekijken. In de Help staat ook het een en ander, maar alles is Engelstalig. Daarom geef ik hier, in de nieuwsbrief, een beknopte uitleg.

Figuur 1



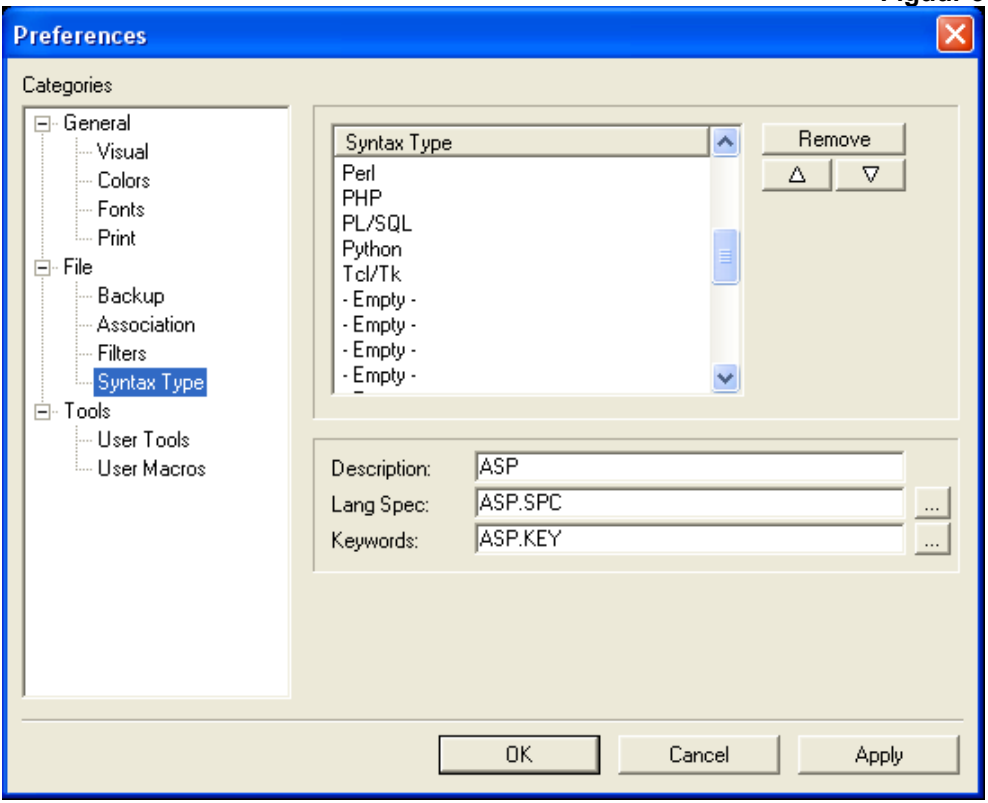
Kies **New** om een nieuwe pagina te maken en als u een pagina bestand hebt dan kunt u die openen met **Open...**.
Verder spreken de menu-items voor zich, **Reload** wil zeggen dat een pagina opnieuw ingeladen (herladen) kan worden.
Het allerbelangrijkste van dit menu is het menu-item **FTP**. In Crimson is er een mogelijkheid om de upload gegevens in te stellen, zodat u uw pagina's kunt publiceren op de internet browser. U moet dan wel een eigen webhosting hebben met een adressite en een gereserveerde webruimte, anders werkt de FTP niet.
Om niet steeds uw pagina's via de keuze **Open...** te hoeven openen kunt u ook de vastgehouden pagina's, de **Recent Files**, uit de lijst kiezen.

Figuur 2



Figuur 2 laat zien dat Crimson een hele lijst met syntax typen kent, onder andere ook Basic. Het is ook mogelijk een eigen taal te kiezen die niet in de lijst staat. Kies daarvoor het menu-item **Customize...**. Na de keuze zal de preferences formulier worden geopend, zie figuur 3.
Standaard is de syntax type HTML al gekozen door Crimson, zie figuur 2. Waarom de keuze XHTML er niet is komt doordat in het sjabloon een verwijzingcode staat. Die code zorgt ervoor dat de extra 'X' samen met HTML zal werken. Zonder die verwijzing zou er alleen maar de taal HTML gebruikt kunnen worden. Figuur 4 geeft de sjabloonlisting.

Figuur 3



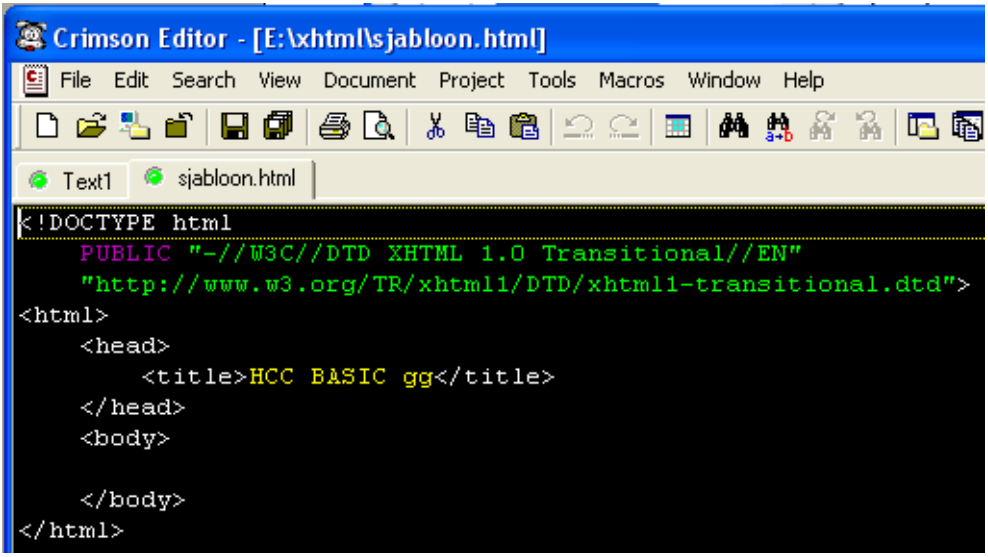
Hier kunt u de syntax type bewerken. In de lijst kunt u zien dat er een aantal 'Empty' zijn, dus leeg zijn. Door een lege te kiezen, kunt u die met een andere taal vullen.

Description: Geef hier de beschrijving van de taal.
Lang Spec: Geef hier de taal specificatie. Zoek die eventueel op in de map, door op de knop met de drie punten te klikken.
Keywords: Geef hier het bestand op met de sleutelwoorden extensie. Zoek die eventueel op in de map, door op de knop met de drie punten te klikken.

Wanneer u klaar bent klikt u op de knop **OK**. Wilt u echter niet dat gelijk het formulier gesloten wordt, klik dan op de knop **Apply**. Apply betekent Toepassen. Het formulier blijft dan actief.

De eerste pagina, de index.
Hieronder ziet u de editor met de HTML code. Dit is het index.html bestand waar eerst mee begonnen moet worden.

Figuur 4



Voordat we het als een index opslaan, is het verstandig om het apart te bewaren als een sjabloon. Zo kunt u makkelijker met uw sites beginnen dan telkens weer een nieuwe index geraamte te moeten maken.

DOCTYPE Geef hier op met welke scripttaal u wilt beginnen. Figuur 4 laat een HTML sjabloon zien, dus staat er achter **html**.

PUBLIC Zoals figuur 4 laat zien staat er tussen aanhalingstekens de verwijzing naar de site waar XHTML te vinden is. Het sleutelwoord PUBLIC zorgt ervoor dat XHTML openlijk in de paginagroep te gebruiken is.

Crimson Editor en het Visual Basic Script.
Hoewel u uit de syntax type lijst de taal **Basic** kunt kiezen, is het toch niet helemaal de Basic programmeertaal die we normaal kennen. Crimson kent namelijk alleen VBScript, zie figuur 5.

Door alleen maar de naam Basic op te geven, zal de browser de hele Basic-code negeren. De browser ondersteund namelijk alleen maar scripts en geen programmeertalen. Dat is niet alleen zo met Basic, ook met Java is dat het geval dus moet er JavaScript worden opgegeven. Elke taal kan wel gebruikt worden, als ze maar een eigen script hebben.

Figuur 5

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <script language="VBScript">
            Function Test()
                MsgBox "Dit is een test."
            End Function
        </script>
        <title>HCC BASIC gg</title>
    </head>
    <body onload="Test()">
        Een Basic voorbeeld.
    </body>
</html>
```

Dit voorbeeld laat zien hoe u VBScript in XHTML gebruikt. In het headblok en voor de titel staat de script code met de aangegeven scripttaal. Binnen in het scriptblok staat een Basic functie die een boodschapvenster weergeeft met de tekst "Dit is een test.".
De functie zal worden aangeroepen door de event **onload**, zie achter body.

Let op! Geef bij variabelen en functies geen typen op. VBScript werkt alleen maar zonder typen.

Wordt uw pagina echter geblokkeerd, zorg er dan voor dat de scripts en ActiveX besturingselementen geaccepteerd worden, door op de waarschuwingsbalk te klikken die boven de browser tevoorschijn komt.

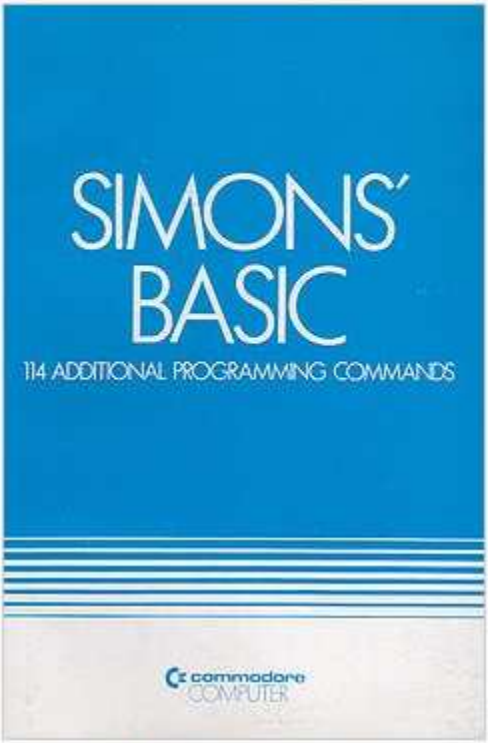
Tip! Wilt u uw pagina's testen? Dat is mogelijk. Klik op het menu **Tools** en kies het menu-item **View in browser**. Uw pagina wordt dan in de browser weergegeven.

Marco Kurvers

In de tijd van Simon's BASIC.

Simon's BASIC was een uitbreiding van BASIC 2.0 voor de Commodore 64 home computer. Geschreven door een 16 jaar oude Britse programmeur David Simons in 1983, het werd gedistribueerd voor Commodore in cartridge formaat.

Mogelijkheden
114 verschillende sleutelwoorden waren in BASIC 2.0 toegevoegd in Simon's BASIC. Deze toegevoegde sleutelwoorden waren vooral geschikt om makkelijk sprites, hoge resolutie, meer kleuren en geluid te coderen. Maar dat niet alleen, commando's waren ook geïmplementeerd om gestructureerd te programmeren. Sleutelwoorden speciaal geschikt voor het schrijven en bewerken van BASIC programma's, net als bij de VIC-20 programmeer cartridge, die ook was toegevoegd.
Simon's BASIC kon ook werken met hexadecimale getallen in toekenningen en berekeningen door het invoegen van een $ prefix, of binaire getallen door te utiliseren met een % prefix.



Simons' BASIC manual front page.

Vanwege een deel van de cartridge gegevens gemapt waren, in het geheugen vanaf adres $8000-$9FFF, welke de standaard C64 BASIC RAM overlapte, was de hoeveelheid overgebleven geheugen voor de BASIC programma's 8 kilobyte minder dan de standaard C64 configuratie.


Simons' BASIC cartridge.

Het Simon's BASIC opstartscherm. Anders dan de normale C64, die blauwe symbolen heeft, en het mindere 8 kb aanwezige BASIC programma geheugen in gebruik door de cartridge.

Marco Kurvers

Het BASIC commando DEF.

Al in de tijd van Commodore was er een mogelijkheid om in BASIC zelf functies te definieren. Twee commando's, `DEF FN`, werden daarvoor gebruikt. De syntaxis is als volgt:

`DEF FN`<functienaam> (<parameter>) =<expressie>

Met een voorbeeld: `DEF FNA(X)=100/X`

In plaats van steeds de expressie te gebruiken, kan men de functie gebruiken samen met bijvoorbeeld `PRINT`. Het beperkte is dan wel dat maar één regel functies gedefinieerd kunnen worden, maar ja, wat is BASIC?! Het probleem is bijvoorbeeld dat parameter `X` waarde nul kan hebben en deelt door nul en dat kan niet. Doordat het één regel functies zijn is het onmogelijk de `DEF FN` regels op fouten te controleren. Functies schrijven in `GOSUB` en `RETURN` subroutines is de enige oplossing, maar dat is weer het nadeel dat geen functienamen gebruikt kunnen worden. Alleen maar regelnummers.

Andere mogelijkheden met DEF.
Wat later in de tijd, zoals in GW-BASIC en later ook in QBASIC en in QuickBASIC, waren er mogelijkheden om variabelen te definiëren met opgegeven naamtypen. Die naamtypen waren wel afgekort en moesten zonder spatie achter de commando `DEF` worden geplaatst. Hieronder een aantal voorbeelden.

`DEFINT A`	Definieert alle variabelen als integers waarvan de namen beginnen met de letter A.
`DEFINT A-Z`	Definieert alle variabelen als integers waarvan de namen allemaal beginnen met A tot en met Z.
`DEFSTR S`	Definieert alle variabelen als strings waarvan de namen beginnen met de letter S.
`DEFDBL D`	Definieert alle variabelen als doubles waarvan de namen beginnen met de letter D.
`DEFLNG X-Z`	Definieert alle variabelen als long integers waarvan de namen beginnen met de letters X, Y en Z.

Het nadeel was dat gelijk alle variabelen, die als eerste letter met de opgegeven letter begonnen, geïnitialiseerd waren. Variabele als 'Anders' was dan niet de enige integer, ook 'Aap' en 'Appel' werden gelijk integers. Daarom is de tweede definitie minder geschikt, omdat het hele alfabet als integer geïnitialiseerd wordt. Wanneer men toch een variabele als een string wilde hebben, moest er een dollarteken '$' achter de variabele worden geplaatst. Op die manier kon één van bovenstaande `DEF` regels genegeerd worden.

Andere BASIC versies herkennen ook variabelen met meerdere symbolen, niet alleen het dollarteken. Ook het uitroepteken '!' en het procentteken '%' kunnen achter de variabelen worden geplaatst. In Commodore BASIC worden minder symbolen herkend.

Marco Kurvers

De nieuwste versies van PowerBASIC.

In de goeie oude tijd van het programmeren in BASIC was alles eenvoudig, we hadden `GOTO` en `GOSUB` waarmee we naar een bepaalde regel of een label konden gaan en dat was het dan zo'n beetje. Maar we gingen vooruit, eerst met `SUB` en `FUNCTION`, dan met parameters als `BYVAL`, `BYREF` en zelfs `BYCOPY` en `OPTIONAL`. Veelvoudige aanroep- en return mogelijkheden kwamen erbij, meer kracht en complexer, vooral in het "inwendige van de compiler".
Sinds die tijd is er veel veranderd, in de nieuwste versies van PowerBASIC hoef je zelfs niets meer te declareren, dat doet het programma automatisch voor je. `ARRAY SORT` is altijd al een essentieel deel van PowerBASIC geweest, snel, efficiënt en veel mogelijkheden. Je kon sorteren op numerieke- of string waarden maar in de nieuwste versies, PBCC5.01 en PBWIN9.01 kan je op meerdere waarden tegelijk sorteren, niet bijvoorbeeld alleen op postcode maar daarbinnen bijvoorbeeld ook op persoonsnaam. Met Custom Sort Array kan je op een aantal sleutelwaarden sorteren, in elke volgorde die je wenst. De custom array kunnen user-defined types, fixed length strings of ASCIIZ strings zijn. Eén sleutel of meerdere,

in de volgorde die je wilt hebben. Hiervoor moet je zelf een functie schrijven en die geeft als resultaat `-1` als de eerste waarde kleiner is dan de tweede, `+1` bij het omgekeerde en nul als beide waarden gelijk zijn. De array `MyType()` bijvoorbeeld is een array van user-defined types die gesorteerd moeten worden en `MyFunction()` is de code die je zelf schrijft om 2 elementen van `MyType()` met elkaar te vergelijken. Het commando wordt dan `ARRAY SORT MyType(), Using MyFunction()`. In de Help-file van beide programma's staat een voorbeeld van een dergelijke functie voor meerdere sleutelwaarden.
Een verdere vereenvoudiging heeft men toegepast bij het `INCLUDE` commando. Om te vermijden dat een bepaalde externe file meerdere malen in hetzelfde programma wordt aangeroepen heeft men nu het commando `#INCLUDE ONCE` toegevoegd hetgeen betekent "Voeg deze file aan het programma toe indien het nog niet eerder daaraan toegevoegd werd". Dit is het voordeel van leesbare code!

In beide versies kan je nu ook met Objecten werken. Je hebt COM Objects, Internal Objects, Methods, Properties, Class Methods, Constructors, Destructors, Inheritance met Method Overrides, Encapsulation, Event Handlers (Dispatch en Direct), Instance variabelen en nog veel meer. Zelfs DISPATCH zonder Variants. Het grootste voordeel van PowerBASIC Objects is, dat zij snel en klein zijn net zoals Subs en Functions. Bovendien zijn Objects facultatief, gebruik ze op het juiste moment, vervolg met normale code of gebruik ze door elkaar als je dat zo uitkomt.
COM Objects maken veel gebruik van het register. Elk Object, elke CLSID, elke IID en anderen behoeven meestal een entry in het register. Maar met PowerBASIC kan dat geheel vermeden worden, ongeacht hoe of waar een COM object werd gecreëerd, je kunt het benaderen met PowerBASIC zelfs als het niet in het register staat. Verder kun je COM objects compleet "private" houden als je dat nodig hebt.

PB/CC is de console compiler van PowerBASIC, gemakkelijk in het gebruik en toe te passen als je "prestaties" wilt hebben met niet veel "rompslomp" daar omheen. Met PowerBASIC 9.01 voor Windows kun je grafische schermen maken zoals Windows die heeft.
Nieuw kosten deze programma's respectievelijk $ 169 en $ 199, upgrades vanaf PB/CC 4 respectievelijk PB/WIN 8 kosten $ 89 respectievelijk $ 99. Ga naar www.powerbasic.com en klik op HELP DESK voor meer informatie of op PURCHASE voor aankoop.

F. Luchsinger
Vertaald uit PowerBASIC Gazette # 64

Variabelengedrag

Met programmeren komen we van alles tegen. In het meeste geval het gebruik van variabelen. Variabelen zijn nodig om

- de resultaten van berekeningen te bewaren;

- de resultaten in nieuwe berekeningen samen te gebruiken en het nieuwe resultaat te bewaren;
- te controleren en, eventueel met meerdere controles, het resultaat te gebruiken op de manier zoals de vorige twee aangeven;
- de code beter te kunnen structureren.

Daarbij kennen we ook nog gedragssoorten die we tijdens gebruik van variabelen tegenkomen:

- globale
- lokale
- geërfde
- geargumenteerde
- geparadeerde
- optionele
- gestructureerde

Er kunnen nog meer gedragssoorten bestaan. Elke BASIC versie heeft er verschillende, meer of minder. Het is helaas niet zo dat alle genoemde gedragssoorten in alle BASIC versies te vinden zijn. Het is bijvoorbeeld niet mogelijk om in oude BASIC versies een geërfd variabelengedrag te gebruiken.

Kritiek krijgen van de computer; wat doen we met het resultaat?
We moeten berekeningen maken om goed werkende programma's te schrijven. Daarbij vergeten we iets dat heel belangrijk is: controle. Controleren op de grammatica (*de syntaxis)* is echter niet voldoende. Het gedrag speelt ook een grote rol, en dat vooral als we het over berekeningen hebben. Het resultaat kan dan op heel wat anders uitkomen dan een programmeur zou denken en/of gewild zou hebben.
Staat er: `Y = X / N`, dan is het resultaat van variabele `N` het belangrijkst. De waarde kan bijvoorbeeld 0 zijn en dan is de kritiek (of ook gezegd: foutmelding of gedrag), dat de computer geeft, niet best. We kunnen dat onderdrukken met code als:

```
IF N > 0 THEN
   Y = X / N
ELSE
   PRINT "Delen door nul kan niet."
END IF
```

Zouden we de controle weglaten, dan kan de computer een melding geven als: *Division by zero error.* Het kan ook gebeuren dat het erger is dan een verkeerde deling. Het programma zal ook direct stoppen met de uitvoering, mocht een foutmelding - als een kritiek - dit fataal onderbreken.
Als we de fouten niet weg kunnen krijgen, zelf niet op kunnen lossen, dan moeten we fout afhandelingcode schrijven. Die is vooral belangrijk wanneer we verschillende gedragssoorten gebruiken zoals objecten typen waarvan we de instanties willen creëren en ook weer willen vernietigen. Er bestaan verschillende soorten foutafhandelingen:

Code	BASIC versie

`TRAP n`
`  ...`
`  REM *HIER BEGINT REGEL N*`
`  ...`
`  IF ER=fc AND EL=rn THEN ...`
`  ...`
`RESUME [NEXT]`
	Commodore 128, Commodore 128D
	BASIC 7.0
	
	C128 BASIC 7.0 was de allereerste BASIC versie met foutafhandeling statements.

`ON ERROR GOTO n | label`
`...`
`REM *HIER BEGINT REGEL N OF DE LABEL*`
`...`
`IF ERR=fc AND ERL=rn THEN ...`
`...`
`ON ERROR GOTO 0`
	GW-BASIC (werkt niet met labels)
	QBASIC, QuickBASIC t/m versie 4.5
	Visual Basic t/m versie 6.0

`Try`
`  ...`
`Catch gedragssoort`
`  If kritiekcontrole Then`
`  foutafhandeling`
`  Else`
`  iets anders`
`  End If`
`  ...`
`[Finally iets anders`
`  ...]`
`End Try`
	Visual Basic .NET
	
	Werkt op dezelfde manier als in C++.
	* gedragssoort: soort exception waaruit de kritiek wordt gehaald en gecontroleerd wordt.
	* kritiekcontrole: eigenschappen die de resultaten van de kritiek hebben. Zo kan worden nagegaan wat er precies aan de hand is.
	* Er kunnen meerdere gedragssoorten in een `Try ... End Try` voorkomen. Voldoet de kritiekcontrole bij elke `Catch` echter niet, dan zal Basic eerst de algemene gedragssoort met de kritiekcontrole proberen. Mocht dat ook niet lukken, dan zal de optionele `Finally` worden uitgevoerd, als die ook aanwezig is anders zal meteen de volgende code na `End Try` worden uitgevoerd.

De algemene gedragssoort, zoals het in de laatste rij is uitgelegd, kan elk gedrag krijgen, of het nou om de printer gaat die plotseling niet afdrukt, of een bestand die niet geopend kan worden. Het nadeel is wel dat de programmeur nu niet zelf kan weten om welke fout het gaat en hoe het afgehandeld kan worden.

Marco Kurvers

Datatypen

In programmeertalen kennen we verschillende soorten datatypen, van eenvoudig tot ingewikkeld. Dat laatste kunnen vooral lijsten en library's zijn, dat kan vooral lastig zijn als de objecten georiënteerd samen kunnen werken (geërfd kunnen worden).

Enumeratietype
Een enumeratie of opsomming is een datatype. Variabelen van een enumeratietype kunnen een vaststaand aantal waarden aannemen, die met een identifier kunnen worden aangeduid.

Bijvoorbeeld:

```
Public Enum TKleur
   Rood = 0
   Oranje = 1
   Geel = 2
   Groen = 3
   Blauw = 4
   Paars = 5
End Enum
```

```
Dim Kleur As TKleur

Kleur = Rood
```

Hierbij gaat het om een aanduiding. Dat betekent dat de namen niet als objectvelden gebruikt kunnen worden. Alleen de variabele kan worden gebruikt die de aanduiding heeft, in principe dus de waarde 0.

Over het algemeen worden de waarden van een enumeratietype intern weergegeven door een (kleine) integer. De enumeratiewaarden en integers kunnen makkelijk naar elkaar geconverteerd worden, en zijn de waarden uit het voorbeeld hierboven equivalent aan de integers 0 t/m 5.

Het sleutelwoord `Enum` wordt binnen programmeeromgevingen veelvuldig gebruikt als afkorting.

Gegevenstype
Een datatype, ook wel gegevenstype genoemd, is een begrip uit de informatica. In een programmeertaal wordt met iedere expressie een datatype geassocieerd. Dit datatype bepaalt welke waarden de expressie kan aannemen en welke bewerkingen erop uitgevoerd kunnen worden.

Primitief en samengesteld
Er zijn twee soorten gegevenstypes. Een primitief type is een type waarvan de waarden primitief zijn. Primitieve waarden kunnen niet opgedeeld worden in eenvoudigere waarden. Een samengesteld type bestaat uit waarden die opgebouwd zijn uit eenvoudiger waarden.

Primitieven
Primitieve gegevens zijn kleine, eenvoudige basisblokken. Primitieve typen bestaan uit deze primitieve gegevens. Iedere programmeertaal heeft een aantal ingebouwde primitieve types, en sommige talen bieden de mogelijkheid om nieuwe primitieve typen te definiëren.

De meest algemeen voorkomende primitieve typen zijn:

- **Boolean**, ook bekend als *bool*, *flag* of *logic*. Kan de waarde *ja* of *nee* bevatten. Andere benamingen voor deze waarden zijn *waar* of *onwaar*, of het Engelse *true* of *false*.
- **Karakter**, ook bekend als *character* of *char*. Deze kan precies één ANSI- of EBCDIC- of Unicode-teken bevatten. Het aantal bytes dat dit type inneemt hangt af van de taal. Historisch was dat meestal 1 byte, maar tegenwoordig ondersteunen veel talen Unicode en worden er meer bytes gereserveerd voor een variabele van type *char*.
- **Integer**, ook bekend als *int*, *short*, *long*, *signed*. Een integer kan een geheel getal bevatten. Het bereik van een integer kan afhankelijk zijn van de hardware waarop gewerkt wordt, bijvoorbeeld in QuickBASIC, PowerBASIC, Visual Basic of in C, of is vastgelegd in een specificatie en is hardware-onafhankelijk zoals in Java.
- **Real**, ook bekend als *float*, *single*, *double*; alle niet gehele getallen. Voor het bereik van een real geldt hetzelfde als dat van een integer. BASIC kent het type Real echter niet en moeten we gebruik maken van een single of een double.

Op elk datatype bestaan diverse varianten. Deze variaties kunnen verschillen in precisie (aantal bytes), interne representatie (in het geheugen) of de functies die erop toegepast kunnen worden. Er kunnen Basic talen bestaan die een onderscheid maken tussen typen die *singned* (deze kunnen negatief zijn) en die *unsigned* zijn (deze kunnen niet negatief zijn). Deze types kunnen in elkaar omgezet worden door middel van een typeconversie. In sommige gevallen kan dit zonder dat er informatie verloren gaat, bijvoorbeeld bij het omzetten van een integer naar een double. In andere gevallen kan er informatie verloren gaan, bijvoorbeeld bij het omzetten van een double naar een integer.

Een variabele van een primitief type wordt vaak afgekort naar zijn type genoemd. Zo noemt men een variabele van het type integer meestal int. Voorbeeld: Variabele `Getal` gedeclareerd als type `Integer` zal dan genoemd worden als: `intGetal`.

Strings
Strings zijn reeksen van karakters. Er zijn verschillende manieren om strings te repesenteren.

- Als primitieve waarden.
- Als arrays van karakters.
- Als pointers naar arrays van karakters.
- Als lijsten *(lists)* van karakters. Merk op dat een lijst een ander datatype is dan een array.
- Als objecten.

Samengestelde typen

In tegenstelling tot primitieve types, die slechts in een beperkt aantal soorten voorkomen, is het aantal mogelijke samengestelde types in principe oneindig. Voorbeelden van benamingen voor samengestelde types in verschillende programmeertalen zijn *classes*, *structures* en *records*.

Afbeeldingen *(mappings)*

Een afbeelding f voegt aan elementen van verzameling A een element uit verzameling B toe. We schrijven:

$$f : A > B$$

In programmeertalen komen we twee soorten afbeeldingen tegen: functies en arrays.

Arrays

Een array is een afbeelding van een eindige verzameling A, de index genoemd, op een verzameling B. In de praktijk is de index altijd een reeks van opeenvolgende discrete waarden. Het laagste element uit de index is de ondergrens van de array, het hoogste element de bovengrens. De lengte van de array is het aantal elementen in de index verzameling A: $| A |$.

Veel talen beperken de index tot een reeks van integers met 0 als ondergrens, zoals wij dat kennen in BASIC. Er zijn echter ook talen die de programmeur vrij laten in het kiezen van een index, zolang deze maar bestaat uit een discreet primitief type, voorbeeld: een enumeratiewaarde.

De meeste talen ondersteunen ook meerdimensionale arrays. In dit geval is de indexverzameling A van een n-dimensionele array een verzameling bestaande uit tupels.

Associatieve array

Een array is in feite een speciale vorm van een associatieve array. Bij een associatieve array hoeft de index geen opeenvolgende reeks te zijn. Vaak is het ook toegestaan om andere (primitieve) typen dan integers als index te gebruiken, bijvoorbeeld strings. Dit kan in BASIC echter niet en moeten we ons beperken tot integers om als index te gebruiken.

Sommige talen ondersteunen wel 'pure' arrays, maar geen associatieve arrays (bijvoorbeeld BASIC en C). Andere talen hebben alleen associatieve arrays en implementeren gewone arrays als associatieve arrays (bijvoorbeeld PHP). Dit heeft een aantal nadelen op het gebied van efficiëntie en geheugengebruik. Er zijn ook talen die beide ondersteunen (bijvoorbeeld Perl).

Functies

Een functie is een afbeelding van een verzameling *A*, nu meestal argument of origineel genoemd, op een verzameling *B*, beeld of functiewaarde genoemd. Als er sprake is van een functie met meerdere argumenten, bestaat *A* weer uit tupels. In tegenstelling tot bij arrays, kan bij functies de bronverzameling *A* oneindig zijn.

Afgezien van het feit dat bij een functie de bronverzameling oneindig kan zijn is het belangrijkste verschil een kwestie van implementatie.

Structures, records en tupels

Veel programmeertalen maken het mogelijk om een nieuw type te definiëren dat bestaat uit (andere) typen. Er zijn verschillende termen voor zo'n samengesteld datatype: records (bijvoorbeeld in Pascal, of een soortgelijk recordtype in BASIC waarmee we alleen met het sleutelwoord `Type` kunnen werken), structures of structs (bijvoorbeeld in C en Visual Basic .NET) of tupels (bijvoorbeeld in functionele programmeertalen zoals Haskell en ook sommige Basic dialecten ondersteunen tupels).

Ongeacht de benaming kan een type gedefinieerd worden als een tupel van zijn samengestelde typen. Een record of structure van de typen *A* en *B* is een type *T* waarvoor geldt:

$$T = \{(a,b)|a \in A, b \in B\}$$

Oftewel: record, structure en tupel types zijn een Cartesisch product van hun samengestelde typen. Neem het volgende BASIC fragment:

```
Public Type t      'BASIC kent niet het sleutelwoord Record.
   a As Integer
   b As Character  'Werkt dit type niet, gebruik dan type String.
End Type
```

NB: In plaats van de hele string te gebruiken, kunt u ook achter String een sterretje met een 1 typen, dus zo: `String * 1`. Niet alle BASIC talen ondersteunen dat.

Een variabele van type `t` kan alle waarden aannemen die bestaan uit een Integer gevolgd door een Character. Elk van deze waarden kan beschreven worden door een tupel *(x, y)* waarbij *x* een integer is en *y* een character. Het verschil tussen records en structures enerzijds en tupels anderzijds, is dat bij de eerste de samenstellende waarden kunnen worden aangeduid met een naam. De samenstellende delen van tupels worden aangeduid met hun positie.

Wat als we twee typen definiëren, die uit dezelfde samenstellende delen bestaan, zoals in onderstaand BASIC fragment?

Public Type t1
  a1 **As Integer**
  b1 **As** Character
End Type
Public Type t2
  a2 **As Integer**
  b2 **As** Character
End Type
	Er zijn nu 2 typen gedefinieerd, die beide uit een tupel van een integer en een character bestaan.
	Dat wil zeggen: voor iedere *(x, y)* waarvoor geldt $(x, y) \in T_1$ geldt ook voor $(x, y) \in T_2$. De typen `t1` en `t2` zijn structureel equivalent. Toch staan niet alle talen het toe dat twee structureel equivalente typen door elkaar gebruikt worden. Het volgende BASIC fragment zal een foutmelding opleveren:

Dim v1 **As** t1, v2 **As** t1 **Dim** v3 **As** t2 v1.a1 **=** 3 v1.b1 **=** "z" v2 **=** v1 v3 **=** v1	De compiler zal meteen met dit fragment met een foutmelding komen, en wel: **Typen komen niet met elkaar overeen.** Die fout zal worden veroorzaakt door de laatste regel, omdat `v3` niet hetzelfde type heeft als `v1` en dit daarom niet is toegestaan.

De reden hiervan is dat BASIC gebruik maakt van *name equivalence*.

Name equivalence en structural equivalence
In het bovenstaande voorbeeld zijn er twee typen gedefinieerd met dezelfde structuur. Een geldige waarde voor het ene type is per definitie een geldige waarde voor het andere type. Of het toegestaan is om deze types door elkaar te gebruiken hangt af van de manier waarop een taal bepaalt of twee typen equivalent zijn.
Als een taal structural equivalence gebruikt, zijn twee types gelijk als ze dezelfde verzameling waarden hebben. Of dit het geval is, wordt bepaald door de structuur van het samengestelde type te vergelijken. Bij name equivalence wordt er gekeken of de twee typen op dezelfde plek gedefinieerd zijn, dat wil zeggen: of ze dezelfde naam hebben.

BASIC is nou net de taal die name equivalence hanteert. Daarom is het in het bovenstaande voorbeeld niet toegestaan om een variabele van het ene type toe te wijzen aan een variabele van het andere type, ook al zijn de twee typen op de naam na equivalent.

Marco Kurvers

Wat zijn programmeertalen?

Een programmeertaal is een taal waarin de opdrachten die een computer moet uitvoeren, worden geschreven. Deze talen hebben een andere syntaxis en grammatica dan natuurlijke talen. Deze laatste zijn te complex en ambiqu om als programmeertaal te fungeren. Code

die in een programmeertaal geschreven is, dient maar op één manier te kunnen worden 'begrepen' door de computer.

Programmeerparadigma
Er zijn in de loop der jaren veel verschillende programmeertalen ontstaan en zij kunnen op verschillende manieren gecategoriseerd worden. Een veel gebruikt onderscheid is dat van programmeerparadigma. Enkele belangrijke voorbeelden zijn het objectgeoriënteerde, imperatieve, functionele en logische programmeerparadigma.

Gebruik
Nadat een programma in de computer is ingevoerd, kan deze het op vier verschillende manieren uitvoeren:

- direct: wanneer het programma is geschreven in de machinetaal van de betrokken computer, kunnen de instructies direct uitgevoerd worden.
- via een interpreter: met behulp van een interpreter voor de programmeertaal worden de instructies sequentieel omgezet in machinecode en direct uitgevoerd.
- na compilatie: met behulp van een compiler voor de programmeertaal worden de instructies eerst alle omgezet in machinetaal die door de processor in de computer direct kunnen worden begrepen, waarna deze instructies kunnen worden uitgevoerd.
- via tussencode: met behulp van een compiler voor de programmeertaal worden de instructies omgezet in een tussencode (bytecode, ook wel *P*-code genoemd), een speciale interpreter voor die bytecode voert dan deze instructies uit.

Een programma dat met een compiler wordt vertaald, kan over het algemeen sneller door de computer worden uitgevoerd dan wanneer gebruikgemaakt wordt van een interpreter, aangezien een interpreter de opdrachten eerst nog moet omzetten naar machinetaal. Programmeertalen worden over het algemeen óf altijd met een compiler óf altijd met een interpreter gebruikt.

Definitie
Onder programmeertalen worden, in de normaal gebruikte definitie, talen verstaan die Turingvolledig zijn. Dat wil zeggen dat het mogelijk moet zijn om in de programmeertaal een interpreter voor een Turingmachine te schrijven, en het moet mogelijk zijn een interpreter te schrijven voor de programmeertaal op een Turingmachine. Het begrip Turing betekent in feite dat een taal zijn eigen commando's kan begrijpen.

In een taal die niet Turingvolledig is, kan een kleiner aantal problemen opgelost worden dan in een Turingvolledige taal. In SQL kan men bijvoorbeeld wel totalen van tabellen met gegevens berekenen, maar men kan bijvoorbeeld niet berekenen wat de kortste route tussen twee punten in een graaf is. Een graaf is een verzameling punten, waarvan elk paar al of niet onderling verbonden kan zijn d.m.v. een lijn.

Geschiedenis
Het is mogelijk om computers direct in hun eigen machinetaal te programmeren: direct de enen en nullen te specificeren die door de processor kunnen worden begrepen. Dit was voor de eerste computers gebruikelijk, met schakelaartjes werden groepen van 8 bits ingesteld. Men ondervond echter snel dat het veel te lastig was om programma's die op die manier waren geschreven, te onderhouden. Snel werd er daarom een symbolische manier bedacht om de machine-instructies als tekst weer te geven in de vorm van mnemonics. Zo werd het mogelijk om instructies veel eenvoudiger te lezen. Deze code, die nog wel een-op-een met de instructiecodes overeen komt, noemt men assembler.

Voor het programmeren van assembler en machinetaal moet de programmeur heel veel weten van de computer die hij wil programmeren. Om programmeren makkelijker te maken, zijn daarna andere programmeertalen, de zogenaamde hogere programmeertalen ontwikkeld. Hoe hoger de orde, hoe verder de taal van de machine-instructies af staat. Een imperatieve programmeertaal (zoals Pascal en C) staat bijvoorbeeld dichter bij de machine-instructies dan een functionele programmeertaal (zoals Scheme, Haskell en BASIC). Een functionele programmeertaal sluit meer aan bij het denken van de mens dan bij de interne werking van de computer. Zo is het in BASIC mogelijk om 'normale' wiskundige definities te gebruiken.

Programmeertalen worden ook wel onderverdeeld in generaties:

- Eerste generatie: machinetaal.
- Tweede generatie: assembler (de kale machine-instructies, maar leesbaar neergezet).
- Derde generatie: BASIC en procedurele talen als COBOL, Algol, C en Fortran, en later ook objectgeoriënteerde talen zoals C++, Java en Visual Basic (.NET).
- Vierde generatie: Talen met een hoger abstractieniveau die voor een bepaald doel zijn ontwikkeld, zoals bijvoorbeeld SQL en VBA voor Office-software.
- Vijfde generatie: Probleemoplossende talen. Hierbij specificeert de programmeur geen algoritme maar het probleem zelf, met een aantal bijbehorende beperkingen. Vijfde generatietalen worden vooral gebruikt op het gebied van kunstmatige intelligentie. Bekende vijfde generatietalen zijn: Scara, Asea, PLC en Prolog.

De generaties worden vaak afgekort als GL, bijvoorbeeld 3GL, als afkorting van 3rd Generation Language(s).

Marco Kurvers

Andere Basic programmeertalen.

Blitz Basic
Blitz Basic is een compiler voor de programmeertaal BASIC.

De Blitz BASIC compilers werden oorspronkelijk ontworpen voor de Amiga maar zijn nu beschikbaar op verschillende platformen. De Blitz producten zijn hoofdzakelijk ontworpen om computerspellen te programmeren.

Compilers
Deze compilers zijn nu beschikbaar:

- Blitz3D
- BlitzPlus
- BlitzMax

Externe links
Wilt u meer weten over deze Blitz compilers, kijk dan eens op de site: www.blitzbasic.com

DarkBASIC

DarkBASIC is een programmeertaal, afgeleid van talen zoals Visual Basic, maar met geïmplementeerde DirectX functies zoals het inladen van een 3D-Model of het creëren van simpele objecten zoals kubussen en bollen en deze scripts laten uitvoeren. Ook zijn er aparte functies voor het simuleren van een wereld met natuurkundige verschijnselen zoals zwaartekracht.

Als een programma wordt gecompileerd wordt het DarkBASIC-PRO formaat omgezet naar Assembly. DarkBASIC is speciaal geschreven voor het ontwikkelen van games.

Externe links

- DarkBASIC Professional, de officiële Engelse website van DarkBASIC http://darkbasicpro.thegamecreators.com/
- The DB Resourcelibrary, een Engelse website met bronnen voor DarkBASIC. http://www.colorarts.de/dbtutor/
- DBTOOLZ, een Engelse website met tools voor DarkBASIC.

Liberty BASIC

Liberty BASIC (LB) is een commerciële programmeertaal en integrated development environment (IDE) die draait op 16 en 32 bit versies van Windows en ook op OS/2.

Achtergrond
Liberty BASIC is geschreven door Carl Gundel en zijn eerste versie werd gepubliceerd in 1992, door zijn bedrijf Shoptalk Systems, en is sindsdien regelmatig veranderd. De laatste gepubliceerde update van de software was in 2006. De huidige versie is v4.03.

Liberty BASIC wordt in diverse gidsen en verwijzingen over programmeertalen als voorbeeld taal gebruikt bij het programmeren van Windows, met inbegrip van het boek *Beginning Programming For Dummies* door Wallace Wang.

Hoewel Liberty BASIC zijn beperkingen heeft in de rol als zeer geavanceerde programmeertaal, is het toch zeer acceptabel als een goede bruikbare inleiding tot het programmeren van IDE voor de gemiddelde en gevorderde gebruikers van Windows en OS/2. De OS/2 versie is zeer oud, maar gratis. Een nieuwe versie die op Windows, Macintosh en Linux zal draaien wordt momenteel actief ontwikkeld.

De compiler herkent zijn eigen dialect van de programmeertaal BASIC.

Het programmeertaal dialect en IDE heeft een eigen marktgebied ontwikkeld voor beginnende en gevorderde programmeurs die de vaardigheden van programmeren willen leren. Hoewel het nog minder bekend staat als commercieel product is het daar ook geschikt voor en betekent dit niet dat Liberty BASIC alleen geschikt is voor onderwijssoftware. Het is een product met een commerciële potentie.

De huidige versie, draait alleen op Microsoft Windows, maar aan versie 5 wordt momenteel druk gesleuteld en die draait eveneens op Mac OS en Linux systemen.

Eigenschappen

- Een interactief leerprogramma geschikt voor de beginner.
- Een visueel ontwikkel hulpmiddel genaamd FreeForm, geschreven in Liberty BASIC en in de loop der jaren zeer uitgebreid door de gemeenschap van Liberty BASIC.
- Een fouten opsporingsprogramma.
- Het gemakkelijke aanroepen van DLL s en Application Programming Interfaces.
- Liberty BASIC is in staat via de ODBC-methode te communiceren met alle denkbare databases.
- Mogelijkheden tot tekeningen programmeren in kleur.
- Kan spelletjes met SPRITE animatie realiseren, met geluid, muziek en joystick controle.
- Een aanvullend pakket genaamd assist met vele nieuwe eigenschappen, zoals een code opmaak, het plaatsen van versienummer bij broncodes, prestaties toetser, makkelijk te gebruiken browser voor het aangeven van code verschillen, en een beter comprimering en distributiesysteem.

Bronnen

- Websites voor broncode en volledig werkende programma's. Nederlandstalig forum. http://www.libertybasic.nl/
- Liberty BASIC Programmer's Encyclopedia (omvat onder andere een archief van de LB – bulletins bestaande uit 143 nieuwsbrieven van verscheidene jaren).
- Helpbestanden en tutorials in het Nederlands. Zie ook in zelfde Liberty BASIC website.

Onderscheidende elementen en eigenschappen van de taal
Liberty BASIC biedt mogelijkheden tot programmeren in een stijl die lijkt op die van DOS BASIC die via het toetsenbord werkten, maar nu wordt daarvoor een "venster" gebruikt waarin geformatteerde tekst staat en gebruikersinput opgevangen wordt. Het ondersteunt daarnaast ook GUI – gebaseerde EVENT/DRIVEN programmering waarbij verscheidene types vensters gebruikt worden die de standaardcontroles zoals knopen, menu's, textboxes, etc. kunnen bevatten.

De kerngedachte bij het creëren van Liberty BASIC was om de behandeling van vensters naar analogie syntaxis van het werken met bestanden te doen. Bijvoorbeeld, uit de helpfile van Liberty BASIC:

```
OPEN device FOR purpose AS #handle {LEN = n}
```

Deze algemeen toepasbare syntaxis is één van de eigenschappen van LB die het tot een eenvoudig te leren taal maakt.
Zodra een apparaat (device) geopend is, kunnen gegevens en ook bevelen voor de besturing van dit device worden geprint. Voor elk type van apparaat is er een reeks bevelen die op deze wijze naar kunnen worden verzonden. In de recentere versies van LB kan het woord "print" uit de `PRINT` statement worden weggelaten, waardoor de syntaxis nog eenvoudiger is geworden.
Eenvoudig staat centraal bij Liberty BASIC. Slechts twee variabelentypes worden ondersteund in LB 4.03: numeriek en karakterreeks. Geen typeverklaringen worden vereist: elke variabele met een $-teken aan het eind van zijn naam is een stringvariabele; anders is het een numerieke variabele. De vereiste nauwkeurigheid van de numerieke variabelen wordt automatisch toegepast. Voor het aanroepen van APIs of het aanroepen van DLLs van derde partijen waren toch extra data types (STRUCT) nodig. Het werken en aanroepen van DLLs is daarom toch niet ook eenvoudig geworden.

Licentie informatie
De GOLD licentie stelt u in staat op zichzelf staande toepassingen te maken die een runtime-engine en enkele hulp bestanden gebruiken (uw programma wordt gecompileerd naar zogenaamde P-code). Op zichzelf staande (standalone) toepassingen vereisen niet dat u uw broncode prijsgeeft.

Externe links
- Liberty BASIC Nederland
  http://www.libertybasic.nl/
- Liberty BASIC
  http://www.libertybasic.com/
- Liberty BASIC programmers Encyclopedia
  http://lbpe.wikispaces.com/
- blog van Carl Gundel auteur van Liberty BASIC
  http://basicprogrammering.blogspot.com/

- Engelstalig forum.
  http://libertybasic.conforums.com/index.cgi

Meer over programmeertalen kunt u zien op de site:
http://nl.wikipedia.org/wiki/Lijst_van_programmeertalen

Marco Kurvers

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. di. 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	di. t/m zo.	19-22	(0342) 424452	m.a.kurvers@hccnet.nl