Nieuwsbrief

18de jaargang juni 2011 **Nummer 2**

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Gordon Rahman	9

Contacten

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

http://www.basic.hcc.nl

In dit laatste cursusdeel van VBA met Excel kijken we terug naar wat we allemaal geleerd en gedaan hebben. Er komen ook weer nieuwe snufjes bij die wel interessant kunnen zijn en misschien zou u wel willen weten welke verschillen er zijn in oude en nieuwe Excel versies.

Hebt u wel eens gehoord van API functies en handles? Gordon Rahman zal u alles vertellen en laten zien over deze functies en waarom de handles daarvoor nodig zijn.

Tegenwoordig kan men niet meer zonder de Control Flow. Programma's die in andere talen geschreven zijn wil men vaak omzetten in een taal waar men mee bezig is. Als wizards niet 100% te vertrouwen zijn dan zit er maar één ding op: een conversietabel raadplegen.

In Excel ben ik bezig met nieuwe conversietabellen. Ik heb er even de tijd voor nodig om deze klaar te krijgen.

Marco Kurvers

BASIC cursus: VBA met Excel (laatste deel).

Deze BASIC cursus in VBA met Excel liet u zien dat Excel niet zomaar een editor is met opmaak- en effectmogelijkheden. U kunt uw eigen project opbouwen tot een zelfstandig programma dat alleen gestart kan worden in Excel. U kunt dus geen applicaties maken binnen in Excel die dan zonder Excel zouden moeten werken.

Om nog eens alles na te lopen van wat we gedaan hebben in deze vijf lessen, zal hieronder een samenvatting staan met alle belangrijke punten.

Samenvatting.

Excel is een programma met werkbladen die verdeeld zijn in vakken. Die vakken zijn cellen waarin waarden en formules ingevoerd worden.
De cellen kunnen zelfstandig werken, maar ook naar elkaar verwijzen. Door de ene cel met een andere cel te laten verwijzen kunnen we waarden kopiëren zonder gebruik te maken van het clipbord. Maar er is wel een verschil tussen normaal kopiëren van de cellen en naar cellen verwijzen, namelijk:

- kopiëren – we kunnen de inhoud van een cel kopiëren en plakken in andere cellen. Veranderen we de inhoud van de originele cel dan zal de gekopieerde inhoud niet veranderen;
- verwijzen – we kunnen de inhoud van een cel in meerdere cellen plaatsen door in die cellen de originele cel op te geven. Dit wordt *verwijzen* genoemd. Het nadeel is dat bij het veranderen van de inhoud van de originele cel ook de inhoud van de andere cellen, die naar de originele cel verwijzen, zal veranderen.

De inhoud in de cellen moeten we niet altijd beschouwen als het resultaat van een formule of een tekst, want ook de opmaak speelt een grote rol wanneer we de cellen kopiëren, verslepen of verwijzen. Het ligt aan de Excel versie hoe de opmaak werkt. In de Excel versies vanaf 2007 zal er een rolmenu verschijnen als we het vakje rechtsonder in de selectie gebruiken en op het icoontje klikken die bij het vakje zal verschijnen. In de oude Excel versies moet u tijdens het slepen op het vakje de rechter muisknop gebruiken om het rolmenu te krijgen.

De inhoud zonder opmaak betekent: de uitvoer tijdens een berekening (kan bestaan uit een formule, expressie of een conditie) of een tekstinvoer. De opmaak staat er buiten en kan apart ingesteld worden door de celeigenschappen te gebruiken. Die kunt u kiezen via het menu Opmaak en menu-item Celeigenschappen of met de rechter muisknop op een aantal geselecteerde cellen te klikken en uit het rolmenu het menu-item Celeigenschappen te kiezen.

We kunnen de formules zelf invoeren of gebruik maken van Excel functies die eventueel hetzelfde werk voor u kunnen doen. Mocht het beide niet lukken dan is er een mogelijkheid gebruik te maken van macro's. Als we daarmee gaan werken dan leren we omgaan met de BASIC programmeertaal. Die programmeertaal is echter geen applicatieprogrammeertaal. Het bestaat alleen uit Excel onderdelen en extra Excel onderdelen die we niet op de werkbladen kunnen gebruiken. Het is dus niet verkeerd om uw werkbladen functioneler te laten werken met functies en subroutines die u in VBA (Visual Basic for Applications) kunt programmeren. Bovendien heeft VBA een klein IDE scherm waarmee u, net zoals in Visual Basic 6, (dialoog)formulieren kunt ontwerpen en via de werkbladen, de cellen of werkbalk knoppen kunt laten werken. Op die manier is het mogelijk om Excel uit te breiden met uw eigen onderdelen.

In Excel kunnen we functies samen laten werken. We kunnen niet zelf de functies kiezen die we samen willen laten werken. Er zijn speciale functies die op die manier werken. Die functies bestaan uit meerdere functienamen gescheiden door punten. Eén daarvan die we bekeken hebben is de SOM.ALS() functie die zowel een conditie controleert en tegelijkertijd de waarde van de ene cel uit de selectie in de som berekent, als de controle waar is, of de waarde van de cel uit de selectie niet in de som berekent, als de controle niet waar is.
We kunnen de functies kiezen uit de functiecategorieën. Het voordeel is dat u hiermee niet de functies handmatig in de cellen hoeft in te voeren. Elke functie die u uit de lijst kiest heeft een eigen dialoogscherm waarmee u de argumentwaarden van de parameters kunt instellen. U kunt ook zien wat er gebeurt voordat u de instellingen accepteert.

In VBA kunnen we ook werken met de werkbladen, maar we kunnen helaas niet de namen van de bladen gebruiken die we in Excel instellen. Dus alleen Blad1, Blad2 enzovoort, maar niet Boodschappen die we in Blad1 zouden hebben genoemd.

Een andere manier om met de bladen te werken is het collection object Sheets. Hiermee kunt u programmatisch werkbladen toevoegen of de inhoud wijzigen met het Item() object. De Count eigenschap bepaalt hoeveel werkbladen er zijn.
Het nadeel met Sheets is dat u niet direct ermee kunt werken. U moet altijd een instantie object declareren voordat u een bepaald blad uit het Item() object kunt kiezen. U kunt wel met de Blad objecten direct werken omdat die los staan van de collectie, maar als u toch een object instantie maakt van Sheets.Item(1) dan werkt dat hetzelfde als wanneer u gewoon werkt met het object Blad1.

Een ander zeer interessant object is Range(). Met dit object kunnen we in de code de gegevens in de werkbladcellen benaderen en wijzigen. Het is zelfs mogelijk in de code een celgebied te selecteren. Het Range() object kent een heleboel methoden en eigenschappen. Wat we dus normaal gesproken op het werkblad doen kunnen we dus ook met het Range() object doen. Het object heeft echter één voordeel ten opzichte van een selectie maken op een werkblad, namelijk: het Range() object heeft zelf ook weer een Range() object, en nog een, en nog een... We kunnen dus selecties in selecties maken. Het verschil is wel dat de selecties relatief werken ten opzichte van de buitenste selectie. Als er zou staan: Range("A1") dan zou de linker bovenhoek van de selectie niet altijd kolom A en rij 1 hoeven te zijn. Dit geldt ook voor de eigenschap Cells() die in het Range() object ook werkt.

Let op! De Cells() eigenschap werkt niet op kolom en rij, maar op rij en kolom!
Cells() is geen object. We kunnen geen extra mogelijkheden na een punt uitvoeren.

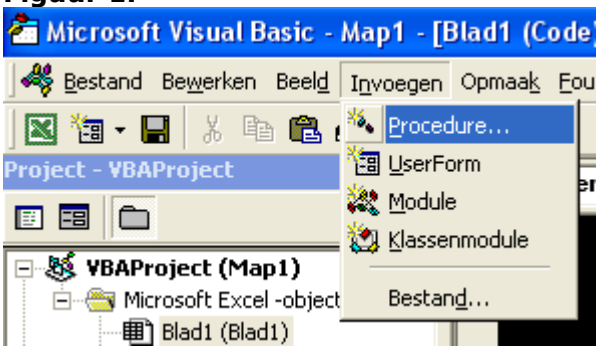
In plaats van een Range() object heeft VBA ook een Range type. Wat kunnen we ermee?

- Met dit type is het mogelijk zelf Range object instanties te maken. We kunnen gebieden opslaan en in andere werkbladen gebruiken.
- We kunnen gebieden, door middel van eigen subroutines, als parameters doorgeven zodat we object gestructureerd in VBA kunnen programmeren. De Range argumenten worden ook wel targets genoemd.
- Een target is een bereik die we als parameter op moeten geven. De naam van het argument ontvangt dan de parameterwaarde, dus het opgegeven bereik.

Meer in VBA.

Figuur 1 laat nogmaals het Invoegen menu zien. In de vorige nieuwsbrief hebt u kunnen lezen wat de menu-items van dit menu doen. Er is echter nog een menu-item dat ik nog niet behandeld heb: Bestand...

Figuur 1.

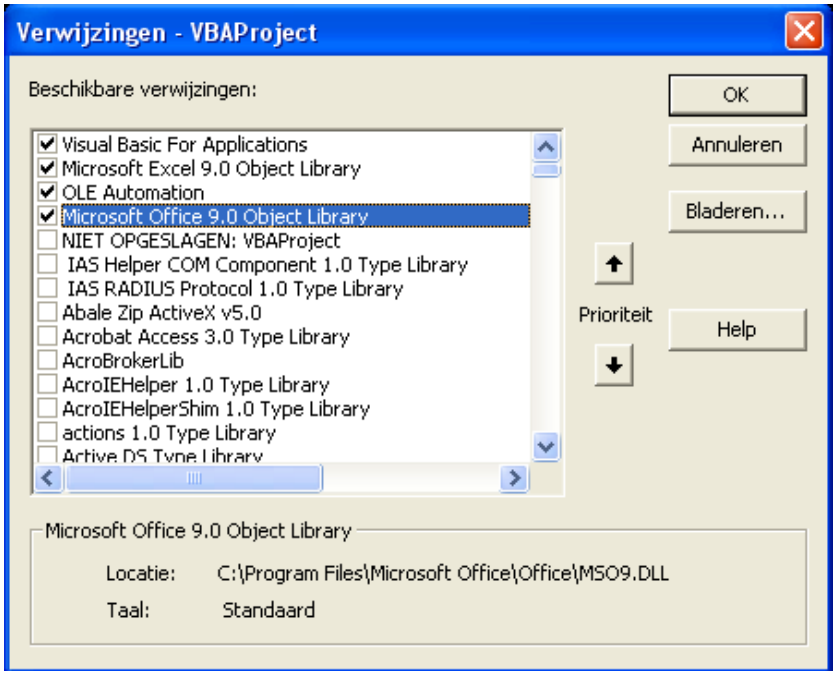


Wat kunnen we met dit menu-item doen?

Zoals u ziet bij Bestandstypen gaat het enkel om externe tekst-, Basic- en klassenbestanden in te voegen in uw VBA project. Een ander soort type bestand is ook toegestaan, maar dan moet uw project wel het bestand kunnen lezen en/of kunnen vertalen. Bijvoorbeeld een XML databestand.

In het VBA menu Extra is er een menu-item: Verwijzingen...
Met het dialoogscherm dat na het kiezen van het menu-item verschijnt, kunnen we externe DLL's in VBA gebruiken. Alleen de DLL's of andere soort library bestanden die beschikbaar zijn staan in de lijst, zie Figuur 2.

Figuur 2.



De beschikbare verwijzingen zijn alle object library's en type library's of API DLL's die overal in de computer kunnen staan en gewoon door Windows gevonden worden. VBA ziet ze daardoor ook en geeft ze weer in de lijst.
De eerste vier verwijzingen zijn aangevinkt. Deze vier zijn belangrijk om VBA en de object library van Excel 9.0 en van Office 9.0 te laten werken.

Met de knop Bladeren... kunt u andere verwijzingen opzoeken indien uw verwijzing niet in de lijst aanwezig is. Controleer eerst of uw verwijzing gecompileerd en geregistreerd is, anders kan VBA de verwijzing niet herkennen of vinden. Vind VBA hem wel, maar werkt de punt niet om de eigenschappen en methoden te gebruiken dan kan dat aan hetzelfde probleem liggen, namelijk: niet geregistreerd.

De volgende keer komt er een nieuwe Excel cursus met VBA, maar dan met de versies XP en 2007. We kunnen dan eens kijken of er veel verschil is tussen deze versies.

Mocht u vragen hebben stuur dan gerust een bericht naar mij.
Woont u ergens in regio Barneveld, Ede, Amersfoort of Scherpenzeel, kom dan eens een keer langs bij Buurthuis Bronveld in Barneveld. Zie www.buurthuisbronveld.nl voor meer informatie. Elke dinsdag geef ik les in tekstverwerking, internet en Office met Word en Excel. Ook geef ik elke 2de en 4de maandagavond workshops over deze onderwerpen.
Elke eerste maandagavond ben ik in De Meern (http://utrecht.hcc-utrecht.nl). Er is dan een bijeenkomst waar u van harte welkom bent en u ook uw vragen kwijt kunt. Op de 3de maandagavond is er dezelfde bijeenkomst, maar dan in Amersfoort (http://amersfoort.hcc-utrecht.nl).

Marco Kurvers

Programmeren – de control flow.

We weten het; er zijn meerdere wegen naar Rome. In de vorige nieuwsbrief heb ik u uitgelegd wat daarvan de voordelen en nadelen zijn en waarom er zoveel uitwegen zijn.
Als we een weg kiezen, kunnen we obstakels tegenkomen. Wegversperringen of files, of de spoorbomen zijn dicht. In de computertaal kunnen we ook obstakels tegenkomen. Waar we tijdens het programmeren vooral mee te maken hebben is de control flow. Een aantal nieuwsbrieven terug heb ik er toen ook wat over verteld. Als we het over de control flow hebben dan hebben we het over:

- uitgelijnde code en structuurcode;
- parameterwaarden doorgeven, voordeel: geen last van zwervende globale variabelen;
- zelf geschreven subroutines en functies, voordeel: we hoeven het wiel niet opnieuw uit te vinden.

De control flow kun je de spieren noemen, de opbouw van de code, die nodig is om de juiste beweging te kunnen maken zodat het programma goed bestuurd wordt.

Elke taal zit weer anders in elkaar en dat hoeven niet speciaal andere soorten programmeertalen te zijn; ook verschillende Basic versies zitten anders in elkaar. Zo kan een PowerBASIC programmeur er geen touw aan vastknopen als deze een Liberty BASIC programma ziet. Het is erg lastig de control flow te lezen en de code in zijn eigen programma en programmeertaal aan te passen. De structuur(volgorde) is daarom ook het allerbelangrijkste dat men moet begrijpen om dit te kunnen doen. Zonder een control flow hebben we alleen maar een eentonige programmalijst met code. BASICA programma's hadden nauwelijks gestructureerde code. De enige spier die nog voor structuur kon zorgen waren de statements GOSUB <regelnummer>|<label> ... RETURN. Het GOTO statement is eigenlijk geen statement dat bij de control flow hoort. Dit statement was altijd al een 'bijdehandje'. Helaas wordt het statement in veel Basic programma's toege-

past, althans in welke versie het nog kan. In de BASICA versie wordt GOTO nog veel gebruikt en PowerBASIC, Liberty BASIC, QuickBASIC en Visual Basic tot en met versie 6 gebruiken het met labels. Ook al werkt het met labels veel beter dan met regelnummers; we krijgen daarmee geen betere control flow.

Laten we eens bovenstaande stappen goed bekijken.

Uitgelijnde code en structuurcode.

Deze code zijn programmaregels die gestructureerd ingesprongen worden. Basic kent geen inspring symbolen of sleutelwoorden, zoals we de accolades in C kennen en de begin ... end sleutelwoorden in Delphi kennen. Sommige Basic versies kennen wel het End of Exit statement met een flow statement erachter. Met een flow statement wil ik zeggen: elk statement die voor structuur zorgt (IF, FOR, WHILE enzovoort). Als we de code niet uitlijnen, is het moeilijk te vinden waar een blokstructuur begint. In de moderne Basic versies wordt voor de programmeur al ingesprongen (de code uitgelijnd) zodra een statement wordt gebruikt die algemeen als een flow statement werkt. Onderstaande codefragmenten zijn van BASICA en QuickBASIC.

```
100 GOSUB 1000
110 REM OVERIGE CODE
500 END
999 REM DE TEST SUBROUTINE
1000 PRINT "HALLO"
1010 REM OVERIGE CODE
1999 RETURN
```

```
Test
' Overige code
End
Sub Test
    Print "Hallo"
    ' Overige code
End Sub
```

De control flow zit in QuickBASIC veel beter in elkaar dan de oude BASICA code. BASIC 7.0 (Commodore 128) was de eerste BASIC versie met een goede control flow. Zelfs code tussen een BEGIN ... BEND zag het voor BASIC 7.0 met regelnummers er stukken beter uit. Helaas kon die blokcode alleen werken na een THEN of een ELSE sleutelwoord.

Laten we nogmaals bovenstaand codefragment bekijken, maar dan in Delphi en C.

```
procedure Test;

// Overige code

procedure Test;
begin
   Writeln('Hallo');
end;

begin
   Test;
end.
```

```
void Test();

// Overige code

void Test()
{
   printf("Hallo\n");
}

void main()
{
   Test();
}
```

Microsoft heeft veel veranderd om de moderne Basic versies net zo een structuur te geven als Delphi en C hebben. Van Visual Basic heb ik veel aan u laten zien en zoals u aan de codefragmenten kunt zien lijkt de control flow van Visual Basic er helemaal niet op. Ook al zouden wij nog prima Basic kunnen gebruiken om te programmeren; projecten schrijven die zo een fantastische control flow hebben als Delphi, C en C++ zou niet kunnen lukken.

Parameterwaarden doorgeven, voordeel: geen last van zwervende globale variabelen.

Microsoft heeft geprobeerd een goede structuur te geven aan Basic. Voor een deel is dat wel gelukt. Er zitten nog haken en ogen aan. We kunnen nog steeds doen wat we vroeger ook gauw deden: in het wilde weg zomaar wat variabelen declareren, of zelfs helemaal niet declareren, en in het hele programma gebruiken. Natuurlijk is dat makkelijk, we hoeven dan niet steeds te onthouden waar we de variabelen gelaten hebben. Basic versies die een IDE hebben (Visual Basic versies) bouwen de programma's op in puzzelstukken. De puzzelstukken bestaan dan uit projecten, klassen, formulieren en modules.

Onthoud! Liberty BASIC heeft ook een control flow als Visual Basic heeft, maar werkt zonder IDE.

Waarom zouden we variabelen als parameters doorgeven aan de argumenten van de subroutines en functies? Als we een variabele in een formulierklasse declareren, is die variabele in de hele klasse beschikbaar.

Maken we een subroutine aan, dat dus een methode wordt van de klasse, dan mogen we de variabele in de methode gebruiken. Maar de spier in de control flow zal niet lekker werken en het programma krijgt kramp in zijn bovenbeen als we de variabele gaan vertellen "ga vooruit lopen", terwijl we juist willen dat de methode achteruit loopt. Met andere woorden, we moeten ervoor zorgen dat de methode een waarde binnenkrijgt van de variabele. De variabele in de methode, dus het argument, ontvangt die waarde. Maar dankzij de parameter kunnen we elke waarde of andere variabele opgeven die met het 'bovenbeen' niks te maken heeft. Zie onderstaande voorbeelden. Het eerste voorbeeld is zonder parameter.

Voorbeeld 1: zonder parameter

```
Dim NaamVar As String

...

Sub Beweging
    NaamVar = "Achteruit lopen"     ' hier wordt de waarde overschreven; foute boel dus
    ...
End Sub

...

Sub Main
    NaamVar = "Vooruit lopen"
    ...
    Beweging
    ...
End Sub
```

Voorbeeld 2: met parameter

```
Dim NaamVar As String

...

Sub Beweging(ByVal Naam As String)
    ' doe iets met de argument Naam
    If Naam <> "Stil staan" Then
        Naam = "Beweging: " + Naam      ' indien mogelijk: Naam = "Beweging: " & Naam
    End If
    ...
End Sub

...

Sub Main
    NaamVar = "Vooruit lopen"
    ...
    Beweging "Achteruit lopen"
    ...
    Beweging NaamVar
    ...
    Beweging "Stil staan"
    ...
End Sub
```

Voorbeeld 2 laat een betere control flow zien. Dankzij de parameter zal variabele NaamVar niet overschreven worden. Ook als we hem zelf meegeven en deze als argument laten veranderen zal er met de oude waarde niks gebeuren. Het handhaven van parameterwaarden en argumentvariabelen zorgt voor een betere doorstroming van code en voorkomt achtergelaten zwerfafval.

Zelf geschreven subroutines en functies, voordeel: we hoeven het wiel niet opnieuw uit te vinden.

Eigenlijk spreekt dit al voor zich, want zoals we bovenstaande code bekeken hebben scheelt het veel programmeerwerk als we zelf subroutines en functies schrijven. We zouden anders telkens hetzelfde codeblok, dat in de subroutine staat, moeten schrijven. Het voorbeeld 'Lichaamsbeweging' laat een goede structuur zien van meerdere keren aanroepen van de subroutine met andere parameterwaarden. Dit is een goed voorbeeld van gestructureerd programmeren, het wiel niet opnieuw te moeten maken en een prima 'gespierd' programma waarmee de control flow goed tot zijn recht komt.

Tot slot.

Wilt u meer weten over de control flow? Ga naar www.wikipedia.nl en typ in het tekstvak *control flow*. Er zal een hele uitleg verschijnen over de control flow.
Wikipedia is een website over alles en nog wat. Als u de pagina's leest over de control flow haal dan niet de verschillende code door elkaar, want het gaat niet alleen maar over BASIC.

Marco Kurvers

API functies en handles.

In de Windows submappen bevinden zich onderdelen die in sommige programmeertalen gebruikt worden. Programmeertalen die zelf nog niet veel kunnen maken er gebruik van. Maar ook Windows zelf maakt gebruik van die onderdelen. Wat zijn het allemaal voor onderdelen, wat is een API en wat is een handle?

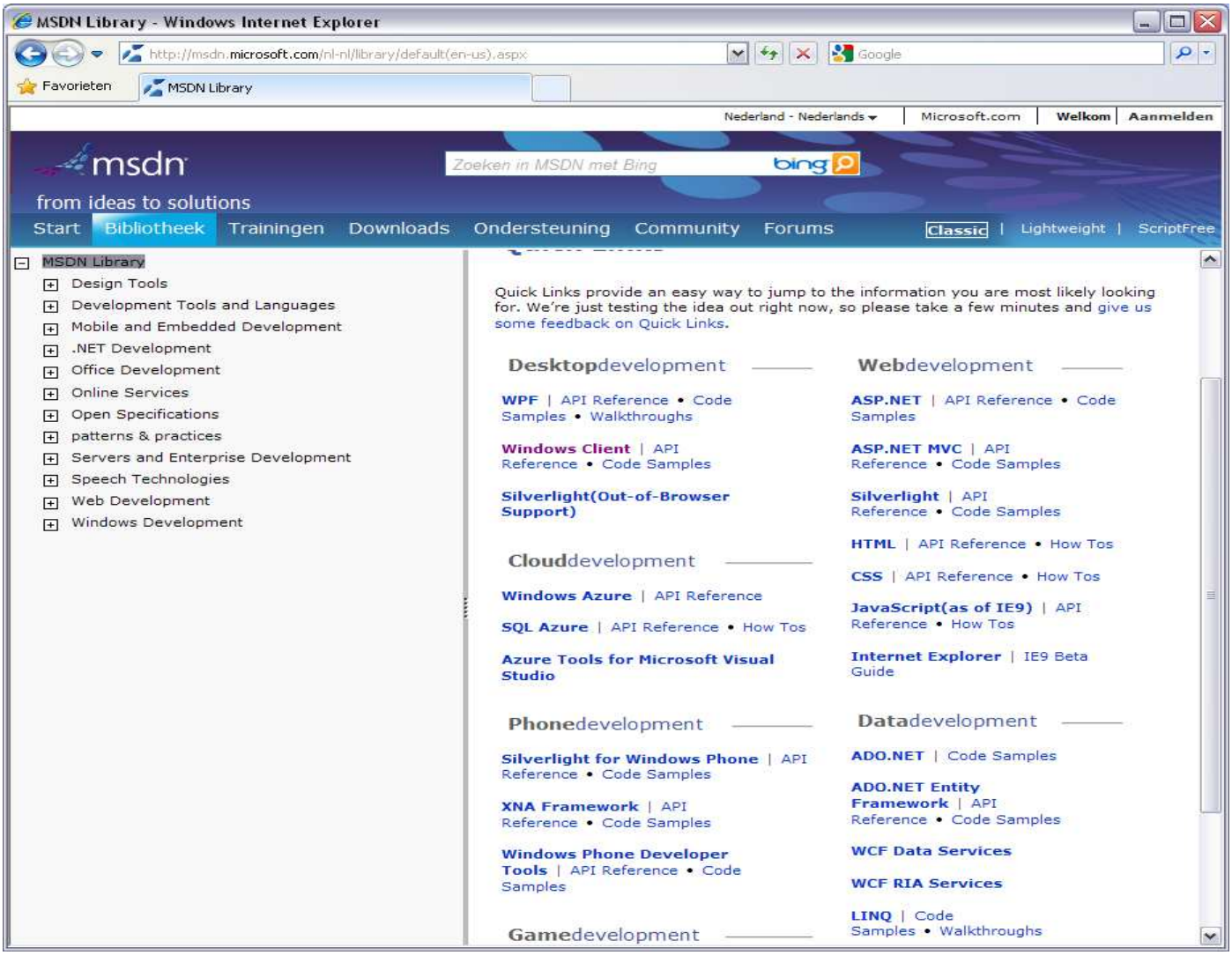
Gordon Rahman zal een uitleg geven over dit onderwerp. Dat is echter nog niet alles. De intermezzo die erachteraan komt is een voorbeeld over hoe u een calculator met behulp van API functies en handle's in Liberty BASIC programmeert. Het is daarom gelijk een goed voorbeeld hoe u Windows applicaties kunt schrijven zonder gebruik van de IDE.

API, een inzicht in Windows.

API (Application Programming Interface) is de systematiek die de toegang tot de functies in de DLL bestanden weergeeft. DLL bestanden zijn bestanden van Microsoft waar veel gebruikte functies voor het besturingssysteem instaan. DLL staat voor Dynamic Link Library. De meeste computertalen maken intensief gebruik van de DLL's als die talen Windows onderdelen gebruiken. Liberty BASIC gebruikte bij de eerste versies zelf voornamelijk de functies uit de Windows DLL's, omdat Liberty BASIC zelf nog weinig eigen native commando's bezat.

Microsoft heeft (onder druk?) zijn DLL functies heel goed gedocumenteerd en publiekelijk toegankelijk gemaakt. Dat is op de MSDN site allemaal te volgen. Die MSDN site heeft behalve informatie over de bestaande software ook info over de nieuwste ontwikkelingen bij MS ten aanzien van het ontwikkelen van nieuwe software. Daarom is de site zeer uitgebreid en voor een beginner bijna niet te doorgronden.

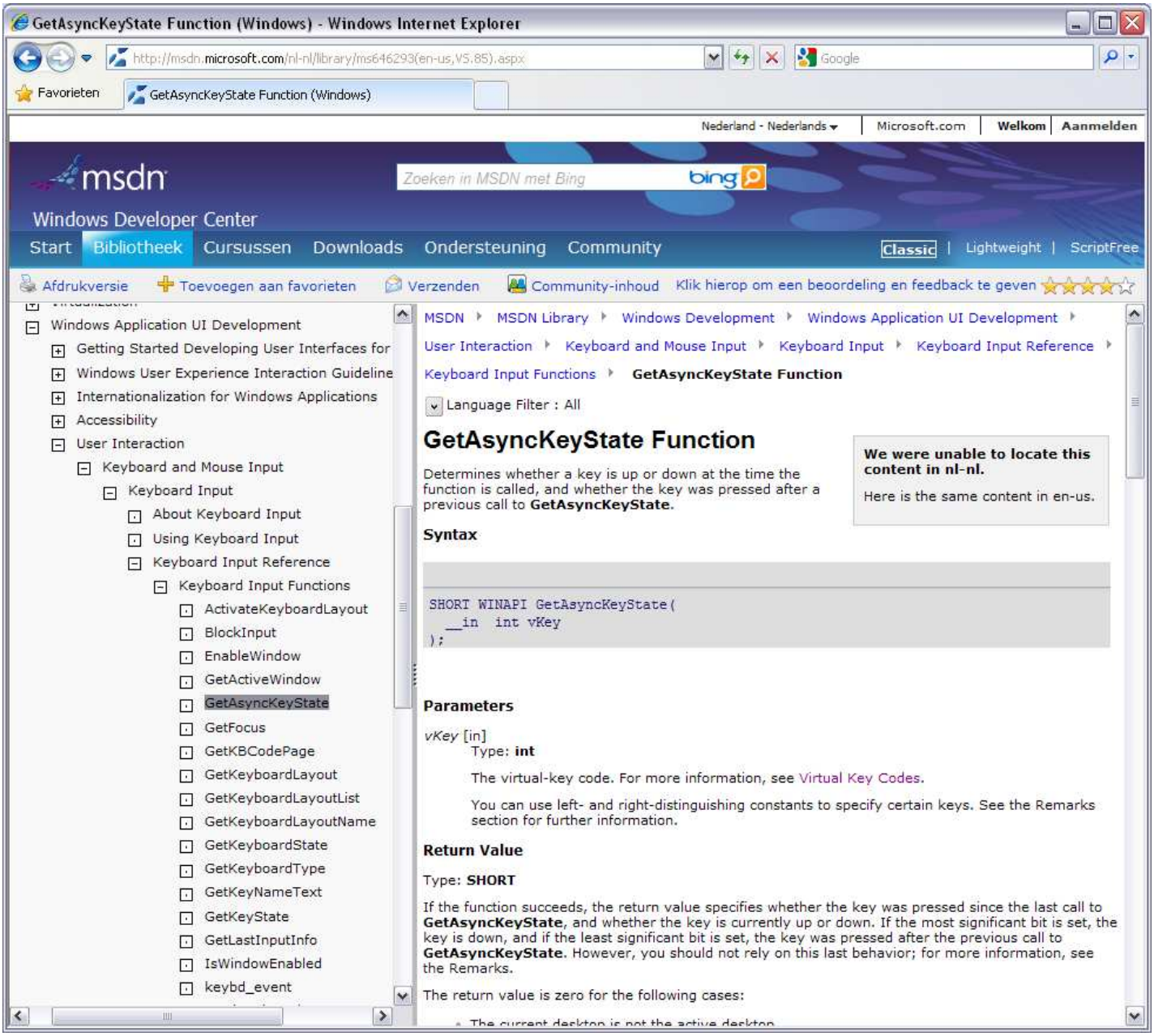
Er zijn dikke boeken over API te verkrijgen. Ik ga in drie delen door de materie vanuit het gezichtsveld van een Liberty BASIC programmeur behandelen. Ga naar http://msdn.microsoft.com.
Laat je niet verleiden tot uitstapjes, maar ga naar de bibliotheek (Library als u op de Engelse site bent). Probeer liever de Engelse site, want de site is in het Engels. Wijzig daarom uw view preferenties direct op [classic], maar als u gewend bent met het moderne [lightweight] of [scriptfree] dan moet u dat maar doen.



Figuur 1. MSDN site

Liberty BASIC programmeurs kiezen nu bijvoorbeeld voor:

- Windows Development
- Windows Application UI Development
- User Interaction
- Keyboard and Mouse Input
- Keyboard Input
- Keyboard Input Reference
- Keyboard Input Functions
- GetAsyncKeyState



Figuur 2. De GetAsyncKeyState functie.

Hier in Figuur 2 ziet u een beschrijving van de functie GetAsyncKeyState.
Deze functie komt overeen met de Inkey$ functie uit Liberty BASIC en andere (Windows) BASIC talen.
In de documentatie op deze MSDN pagina staan, naast een voorbeeld in de taal C ook listing voorbeelden in Visual Basic 9 (versie 2008), VB.NET en C#. Deze voorbeelden komen uit het forum van MSDN gebruikers. Er staat op de pagina ook in welke DLL deze GetAsyncKeyState functie staat. Het staat in User32.DLL. De Liberty BASIC gebruikers hoeven deze DLL niet apart te openen. Deze veel gebruikte DLL staat constant geopend als Liberty BASIC draait. De zogenaamde 'handle' van de DLL is voor Liberty BASIC #user32. De functie geeft een resultaat als gevraagd wordt of een bepaalde toets (Virtual Key Code) ingedrukt is.

Oke, DLL's bevatten functies die we kunnen benutten.
We openen de DLL file met Liberty BASIC en doen een Call naar de functie in die DLL. We geven daarbij de nodige parameters mee en lezen het antwoord van de DLL (return waarde) of we merken de uitwerking van de DLL functie.

Voordat we verder duiken in de materie zullen we eerst het handle concept bespreken. Het handle concept van Windows geeft aan elk object (vensters en controls) een unieke identiteit waaraan eigenschappen gekoppeld kunnen worden.

Windows van Microsoft gebruikt het handle concept voor al zijn devices (apparaten) en objecten (bestanden, vensters en controls, de controls zijn weer knoppen, messageboxen, listboxen enzovoort).

Een handle is een uniek nummer waaraan Windows kan zien welk apparaat- of programmaonderdeel bedoeld wordt. Omdat Liberty BASIC in feite steeds op de achtergrond gewoon ook de DLL functies van Windows gebruikt, geeft Windows aan elke Liberty BASIC device ook gewoon een Windows handle nummer mee. Liberty BASIC gebruikt daarnaast zijn eigen handle nummers. De handles van Liberty BASIC zijn geen cijfers maar namen (strings) voorafgegaan door het teken #.

Een handle voorbeeld.
`#WillekeurigGekozenNaam` is een voorbeeld van een handle in Liberty BASIC. De Liberty BASIC taal heeft een functie waarmee we kunnen zien wat het overeenkomstige Microsoft Windows handle nummer is.
In ons voorbeeld zouden we kunnen programmeren:

```
HW = hwnd(#WillekeurigGekozenNaam)
PRINT HW
```

Als het goed is zien we nu het getal dat Windows gebruikt voor deze handle.
De Liberty BASIC functie HWND() staat als volgt beschreven in de help file van Liberty BASIC:

This function returns the Windows handle (a numeric value) for the window referred to by the Liberty BASIC #handle.

Nog een voorbeeld van het gebruik van de handle, spreek uit: hendel, is de functie van een knop (button) in Liberty BASIC. De volledige handle naam wordt dan: #venster.knop1. Hieronder enkele kanttekeningen:

1. Wat bij de handle #venster.knop1 opvalt is de . (punt) in de handle naam. Het eerste deel van de handle naam, tot de punt, is gelijk aan de handle naam van het venster object waarvan knop1, de eigennaam van de control, deel uitmaakt. Liberty BASIC gebruikt ook het OOP concept waarbij de objecten uit de naamstelling reeds aan elkaar gekoppeld zijn. Elke naam geeft de familienaam weer.
2. De handle naam kent geen spaties of liggende strepen (underscore), daarom gebruiken we het kamelenbult schrift. Hoofdletters geven de aparte woorden aan.

```
Hw = hwnd(#venster.knop1)
```

Dus als een handle in een DLL functie als argument gevraagd wordt dan kunnen we in het vervolg het handle nummer vooraf bepalen met de hwnd() functie en daarna dat gevonden getal (handle nummer) aan die DLL meegeven. De Windows handles (getallen) zijn 32 bits positieve integere getallen. Dat heet in het jargon: unsigned 32 bit integer, dat voor dergelijke getallen het type ULONG geldt. Tot Windows 95 werkten we nog met 16 bits getallen. Intussen zijn de 64 bits microprocessors reeds gewoon goed geworden. Een byte bestaat uit 8 bits dus hebben we voor Windows XP handles 4 bytes nodig om het getal op te slaan. Tweeëndertig bits getallen reiken van 2,1 miljard tot −2,1 miljard. De UNSIGNED integers zijn alleen de positieve waarden.

Met het CALLDLL commando roepen we een functie uit een DLL aan.
Eerst moeten we de DLL openen voordat we de functies kunnen gebruiken. We moeten ook de DLL's weer netjes sluiten als we klaar zijn. Het openen van een DLL gaat natuurlijk in Liberty BASIC volgens het bekende patroon: OPEN <DLLnaam.DLL> FOR DLL AS #<dllnaam>

De meeste standaard DLL's (8 stuks) zijn reeds open bij Liberty BASIC. Openen hoeft niet, maar je moet wel de afgesproken #dllhandleNaam gebruiken als je een functie uit dergelijke reeds geopende DLL's wilt gebruiken.

#user32	**#kernel32**	**#gdi32**	**#winmm**
#shell32	**#comdlg32**	**#comctl32**	

De bijbehorende DLL's staan in de map Windows\System32 (user32.dll, kernel32.dll enzovoort).

Als je bijvoorbeeld URLMON.DLL wilt openen dan doe je dat natuurlijk het liefst met een herkenbare handlenaam als #urlmon of #url.
Uiteraard sluit je een dergelijke DLL na gebruik, bijvoorbeeld bij het beëindigen van het programma, weer netjes af.

```
Close #urlmon
```

Sluit geen DLL af als je het niet geopend hebt.

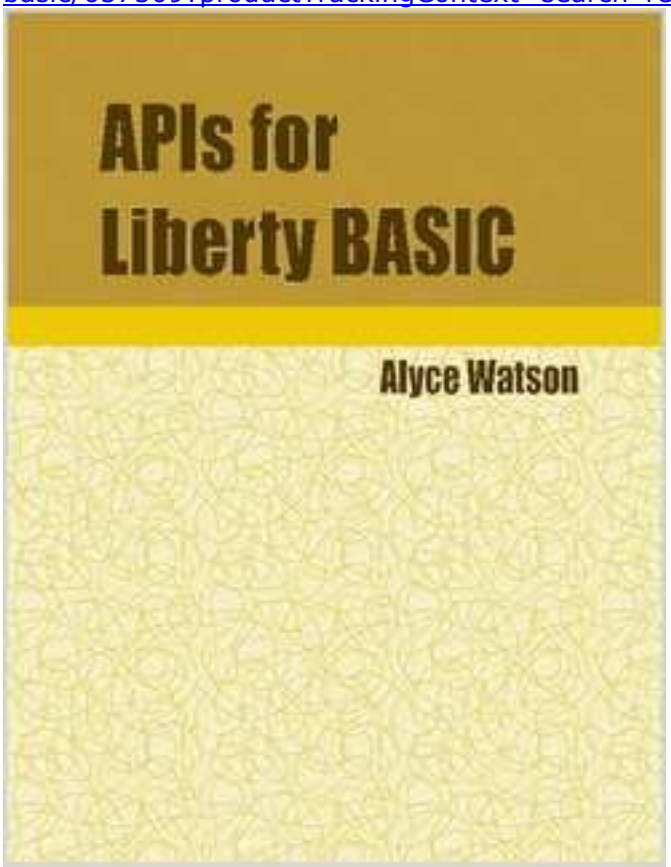
Het CALLDLL statement moet op één regel geschreven worden. Voor het leescomfort kun je de regel wel door middel van de underscore, of ook wel genoemd: onderstrepingsteken, in delen afbreken. Hier volgt een willekeurige CALLDLL als voorbeeld.

Op één regel:
CallDll #<handle>, FunctionName, Parm As <Type>, Parm2 As <Type>, Result As <ReturnType>

Getoond over meerdere regels met gebruik van het onderstrepingsteken:
```
CallDll #<handle>, FunctionName, _
    Parameter1 As <Type>, Parameter2 As <Type>, _
    Result As <ReturnType>
```

Bovengenoemd voorbeeld roept een DLL aan en vraagt daarbij om een functie genaamd FuntionName en geeft daarbij twee argumenten, Parameter1 en Parameter2, van bepaalde types, zoals integers, pointers en strings, aan die functie mee. De call ontvangt van de functie een waarde terug in de variabele Result.

Er zijn ook voor Liberty BASIC veel boeken, twee stuks over API's, verkrijgbaar. Het beste boek is volgens mij het boek van Alyce Watson. Dit boek is te koop bij Lulu.com.
http://www.lulu.com/product/file-download/apis-for-liberty-basic/657509?productTrackingContext=search_results/search_shelf/center/2



Figuur 3 Boek van Alyce Watson.

Bijna 300 pagina's met gedetailleerde uitleg en voorbeelden van het gebruik van functies uit DLL's worden in dit boek in beschouwing genomen. Alyce heeft op de 'Liberty BASIC Programmers Encyclopedia'-wikispaces site een uit negen delen bestaande tutorial geplaatst.
http://lbpe.wikispaces.com/advanced

Terug naar de CALLDLL.
CALLDLL #handle, "FunctieNaam", Par as TYPE, Par2 as TYPE, Result as RtnTYPE

#handle
Het handle concept is reeds eerder uitvoerig besproken.

FunctieNaam
De namen van de functies in de DLL moet je kennen en nauwkeurig toepassen. Er bestaan gelukkig zelfs programma's, ook in Liberty BASIC geschreven, waarmee elke willekeurige DLL geopend kan worden en waarna alle erin staande functienamen uitgelezen kunnen worden.

Parameter as TYPE

Denk bij de lijst van argumenten of parameters aan een lijst met instructies. De functie die we in de DLL aanroepen zal met deze parameters de gevraagde acties uitvoeren. De parameters moet je in de juiste volgorde opgeven. Ook moet je de juiste nauwkeurigheid (TYPE) opgeven. Je kunt geen array elementen of strings eenvoudig meegeven. Je kunt in feite alleen getallen aan een DLL meegeven. Getallen van 2 of 4 bytes. Dat zijn integers met of zonder teken. Voor array elementen moet je een STRUCT opzetten en voor een string moet je een POINTER (PTR) gebruiken. Dat leg ik straks verder uit.

Het AS TYPE gedeelte van de CALLDLL functie stelt Liberty BASIC in staat de verschillende argumenten om te zetten in types die verwacht worden door de functie. In feite sturen we gewoon enkele gehele getallen naar de DLL. Dus ook gebroken getallen (decimale getallen) kun je niet aan een DLL meegeven.
Het argument van de functie, die als parameter zal dienen, kun je ook gewoon als variabele naam opgeven, in plaats van het getal eerst te bepalen en dan een getal mee te geven. Dergelijke variabelen, de parameters, moet je natuurlijk ook aangeven en ze voorzien van een type aanduiding. Je kunt natuurlijk niet rekenen of iets dergelijks met parameters, maar dat lijkt me logisch.

De argumenten komen waarschijnlijk in een vooraf gereserveerde plek in de header van de DLL te staan. In ieder geval is daar geen ruimte bedacht voor strings en arrays, want mij lijkt het dat de lengte en grootte van dergelijke variabelen niet of nauwelijks vooraf te bepalen is. Daarom zullen we zien dat je alleen het geheugenadres op kunt geven waar de string of de array gegevens in het computergeheugen staat.

Liberty BASIC gebruikers zijn altijd verwend door Carl Gundel, de schrijver van Liberty BASIC, doordat Carl automatisch de nauwkeurigheid van de variabelen, die Liberty BASIC gebruikt, bepaalt en of aanpast. Dus Liberty BASIC kent in feite geen type voor zijn variabelen. Standaard zijn de getalswaarden van alle variabelen gehele getallen, integers. Voor een integer getal heeft de computer 2 bytes geheugenruimte nodig. Als er in Liberty BASIC een berekening gemaakt wordt met een decimaal getal (kommagetal) of als de integere waarde te groot wordt dan gaat Liberty BASIC automatisch over op de float variabelen. Dat gaat dus niet bij de DLL's, want die zijn door Microsoft ontwikkeld en daar gelden de regels van Bill Gates.

Dus Liberty BASIC kent geen speciale nauwkeurigheid vooraf aan numerieke variabelen, maar API functies vereisen dat de numerieke variabelen wel een type krijgen.

De types die Liberty BASIC kent als we met DLL's werken zijn:

SHORT	2 bytes dus 16 bits lang	-32768 t/m +32767
USHORT	2 bytes dus 16 bits lang	0 t/m 65535
LONG	4 bytes dus 32 bits lang	-2.147.483.648 t/m +2.147.483.647
ULONG	4 bytes dus 32 bits lang	0 t/m 4.294.967.295

Je begrijpt nu dus ook waarom ook handles bij Microsoft getallen zijn. Strings kun je moeilijk in een DLL stoppen. Getallen gaan er eenvoudiger in.

Voor strings heeft Microsoft het volgende bedacht. Het begin adres waarop de string in het geheugen van de computer staat wordt in de DLL geplaatst. Liberty BASIC noemt dit geheugenstartpunt van de string een *pointer*.

PTR 4 bytes dus 32 bits lang
Een voorbeeld van een pointer in een DLL:

```
CALLDLL #user32, "SetWindowTextA", winHandle as ulong, "Nieuwe opschrift" as ptr, _
                 Result as void
```

De 4 bytes geven dus het adres aan van waar het karakter N(ieuwe) staat. Er wordt daar vandaan gelezen tot een CHR$(0) symbool. Result as VOID staat hier gewoon omdat elke CALLDLL bij Liberty BASIC een returnwaarde moet hebben. VOID betekend dat er geen bytes gereserveerd zijn.

Arrays geven een groter probleem. We kunnen niet met een pointer volstaan. Voor elk element van de array moet apart geheugen vrij gemaakt worden. Daarom kent Liberty BASIC voor arrays het begrip STRUCT.

STRUCT

Structs worden bij DLL's gebruikt om een hele lijst aan informatie door te geven. Denk direct maar aan een array. We gebruiken een struct natuurlijk ook om een lijst gegevens terug te ontvangen. Voor string variabelen gebruiken we een variabele waarachter we het dollarteken plaatsen. Voor een struct variabele plaatsen we gewoon .struct (punt struct) achter elk element van de struct. Dus stel dat de struct A heet, dan zouden het eerste en het tweede element bijvoorbeeld x en y kunnen heten. Deze elementen krijgen elk hun eigen nauwkeurigheid.

Hier volgt een struct genaamd A met twee elementen, y en x:

```
STRUCT A, _
   y as SHORT, _
   x as ULONG
```

Stel dat we het 2de element willen bekijken, dan vragen we naar A.x struct. Dit is natuurlijk naar analogie van de ARRAY. Voor arrays gebruiken we () achter de variabelennaam. Dus ARRAY B met 3 elementen wordt B(0), B(1) en B(2).

Tip! Struct elementen worden ook wel *recordvelden* genoemd.

Net zoals we een array eerst in het geheugen vastleggen met het DIM statement (Declare In Memory), zullen we dat ook met structures moeten doen. Daarom zul je vooraf een STRUCT moeten opbouwen nog voordat je de CALLDLL doet. Hier volgt een voorbeeld van een STRUCT met een andere STRUCT erin:

```
STRUCT BITMAPINFOHEADER, _
   biSize As Long, _
   biWidth As Long, _
   biHeight As Long, _
   biPlanes As Short, _
   biBitCount As Short, _
   biCompression As Long, _
   biSizeImage As Long, _
   biXPelsPerMeter As Long, _
   biYPelsPerMeter As Long, _
   biClrUsed As Long, _
   biClrImportant As Long

STRUCT BITMAPINFO, _
   BITMAPINFOHEADER As Struct, _
   bmiColors As Char[256]
```

We mogen wel een element (veld of member) van een struct direct gebruiken zonder tussenkomst van een variabele. In ons voorbeeld:

```
Print BITMAPINFOHEADER.biPlanes.struct
```

Pointer en struct moet je vaak gezien en gedaan hebben om daar zeer vertrouwd mee te geraken.

In de volgende nieuwsbrieven zullen we daar zeker dieper op ingaan.

Gordon Rahman

Grafisch programmeren in GW-BASIC (7).

In dit en het volgende hoofdstuk, waarschijnlijk zullen ze in meerdere nieuwsbrieven worden verdeeld, gaan we ons bezighouden met het tekenen van driedimensionale lichamen (zo heet dat in de wiskunde) zoals kubussen, prisma's, piramiden, kegels en bollen, en met het tekenen van driedimensionale grafieken van functies. Zo'n driedimensionaal grafiek is een vlak in de ruimte met een vergelijking van de vorm z = f(x, y). Zo krijgen we de bekende 'Mexicaanse hoed' (zie later in de komende nieuwsbrieven) als we de functie $z = e^{-(x2 + y2)}$ tekenen (e is het grondtal van de natuurlijke logaritme: e = 2,71828...).

Er bestaan veel programma's waarmee driedimensionale figuren getekend kunnen worden. Waarom we toch ook hier in de nieuwsbrieven dergelijke programma's laten zien komt, omdat er aan de meeste van deze programma's drie nadelen kleven:

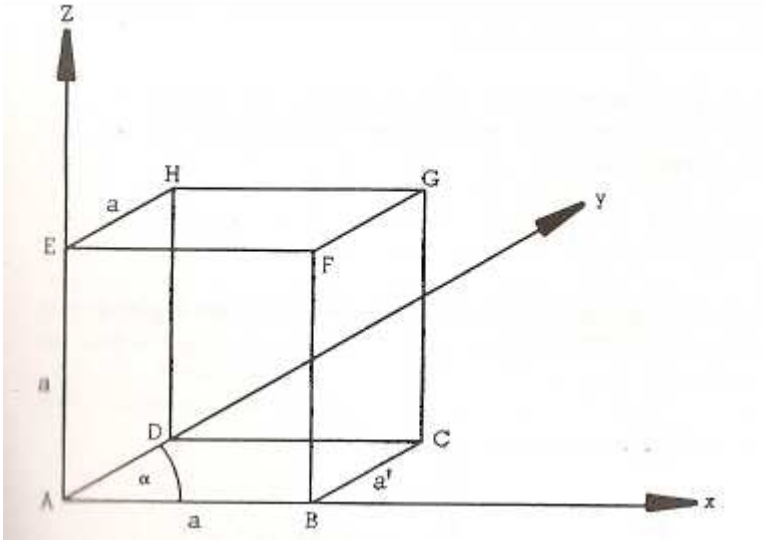
1. De meeste programma's voor driedimensionaal tekenen zijn voor een bepaald graphics-systeem ontworpen en zijn slechts met veel inspanning geschikt te maken voor andere systemen.
2. De wiskundige theorie die aan het driedimensionaal tekenen ten grondslag ligt wordt zelden of zeer summier behandeld.
3. Veel programma's zijn uiterst ingenieus ontworpen en draaien op de kleine microcomputers, in BASIC, heel langzaam.

Met de volgende programma's en stukjes theorie hopen we deze drie nadelen uit de weg te ruimen.

TIP! Hoewel de computers tegenwoordig een stuk sneller werken en de BASIC programmeertalen snelle compilers hebben, is het toch nuttig om te weten hoe we vroeger hiermee moesten omgaan. Het is nog steeds niet verkeerd om die theorie te gebruiken.

Er bestaan verschillende methoden om een driedimensionaal lichaam, bijvoorbeeld een kubus, in een plat vlak te projecteren. Wij hanteren de projectiemethode, die u wellicht op school gebruikt hebt (of nog gebruikt). Stelt u zich een doorzichtige kubus voor met ribben van draadijzer. U staat voor deze kubus een beetje rechts van het midden en kijkt er zo'n beetje schuin bovenop. Uw gezichtsstralen projecteren elke hoek, met de daaraan verbonden ribben, van de kubus in een, achter de kubus liggend, projectievlak. Zo ontstaat de bekende 'schuine' afbeelding van een kubus.

Figuur 1.



De verticale en horizontale ribben worden op ware grootte getekend.
De ribben die naar achteren lopen worden verkort weergegeven.

a' = k.a k = verkleiningsfactor

De rechte hoek tussen de ribben BA en AD in het grondvlak lijkt eveneens kleiner te zijn geworden (zie α in Figuur 1). U kunt gemakkelijk nagaan dat door het vastleggen van alpha (bijvoorbeeld 45^0) en k (bijvoorbeeld 0,5) de projectierichting eenduidig bepaald is.

Wat we nodig hebben zijn transformatievergelijkingen die voor een bepaalde α en k de coördinaten x,y,z van een punt op de ruimtelijke kubus projecteren op de coördinaten x' en y' van het geprojecteerde punt in het platte (projectie)vlak. Deze formules vormen de kern van alle volgende programma's.

Afleiding van de transformatievergelijkingen.

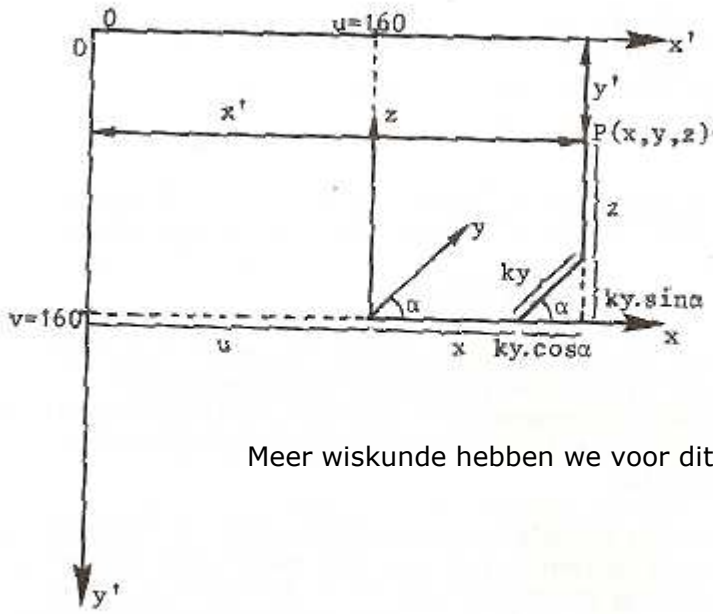
Het projectievlak is ons beeldscherm. We gebruiken ook nu een schermresolutie van 320 bij 320 punten (GW-BASIC). De oorsprong van het ruimtelijk coördinatensysteem (xyz) moet precies in het midden van

het beeldscherm geprojecteerd worden. De oorsprong van het beeldscherm coördinatenstelsel ligt weer in de linkerbovenhoek. De x-as wijst naar rechts; de y-as naar beneden.

De afbeelding van het ruimtelijk punt P(x,y,z) op het punt P'(x',y') in het platte vlak komt tot stand met behulp van de volgende transformatievergelijkingen:

$x' = U + x + k.y.\cos\alpha$ èn
$y' = V - (k.y.\sin\alpha + z)$

Hoe we aan deze vergelijkingen komen is uit Figuur 2 direct (nou ja, direct.....) duidelijk.



Figuur 2.

Nemen we C = K*COS(A), S = K*SIN(A) en H = 0.5, dan zijn de transformatieformules in BASIC:

```
X2 = INT(U+X+C*Y+H)
Y2 = INT(V-S*Y-Z+H)
```

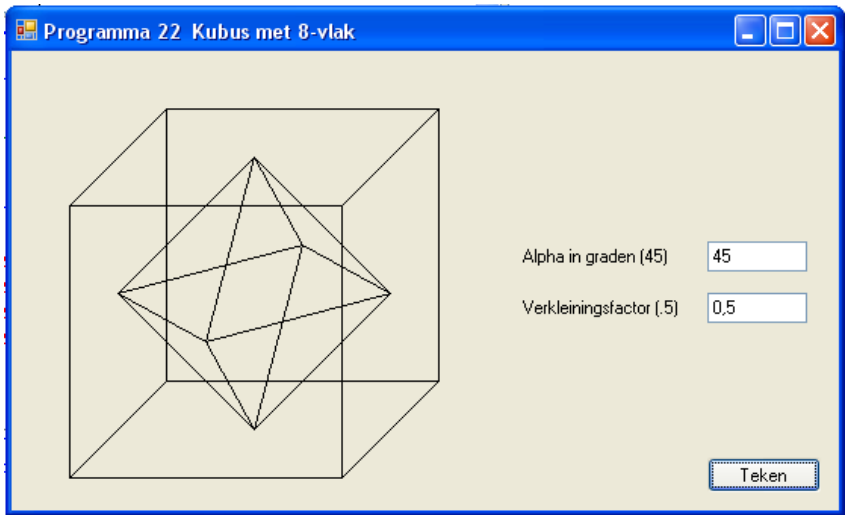
Meer wiskunde hebben we voor dit niet nodig.

Programma 22 laat zien hoe je van elk, door rechte lijnen begrensd lichaam (polyeder) een projectie kan maken. Hiervoor is het noodzakelijk dat de waarden van de coördinaten x,y,z in alle hoekpunten bekend zijn. Als voorbeeld van het gebruik van dit programma tekenen we een kubus met een ingeschreven achtvlak (octaëder). De gevolgde programmeertechniek leidt ook bij een viervlak, prisma of piramide, tot het gewenste resultaat.

We nemen aan dat het middelpunt van de kubus samenvalt met de oorsprong van het ruimtelijk coördinatensysteem. Om het programma 'sneller' te maken nemen we de coördinaten van de acht hoekpunten ABCDEFGH van de kubus en de coördinaten van de zes hoekpunten IJKLMN van de achthoek op in DATA regels. Het tekenen van de ribben wordt door de letterstring Z$ bestuurd. Zo betekent Z$ = "ABBCCDDA" dat de computer van A naar B, van B naar C, van C naar D en van D naar A rechte lijnen moet trekken. Met de MID$ functie halen we steeds één letter uit de string Z$ en maken er vervolgens met de ASC functie een getal van (A=1, B=2, C=3, ...). Hiermee moet de werking van het programma duidelijk zijn. Kies voor alpha 45^0 en voor k 0,5; dat geeft de mooiste projectie.

```
100 '       programma 22     KUBUS MET  8-VLAK
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "ALPHA IN GRADEN    (45)  "; A
150 INPUT "VERKLEININGSFACTOR (.5)  "; K
160 U=160: V=160: H=.5 : RD=4*ATN(1)/180
170 W=A*RD: C=K*COS(W) : S=K*SIN(W)
180 DIM X(14), Y(14), Z$(2)
190 FOR J=1 TO 14
200     READ X,Y,Z
210     X(J)=INT(U+X+C*Y+H):Y(J)=INT(V-S*Y-Z+H)
220 NEXT J
230 CLS
240 FOR N=1 TO 2
250    READ Z$(N) : L=LEN(Z$(N))
```

```
260     FOR M=1 TO L-1 STEP 2
270         A$=MID$(Z$(N),M,1)   : I=ASC(A$)-64
280         B$=MID$(Z$(N),M+1,1): J=ASC(B$)-64
290         X1=X(I): Y1=Y(I): X2=X(J): Y2=Y(J)
300         LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
310     NEXT M
320 NEXT N
330 A$=INKEY$: IF A$="" THEN 330
340 CLS: KEY ON: END
350 '
360 DATA -90,-90,-90,+90,-90,-90
370 DATA +90,+90,-90,-90,+90,-90
380 DATA -90,-90,+90,+90,-90,+90
390 DATA +90,+90,+90,-90,+90,+90
400 DATA 0,0,-90,0,-90,0,+90,0,0
410 DATA 0,+90,0,-90,0,0,0,0,+90
420 DATA "ABBCCDDAAEBFCGDHEFFGGHHE"
430 DATA "IJIKILIMJKKLLMMJJMKNLNMN"
```



Er is in Programma 22 echter een probleem. Toen ik de code had overgenomen en in Visual Basic 2008 had uitgevoerd, kwam ik tot de ontdekking dat er een lijn minder werd getekend. Volgens het boek moeten we de juiste data hebben om de kubus met 8-vlak te kunnen tekenen. Dat is dus niet het geval.
Indien u GW-BASIC heeft probeer dan eens bovenstaande listing en voer hem uit. Mocht het inderdaad zo zijn dat de tekening niet overeenkomt met bovenstaande tekening op het formulier, neem dan de extra gegevens over die u in onderstaand Visual Basic project kunt vinden. In de DATA regel 430 moeten twee letters bijkomen om ervoor te zorgen dat de laatste schuine lijn getekend wordt. Dat is voor mij een beste klus geweest om de juiste letters te vinden.

```
Public Class frmProg22

    Private Z() As String = _
    { _
        "ABBCCDDAAEBFCGDHEFFGGHHE", _
        "IJIKILIMJKKLLMMJJMKNLNMNNJ" _
    }
    Private Coördinaten() As Integer = _
    { _
        -90, -90, -90, +90, -90, -90, _
        +90, +90, -90, -90, +90, -90, _
        -90, -90, +90, +90, -90, +90, _
        +90, +90, +90, -90, +90, +90, _
        0, 0, -90, 0, -90, 0, +90, 0, 0, _
        0, +90, 0, -90, 0, 0, 0, 0, +90 _
    }
    Private Sub frmProg22_Paint(...) ...
        If txtA.Text().Length() > 0 And txtK.Text().Length() > 0 Then
            Dim A As Single = CSng(txtA.Text())
            Dim K As Single = CSng(txtK.Text()
            Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5
            Dim RD As Single = 4 * Math.Atan(1) / 180
            Dim W As Single = A * RD, C As Single = K * Math.Cos(W)
            Dim S As Single = K * Math.Sin(W)
            Dim X(14), Y(14) As Integer
            Dim J As Integer = 0, PC As Integer = 0
            Do
                Dim PX As Integer = Coördinaten(PC)
                Dim PY As Integer = Coördinaten(PC + 1)
                Dim PZ As Integer = Coördinaten(PC + 2)
                X(J) = Int(U + PX + C * PY + H)
                Y(J) = Int(V - S * PY - PZ + H)
                PC += 3
```

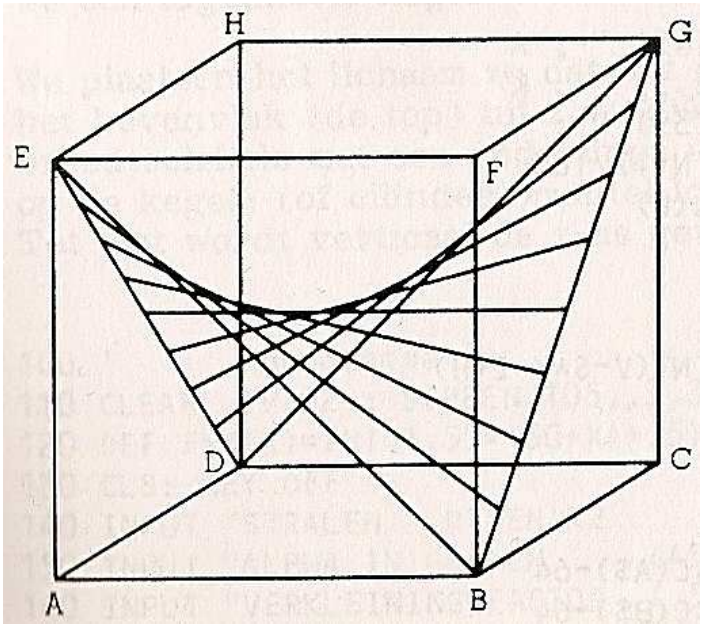
```
            J += 1
        Loop Until J >= 14
        For N As Integer = 0 To 1
            Dim L As Integer = Z(N).Length()
            For M As Integer = 1 To L - 1 Step 2
                Dim strA As String = Mid(Z(N), M, 1), I As Integer = Asc(strA) - 65
                Dim strB As String = Mid(Z(N), M + 1, 1) : J = Asc(strB) - 65
                Dim X1 As Integer = X(I), Y1 As Integer = Y(I)
                Dim X2 As Integer = X(J), Y2 As Integer = Y(J)
                e.Graphics.DrawLine(Pens.Black, X1, Y1, X2, Y2)
            Next
        Next
    End If
End Sub

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

In Visual Basic moet het programma goed aangepast worden. Bijvoorbeeld de FOR lus met variabele J die de gegevens moet ophalen en in de arrays X() en Y() moet plaatsen. Omdat we geen READ statement kunnen gebruiken en we niet variabele J kunnen nemen om uit de Coördinaten() array te lezen, moeten we per drie elementen deze eerst in drie aparte variabelen PX, PY en PZ opbergen. Bovendien moeten we een loop variabele hebben die per drie elementen telt, vandaar dat variabele PC telkens drie stappen neemt per keer. Voor deze oplossing heb ik een DO ... LOOP UNTIL lus gebruikt.
Andere veranderingen zijn de getallen die afgetrokken worden van de waarden die de ASC functies terug-geven met de opgegeven variabelen strA en strB. Waarom 65 en geen 64? Dat komt doordat GW-BASIC een basis element van een array als element 1 beschouwt en niet als element 0. Zie ook waarom ik de FOR lus met variabele N laat beginnen met waarde 0 tot en met 1 om de Z string array te kunnen lezen.

Ik hoop dat ik u voldoende informatie heb gegeven omtrent de veranderingen van Programma 22.

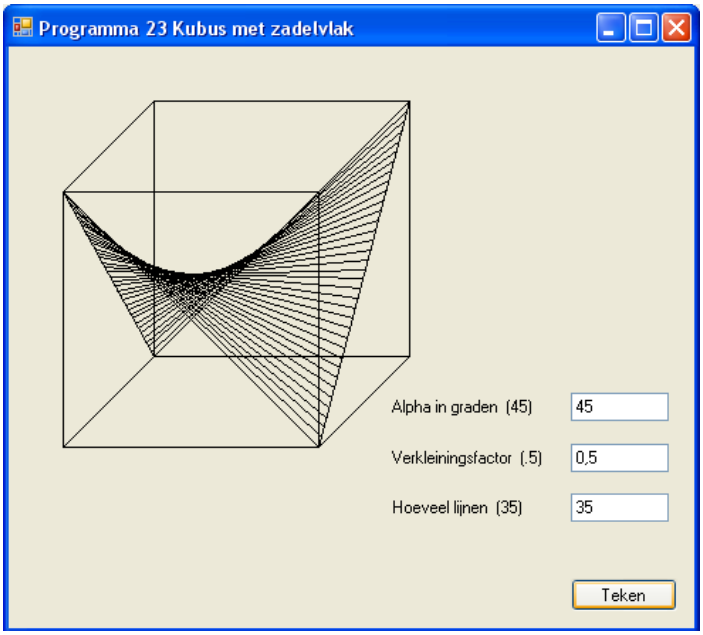
In **Programma 23** wordt in een kubus een zadelvlak getekend. Het geeft het idee van een gekromd vlak in een kubus.



Bekijk de figuur. De diagonalen BG en ED zijn elk in acht (N) gelijke stukken verdeeld en de zo ontstane punten zijn paarsgewijs door een rechte lijn met elkaar verbonden. Kies in het programma voor N de waarde 18 of 15. De figuur is een normale tekening en daarom een projectie uit het boek. De onderstaande figuur is getekend door de computer.

```
100 '    programma 23     KUBUS MET ZADELVLAK
110 CLEAR ,19202 : SCREEN 105,,3,3
120 DEF FNX(X)=INT(1.55*(50+X)+.5)
130 CLS: KEY OFF
140 INPUT "ALPHA IN GRADEN      (45) "; A
150 INPUT "VERKLEININGSFACTOR   (.5) "; K
160 INPUT "HOEVEEL LIJNEN       (35) "; N
170 U=160: V=160: H=.5 : RD=4*ATN(1)/180
```

```
180 W=A*RD: C=K*COS(W) : S=K*SIN(W)
190 DIM X(8), Y(8)
200 FOR J=1 TO 8
210    READ X,Y,Z
220    X(J)=INT(U+X+C*Y+H):Y(J)=INT(V-S*Y-Z+H)
230 NEXT J
240 CLS
250 READ Z$ : L=LEN(Z$)
260 FOR M=1 TO L-1 STEP 2
270    A$=MID$(Z$,M,1)    : I=ASC(A$)-64
280    B$=MID$(Z$,M+1,1) : J=ASC(B$)-64
290    X1=X(I): Y1=Y(I): X2=X(J):Y2=Y(J)
300    LINE (FNX(X1),Y1) - (FNX(X2),Y2),1
310 NEXT M
320 FOR J=0 TO N
330    X1=INT(X(2)+J*(X(7)-X(2))/N+H)
340    Y1=INT(Y(2)+J*(Y(7)-Y(2))/N+H)
350    X2=INT(X(5)+J*(X(4)-X(5))/N+H)
360    Y2=INT(Y(5)+J*(Y(4)-Y(5))/N+H)
370    LINE (FNX(X1),Y1)-(FNX(X2),Y2),1
380 NEXT J
390 LINE (FNX(X(2)),Y(2))-(FNX(X(7)),Y(7)),1
400 LINE (FNX(X(5)),Y(5))-(FNX(X(4)),Y(4)),1
410 A$=INKEY$: IF A$="" THEN 410
420 CLS: KEY ON: END
430 DATA -90,-90,-90,+90,-90,-90
440 DATA +90,+90,-90,-90,+90,-90
450 DATA -90,-90,+90,+90,-90,+90
460 DATA +90,+90,+90,-90,+90,+90
470 DATA "ABBCCDDAAEBFCGDHEFFGGHHE"
```




```
Private Coördinaten() As Integer = _
{ _
   -90, -90, -90, +90, -90, -90, _
   +90, +90, -90, -90, +90, -90, _
   -90, -90, +90, +90, -90, +90, _
   +90, +90, +90, -90, +90, +90 _
}
Private Z As String = "ABBCCDDAAEBFCGDHEFFGGHHE"

Private Sub frmProg23_Paint(..., ByVal e As System.Windows.Forms.PaintEventArgs) ...
    If txtA.Text().Length() > 0 And txtK.Text().Length() > 0 And _
       txtN.Text().Length() > 0 Then
        Dim U As Integer = 160, V As Integer = 160, H As Single = 0.5
        Dim RD As Single = 4 * Math.Atan(1) / 180
        Dim A As Single = CSng(txtA.Text()), K As Single = CSng(txtK.Text())
        Dim N As Single = CSng(txtN.Text())
        Dim W As Single = A * RD, C As Single = K * Math.Cos(W), _
            S As Single = K * Math.Sin(W)
        Dim X(8), Y(8) As Integer
        Dim X1, Y1, X2, Y2 As Integer
        Dim J As Integer = 0, PC As Integer = 0
        Do
            Dim PX As Integer = Coördinaten(PC)
            Dim PY As Integer = Coördinaten(PC + 1)
            Dim PZ As Integer = Coördinaten(PC + 2)
            X(J) = Int(U + PX + C * PY + H)
            Y(J) = Int(V - S * PY - PZ + H)
            PC += 3
            J += 1
        Loop Until J >= 8
        With e.Graphics
            For M As Integer = 1 To Z.Length() - 1 Step 2
                Dim strA As String = Mid(Z, M, 1), I As Integer = Asc(strA) - 65
                Dim strB As String = Mid(Z, M + 1, 1) : J = Asc(strB) - 65
                X1 = X(I) : Y1 = Y(I) : X2 = X(J) : Y2 = Y(J)
```

```
                .DrawLine(Pens.Black, X1, Y1, X2, Y2)
            Next
            For J = 0 To N
                X1 = Int(X(1) + J * (X(6) - X(1)) / N + H)
                Y1 = Int(Y(1) + J * (Y(6) - Y(1)) / N + H)
                X2 = Int(X(4) + J * (X(3) - X(4)) / N + H)
                Y2 = Int(Y(4) + J * (Y(3) - Y(4)) / N + H)
                .DrawLine(Pens.Black, X1, Y1, X2, Y2)
            Next
            .DrawLine(Pens.Black, X(1), Y(1), X(6), Y(6))
            .DrawLine(Pens.Black, X(4), Y(4), X(3), Y(3))
        End With
    End If
End Sub
```

Bekijk weer de verschillen tussen de GW-BASIC listing en het Visual Basic project. Er zijn weer veranderingen aangebracht om de juiste projectie te krijgen. De laatste lus J is het belangrijkste. De elementwaarden moeten allemaal één waarde lager om de juiste berekeningen te krijgen. Dat geldt ook voor de laatste twee DrawLine() methoden.

In Programma 23 laat ik het geraamte van de klasse weg en ook de Click event van de Teken knop laat ik weg, omdat deze hetzelfde zijn. Vergeet dus niet de Click event te maken om de methode Refresh() te kunnen aanroepen.

De volgende voorbeelden gaan omvangrijker worden en de code wat ingewikkelder. Ik hoop dat ik deze kan converteren naar Visual Basic. Ik zal in ieder geval mijn uiterste best doen. Tot nu toe is het steeds gelukt!

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

Websites programmeren met Crimson (3).

Enkele basiskenmerken van CSS.

Na ik u in de vorige nieuwsbrief verteld heb, met een stukje code, wat CSS inhoudt, kunnen we beginnen met het beschrijven van enkele basiskenmerken van CSS zodat u weet wat het gebruik van CSS kan betekenen voor uw site.

Eenvoudige en eenduidige syntaxis.
De syntaxis is de schrijfwijze van CSS. Het geeft aan hoe de taal CSS genoteerd wordt en deze is bepaald eenvoudig te noemen. Alle stijlen die u schrijft hebben dezelfde opbouw:

- eerst wordt een selector genoemd;
- vervolgens staan tussen accolades de attributen of properties van de selector die u wilt beschrijven;
- na de attribuutnaam volgt een dubbelepunt;
- daarna volgt de waarde die het attribuut krijgt;
- elk attribuut wordt afgesloten met een puntkomma;
- als u alle attributen hebt gehad, volgt een sluitaccolade die de stijl afsluit.

Dat is het. Niet meer en niet minder, geen uitzonderingen. Deze notatie geldt daarom voor elke stijl die u in het stijlenbestand wilt opnemen. De declaratie in het codefragment is daarom volkomen correcte CSS.

```
body
{
    color : #FF99CC;
    background : #FFFFFF;
    font-family : arial, sans-serif;
    margin : 10;
    padding : 10;
}
```

Wat nu weer: selector, attributen?
Onderstaand tabel geeft de begrippen als selector, declaratie en keyword. Dan hebt u wat meer houvast.

	keyword	waarde	keyword	waarde
body	{color:	blue;	border-bottom:	1px; }
selector	Declaratie		Declaratie	

CSS is net als HTML niet gevoelig voor spaties, tabs en regelovergangen (Enters). Witruimte wordt genegeerd, zolang maar aan de hiervoor beschreven syntaxiseisen is voldaan. De volgende declaratie heeft daarom precies hetzelfde effect als de hiervoor genoemde code:

```
body{color:#FF99CC;background:#FFFFFF;font-family:arial,sans-serif;margin:10;padding:10;}
```

De mogelijkheden van CSS voor alle elementen.

In ouderwets HTML bent u afhankelijk van het feit of er bepaalde attributen voor een tag beschikbaar zijn die het uiterlijk van de tag beïnvloeden. Zo kent een afbeelding bijvoorbeeld het attribuut border, <img border="1">, maar <p border="1"> heeft geen enkel effect en de alinea kent deze dan ook niet.
Voor tabellen is border="1" wel weer beschikbaar, maar zien de randen er anders uit dan wanneer border wordt gebruikt bij <img>. En wat wil die "1" eigenlijk zeggen. Centimeters? Pixels? Inches? Dat is niet aan te geven. Kortom, het is willekeur troef.

CSS verandert dit. Alle mogelijkheden van CSS zijn beschikbaar voor alle elementen.

- Met CSS kunt u de kleur en achtergrond bepalen voor elk element op de pagina.
- U kunt elk gewenst element een rand geven en zeer nauwkeurig aangeven hoe die rand er moet uitzien, welke dikte hij moet hebben en welke kleur.
- Desgewenst kunt u zelfs voor één element de boven-, linker-, onder- en rechterrand apart instellen. Dit is bijvoorbeeld handig om lijnen tussen kolommen te plaatsen.
- U kunt een achtergrondafbeelding koppelen aan elk gewenst element.
- Het is mogelijk precies de witruimte binnen elementen en rondom elementen in te stellen met padding en margin. En zo kunnen we nog wel even doorgaan. Om precies te zijn: in de komende nieuwsbrieven zal ik veel meer van CSS laten zien.

Samenvattend: alles wat met HTML mogelijk was, kan ook met CSS. Alles wat u met CSS kunt bereiken, kan bij lange na niet met HTML.

Stijlen hergebruiken.
Een van de grootste winstpunten van CSS is waarschijnlijk dat een stylesheet op elke gewenste pagina van de site kan worden gebruikt. Geen dubbele, tien- of honderdvoudige code meer voor elke pagina in de site; geen uitgebreide 'zoek en vervangopdrachten' meer als een lettertype of letterkleur site-wide moet worden aangepast of een kolom 10 pixels breder moet worden. Het aanpassen van de stylesheet volstaat.

Staat in een stylesheet de opdracht om alle kopteksten blauw te maken? Dan ziet deze code er ongeveer zo uit als:

```
h1, h2, h3 {color: blue;}
```

Als uw opdrachtgever besluit liever toch rood te gebruiken, dan hoeft u de code alleen als volgt aan te passen...

```
h1, h2, h3 {color: red;}
```

...en elke pagina waarop deze stylesheet van toepassing is, zal automatisch goed worden weergegeven in de browser. Het spreekt voor zich dat dit het bijwerken van een site enorm versnelt en nuttig is voor de kosten die zeer laag zullen zijn. Het sneller kunnen werken aan een eigen site met gebruik van een stylesheet is het grote voordeel waardoor de kosten laag worden.

Ruimte- en bandbreedtebesparend door caching in de browser.
Als u inderdaad een basisstylesheet gebruikt voor uw site, dan is deze stylesheet na het opvragen van de eerste pagina door een bezoeker aanwezig in de cache van zijn/haar browser. Dat betekent dat bij het opvragen van een volgende pagina van de site de stylesheet niet opnieuw meegezonden hoeft te worden. Dit heeft verschillende voordelen:

- De weergave van de pagina in de browser wordt versneld omdat minder code gedownloaded hoeft te worden (client-sided voordeel).
- De opmaakcode hoeft niet met elk document worden meegezonden waardoor minder bandbreedte wordt gebruikt door de webserver (server-sided voordeel). Uw server presteert daardoor beter, tegen lagere kosten. Of andersom geredeneerd: tegen dezelfde kosten kunt u een groter aantal bezoekers bedienen.

Heeft een specifieke pagina van uw site toch afwijkende, of aanvullende opmaak nodig? Koppel in dat geval gewoon een extra stylesheet aan de pagina waarin deze opmaak wordt beschreven. Het is zonder meer mogelijk verschillende stylesheets te koppelen aan een pagina. De opdrachten in de extra stylesheets vullen elkaar aan of overschrijven elkaar.

Trapsgewijze invulling (cascading).
Het woord *cascading* in Cascading Style Sheets is niet voor niets gekozen. Dit refereert namelijk aan de wijze waarop stylesheets elkaar kunnen aanvullen of overschrijven. De term cascading wordt in het Nederlands vaak vertaald met 'trapsgewijs'. En dat geeft de werking eigenlijk wel goed weer.

Stel dat u een basisstylesheet hebt gemaakt waarin de stijl voor <body> wordt gedefinieerd met een witte achtergrond. Het kan echter zijn dat u voor bepaalde pagina's binnen de site een anders gekleurde achtergrond nodig hebt. In de stylesheet voor die pagina schrijft u dan een declaratie zoals:

```
body
{
    background: #5E5E5E;
}
```

Op de pagina's waarin deze stylesheet aanvullend wordt gekoppeld in combinatie met de basisstylesheet, zal de pagina een donkergrijze (#5E5E5E) achtergrond hebben omdat de eigenschap background is overschreven. Tegelijkertijd hebben echter de eigenschappen van <body> die niet zijn overschreven, zoals het lettertype en de tekstkleur hun eigenschappen uit de basisstylesheet behouden! Deze hoeven dus niet nogmaals te worden gedeclareerd.

CSS in de praktijk.
Na de theorie weten we dat de praktijk de beste leermeester is. Daarom nemen we nog eens een voorbeeld dat ook in de vorige nieuwsbrief stond.

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


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

Hieronder staat een selector met tussen accolades de declaraties. Typ de regels in bovenstaand voorbeeld tussen de regels <style type="text/css">...</style> in de lege ruimte:

```
body
{
    color: #FF99CC;
    background: #5E5E5E;
    font-family: arial, sans-serif;
    margin: 10;
    padding: 10;
}
```

We mogen meer selectors toevoegen, zolang we maar tussen de openingsstyletag en de sluitstyletag blijven. Iedere selector heeft zijn eigen eigenschappen. Zo kunnen we bepalen hoe we een titel met tekstgrootte h1 willen opmaken door onderstaande selector achter de body selector toe te voegen:

```
h1
{
    color: white;
    font-size: 220%;
    font-family: Times New Roman;
}
```

Kunt u zelf ontleden wat in deze regel wordt ingesteld? Inderdaad, de letterkleur wordt ingesteld op wit, de lettergrootte wordt ingesteld op 220% ten opzichte van de originele lettergrootte en het lettertype wordt ingesteld op Times New Roman. U mag zowel hexadecimale kleurwaarden gebruiken als de namen van de vaste standaardkleuren (black, white, red, blue enzovoort). Meer over HTML kleurwaarden leest u op www.w3schools.com/HTML/html_colornames.asp.

Als u alles ingevoerd hebt dan kunt u het testen door via het menu Tools te klikken op View in Browser. U kunt, wat u schrijft in de Crimson Editor, direct testen door uw project uit te voeren via het menu-item. Hebt u nog geen project aangemaakt dan is het uitvoeren van enkele pagina's ook mogelijk.
Hoe we meer pagina's tot één website kunnen bouwen lijkt een grote klus, maar Crimson geeft ons een mogelijkheid om, door middel van een project en een map waar alle pagina's, stylesheets, logs en andere informatie in opgeslagen wordt, alles op die manier bij elkaar te houden. Dat betekent dat in elk project dat u in Crimson opent direct alle pagina's en stylesheets aanwezig zijn. U hoeft dus niet alles zelf apart te openen. Gebruikt u niet de Crimson Editor maar een normaal schrijfblok (wordpad of kladblok) dan zal het veel lastiger worden om alles bij elkaar te houden wanneer uw site groter wordt.
Meer over Crimson projecten behandel ik in de volgende nieuwsbrief.

Marco Kurvers

Appendix – nieuwe conversietabellen.

De conversietabellen die ik een aantal nieuwsbrieven terug ook had geplaatst, zullen nogmaals worden gegeven. De ruimte was toen zeer gering omdat de nieuwsbrieven nog in A5 formaat werden geschreven. Nu is er meer ruimte en kunnen de tabellen veel duidelijker worden weergegeven.

Toch zullen ze niet meer hier in komen te staan. In vervolg zullen het aparte bestanden worden en worden ze steeds bijgewerkt in Excel.

Er zal ook een nieuw tabel bijkomen. Nu met alle Basic mogelijkheden aan de linkerkant en aan de bovenkant alle andere soorten scripts en programmeertalen. Dat ga ik doen zodat u programma's uit andere talen kunt converteren naar Basic. Let wel op dat de Basic mogelijkheden van allerlei soorten versies er zullen staan. Raadpleeg de Basic versie converteertabellen als u wilt weten welk bepaald statement in welke Basic versie bestaat.

De conversietabellen komen er dus aan.

Cursussen

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

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

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

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

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

Software

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

Hoe te bestellen

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

ABN-AMRO nummer 49.57.40.314
HCC BASIC ig
Haarlem

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

Alle prijzen zijn inclusief verzendkosten voor Nederland en België.

Vraagbaken

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

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