Basic Bulletin

21ste jaargang juni 2014 Nummer 2

HCC BASIC interessegroep

Inhoud

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	06-30896598	m.a.kurvers@live.nl
Ledenadministratie	Fred Luchsinger	0318-571187	f.luchsinger@kader.hcc.nl
Webmaster	Jan van der Linden	071-3413679	j.vd.linden@kader.hcc.nl

http://www.basic.hcc.nl

Redactioneel

Voor wie de vraagbaak nog gebruikt, zal ik het volgende meedelen. Sinds kort maak ik haast geen gebruik meer van hccnet.nl. Als u wat naar mij toe wilt sturen, dan reageer ik in vervolg alleen nog maar op berichten met het email adres: m.a.kurvers@live.nl

Nu ik gebruik maak van dat email adres, is het niet meer van belang wat voor vragen het zijn. Wilt u bijvoorbeeld weten hoe een website wordt gebouwd of hoe de programmeertaal Delphi werkt of misschien heeft u meer interesse in Game Maker? Dan zijn zulke vragen geen probleem.

Maar omdat ik redacteur blijf hoeft u zich geen zorgen te maken dat Basic vragen niet meer kunnen, want ook daar blijf ik op reageren.

In dit Bulletin zijn weer artikelen binnen gekomen. Ik wens u weer heel veel lees- en uitprobeer plezier.

Marco Kurvers

BBC BASIC for Windows – De library's (1).

Een library installeren en gebruiken.

Voordat we de Windows functies en procedures kunnen gebruiken, moeten we het juiste library opgeven. Dit moeten we doen met het sleutelwoord INSTALL.

Het is een instructie waarmee u een bibliotheekbestand met een of meer vooraf gedefinieerde functies en procedures in kunt laden. Deze functies en procedures kunnen worden aangeroepen door de naam vanuit uw programma, maar ze worden niet weergegeven in de lijst van het programma. Als u, of iemand anders, een aantal handige functies vanuit verschillende programma's geschreven heeft aan kunt roepen, is dit een handige manier van het verpakken en verdelen ervan.

U kunt zoveel bibliotheekbestanden op hetzelfde moment installeren als u wilt, zolang er nog voldoende geheugen over is. Deze bibliotheken blijven resident aanwezig totdat het BASIC uitvoerbare programmavenster wordt gesloten (ze worden niet gewist door een CHAIN statement). Als twee of meer bibliotheken een procedure of functie met dezelfde naam bevatten, zal deze worden gebruikt in de laatste geladen bibliotheek.

Het HIMEM statement
De bibliotheken zijn altijd boven HIMEM geladen, dus als u HIMEM met een waarde hoger uitvoert dan de waarde van de eerste bibliotheek die geïnstalleerd werd, zullen alle bibliotheken worden verwijderd uit het geheugen.

Een bibliotheekbestand is een standaard BBC BASIC interne-indeling (tokenised) programmabestand, behalve dat het normaal gesproken alleen procedure- en functie definities moet bevatten. Regelnummers en labels worden niet erkend in bibliotheekbestanden, vandaar dat GOTO en GOSUB statements niet gebruikt kunnen worden, en de enige variant van de RESTORE instructie die nuttig is, is het RESTORE + n vorm dat wordt gebruikt in combinatie met lokale gegevens en gegevens herstellen.

Een bibliotheek laden en gebruiken
U moet het voorvoegsel van de bestandsnaam met elke @lib (voor standaard bibliotheken) of @dir$ (voor bibliotheken die specifiek aan uw programma gericht zijn) opgeven, zodat het bestand vanuit de juiste locatie geladen zal worden, ongeacht de huidige instelling van de directory. Zie de sectie van de bibliotheekroutines voor details van de meegeleverde bibliotheken.

```
INSTALL @lib$+"arraylib"
INSTALL @dir$+"mylib"
```

Alleen BBC BASIC for Windows versie 5.91a of later
INSTALL controleert om te zien of een bibliotheek al geladen is, en zo ja, dan zal er niets gebeuren. Merk op dat een hoofdlettergevoelige vergelijking van de meegeleverde pad/bestandsnaam anders wordt uitgevoerd, zodat dezelfde opgegeven bibliotheek niet zal worden gedetecteerd.

Het HIMEM statement in details
Eigenlijk is HIMEM geen statement, maar een pseudo-variabele. Een pseudo-variabele die het adres van de eerste byte boven het programmageheugen voor de gegevens en stack van BBC BASIC for Windows bevat. Het programma en de gegevens van de gebruiker gaan opwaarts vanuit LOMEM en de stack benedenwaarts vanuit HIMEM. Als de twee elkaar ontmoeten, geeft dat een untrappable 'No room' foutmelding. Meer details daarover, zie in de volgende Bulletin.

Standaard is HIMEM ingesteld op iets minder dan 1 Mbyte boven LOMEM (dit wordt bepaald door de instelling van het oorspronkelijke gebruiker geheugen). Als dit onvoldoende is voor uw programma- en gegevensbestanden, kunt u het met HIMEM verhogen (afhankelijk van voldoende geheugen in de computer). Dit is een belangrijk verschil van andere versies van BBC BASIC, waarin u nooit met HIMEM boven de oorspronkelijke waarde verhogen moet.

Laat je niet verleiden om HIMEM te verhogen met een *absolute* waarde (geheugen adressen zijn toegewezen door Windows ™ en variëren van sessie naar sessie). Stel het in plaats daarvan altijd *relatief* in met PAGE, LOMEM of TOP. Stel HIMEM in met een exact veelvoud van vier voor de beste prestaties, u kunt dat bereiken door te ANDen met -4 (&FFFFFFFC).

Als u een geheugengebied dat niet zal worden aangetast door CHAIN of CLEAR (bijvoorbeeld gegevens tussen programma's) nodig hebt, kunt u HIMEM verlagen van de oorspronkelijke waarde. Het gebied van geheugen tussen de nieuwe waarde van HIMEM en de oude waarde van HIMEM (–1) zal niet worden gebruikt door BBC BASIC for Windows (behalve wanneer de INSTALL instructie de eerste bibliotheek laadt).

HIMEM mag niet worden gewijzigd binnen een subroutine, procedure of functie; in een FOR … NEXT, REPEAT … UNTIL, of WHILE … ENDWHILE lus evenmin binnen een meer-regel IF … ENDIF blok. Als u toch HIMEM wilt wijzigen dan is het beter om dat aan het begin in uw programma te doen.

```
HIMEM = PAGE+10000000
HIMEM = (LOMEM+10000000) AND -4
HIMEM = HIMEM-40
```

Syntax

```
HIMEM=<numeric>
<n-var>=HIMEM
```

Array en matrix functies

De ARRAYLIB bibliotheek bevat een set van procedures en functies voor het uitvoeren van rekenkundige en matrix bewerkingen op 1- en 2-dimensionale arrays. Deze omvatten het toevoegen van twee matrices, het vermenigvuldigen van twee matrices, omzetting van een matrix en een matrix omkeren.

De bibliotheek moet worden geladen vanuit uw programma met behulp van de opdracht:

```
INSTALL @lib$+"ARRAYLIB"
```

De functies in de bibliotheek zijn:

- PROC_add
- PROC_mul
- PROC_sum
- PROC_dot
- PROC_transpose
- PROC_invert
- FN_mod
- FN_det

In BBC BASIC for Windows versie 3.00a en later zijn verscheidene operaties opgenomen binnen de interpreter; zie de sectie '***Rekenkundige arrays'*** voor details. Met behulp van de ingebouwde operaties zal het allemaal veel sneller werken dan met behulp van de bibliotheek routines.

De bibliotheek routines die niet worden ondersteund als ingebouwde bewerkingen zijn PROC_transpose, PROC_invert en FN_det.

PROC_add(A(), B)

PROC_add voegt een scalaire waarde B toe aan alle elementen van de 1ste dimensie of 2de dimensie (numerieke) matrix A() en resulteert het resultaat in A():

```
DIM N(3)
N() = 1, 2, 3, 4
PROC_add(N(), 5)
PRINT N(0),N(1),N(2),N(3)
```

De uitvoer van dit programma zal zijn:

PROC_mul(A(), B)

PROC_mul vermenigvuldigt alle elementen van de 1ste dimensie of 2de dimensie (numerieke) matrix A() door de scalaire waarde B en geeft het resultaat in A():

```
DIM N(3)
N() = 1, 2, 3, 4
PROC_mul(N(), 2)
PRINT N(0),N(1),N(2),N(3)
```

De uitvoer van dit programma zal zijn:

```
2      4      6      8
```

PROC_sum(A(), B())

PROC_sum telt de 1ste dimensie of 2de dimensie (numerieke) arrays A() en B() samen op en geeft het resultaat in A(). A() en B() moeten dezelfde dimensies hebben.

```
DIM N(3), S(3)
N() = 1, 2, 3, 4
S() = 5, 6, 7, 8
PROC_sum(N(), S())
PRINT N(0),N(1),N(2),N(3)
```

De uitvoer van dit programma zal zijn:

```
6      8      10     12
```

PROC_dot(A(), B(), C())

PROC_dot vermenigvuldigd 2D matrices A() met B() en geeft het resultaat in C(). Het aantal kolommen voor A() moet gelijk zijn aan het aantal rijen van B(), het aantal kolommen voor C() moet gelijk zijn aan het aantal kolommen voor B() en het aantal rijen van C() moet gelijk zijn aan het aantal rijen van A().

```
DIM N(0,2), S(2,1), D(0,1)
N() = 1, 2, 3
S() = 4, 5, 6, 7, 8, 9
PROC_dot(N(), S(), D())
PRINT D(0,0) D(0,1)
```

De uitvoer van dit programma zal zijn:

```
40     46
```

PROC_transpose(A(), B())

PROC_transpose zet 2D matrix A() om en het resultaat komt in B(). Het aantal kolommen voor A() moet gelijk zijn aan het aantal rijen van B() en het aantal rijen van A() moet gelijk zijn aan het aantal kolommen voor B().

```
DIM N(1,2), T(2,1)
N() = 1, 2, 3, 4, 5, 6
PROC_transpose(N(), T())
PRINT T(0,0) T(0,1)
PRINT T(1,0) T(1,1)
PRINT T(2,0) T(2,1)
```

De uitvoer van dit programma zal zijn:

```
1      4
2      5
3      6
```

PROC_invert(A())

PROC_invert keert de vierkant matrix A() om en het resultaat komt in A().

```
DIM M(2,2)
M() = 2,0,6,8,1,-4,0,5,7
PROC_invert(M())
PRINT M(0,0) M(0,1) M(0,2)
PRINT M(1,0) M(1,1) M(1,2)
PRINT M(2,0) M(2,1) M(2,2)
```

De uitvoer van dit programma zal zijn:

```
0.09184      0.10204      -0.02041
-0.19048     0.04762      0.19048
0.13605      -0.03401     0.00680
```

FN_mod(A())

FN_mod geeft als resultaat de absolute waarde (de vierkantswortel van de som van de kwadraten van alle elementen) van een 1D- of 2D-matrix.

```
DIM M(2,2)
```

```
M() = 2,0,6,8,1,-4,0,5,7
PRINT FN_mod(M())
```

De uitvoer van dit programma zal zijn:

```
13.96424
```

FN_det(A())

FN_det geeft als resultaat de determinant van een vierkante matrix.

```
DIM M(2,2)
M() = 2,0,6,8,1,-4,0,5,7
PRINT FN_det(M())
```

De uitvoer van dit programma zal zijn:

```
294
```

Visual Basic 2010 – Het verschil met de oudere versies.

De zwarte dozen

Visual Basic 2010 is al veel verder dan 2003. Versie 2003 lijkt dus sterk verouderd, maar toen al bestond de allereerste Microsoft .NET Framework waarmee we voor het eerst zonder zwarte dozen konden werken.
Wie niet die uitdrukking kent, zou graag willen weten wat ik met een zwarte doos bedoel. Wie dat wel kent, mag best de onderstaande alinea's overslaan.

In Visual Basic 6 bestonden er alleen maar losse componenten en formuliersjablonen. Er waren geen object instanties ervan, dus konden we ook niets laten erven van elkaar. Er bestond geen OOP (Object Oriented Programming) waarmee we dus niet alles georiënteerd konden laten werken.

Hoewel de componenten als objecten gebruikt moesten worden, was het onmogelijk zelf een instantie te creëren van een component. Gelukkig was er een mogelijkheid een component array te maken, maar dat moest altijd in de design worden aangemaakt, in de IDE modus dus.
Met de formulieren is het precies zo het geval. Alles van een formulier wordt verborgen gehouden als een zwarte doos zoals in een vliegtuig. Onderstaande code is bijvoorbeeld onmogelijk en Visual Basic gaf hier dan ook direct een foutmelding:

```
Dim NewForm As New Form
```

En we hoeven ook niet te proberen het zonder New te doen, want om een object te gebruiken moet deze geset (gecreëerd) worden, zodat we weer die foutmelding krijgen.
We kunnen wel een formulier aanmaken, maar dan kan dat alleen van een bestaand formulier, bijvoorbeeld van Form1. Natuurlijk zal alles wat erop staat, zoals de componenten, ook in het object aanwezig zijn. Daarom is de conclusie:

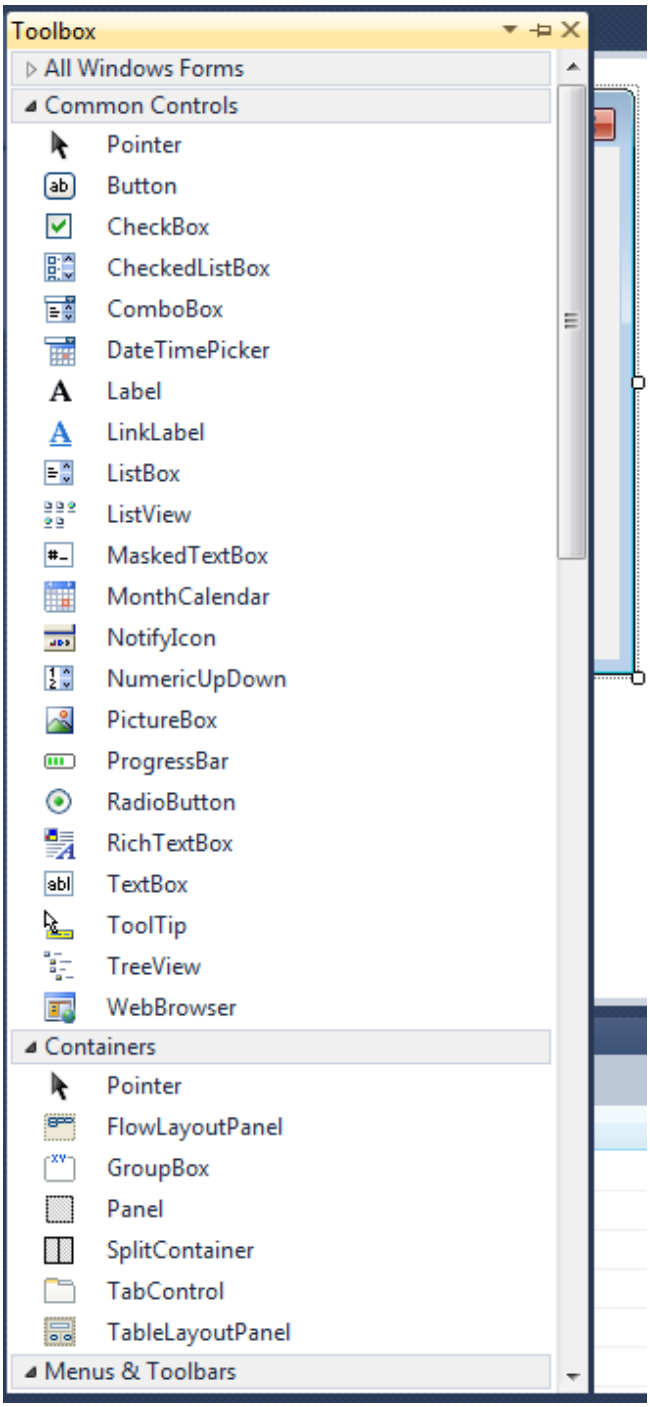
In Visual Basic 6 kunnen we **geen** instanties van formuliersjablonen aanmaken in de code, maar we kunnen wel bestaande formulieren klonen. We mogen ook een heel ander object naar een bestaand formulier laten verwijzen, waardoor het mogelijk is formulierobjecten als parameters te gebruiken.

Omdat het gebruik toen van componenten en formulieren zeer beperkt was, liep het programmeren van applicaties niet altijd op rolletjes. We moesten soms wel onderdelen hebben die functies hadden die in VB 6 gewoon niet te maken waren.

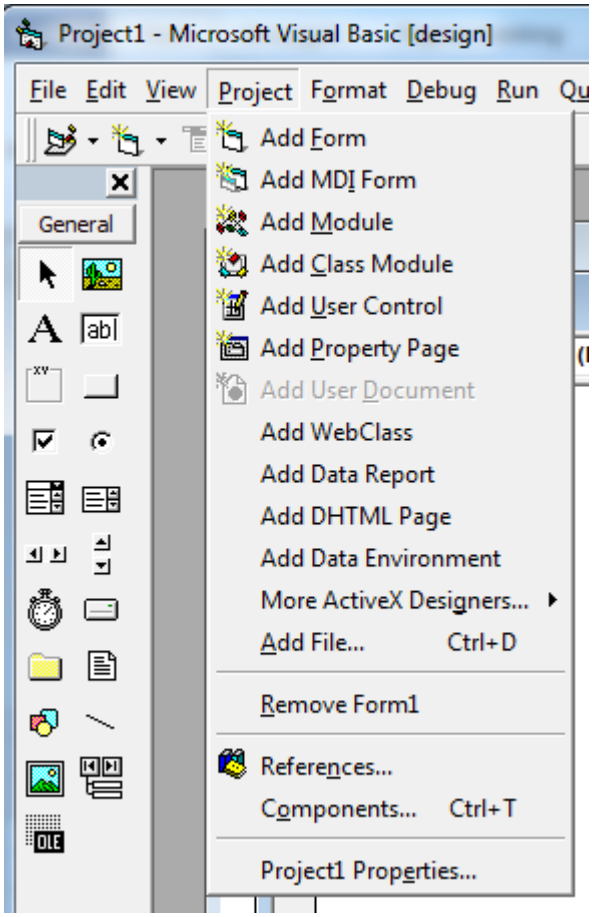
De moderne Basic versies

Vanaf 2003 hebben we nu moderne Basic versies die nu alles hebben wat we gebruiken willen. Dit betekent dus geen zwarte dozen meer en we mogen nu zelfs eigen formulieren creëren. Maar dat betekent ook dat we ons aan de nieuwe mogelijkheden aan moeten passen. Een aantal oude mogelijkheden zijn niet meer geldig. Oude programma's moeten correct omgezet worden, ook al bestaat er een converteerwizard.

De design op de IDE



De IDE lijkt nog steeds op die van versie 6, maar al snel is te zien dat de componentenlijst behoorlijk is veranderd. Alles staat nu in groepen onderverdeeld die in- en uitgeklapt kunnen worden. Gaan we met de muis over de toolbox, dan kunnen we de hele toolbox met alle groepen in- en uitklappen, zie afbeelding links.

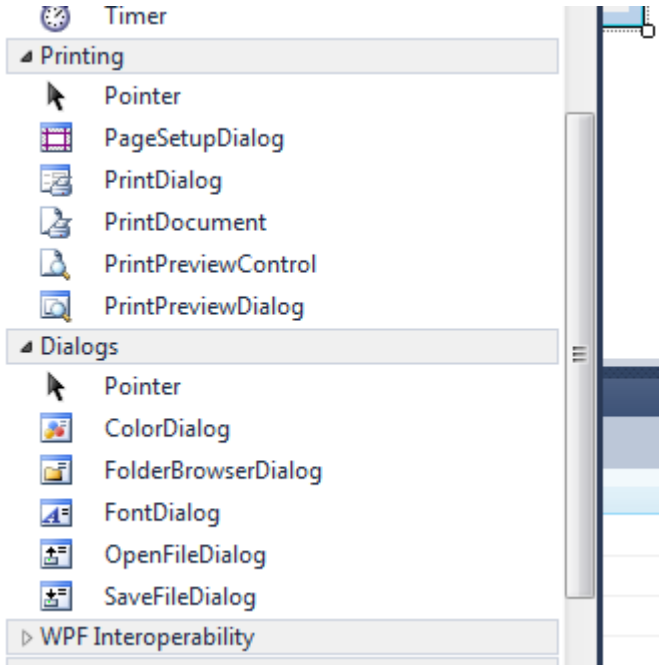


Dat is, zoals we de bovenste afbeelding zien, niet het geval. Visual Basic 6 geeft een lijst met alleen de belangrijkste componenten. De andere componenten en ActiveX onderdelen moeten opgezocht worden in een apart dialoogvenster. De componenten vindt u bij **Components…** en de ActiveX onderdelen en de library's vindt u bij **References…**. Maar denk maar niet dat de toolbox alles heeft wat er in VB 6 ge-

bruikt kan worden. Bepaalde oude componenten en componentbesturingen bestaan niet meer, maar zijn bijvoorbeeld vervangen door nieuwe componenten met betere functionaliteiten.

De CommonDialog control
Een voorbeeld is de oude CommonDialog control. Deze control was zelf geen component, maar had verborgen dialoogformulieren voor het besturen van het 'bestanden openen' venster, het 'bestanden opslaan' venster, het 'kleuren' venster, het 'lettertype' venster en het 'afdruk' venster. Deze dialoogvensters bestaan nog steeds, maar staan nu als aparte besturingscomponenten in de toolbox.

Hiernaast ziet u een deel van de toolbox met twee groepen waarvan de meeste besturingscomponenten aanwezig waren in de CommonDialog control. Er zijn echter nieuwe bijgekomen in de Printing groep, behalve de PrintDialog die we allemaal nog wel in VB 6 kennen.



De dialoogvensters van de besturingscomponenten werken nu veel gemakkelijker. De gegevens worden veel sneller doorgegeven en teruggegeven. Alleen de MessageBox methode is al voldoende om te bepalen op welke knop geklikt is. De oude MsgBox kan dat ook bepalen, maar het resultaat is altijd een muisklik van het berichtvenster en niet van een dialoogvenster. MsgBox is daarom altijd een op zichzelf werkend commando.

De converteerwizard
Jammer genoeg kan de converteerwizard een oud project niet 100% correct converteren. Er zullen altijd fouten in blijven, zoals het gebruik van de variabelen. De wizard zal niet de declaraties van de variabelen op de nieuwe manier omzetten en dat komt doordat de oude manier ook nog steeds werkt, maar de oude manier zal ook niet altijd relevant zijn. Een ander nadeel is ook dat de wizard niet controleert of bepaalde variabelen wel gedeclareerd zijn. Vanaf VB 2003 is het verplicht variabelen te declareren. Zomaar plotseling een variabele gebruiken is niet meer toegestaan en dat geldt ook voor de symbolen. Stringvariabelen kunnen niet meer met een dollarteken geïnitialiseerd worden, en dat geldt voor alle soorten variabelentypen. Functies geven nu een waarde terug van een bestaand type, ze hebben dus geen symbool meer, zoals de Mid$ functie had, en ze kunnen ook niet meer variant zijn. Ook bij de variabelen trouwens niet. Het Variant type wordt niet meer ondersteund.

Enkele voorbeelden

Wilden we in VB 6 drie variabelen declareren van het type Integer, dan dachten we dat onderstaande regel juist was:

```
Dim I, J, K As Integer
```

Maar dat is echter niet het geval. De variabelen I en J werden variant variabelen en alleen K werd een Integer. Hoe gek dat ook eruit ziet, we moeten in versie 6 echt onderstaande regel gebruiken:

```
Dim I As Integer, J As Integer, K As Integer
```

Gelukkig hoeven we in modern Basic dit niet meer te doen. Nu is de eerste regel wel juist en worden alle drie variabelen Integer variabelen. Zouden we de converteerwizard gebruiken, dan hoeven we niet te denken dat dan de tweede regel netjes geconverteerd wordt in de eerste regel. De tweede regel is niet fout en daarom doet de wizard er ook niets aan.

Het gebruik van eigen typen is echter een andere zaak. Het Type sleutelwoord kan niet meer gebruikt worden. Om eigen typen te maken, moeten we gebruik maken van het sleutelwoord Structure.
Als de wizard een recordtype ziet staan, wordt deze helaas niet geconverteerd in een structure en daarom is de wizard niet 100% oké.

Hebben we een variabele gedeclareerd, dan mogen we die initialiseren:

```
Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBa-
se.Load
      Dim I As Integer

      I += 3
      Debug.Write(I)
End Sub
```

In Visual Basic 6 ziet dat er zo uit:

```
Private Sub Form1_Load()
      Dim I As Integer

      I = I + 3
      Debug.Print I
End Sub
```

Eigenlijk wil Visual Basic 2010 dat gelijk achter de declaratie een initialisatie gegeven wordt. Daarom zal altijd, wanneer een variabele gedeclareerd wordt, een kronkelend streepje onder de variabele liggen omdat gelijk een toekenning verwacht wordt. Doen we dat niet, dan vindt Visual Basic 2010 het prima, maar alsnog wordt er een initialisatie verwacht. Een variabele controleren met een If … Then zonder dat de variabele een waarde heeft, leidt tot het stoppen van de uitvoer van het programma.

We kunnen bijvoorbeeld zeggen:

```
Dim I As Integer = 3     ' Dit kan niet in Visual Basic 6
```

We zouden dan denken dat dit ook kan:

```
Dim I As Integer += 3    ' Helaas, dit is niet toegestaan
```

We weten nu dat we direct erachter de toekenning mogen opgeven, maar er geldt ook weer een wet:

```
Dim I, J As Integer = 10 ' Dit is niet toegestaan
```

We kunnen maar één variabele per declaratie initialiseren.

Maar we mogen ook variabelen op een andere manier declareren, bijvoorbeeld in een For lus:

```
For N As Integer = 1 To 10

Next N
```

Er is heel wat veranderd en bijgekomen, zoals nieuwe operatoren die al veel eerder in andere programmeertalen bestonden. Basic gaat dus achter de feiten aan.

Een calculator programma in Liberty BASIC.

Een calculatorprogramma schrijven.

Vaak geef ik aan de cursisten huiswerkopgaven. Daarna laat ik een van de beste voorbeelden zien die door een cursist geschreven is. Tevens geef ik ook uitleg over de code. Ik gaf de volgende huiswerk opgave: Schrijf een calculatorprogramma in Liberty Basic waarbij je gebruik mag maken van de volgende afbeeldingen en een wave bestand.

Het valt op dat dit slechts drie toetsen zijn met een achtergrond. De cursisten moesten zelf de tekens op de toetsen plaatsen. Dat moest met enig beleid gedaan worden, anders krijg je toetsen zoals de toets hier rechts. De achtergrond van het teken 4 op de toets is wit en dat staat lelijk op de mooie toets. Gelukkig heeft Microsoft Windows daar wat op gevonden. Je kunt de achtergrond van de tekens doorzichtig maken. Daar heb je de SetBKmode functie voor nodig. Je moet ook het nummer van de DC (Device Control) kennen. Deze gegevens worden op de MSDN site goed besproken. Gevorderde Liberty BASIC programmeurs hebben dat natuurlijk al uitgezocht.

Ik gaf een testbestand aan de cursisten mee. Niet elke cursist is een gevorderde. Met het testbestand konden de toetsen naar eigen smaak opgemaakt worden. Hier volgt de uiteindelijke listing van een cursist. Ik zal de listing bespreken.

```
nomainwin

      WindowWidth = 267
      WindowHeight = 420
      UpperLeftX=int((DisplayWidth-WindowWidth)/2)
      UpperLeftY=int((DisplayHeight-WindowHeight)/2)

      global R$

      loadbmp "calcBG", "calculatortest1.bmp"

      bmpbutton #main.button7,"CijferKnop7.bmp",cijferClick, UL, 020, 130
      bmpbutton #main.button8,"CijferKnop8.bmp",cijferClick, UL, 080, 130
      bmpbutton #main.button9,"CijferKnop9.bmp",cijferClick, UL, 140, 130

      bmpbutton #main.button4,"CijferKnop4.bmp",cijferClick, UL, 020, 180
      bmpbutton #main.button5,"CijferKnop5.bmp",cijferClick, UL, 080, 180
      bmpbutton #main.button6,"CijferKnop6.bmp",cijferClick, UL, 140, 180

      bmpbutton #main.button1,"CijferKnop1.bmp",cijferClick, UL, 020, 230
      bmpbutton #main.button2,"CijferKnop2.bmp",cijferClick, UL, 080, 230
      bmpbutton #main.button3,"CijferKnop3.bmp",cijferClick, UL, 140, 230

      bmpbutton #main.button0,"CijferKnop0.bmp",cijferClick, UL, 020, 280
      bmpbutton #main.buttonP,"CijferPunt1.bmp",cijferClick, UL, 080, 280
```

```
     bmpbutton #main.buttonD,"PowerKnop3.bmp",powerClick, UL, 200, 130
     bmpbutton #main.buttonV,"PowerKnop2.bmp",powerClick, UL, 200, 180
     bmpbutton #main.buttonA,"PowerKnop0.bmp",powerClick, UL, 200, 230
     bmpbutton #main.buttonO,"PowerKnop1.bmp",powerClick, UL, 200, 280

     bmpbutton #main.buttonI,"IsGelijkKnop1.bmp",[buttonIClick], UL, 145, _
               330

     bmpbutton #main.buttonC,"PowerKnop5.bmp",[buttonCClick], UL, 020, 330
     bmpbutton #main.buttonB,"PowerKnop6.bmp",[buttonBClick], UL, 140, 280
     'bmpbutton #main.button20,"PowerKnop4.bmp",[buttonPMClick], UL, 95, 80

     TextboxColor$ = "green"
     textbox #main.textbox21, 20, 40, 230, 40
     stylebits #main.textbox21, _ES_RIGHT,0,0,0

     graphicbox #main.g, 0,0,267,420
     stylebits #main.g, 0,_WS_BORDER,0,0

     open "untitled" for window _popup as #main
     #main.g "down; drawbmp calcBG 0 0"
     #main.textbox21 "!font 15"
     #main "trapclose [quit.main]"
     #main.g "flush"
     wait

sub cijferClick handle$
     playwave "tick0.wav", async
     #main.textbox21 "!contents? Number$"
     if right$(handle$,1)="P" then
          #main.textbox21 Number$ + "."
     else
          #main.textbox21 Number$ + right$(handle$,1)
     end if
end sub

sub powerClick handle$
     #main.textbox21 "!contents? lastR$"
     #main.textbox21 ""
     R$ = R$ + lastR$
     select case right$(handle$,1)
          case "V"
               R$ = R$ + "*"
          case "O"
               R$ = R$ + "+"
          case "A"
               R$ = R$ + "-"
          case "D"
               R$ = R$ + "/"
     end select
end sub

[buttonCClick]
print "c"
#main.textbox21 ""
```

```
R$ = "" : a$ = ""
#main.textbox21 a$
wait

[buttonBClick]
confirm "Quit?";yn$
if yn$ = "yes" then [quit.main]
notice "Calculator by Cursist 2011" 'Insert your Backspace
wait

[buttonIClick]
playwave "Bounce.wav"
#main.textbox21 "!contents? varName$"
TR$ = TR$ + varName$
print TR$
a$ = eval$(TR$)
print a$
#main.textbox21 ""
#main.textbox21 a$
R$ = ""
wait

[buttonPMClick]
print "pm"     'Insert your plus or minus
wait

[quit.main] 'End the program
unloadbmp "calcBG"
close #main
end
```

Hier zie je het resultaat van het bovenstaand programma, geschreven door een Liberty BASIC cursist uit Amstelveen.

De uitwerking en uitleg van de listing.
Het commando NOMAINWIN zorgt ervoor dat het console venster (mainwindow), dat Liberty BASIC standaard opent, nu niet wordt getoond.
De eerste vier opdrachten geven de positie van het venster (onze calculator) op het monitorscherm aan. De derde regel ziet er zo uit:

```
UpperLeftX=int((DisplayWidth-WindowWidth)/2)
```

Hiermee positioneren we ons venster precies in het midden van ons monitor scherm (display). De variabele UpperLeftX is de X-coördinaat van de linkerbovenhoek van ons venster.

Daarna maken we R$ globaal. Globale variabelen zijn overal in het programma te gebruiken. Variabelen die in subroutines gebruikt worden zijn lokaal, waardoor de waarden alleen binnen de subroutines toegankelijk zijn. De waarde van R$ moet ook buiten de subroutine toegankelijk zijn, daarom is die variabele hier globaal gemaakt.

Onderstaande regel laadt het achtergrondplaatje in het geheugen.

```
loadbmp "calcBG", "calculatortest1.bmp"
```

De volgende regel laat één van de negen buttons zien die op het venster moet komen.

```
bmpbutton #main.button3,"CijferKnop3.bmp",cijferClick, UL, 140, 230
```

#main.button3 is de handle, ofwel de naam van de control. Het deel #main komt overeen met de handle-naam van het venster waarin de button staat. De eigen naam van de button (bmpbutton) is button3 omdat het een plaatje is met button eigenschappen.
Let op dat de handle-namen gelijk zijn en alleen het laatste karakter, in dit geval 3, per button verschilt. Dat komt straks goed van pas.

cijferClick is de naam van de subroutine die aangeroepen wordt als op de bmpbutton wordt geklikt. Het klikken op een button met de muis geeft een *event*. Windows registreert dat en geeft de event door aan Liberty BASIC. Het programma vervolgt dan bij de subroutine cijferClick:

```
sub cijferClick handle$
     playwave "tick0.wav", async
     #main.textbox21 "!contents? Number$"
     if right$(handle$,1)="P" then
          #main.textbox21 Number$ + "."
     else
          #main.textbox21 Number$ + right$(handle$,1)
     end if
end sub
```

De naam van de handle komt in de variabele handle$ te staan. Dan wordt het geluidsfragment, een wave bestand, tick0.wav gespeeld. Dat is een click klank. Daarna wordt de inhoud van textbox21 uitgelezen en die inhoud wordt in de variabele Number$ geplaatst.
Als het laatste teken van handle$ een P is dan wordt het resultaat van de expressie Number$ + "." in de textbox geplaatst, anders wordt het resultaat van de expressie Number$ + right$(handle$,1) in de textbox geplaatst. De expressie zal het aanwezige getal plus het laatst ingevoerde cijfer resulteren.

Na de bmpbuttons staan de knoppen voor rekenkundige bewerkingen, daarna de knoppen voor resultaat, daarna de knop clear en dan de knop aan/uit.

De rekenkundige knoppen zorgen voor een event aanroep naar de subroutine powerClick:

```
sub powerClick handle$
     #main.textbox21 "!contents? lastR$"
     #main.textbox21 ""
     R$ = R$ + lastR$
     select case right$(handle$,1)
          case "V"
```

```
                R$ = R$ + "*"
            case "O"
                R$ = R$ + "+"
            case "A"
                R$ = R$ + "-"
            case "D"
                R$ = R$ + "/"
        end select
    end sub
```

Bovenstaande subroutine spreekt voor zich, die is wel te volgen.

De belangrijkste knop is de knop met het = teken.

```
    bmpbutton #main.buttonI,"IsGelijkKnop1.bmp",[buttonIClick], UL, 145, _
              330
```

Deze knop veroorzaakt een event naar een “label” (Engels voor merkteken). De label staat tussen rechte haakjes [buttonIClick].

```
    [buttonIClick]
    playwave "Bounce.wav"
    #main.textbox21 "!contents? varName$"
    TR$ = R$ + varName$
    a$ = eval$(TR$)
    #main.textbox21 ""
    #main.textbox21 a$
    R$ = ""
    wait
```

Eerst wordt er een wave bestand (Bounce.wav) afgespeeld. Dan wordt de inhoud van textbox21 gelezen en in variabele varName$ geplaatst. De variabele R$, uit de vorige subroutine, bestond uit een getal en een operator (getal + of getal – of getal / of getal *). Deze wordt nu aan de variabele uit de “label” subroutine gekoppeld. Variabele TR$ wordt bijvoorbeeld “123 + 456” of “123 / 456” enzovoorts. Deze TR$ string wordt nu met een speciale Liberty BASIC functie geëvalueerd, eval(TR$). Daarna wordt de uitkomst in het venster textbox21 geplaatst.

De textbox en het grafische venster zijn van speciale zogenaamde stylebits voorzien. De stylebits van de textbox zorgt ervoor dat de ingegeven tekst aan het rechter uiteinde van de textbox wordt geplaatst tijdens het invoeren. De stylebits van het grafische venster zorgt ervoor dat er geen rand om het venster verschijnt.

Okay, je hebt geen Liberty BASIC maar wel Just BASIC. Wat nu?

- Just BASIC kent geen stylebits.
- Just BASIC kent de EVAL functie niet.
- Just BASIC kan de DLL’s (voor het tekenen op de knoppen) niet openen.

Hier volgt een voorbeeld van dezelfde opgave, maar nu in Just BASIC geprogrammeerd.

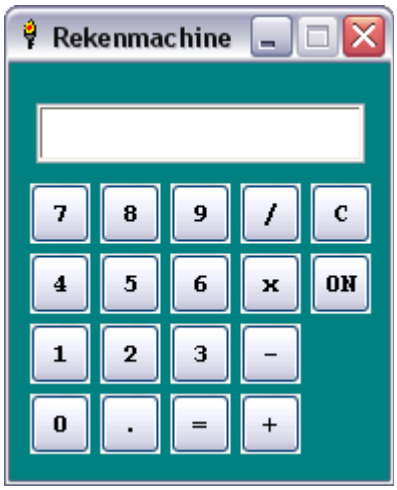
```
'** rekenmachine

    NOMAINWIN
    WindowWidth = 195 : WindowHeight = 240
    UpperLeftX = INT((DisplayWidth-WindowWidth)/2)
```

```
    UpperLeftY = INT((DisplayHeight-WindowHeight)/2)


[Display]
    reken$="":getal=0:vgetal=0:getal$=""
    textbox     #m.textbox1, 13,20,164,30
    button      #m.b1,"9",[k9],ul,80,60,30,30
    button      #m.b2,"8",[k8],ul,45,60,30,30
    button      #m.b3,"7",[k7],ul,10,60,30,30
    button      #m.b4,"6",[k6],ul,80,95,30,30
    button      #m.b5,"5",[k5],ul,45,95,30,30
    button      #m.b6,"4",[k4],ul,10,95,30,30
    button      #m.b7,"3",[k3],ul,80,130,30,30
    button      #m.b8,"2",[k2],ul,45,130,30,30
    button      #m.b9,"1",[k1],ul,10,130,30,30
    button      #m.b10,"0",[k0],ul,10,165,30,30
    button      #m.b11,".",[kk],ul,45,165,30,30
    button      #m.b12,"x",[kx],ul,115,95,30,30
    button      #m.b13,"/",[kd],ul,115,60,30,30
    button      #m.b14,"+",[kp],ul,115,165,30,30
    button      #m.b15,"-",[km],ul,115,130,30,30
    button      #m.b16,"=",[ks],ul,80,165,30,30
    button      #m.b17,"C",[kc],ul,150,60,30,30
    button      #m.b18,"ON",[ko],ul,150,95,30,30
                BackgroundColor$ = "darkcyan"
    Open "Rekenmachine" for window_nf as #m
      #m "trapclose [einde]"
      #m "font courier_new 10 bold"
    Wait

[einde]
    close #m
    END

[ko]
    if aan=0 then
       #m.textbox1 "!font courier_new 10 bold"
       print #m.textbox1,space$(14)+"0"
       aan=1:reken$="":getal$="":getal=0:vgetal=0
    else
       print #m.textbox1,"":aan=0
    end if
    wait

[kc]
    if aan=1 then
       print #m.textbox1,space$(14)+"0"
       reken$="":getal$="":getal=0:vgetal=0
    end if
    wait

[k0]
    if aan=1 then
        reken$=reken$+"0"
        print #m.textbox1,space$(15-len(reken$))+reken$
```

```
    end if
    wait

[k1]
    if aan=1 then
        reken$=reken$+"1"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k2]
    if aan=1 then
        reken$=reken$+"2"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k3]
    if aan=1 then
        reken$=reken$+"3"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k4]
    if aan=1 then
        reken$=reken$+"4"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k5]
    if aan=1 then
        reken$=reken$+"5"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k6]
    if aan=1 then
        reken$=reken$+"6"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k7]
    if aan=1 then
        reken$=reken$+"7"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k8]
    if aan=1 then
        reken$=reken$+"8"
```

```
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[k9]
    if aan=1 then
        reken$=reken$+"9"
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[kk]
    if aan=1 then
        reken$=reken$+"."
        print #m.textbox1,space$(15-len(reken$))+reken$
    end if
    wait

[kx]
    if aan=1 then
        gosub [bereken]:maal=1
        print #m.textbox1,space$(15-len(getal$))+getal$
        reken$=""
    end if
    wait

[kd]
    if aan=1 then
        gosub [bereken]:deel=1
        print #m.textbox1,space$(15-len(getal$))+getal$
        reken$=""
    end if
    wait

[kp]
    if aan=1 then
        gosub [bereken]:plus=1
        print #m.textbox1,space$(15-len(getal$))+getal$
        reken$=""
    end if
    wait

[km]
    if aan=1 then
        gosub [bereken]:min=1
        print #m.textbox1,space$(15-len(getal$))+getal$
        reken$=""
    end if
    wait

[ks]
    if aan=1 then
        gosub [bereken]
        print #m.textbox1,space$(15-len(getal$))+getal$
        reken$=""
```

```
    end if
    wait

[bereken]
    if reken$<>"" then
       getal=val(reken$)
       if vgetal<>0 then
           if maal=1 then
               getal=getal*vgetal:maal=0
           end if
           if deel=1 then
               getal=vgetal/getal:deel=0
           end if
           if plus=1 then
               getal=getal+vgetal:plus=0
           end if
           if min=1 then
               getal=vgetal-getal:min=0
           end if
       end if
       vgetal=getal
       getal$=str$(getal)
    end if
    return
```



Dit is het resultaat van de uitvoering van de Just BASIC listing. De cursist heeft geen gebruik gemaakt van de knoppen (bmp plaatjes) die bij de opgave horen. Bestudeer de listing zelf.

Alle listings en plaatjes staan op het Liberty BASIC forum. Om jullie kennis te laten maken met Liberty BASIC en de workshop, verklap ik dat ook de oplossingen van de andere cursisten op het forum te vinden zijn.

http://www.libertybasic.nl/viewtopic.php?f=15&t=542

Gordon Rahman

QB64 Basic voor Windows, Apple OSX en Linux.

Om een computerprogramma te maken was het in het hele begin van het computertijdperk bijna altijd cijfertjes inkloppen in Hex notatie, A7B3 enz. Hele listings met cijfertjes verschenen in de computer-bladen. Ken je die instructies nog? Load A,C. Compare B,D
Het was dus een verademing toen er compilers kwamen waardoor je gebruik kon maken van simpele in de 'normale' taal geschreven opdrachten die de compiler dan weer ging omzetten naar de gewenste machinecode.
Sinclair was er zo een met een compiler in de ROM. Applesoft, BBC herinneren we ons nog. Ieder op zijn eigen wijze. Enfin, je kunt er alles over lezen in deze nieuwsbrieven.
Toen het IBM en daarna het Windows tijdperk aanbrak kwam er QB op de markt. Je kon er het nodige mee. Er was een compiler bij, dus je kon van je basic programmaatje een uitvoerbaar bestand maken wat prima in Windows werkte. QB 4.5 was de laatste versie ervan. Of er daarna nog verder aan QB is ontwikkeld is mij niet bekend. Het 'echte' Windows verscheen namelijk met zijn mooie grafische schermen en de mogelijkheden daarvan kon je niet benutten met QB. Zo ontstonden er nieuwe programmeertalen die daar op aansloten. QB was afgedaan. DOS kwam langzamerhand in de vergetelheid.

Toch bleken er nog vele DOS programma's in gebruik te zijn. De mogelijkheden om deze te gebruiken bleven in de diverse Windows versies aanwezig. QB is gebaseerd op 16 bits grootte en draaide dank zij de gebruikte processoren in de Windows computers ook op de (standaard) 32 bits systemen. Totdat de 64 bitters in omloop kwamen. Einde verhaal met de 16 bits DOS programma's toch?

Knappe koppen realiseerden z.g. DOS boxen programma's. Zo kon je via een simulatie programma toch nog DOS programma's draaien zij het met de nodige problemen en beperkingen.
Een programma in een programma om het zomaar te zeggen.

In mijn dagelijkse praktijk gebruikte ik diverse zelfgemaakte DOS programma's die uitstekend voldeden. Weliswaar geen grafische hoogstandjes maar wel heel effectieve databases en gemakkelijke te gebruiken tooltjes. De computer moest vernieuwd worden en ja, je kon er nog een 32 bits Windows systeem op laten zetten. Uiteindelijk houdt het toch een keer op. Dus wat nu?
Door speuren op internet kwam plotseling de site van QB64 op het scherm. Er bleek een groep USA programmeurs te zijn met hetzelfde probleem, die daar een oplossing voor hadden gevonden.

De makers van het programma verwoorden het zo:

QB64 is a BASIC compatible C++ compiler that creates working Executable files from Qbasic BAS files that can be run on 32 or 64 bit PC's using WINDOWS(XP, Vista and newer), LINUX or MAC(OSX only). The goal is to be 100% compatible with QuickBasic 4.5 plus add hundreds of new abilities such as program icons and custom sized windows and a great Editor with a new Help Menu.

The new keywords add some new features such as playing music or sound files and instant access to 32 bit graphics file images. Also TCP/IP internet communication is available to download files, email messages over the web or play internet games. DLL Libraries can add more programming options and QB64 can access all of the new USB gaming controllers and printers.

QB is an abbreviation for QBasic or QuickBASIC which is an easy to learn language that grew very popular in the 90's. It uses simple syntax but holds great potential as there are methods to achieve nearly anything. Qbasic is NOT DEAD thanks to QB64!

Dat zag er goed uit dus toch maar even proberen.

Het resultaat was bemoedigend. Je kon opeens een uitvoerbestand maken van je zelfgemaakte QB programma's van je source code dat ook werkte op een 64 bits systeem! Daarbij bleek het ook nog open source te zijn dus iedereen kan en mag het volledig gebruiken.

Bovendien kan QB64 nu ook gebruikt worden in Linux en Apple OSX systemen!

Maar is het zo simpel als het lijkt? In zekere zin wel, echter er zitten wel enkele (kleine) haken en ogen aan. De gecompileerde QB64 programma's werken niet zomaar. Er moeten een aantal DLL files aanwezig zijn op de uitvoerende computer, die overigens bijgevoegd zijn.

Nu is men bezig om dat te integreren in de compilatie zodat dat dan niet meer nodig is. Zelfs voor Android zijn er ontwikkelingen! Er staat op hun site ook al een compiler die dat doet. Echter dat is nog in experimenteel stadium en bij het testen bleek een en ander (nog) niet goed stabiel te zijn.

Er zijn op dit moment (feb 2014) twee versies:
QB64 [GL] versie 0980 is de versie waarbij je geen DLL's meer nodig hebt. Maar die is nog in ontwikkeling.
Versie QB64 (SDL) versie 0954 is de laatste versie die gebruik maakt van DLL files. (Op dit moment de stabielste).

De GL versie kan ook experimenteel gebruikt worden voor Android.
Op de QB64 site (www.qb64.net) kan je alle hardware voorwaarden lezen en vrij downloaden.

Het gehele programma bevat alles wat er nodig is. Voorbeeld programma's zijn er in aanwezig. Je kunt dus zo aan de slag. QB64 is een losstaand programma. Je kunt het (compleet) in elke map uitpakken. Speciale installatie is niet nodig.

Run QB64.exe. Je ziet het ´bekende´ QB4.5 Edit scherm.

Ga nu naar Options > Display om het editor scherm een beetje aan te passen.
Instellen:
Windows sizewidth: 80
Height: 40 (als het op het scherm kan)
Kruisje bij Custom font en bij Row Height > Pixels: 20.
Klikken op OK en je kunt aan de slag.

Laadt nu een (oud) QB basic source code programma in. Toets op F5 en zie wat er gebeurt!

Nieuwsgierig geworden?
Daarom nog wat basic source code om eens kennis te maken met QB64.
De in dit artikel toegepaste (QB) basic code spreekt voor zich zelf en behoeft geen toelichting.
Daarom als voorbeeld het gebruik van een aantal (nieuwe) QB64 keywords.
Maar eerst iets over de Editor. Er is een belangrijke verbetering in de edit mogelijkheden.
Je kunt ´gewoon´ kopiëren met CTRL+C en weer terugzetten met CTRl+V. Ook vanuit andere documenten!
Als je dus source code tekst in Word of een andere tekstverwerker ziet kan je het zo in QB 64 kopiëren. Dat is nog eens makkelijk!
Verder is er GEEN limiet in de source code. Je kunt het net zo lang maken als je wilt!

De syntax corrector van de editor reageert wel iets anders dan bij QB 4,5. Maar het is wel gebaseerd op logische functionaliteit.
Voorbeeld: Toets maar eens in IF a = 1 THEN
Op dat moment komt het bericht: *missing end if*. Daardoor wordt je eigenlijk wel gedwongen als volgende regel direct end if in te toetsen. Zo vergeet je het nooit wat nog wel eens gebeurt bij complexe IF-THEN of FOR-NEXT loops.

Nu je dit weet kunnen we aan de slag.

QB64 bevat een groot aantal eigen keywords. Die beginnen allemaal met een underscore. Zo zie je direct dat het gaat om een (nieuwe) QB64 instructie.

QB64 regelt zijn zaakjes zelf. Dat wil zeggen dat je niet zoals met QB4.5 achteraf via Windows de schermgrootte kan regelen. Nu wordt dit in het programma zelf gedaan en dat gaan we nu maken. We gebruiken daar bij voorkeur Sub procedures voor. In tegenstelling tot Qb 4.5 heb je hier niet bij EDIT de keuze Create SUB. Dat doe je zelf. Je hoeft het ook niet te declareren zoals in QB 4.5
Dus de DECLARE SUB (procedurenaam) hoef je niet bovenin je programma te zetten. Je kunt ze bij de oude programma's gewoon weghalen.
Denk er daarbij wel om dat de SUB's en Function procedures NA het hoofdprogramma moeten komen. Daarom altijd een streep zetten na het einde van de programma code. Denk aan de apostrof voor de streep. Je zult wel weten waarom.
END zetten we er meteen maar voor want het programma mag hier niet meer verder gaan.

```
END
'-------------------------------- Einde programma code ----------------------
```

De SUB's, zo noemen we ze maar, staan verderop in het kort beschreven. We kijken nu naar de in het programma gebruikte speciale QB64 keywords.
_FILEEXISTS (filename) is een Functie. Als het aangegeven file bestaat geeft het -1 anders een 0. Je kunt dus direct zien of die file aanwezig is.

De _LOADFONT functie laadt een gespecificeerd TTF font in en geeft een LONG > 0 retour.
_FONT is een functie die het font op het scherm zet via de LONG variable.
_SCREENMOVE zet het programmavenster op de aangegeven plaats.
_LIMIT (Loops per seconde) beperkt sterk het CPU gebruik. (Stabiliteit).

Voor het geluid zijn er een groot aantal keywords.
_SNDOPEN . Geluidsbestand openen (heeft meerdere syntaxis)
_SNDPLAY Begint met afspelen
_SNDPAUZE Pauzeert
_SNDPAUZED (functie) . Geeft -1 als het geluidsbestand in de pauze staat.
_SNDGETPOS Geeft waarde af in seconden waar het afspelen nu is.
_SNDSTOP Stopt het afspelen.

Er zijn nog meer keywords bestemd voor het geluid. zodat je een geluidsbestand op alle mogelijkheden bij het afspelen kan besturen.

Toelichting bij de SUB's

SUB Fontkeuze
Je kunt met QB64 ook zelf bepalen op welke plaats het programma venster op het scherm komt. Leuk is als je bijvoorbeeld 2 monitoren hebt en je wilt het programma op de 2e monitor openen.

Dat doe je zo:

```
vstand = 10  '10 van de bovenkant af
hstand = 15 '15 van de linkerkant af.
_SCREENMOVE vstand, hstand
```

Je mag elke waarde ingeven. Wil je het in het 2e scherm openen dan geef je een grote hstand waarde. Je kunt het scherm ook met de muis ergens naar toe dirigeren om hem dan later op die plaats weer te openen. Maar dat moeten we wel zelf in het programma maken. Daarover misschien later.

We kunnen het ook precies in het midden laten openen dan gebruiken we de instructie:

```
_SCREENMOVE _MIDDLE
```

Dat is in het programma toegepast.

De lettergrootte van het werkscherm wordt al bepaald. We slaan dat op in het bestand *Letter.ini.* Je hoeft dan niet steeds de schermgrootte in te instellen.

SUB Box is een nostalgische QB procedure dat nog steeds goed zijn werk doet.

SUB Center kennen we allemaal.

SUB Getrout. Bestemd voor detectie van toetsaanslagen. In dit programma zorgt het ook voor de tijd op het scherm. We willen het toch een beetje mooi maken? Kan ook muisroutines bevatten. Misschien later nog eens. Het programma geeft de ingetoetste toets als string terug.

SUB Klok. Zet de tijd op het scherm.

SUB Olijn. Zet op de onderste schermregel de Help tekst.

SUB Geluid. Bevat de afspeelroutine voor het geluid. Een combinatie van de 'nieuwe' QB64 statements en de good old QB!

SUB Letteropslag voor de schermgrootte.

Denk overigens wel om de vereiste syntax en vergeet de underscore hierbij niet!
Net als bij de oude QB Editor zet je de muis op het keyword en toets op F1 als je Help info over het keyword wilt..

Met het onderstaand programma heb je een programma waar je heel wat mee kan experimenteren.

Hoe te handelen
Ga naar www.qb64.net en download het programma QB64 (SDL) v. 0954
Pak het uit in een map naar eigen keuze. Installatie is verder niet vereist. (zie boven)
Zet in dezelfde map een WAV geluidsbestand.
Ga naar QB64.exe en run het.

Kopieer onderstaande programma tekst met Ctrl+C en zet het in het QB64 werkscherm met Ctrl+V.
Toets op F5.

De werking:
Met F1 wordt het werkscherm kleiner
Met F5 wordt het groter
Met F10 kan je een geluidsbestand afspelen
Met ESCAPE eindigt het programma.

En je kunt je eigen code verder toevoegen.

Als er belangstelling voor is kunnen we dat volgende keer verder uitbreiden met een fraaie input SUB procedure en muisgebruik gaan toepassen.

```
' Begin programma voor QB64.
' 2014
' Houd hier wat ruimte voor toekomstige declaraties en DIM functies
DIM SHARED Lettergrootte ' Lettergrootte bestemd ook voor Sub procedure dus globaal
'
_TITLE "Basic programma met QB64" 'Geeft titel
CLS
start:

Fontkeuze

COLOR 7, 0
LOCATE 1, 1: PRINT SPACE$(80)
LOCATE 1, 2: PRINT DATE$
center 1, " Basisprogramma met QB64 "
COLOR 1, 1
box 2, 1, 24, 80
COLOR 4, 3
box 3, 3, 23, 78
COLOR 15, 4: center 3, " Demo BASIC magazine"
COLOR 1, 3
center 11, "Demo van QB64."
center 13, "Zorg voor een .wav bestand. Zet het in de QB64 (SUB)map"
center 15, "Toets op F10 voor geluid afspelen."
Olijn "[F1]Kleiner. [F5]Groter. [F10]Afspelen [ENTER]Verder [ESC]einde"

getrout k$

IF k$ = CHR$(0) + CHR$(59) THEN Lettergrootte = Lettergrootte - 1: letteropslag
IF k$ = CHR$(0) + CHR$(63) THEN Lettergrootte = Lettergrootte + 1: letteropslag
IF k$ = CHR$(0) + CHR$(68) THEN

COLOR 1, 1
box 12, 23, 21, 58
COLOR 4, 0: box 13, 25, 20, 56
COLOR 15, 4
center 13, " Geluid "
COLOR 7, 0
Locate 17,30: Print "WAV file: ";

Line  input ""; Bestand$
Bestand$ = Bestand$ + ".WAV"
If _FILEEXISTS (bestand$) then Geluid bestand$
END IF

IF k$ = CHR$(27) THEN END
IF k$ <> CHR$(13) THEN GOTO start

' Ga hier verder met je eigen Basic programma

COLOR 4, 1
box 2, 1, 24, 80
COLOR 6, 6
box 3, 3, 23, 78
COLOR 15
center 5, "Dit is een basis programma voor Qb64 in de standaard mode"
COLOR 1
center 8, "Hier kun je al je verdere activiteiten in programmeren.  "
center 10, "Indien er belangstelling voor is kunnen we nog een aantal"
center 12, "belangrijke en interessante routines hiervoor publiceren."
center 14, "Vooral het Muisgebruik is eenvoudig te programmeren.     "
center 16, "Alle informatie vind je in www.QB64.net. Ook een mailbox "
center 18, "is daar aanwezig met honderden vragen en antwoorden.     "
COLOR 14
```

```
center 20, "Veel plezier met het maken van BASIC programma's."
Olijn "[Elke toets] terug"
getrout i$
GOTO start
END
'---------------------- Einde programma ------------------

SUB Fontkeuze
Rootpath$ = ENVIRON$("SYSTEMROOT")    ' is dus normaal "C:\WINDOWS"
Font$ = Rootpath$ + "\Fonts\lucon.ttf"      'TTF file in Windows

IF _FILEEXISTS("Letter.ini") THEN                'Controle of dit file aanwezig is
    OPEN "Letter.ini" FOR INPUT AS #1
    INPUT #1, Lettergrootte
    CLOSE #1
END IF

IF Lettergrootte < 10 THEN Lettergrootte = 10 'Minimaal 10

f& = _LOADFONT(Font$, Lettergrootte, "monospace")
_FONT f&
_SCREENMOVE _MIDDLE
END SUB

SUB box (Row1, Col1, Row2, Col2) STATIC  'Bekend van QB
BoxWidth = Col2 - Col1 + 1
LOCATE Row1, Col1
PRINT CHR$(218); STRING$(BoxWidth - 2, CHR$(196)); CHR$(191);
FOR a = Row1 + 1 TO Row2 - 1
    LOCATE a, Col1
    PRINT CHR$(179); SPACE$(BoxWidth - 2); CHR$(179);
NEXT a
LOCATE Row2, Col1
PRINT CHR$(192); STRING$(BoxWidth - 2, CHR$(196)); CHR$(217);
END SUB

SUB center (Regel, text$)
LOCATE Regel, 41 - LEN(text$) / 2
PRINT text$;
END SUB

SUB getrout (Kb$)
lc = CSRLIN: hc = POS(0)        'Cursorplaats opslaan
WHILE INKEY$ <> ""
WEND
Kb$ = ""
WHILE Kb$ = ""
    Kb$ = INKEY$
    _LIMIT 100 ' Gebruikt voor stabiliteit.
    Klok          'Tijd op het scherm. Zie Sub Klok procedure.
WEND
LOCATE lc, hc, 0            'Cursorplaats weer terugzetten
Kb$ = UCASE$(Kb$)    'Altijd hoofdletter retour
END SUB

SUB Klok
SHARED tb
te = VAL(MID$(TIME$, 7, 2))
IF te <> tb THEN tb = te: LOCATE 1, 71, 0: COLOR 7, 0: PRINT TIME$;
END SUB
```

```
SUB Olijn (oL$)
COLOR 3, 0: LOCATE 25, 1: PRINT SPACE$(80); 'Onderste schermregel
DEFINT K
kl = 0
kr = LEN(oL$)     'Lengte tekst
FOR k = 1 TO kr 'Centreren
    IF MID$(oL$, k, 1) = "[" THEN kl = kl + 1
NEXT
LOCATE 25, 41 - (kr - kl * 2) / 2 'Centreren op onderste schermregel
FOR k = 1 TO kr 'Letterkleur aanpassen
    IF MID$(oL$, k, 1) = "[" THEN COLOR 14: k = k + 1
    IF MID$(oL$, k, 1) = "]" THEN COLOR 3: k = k + 1
    PRINT MID$(oL$, k, 1);
NEXT
END SUB

SUB Geluid (callfile$)
PCOPY 0, 4  'Scherm bewaren
h& = _SNDOPEN(callfile$, "sync,vol,len,pause")
_SNDVOL h&, 1
_SNDPLAY h&
_SNDLOOP h&   ' Zet in een loop

Olijn "[SPACE] Pauze"
center 17, space$ (29)
COLOR 15, 0
center 17, "Elke toets stop "
eenmaal% = 0: Pause% = 0
WHILE K$ = "" OR k$ = " "
    _LIMIT 300
    Klok
    K$ = ""
    K$ = INKEY$
    IF K$ = " " THEN Pause% = 1 - Pause%: 'Space voor pauze en weer starten

    IF Pause% = 1 AND eenmaal% = 0 THEN
        _SNDPAUSE h&                                          ' Zet in pause stand
        IF eenmaal% = 0 THEN
            center 20, " Pause ": eenmaal% = 1
            Olijn "[SPACE] Afspelen "
        END IF
    END IF

IF Pause% = 0 THEN
        z% = _SNDPAUSED(h&)         'z% = 1 indien in pauzestand
        IF z% THEN
            _SNDPLAY (h&) 'Speelt weer verder
            eenmaal% = 0
            Olijn "[SPACE] Pauze"
            COLOR 4, 0
            center 20, STRING$(10, CHR$(196))
        END IF
    END IF

    IF K$ <> "" AND K$ <> " " THEN _SNDSTOP h& ' Stopt afspelen

    j% = _SNDGETPOS(h&) 'Teller
    COLOR 14, 0
    LOCATE 15, 39: PRINT j%
WEND
PCOPY 4, 0 'Beginscherm weer terug
END SUB
```

```
SUB letteropslag ' Bewaren schermbreedte
OPEN "Letter.ini" FOR OUTPUT AS #12
PRINT #12, Lettergrootte
CLOSE #12
END SUB
```

Evert Beitler

Cursussen

Liberty BASIC:
Cursus en naslagwerk, beide met voorbeelden op CD-ROM, € 6,00 voor leden. Niet leden € 10,00.

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 programmeertalen, waaronder ook Basic, die u zou willen leren.
Elke dinsdag, woendag en vrijdag in buurthuis Bronveld in Barneveld van 19:00 uur tot 21:00 uur op de dinsdag en van 9:00 uur tot 11:00 uur op de woensdag en vrijdag. Kosten € 5,00 per week.
Meer informatie? Kijk op 'http://www.i-t-s.nl/rdkcomputerservice/index.php' of neem contact op met mij.

Computerworkshop voor iedereen; heeft u vragen over tekstverwerking of BASIC, dan kunt u elke 2de en 4de week per maand terecht in hetzelfde buurthuis Bronveld in Barneveld van 19:00 uur tot 21:00 uur. Kosten € 5,00.
Meer informatie? Kijk op 'http://www.buurthuisbronveld.nl' of neem contact op met mij. Voor overige informatie: http://www.tronicasoftware.nl

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.

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	Jeroen v. Hezik	ma. t/m zo.	19-21	(0346) 214131	j.a.van.hezik@kader.hcc.nl
Visual Basic .NET	Marco Kurvers	ma. t/m zo.	19-22	06 30896598	m.a.kurvers@live.nl
Basic algemeen, zoals VBA Office	Marco Kurvers	ma. t/m zo.	19-22	06 30896598	m.a.kurvers@live.nl
Web Design, met XHTML en CSS