Nieuwsbrief

16de jaargang september 2009 Nummer 3

HCC BASIC interessegroep

Inhoud

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Gordon Rahman kwam met een	
bijdrage.	23

Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Willem Gobel	0118-850837	voorz@basic-gg.hcc.nl
Secretaris	Gordon Rahman Tobias Asserstraat 6 2037 JA Haarlem	023-5334881	secr@basic-gg.hcc.nl
Penningmeester	Piet Boere	0348-473115	penm@basic-gg.hcc.nl
Bestuurslid	Titus Krijgsman	075-6145458	t.krijgsman8@upcmail.nl
Redacteur	M.A. Kurvers Schaapsveld 46 3773 ZJ Barneveld	0342-424452	m.a.kurvers@hccnet.nl
Ledenadministratie	Fred Luchsinger	0318-571187	f.luchsinger@kader.hcc.nl
Webmaster	Jan van der Linden	071-3413679	j.vd.linden@kader.hcc.nl

http://www.basic.hccnet.nl

Redactioneel

In de vorige Nieuwsbrief heb ik het over een aantal onderwerpen gehad, waar ik best wel veel over verteld heb, daarom komt er nogmaals een deel over XHTML en over het maken van de projectbesturing in Visual Basic. Het zal mooi zijn als u Basic goed begrijpt, maar om de besturing in een project goed te begrijpen is toch andere koek. In het hoofdstuk daar meer over.

In deze Nieuwsbrief komt Gordon Rahman met de eerste les van de mini cursus Liberty BASIC. De volgende lessen zullen in de komende nieuwsbrieven staan.

Marco Kurvers

Het verschil tussen condities en expressies.

Vaak gebruiken we formules waar binnen in zo'n formule nogmaals berekend moet worden. De binnen berekening is dan geen variabele. We noemen die ook wel een expressie. Een expressie hoeft niet persé in een formule te staan. Ze komen ook wel voor in condities. Een conditie kan net als een formule in een toekenning worden gezet. Het verschil is dan wel dat het resultaat van een conditie alleen `True` of `False` kan zijn. BASIC gaat hier heel makkelijk mee om. Voor zowel het toekennen van het resultaat als het controleren van de conditie kent BASIC maar één operator: de '='. Andere programmeertalen hebben verschillende operatoren: de toekenningsoperator '=' en de controleoperator '=='.

We kunnen dus in BASIC een regel tegenkomen als: `G1 = A = 10`
Ook al hebben we het over BASIC; de volgende regel zou toch wat duidelijker zijn:
`G1 = A == 10`, of het gebruik van haakjes, dat in BASIC ook mag:
`G1 = (A == 10)`, maar wat de regel code niet doet is om gelijk te kunnen controleren en een keuze te kunnen maken, bijvoorbeeld een codeblok uitvoeren. We kunnen dit op twee manieren doen: 1. De toekenning met conditie laten staan en dan alleen met een `If` statement de variabele `G1` controleren, of: 2. Geen variabele gebruiken maar direct de conditie in een `If` statement opnemen.
Willen we echter op meerdere plaatsen in de code dezelfde conditie controleren, dan is het gebruik van punt 1 best handig. U hoeft alleen maar de variabele te gebruiken en u hoeft niet om te kijken wat voor een conditie, met misschien zelfs een expressie erbij, het ook alweer was.
Met een expressie bij de conditie kan de regel wat moeilijker maken, daarom nogmaals een voorbeeld.

`C = **(B =** (A + 20 * I)**)**` Dit voorbeeld is verdeeld in 3 delen.

Normaal: de toekenning.
Vet: de conditie.
Onderstreept: de expressie.

Uit het voorbeeld ziet u waarschijnlijk dat een conditie wat nodig heeft om het resultaat te kunnen geven. Zouden we de expressie weglaten, dan kan bovenstaande regel niet werken. Er moet wat achter '**B =**' staan. Dat wil niet zeggen dat expressies altijd achter condities moeten staan. U kunt ook op die plaats een variabele gebruiken of een soortgelijke functie.

Niet alleen de '=' operator kunt u gebruiken in een conditie, ook de andere operatoren die u normaal gesproken in een `If` statement hebt staan mag u in een conditie opnemen:

=	controleert of de linkerkant van de operator gelijk is aan de rechterkant van de operator. Het kunnen allebei aangeroepen functies, expressies of variabelen zijn;
>	controleert of de linkerkant van de operator groter is dan de rechterkant van de

	operator, het kunnen aangeroepen functies, expressies of variabelen zijn;
<	controleert of de linkerkant van de operator kleiner is dan de rechterkant van de operator, ook weer kunnen het aangeroepen functies, expressies of variabelen zijn;
=>	controleert of de linkerkant van de operator groter dan of gelijk is aan de rechterkant van de operator;
<=	controleert of de linkerkant van de operator kleiner dan of gelijk is aan de rechterkant van de operator;
<>	controleert of de linkerkant van de operator ongelijk is aan de rechterkant van de operator;
Not	veranderd het resultaat in tegengestelde waarde, True wordt False en False wordt True;
And	met dit operator, die u tussen één of meerdere condities moet gebruiken, zullen meerdere condities samen een resultaat True geven als ze allemaal waar zijn; geeft het resultaat echter een False dan is één conditie, of zijn meerdere condities, onwaar;
Or	dit operator gebruikt u net zoals de And operator tussen één of meerdere condities, echter zal het resultaat van maar één conditie die waar is worden teruggegeven;
XOr	dit operator werkt op twee manieren, logische exclusies met condities en logische exclusies met variabelen; condities gebruiken met deze operator zorgt alleen voor True, False of Null resultaten; gebruik van expressies en variabelen zorgt voor een bitgewijze waarde als resultaat.

Extra haakjes.

Zoals u gezien hebt geven extra haakjes meer duidelijkheid in formules. Dat betekent niet dat extra haakjes geen functie hebben. Als u ze gebruikt, houd dan rekening mee dat het verloop van 'Meneer van Dalen Wacht op Antwoord' veranderd kan worden. Onderstaande twee voorbeelden laten zien dat ze beide niet hetzelfde resultaat geven, doordat het tweede voorbeeld haakjes gebruikt:

```
A = B + C * 2      'Als B = 5 en C = 10, dan is A = 25
A = (B + C) * 2    'Nu is A = 30
```

Haakjes gebruiken bij condities geeft geen functie, want dan doen ze niks. Dan geven ze wel alleen maar duidelijkheid, zoals de voorgaande conditievoorbeelden lieten zien. Expressies worden altijd van binnenuit uitgevoerd indien er haakjes aanwezig zijn. Er kunnen meerdere extra haakjes in voorkomen, maar houd de formule niet te ingewikkeld. Desnoods kunt u een deel ervan eerst opslaan in een variabele. Denk maar eens aan het voorgaande voorbeeld. Daar zei ik in commentaar dat variabele `B = 5`, maar die variabele kan net zo goed uit een expressie bestaan en dan niet te spreken over de andere variabelen wat die niet allemaal voor expressies kunnen hebben. Denkt u eens in wat voor een ramp er kan ontstaan als u al die expressies in één formule zou gebruiken?!

Keuzestructuren nesten met een `ElseIf`.

In plaats van meerdere condities in één `If` statement te controleren, willen we ook wel eens bepaalde condities laten controleren als aan een andere voorwaarde wel of niet werd voldaan. We kunnen dan onderstaande code gebruiken:

```
If A = 10 Then
   'codeblok
Else
   If B > 30 Then
      'Codeblok
   End If
End If
```

Wilt u echter proberen nog een `Else` statement te plaatsen na de laatste `End If`, dan krijgt u een foutmelding. U zou eventueel nog een keer de hele structuur met de conditie `A = 10` kunnen herhalen met telkens een andere conditie na het `Else` statement, maar dat kost erg veel code en tijd. BASIC heeft echter een oplossing voor dat probleem:

```
If A = 10 Then
   'Codeblok
ElseIf B > 30 Then
   'Codeblok
ElseIf conditie Then
   'Codeblok
Else
   'Codeblok
End If
```

Nu kunt u na elke `ElseIf` een andere conditie laten testen, zoveel u maar wilt. Ze worden alleen getest als alleen de allereerste conditie waar is, anders zal alles worden overgeslagen en zal het codeblok na de `Else` worden uitgevoerd.

We nesten veel keuzestructuren door gebruik van een `ElseIf`. Dat kan handig zijn, maar het kan ook de code rommelig maken. Bovendien, zoals u dit voorbeeld ziet, kost het ontzettend veel tijd en ruimte.

```
Select Case A
   Case 10
      'Codeblok
   Case 30
      'Codeblok
   Case Else
      'Codeblok
End Select
```

Het gebruik van een `Select Case ... End Select` blok laat een minder rommelige keuzelijst zien. Hoewel het ongeveer lijkt op voorgaande voorbeeld, heeft een `Select Case` ook een nadeel. Nu kunnen we niet op verschillende condities controleren, alleen maar op meerdere resultaten van de conditie die in de `Select Case` regel opgegeven moet worden. We kunnen ook niet in die keuzelijst zeggen: `Case B > 30`, want dan zal namelijk die conditie vergeleken worden met variabele `A` en niet zoals het met een `ElseIf` werkt.

Rare situaties met een `Select Case`.

Bepaal zelf hoe u expressies, condities en de resultaten ervan gebruikt. Controleer ze wanneer u denkt dat bepaalde code wel of niet uitgevoerd mag worden. Maak gebruik van operatoren als meerdere condities gecontroleerd moeten worden.

Wat gebeurt er eigenlijk als we `Case B > 30` zouden opgeven? Die conditie zou vergeleken worden met de opgegeven variabele `A` die in de `Select Case` regel staat. Dan zou het volgende gebeuren:

```
If A = (B > 30) Then 'twee condities in één
```

Met andere woorden, en daar zijn de extra haakjes weer nuttig voor, twee condities zouden worden gecontroleerd. De codeblok achter `Case B > 30` wordt alleen uitgevoerd als variabele `A` waar is, ook al zou variabele `B` groter zijn dan 30!
Bent u benieuwd? Kijk eens naar het volgende voorbeeld:

```
Public Sub Test()
    Dim A As Boolean, B As Integer

    B = 50
    A = True 'bij tweede test: A = False
    Select Case A
        Case B > 30
            MsgBox "Wordt uitgevoerd."
        Case Else
            MsgBox "Toch niet."
    End Select
End Sub
```

Voer de code twee keer uit met als eerste keer de waarde `True` en als tweede keer de waarde `False`. U zult zien dat wanneer `A = False`, de eerste `Case` zal worden overgeslagen ook al is `B = 50` en dus groter dan 30. De twee uitvoeringen zijn dus:

1. Wordt uitgevoerd.
2. Toch niet.

Zoals u tot nu toe hebt kunnen zien, kan zulke code tot rare situaties leiden. Gebruik daarom zoveel mogelijk commentaarregels zodat u de volgende keer weet wat er staat. Ook is het een rare situatie als we variabele `A` zouden veranderen in een integer en de waarde `40` zouden toekennen. U denkt misschien: nu zal de tekst 'Wordt uitgevoerd.' worden weergegeven. Helaas is dat niet het geval, ook al is getal 40 groter dan 30! Het is daarom beter om geen conditie achter een `Case` te plaatsen. Wilt u toch weten of de opgegeven variabele `A` groter is dan het resultaat van een functie of expressie, gebruik dan de regel:

`Case Is >` expressie

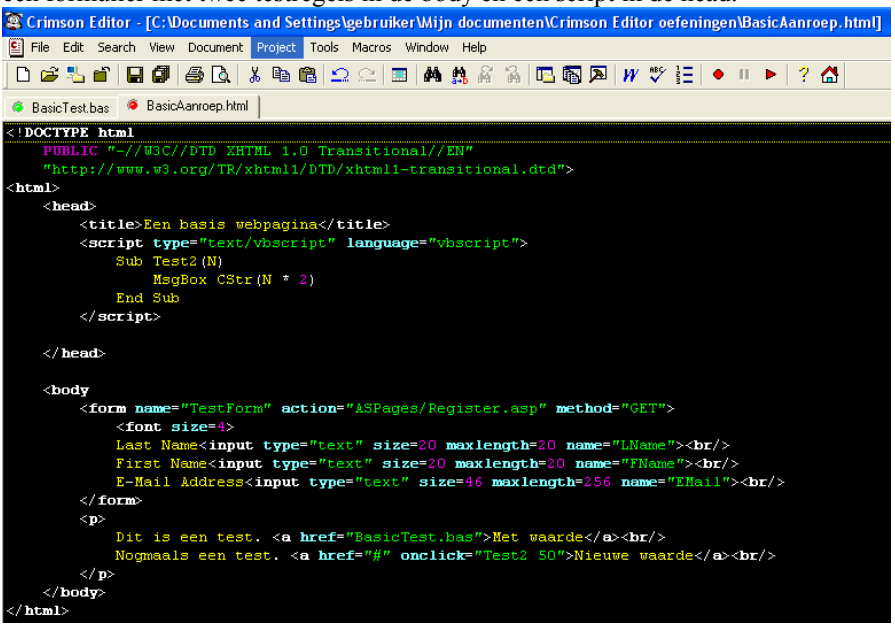
Nu zal het wel correct werken zonder enige rare situaties. Het sleutelwoord `Is` moet u niet vergeten en is alleen nodig als u operatoren wilt gebruiken. U kunt het sleutelwoord niet gebruiken bij een vergelijkingsoperator '='.

Marco Kurvers

Verder met XHTML en Scripts.

In de vorige Nieuwsbrief liet ik een afbeelding zien met een documentenstructuur met verschillende documenttypen en liet ik zien hoe controls – oftewel besturingselementen – weergegeven kunnen worden met behulp van de tag `<Form> ... </Form>`. We weten nu ook dat we met de tag `<script> ... </script>` een programmeertaal kunnen gebruiken. We kunnen dan code schrijven dat normaal gesproken in XHTML niet aanwezig is. Hoewel de interactiviteit tamelijk eenvoudig in de pagina's in te bouwen is, ontbreekt toch enige mogelijkheid om werkelijk dynamische documenten te kunnen schrijven. De middelen om via XHTML iemand naar zijn naam te vragen en vervolgens de website daarop te baseren of om een animatie te maken, zijn eenvoudig niet aanwezig. De programmeur zou dan zijn toevlucht zoeken tot formulieren die gekoppeld zijn aan scripts, of tot gif-animaties. Beide zijn toepassingen die buiten het eigenlijke XHTML-gebied vallen.

Onderstaand afbeelding is de Crimson Editor met een XHTML document dat bestaat uit een formulier met twee testregels in de body en een script in de head.



In het document ziet u het formulier zoals onderstaande structuur die ook in de vorige Nieuwsbrief stond, maar voor de zekerheid laat ik u de structuur van de `Form` tag nog eens zien:

```
<FORM NAME="Form1" [ACTION="ASPages/Register.asp" METHOD="GET"]>
   <FONT SIZE=grootte lettertype>
   ...
</FORM>
```

De opgegeven actie is optioneel en is alleen nodig als gegevens verzonden of ontvangen moeten worden. In het voorbeeld wordt gebruik gemaakt van het invoeren van het e-mail adres.
Zoals ik u in de vorige Nieuwsbrief heb laten zien kunt u van alles in een formulier plaatsen, net zoals u kunt doen in de IDE van de programmeertaal Visual Basic 6.

Onder de `Form` tag, te zien op bovenstaande Crimson Editor afbeelding, ziet u de tag code `<p> ... </p>`. Dit is een alinea-tag. Hoewel we ook zonder de alinea-tag de tekst mogen schrijven, is de tag zeer handig voor tekst acties. Zonder gebruik van `<p>` zou u geen align op kunnen geven. Onderstaande voorbeelden aligneren de tekst op drie manieren en eigenlijk is de eerste actie standaard en werkt op dezelfde manier als bovenstaande afbeelding:

```
<p align="left">Dit is links uitgelijnd.</p>
<p align="middle">Dit is gecentreerd.</p>
<p align="right">Dit is rechts uitgelijnd.</p>
```

Als de browser de drie zinnen overlappend weergeeft of niet goed onder elkaar plaatst, gebruik dan voor de zekerheid de tag `</br>`, zoals u kunt zien op bovenstaande afbeelding. De tag zorgt voor een extra carriage return.

Verwijzingen en gebeurtenissen; het aanroepen van scripts.

In het document van de afbeelding wordt ook de tag `<a> ... </a>` gebruikt met een actie `href`, **H**yperlink **REF**erence. Hiermee kan een verwijzing worden gemaakt naar een internetadres of naar een ander bestandsdocument, zoals het voorbeeld laat zien waarmee een verwijzing wordt gemaakt naar het bestand 'BasicTest.bas'. Helaas doet de verwijzing niet wat eigenlijk het plan is: het uitvoeren van de script die in het bestand staat. Sterker nog; als u Visual Basic 6 hebt, zal de applicatie worden gestart en een nieuwe module worden geopend met de inhoud van BasicTest.
De tweede regel wordt wel correct uitgevoerd met twee acties:

```
<a href="#id-key" event=script>textline</a>

<a href="#" onclick="Test2 50">Nieuwe waarde</a>
```

Het hekje met de id-key is optioneel en is alleen nodig als we naar bepaalde documentdelen willen springen, zoals een menu-item. Zoals in u het voorbeeld ziet zal de tekst 'Nieuwe

waarde' de klikgebeurtenis zijn. Bij het uitvoeren van het document zal de tekst in de browser dan ook een andere kleur hebben. Met een klik op de tekst zal de script `Test2` met argumentwaarde `50` worden aangeroepen. Doordat de scripttaal VBScript is opgegeven, wordt de Basic code goed begrepen en uitgevoerd.

Optimalisaties en opwaarderingen.

Het world wide web is geoptimaliseerd om XHTML-pagina's en hyperlinks weer te geven. Er zijn echter inmiddels ook vele andere bestandstypen in omloop. We kunnen niet verwachten dat browserfabrikanten bij elk nieuw product direct hun browsers opwaarderen. De inhoud van dergelijke nieuwe bestandsformaten kan niet rechtstreeks door de browser worden verwerkt.
De Crimson Editor verwacht zelf ook niet dat het vanzelf kan opwaarderen als er telkens maar weer nieuwe bestandsformaten verschijnen. Daarom heeft de editor vele scripttalen die men kan kiezen, waaronder ook VBScript. Toch kan het gebeuren dat geeneen VB-Script code correct werkt en dat heeft te maken met de opwaardering van de browser. Liever zou de browser willen dat u gebruik maakt van JavaScript. De meeste browsers ondersteunen namelijk die taal. Als u die taal wilt proberen; er zijn beginners boeken. De editor kent ook die script, maar het is voldoende om dan alleen de acties `type` en `language` te wijzigen. Vergeet ook dan niet de juiste syntaxis te gebruiken.

De tag `<object>` en plug-ins.

Met de tag `<object>` kan een programma in een webpagina opgenomen worden, zoals de plug-in Flash-player. Daarmee kunnen dan Flash-movies in de browser worden afgespeeld. Hieronder een voorbeeld:

```
<object classid="clsid:id-key"
   codebase="verwijzingsregel, bijvoorbeeld http://"
   width="breedte" height="hoogte"
   id="id-naam" align="optie uitlijning: left, middle of right">

   <param name="deze zijn nodig voor het instellen van het object"
          value="er kunnen meerdere parameters zijn" />
   <embed src="id-naam"
      hier moeten de eigenschappen van het object ingesteld worden met de waarden van de parameters
   />
</object>
```

Helaas kan ik geen volledig voorbeeld uit het boek halen vanwege de copyright rechten. Bovendien zal bij iedereen het bovenstaand objectvoorbeeld anders ingevoerd moeten worden, omdat de gegevens nou eenmaal niet hetzelfde zijn. Als u objecten wilt gebruiken om programma's in uw webpagina op te nemen, vergeet dan niet eerst de juiste registersleutel van het programma op te zoeken. Die moet bij `clsid:` opgegeven worden. Maak geen

enkele fout met het invoeren van de sleutel, anders werkt het programma of de plug-in zeker niet.

We kunnen meerdere documenten samen gebruiken in één project die in de Crimson Editor aangemaakt kan worden. U hebt gezien dat we gebeurtenissen kunnen gebruiken zonder deze op te moeten geven in de subroutine script. In de listing van de vorige Nieuwsbrief moest dat wel toen er via een `INPUT BUTTON` verwezen werd naar een subroutine in de script.
We kunnen met een `href=` een id-naam opgeven en als verwijzing naar een ander deel van het document of naar een ander deel van een ander bestand laten springen, maar we kunnen ook zonder id-naam een verwijzing maken naar een internetadres of door het aanroepen van een scriptfunctie, die alleen in JavaScript begrepen wordt, of een scriptsubroutine van een andere programmeertaal die in Crimson Editor (of in een eenvoudige teksteditor) gekozen kan worden, o.a. VBScript, door deze als type en language in de script-structuur op te geven. Wanneer u scripts wilt proberen te maken, kunt u prima het bovenstaande voorbeeld ervoor gebruiken. Ook zullen er heus wel helppagina's op internet bestaan waar meer informatie over scripts in te vinden zijn.

Meer hierover kunt u vinden bij:
Sdu klantenservice
Tel: 070-3789880, fax: 070-3789783
E-mail: bestelling@sdu.nl
Website: http://www.academicservice.nl

Marco Kurvers

Het project en de besturing.

Iemand rijdt in de auto en heeft een wegwijzer die de persoon helpt om in de goede wegen en straten te kunnen rijden. De wegwijzer zegt: 'De volgende straat links na 200 meter.' De automobilist doet wat de wegwijzer heeft gezegd en wil links inrijden, maar tot grote verbazing blijkt dat de straat te smal is en er staat een blauw bord met een fiets erop.

De automobilist 'weet' dat via andere straten ook daarheen gereden kan worden, maar de wegwijzer stribbelt echter tegen en bij elke rotonde zegt de wegwijzer dat de automobilist om moet draaien zodat de route gereden kan worden die alleen de wegwijzer kent. De wegwijzer heeft echter niet de data met de wijzigingen die de laatste keren in de straten zijn gemaakt. Natuurlijk bestaan er wegwijzers die geüpload kunnen worden met nieuwe gegevens, maar als iemand een wegwijzer heeft die dat niet kan is de persoon verder van huis.

Wat ik met bovenstaande verhaal wil zeggen, heeft te maken met het hoofdstuk 'De module Main', dat in de vorige Nieuwsbrief staat. Waarom, zou u denken? Dat is waar dit hoofdstuk over gaat: Het project en de besturing.

Basic niet alleen.

Om te leren programmeren moet er een programmeertaal gekozen worden die voor u het beste te begrijpen is en het makkelijkste te leren is. Basic lijkt een programmeertaal waarvan de syntaxis het eenvoudigst lijkt. Het is Engelstalig met weinig symbolen, dus makkelijk te gebruiken om programma's te schrijven.

Als u denkt dat u de programmeertaal onder de knie hebt en u denkt: 'Ha, nou kan ik prima software schrijven!' dan zou u dat inderdaad kunnen doen, maar houd er rekening mee dat programmeren niet alleen Basic is. Er komt veel meer bij kijken dan Basic alleen.
Waarom is Basic niet alleen en was dat dan vroeger net zo? Kwam er in de oude BASIC versies dan ook meer bij kijken? Niet helemaal. Vroeger moesten er omwegen worden gemaakt in de oude BASICA versies die we nu direct kunnen gebruiken. We kunnen ze bijvoorbeeld klassen en databases noemen. Om terug te gaan naar bovenstaand verhaal over de wegwijzer met de automobilist; om aan de wegwijzer te kunnen vertellen dat er een straat is gewijzigd, moeten er nieuwe gegevens ingeladen of geüpload worden. Dat kan alleen als de geprogrammeerde code van de wegwijzer een openstaande deur heeft. Als we dus een programma willen schrijven dat bijvoorbeeld NAW gegevens bij moet houden, moeten we rekening houden dat er opwaarderingen (verbeteringen) plaats moeten kunnen vinden. We moeten dus een programma niet rechtuit programmeren, maar juist ervoor zorgen dat het elke vernieuwing kan aanpassen zonder dat u zelf steeds het geraamte van de code moet veranderen.

De juiste besturing.

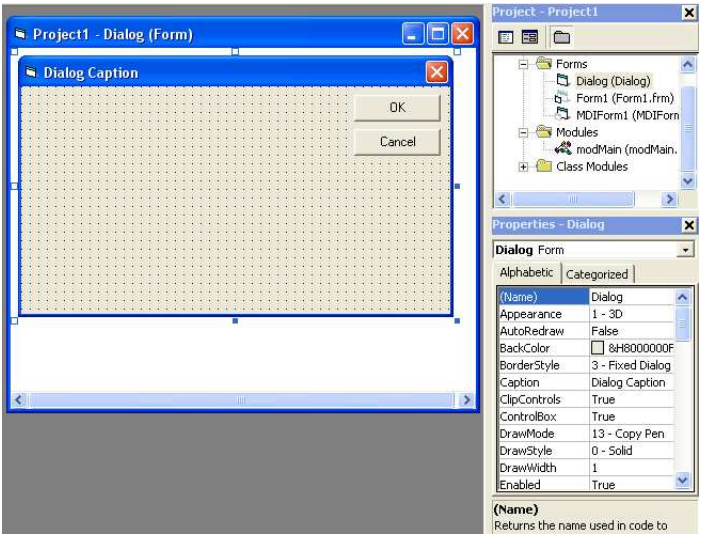
Niet rechtuit programmeren is niet makkelijk. Vele wegen gaan naar Rome, dat weten we. Van alles is te proberen en van alles zou kunnen werken. Als de besturing op precies de juiste manier gemaakt is en in het project is ingebouwd, zal de software correct werken. Hoe u het ook in elkaar zet, de programmeertaal zal echter daardoor niet veranderen. De conclusie is dus: alleen leren hoe de Basic programmeertaal in elkaar zit is niet leren hoe we een programma aan kunnen sturen. We moeten dus extra leren hoe we de juiste besturing in een programma in kunnen bouwen en reken er maar niet op dat u dezelfde besturing in elke programma kunt gebruiken, tenzij u een hoofdaansturing gebruikt.

De subroutine `Main` in de hoofdmodule.

In de vorige Nieuwsbrief hoofdstuk 'De module Main' werd er uitgelegd dat het beter is het project te starten met een subroutine, die het MDI formulier moet laden. Dit is in principe de hoofdaansturing, want zonder zo'n opstart module kan het project niet starten. Er zijn ook andere projecten die helemaal geen opstart module nodig hebben, zoals één formulier applicaties of games. Games kunnen meerdere rooms of schermen hebben, maar een MDI formulier is daarbij niet nodig. Hoe dan de belangrijke gegevens geïnitialiseerd worden, gaat met behulp van datamodules en scripts.

De extra formulieren en dialoogformulieren.

De formulieren zijn de schermen waar we de besturingselementen op plaatsen. Zoals de controls genoemd worden, zo moeten ze ook werken; ze moeten bestuurd worden. Er is één belangrijk aspect dat we goed moeten weten en moeten onthouden: bewaar nooit de gegevens van de controls in globale modules. Zorg ervoor dat elke formulier zijn eigen besturing heeft; laat ze niets van elkaar weten, behalve het MDI formulier en het kind formulier. Dat zijn de enige formulieren die samen werken. Het kind formulier draait altijd onder het MDI formulier. Sommige programmeertalen, en nu ook de nieuwe Visual Basic versie .NET, hebben zelfs een vaderkind relatie onder die formulieren. Omdat het in versie 6 dus nog niet kan, moeten we zelf het kind formulier na het laden van het MDI formulier inladen. Het laden van het kind formulier moet gedaan worden in het `Load` event van het MDI formulier.
Onderstaande figuren laten zien hoe de eigenschappen van het dialoogformulier en het normale formulier eruit zien.

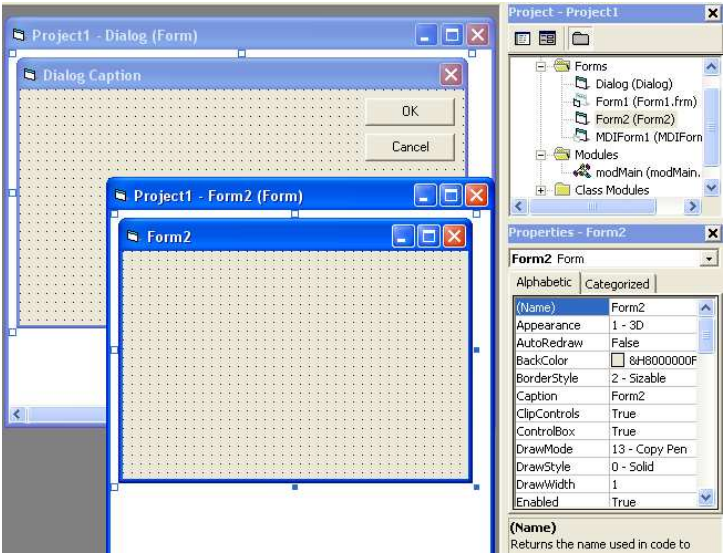


Figuur 1.

Visual Basic 6 kent helaas niet alle formulieren als aparte objecten. Dat komt doordat een formulier een sjabloon is. Hoe ook een formulier is ingesteld, alle sjablonen zijn hetzelfde. Figuur 1 laat daarom ook zien dat een dialoogformulier alleen maar ingesteld hoeft te worden met de eigenschap `BorderStyle`.

Dit dialoogformulier is gekozen uit het Project menu van Visual Basic, waar nog meer vooraf ingestelde formulieren gekozen kunnen worden. Heeft u per ongeluk een normaal formulier gekozen en u wilde een dialoogformulier? Geen nood, u hoeft alleen maar de eigenschap `BorderStyle` te wijzigen in optie '3 – Fixed Dialog' en u hebt een dialoogformulier. Het formulier verwijderen en dan een nieuw dialoogformulier uit het Project menu kiezen is dus helemaal niet nodig. Zie eens Figuur 2 welke optie de eigenschap heeft bij een normaal formulier.

Figuur 2.



Hier bij Figuur 2 kunt u het resultaat zien, zoals hierboven uitgelegd. Nu kunt u zien dat het actieve formulier Form2 geen dialoogformulier is, maar u hoeft maar de eigenschap `BorderStyle` te wijzigen in optie 3 en u hebt een dialoogformulier. Wat er dus eigenlijk gebeurt, is een heleboel code die verborgen zit en niet te achterhalen is. De conclusie is dus: alle formulieren worden ingesteld en gevuld met besturingselementen met maar één soort formuliersjabloon.

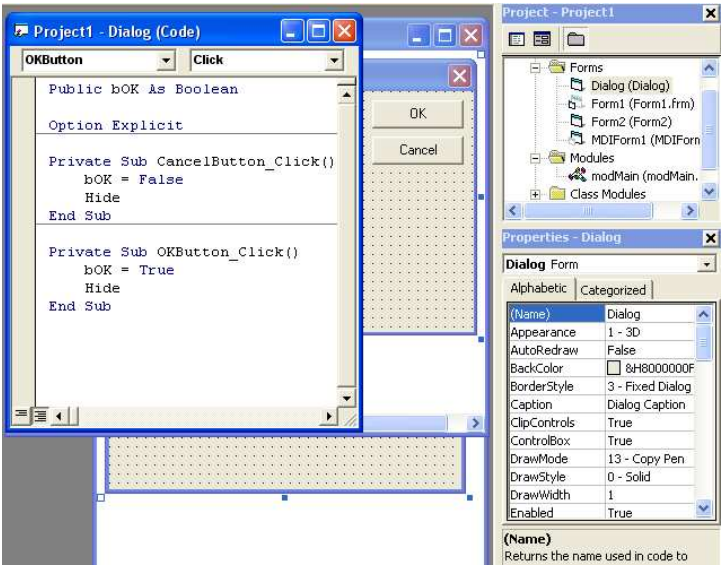
De OK en Cancel knoppen.

Het bepalen van wat er met de gegevens moet gebeuren, wordt niet direct in de besturingselementen gedaan. De OK en Cancel knoppen zorgen ervoor dat er een waarde wordt ingesteld die bepaalt of de gegevens bewaard moeten worden, maar ze doen het niet zelf. Bovendien moet het dialoogformulier niet in de knoppen al afgesloten worden, doe dat altijd pas na het uitvoeren van de keuzestructuur die de waarde van het dialoogformulier controleert, zie onderstaande code:

```
'zorg ervoor dat u hier de gegevens hebt, die eerder al opgehaald
'zijn
Load Dialog
'stuur de gegevens naar Dialog, alleen als er al gegevens zijn
Dialog.Show vbModal
If Dialog.bOk Then 'u mag ook gebruiken: If Dialog.bOk = True Then
   'bewaar hier de gegevens vanuit Dialog
End If
Unload Dialog
```

Figuur 3 laat zien hoe u ervoor kunt zorgen dat bovenstaande code goed werkt met het dialoogformulier. U zou natuurlijk ook willen weten waarom na de keuzestructuur pas het dialoogformulier ontladen mag worden en niet in de knoppen van het dialoogformulier zelf, hoewel dat mogelijk is.

Figuur 3.



En misschien zou u ook willen weten waarom de gegevens gewoon niet in de OK knop bewaard wordt. Ook dat is mogelijk, en Visual Basic stribbelt helemaal niet tegen als u op die manier de besturing programmeert. Ieder heeft zijn eigen manier om te programmeren. De manier die ik hier echter laat zien is de kortste en makkelijkste methode om een besturing, samen met een eventueel menu-item en dialoogformulier, te laten werken.
Nu kunt u ook zien waarom na de keuzestructuur pas het dialoogformulier ontladen mag worden, namelijk door het commando `Hide`. Dit commando zorgt ervoor dat het dialoogformulier gesloten wordt, maar nog wel in het geheugen blijft, zodat in de keuzestructuur de besturingselementen van het dialoogformulier herkenbaar blijven.
Hoe de gegevens bewaard moeten worden is van minder belang en heeft dus even niet met de besturing te maken. Weer als voorbeeld: wat maakt het uit of de wegwijzer een vrouwelijke of mannelijke stem heeft? Het gaat erom of de automobilist de straten kan nemen die de wegwijzer zegt.

Marco Kurvers

Het Microsoft .NET Framework.

Prima kunnen werken met de formulieren, besturingselementen, klassen of korter gezegd: prima kunnen werken met de algemene onderdelen dat Windows ons biedt, is natuurlijk heel fijn. Eerst moesten we in de Windows events zelf de formulieren en controls aanmaken en overal voor zorgen, maar dankzij het Visual Integrated Dynamic Environment (IDE) kunnen we nu zien wat we aan het doen zijn.

Een nieuwe boomstructuur.

Het Microsoft .NET Framework is een structuur die uit een heleboel takken bestaat met één basisstam, daarom wordt het een boomstructuur genoemd. Elke tak kan zelf ook weer een

basistak zijn met een heleboel takken. De basis van de hele structuur is het systeem zelf. Van hieruit zijn in de takken weer onderdelen te vinden die we in de basisstam niet vinden. Dat betekent niet dat een tak een onderdeel niet zou hebben dat wel in de basisstam aanwezig is. Ook al werkt alles als een boom, zelf hoeven we niet steeds onderdelen uit de basisstam te halen die de kleinste takjes niet zouden hebben. Eigenlijk is dat ook zo. De kleinste takjes hebben inderdaad niet zelf de onderdelen die in de basisstam aanwezig zijn. Toch kunnen we die onderdelen gebruiken, maar hoe? We kunnen dan beter de vraag stellen: hoe zit het .NET Framework in elkaar?

Klassen en objecten.

Het Framework is een hiërarchie dat uit klassen en objecten bestaat. Het is erg groot, zo groot dat het hier niet eens op de pagina past om te laten zien. Daarom hieronder een deel ervan, een deeltak die met bestanden lezen en schrijven te maken heeft.

		Object		
Stream			**TextReader**	
	BufferedStream			**StreamReader**
	FileStream			**StringReader**
	MemoryStream		**TextWriter**	
	NetworkStream			**StreamWriter**
	CryptoStream			**StringWriter**

Deze takken hebben, zoals eerder gezegd, niet alles zelf. Het meeste dat ze nodig hebben komt uit de basis klasse waar de Stream, TextReader en TextWriter klassen van afgeleid zijn. De stream klassen in de tweede kolom zijn weer afgeleid van de hoofdklasse Stream, en zo zit dat ook in de vijfde kolom in elkaar. De klasse StreamWriter is echter alleen afgeleid van de klasse TextWriter en zal niet de onderdelen hebben die in de klasse TextReader aanwezig zijn.
Het Object uiteindelijk, zal de initialisatie bevatten naar de juiste klasse. We hebben dat nodig voordat we de instanties kunnen gebruiken en de gegevens kunnen inlezen en kunnen bewaren.

Als we Visual Basic 6 en Visual Basic .NET naast elkaar openen dan zullen we haast geen verschil zien. Zeker niet als we lekker de besturingselementen op de formulieren aan het plaatsen zijn. Gaan we echter de code bekijken in .NET, dan zien we allemaal code van het formulier waar we mee bezig zijn. Zelfs de besturingselementen staan erbij met declaratie- en initialisatiecode waar we niet aan mogen komen. En er staat nu ook bij dat het formulier een klasse is, want deze wordt als een `Public Class` aangemaakt.

Klassen: wie of wat wordt er geërfd?

Werken met besturingselementen en klassen kan al in eerdere Basic versies, zelfs in Liberty BASIC. In Visual Basic 6 kunnen we alles zien wat we in de omgeving aan het doen zijn. Helaas kunnen we niet de techniek gebruiken die wel aanwezig is in Visual Studio .NET. Dankzij het .NET Framework kunnen we methoden en eigenschappen van de besturingselementen en de klassen 'lenen' of in computertaal gezegd: we kunnen code erven waar de klasse van afgeleid is. Een voorbeeld zijn de besturingselementen die de eigenschap `Caption` hebben. In Visual Basic 6 hebben de besturingselementen hun eigen eigenschap `Caption`. Als u een eigen besturingselement wilt maken, moet u weer dezelfde eigenschappen opnieuw maken. In Visual Basic .NET is dat niet meer nodig, want u hoeft maar de klasse in te stellen van waar u wilt afleiden en u erft meteen alles van wat er in die ingestelde klasse staat. Uiteindelijk zullen alle controls ook afleiden van de basiscontrol, die de `System.Control` klasse wordt genoemd. Er wordt afgeleid van de `Control` klasse. Andere onderdelen van het Framework worden afgeleidt van de `System` klasse zelf. Net zoals u in het tabel zag heeft ieder doel zijn eigen hoofddoel, dus u hoeft alleen maar de hoofdklasse te nemen. Telkens afleiden vanuit de basiscontrol is ook niet nodig. Wilt u de TextBox uitbreiden? U kunt dan uw nieuwe besturingselement maken en die afleiden van de originele TextBox. Als de eigenschap `Text` niet veranderd hoeft te worden, hoeft u ook geen nieuwe eigenschap te maken, want uw nieuwe besturingselement zal zelf de eigenschap `Text` herkennen. Het Framework kan dus uitgebreid worden met uw eigen klassen. Denk eraan dat nu ook de besturingselementen en ook de formulieren klassen zijn. Wat eerder niet kon, kan nu wel: zelf formulieren ontwerpen in code, maar het is altijd beter om gewoon de visuele omgeving te gebruiken.

Marco Kurvers

Mijn BASIC keuze: GW-BASIC.

GW-BASIC is een interpreter die door Microsoft in 1985 op de markt gebracht werd voor IBM-PC klonen en werd later meegeleverd met besturingssysteem MS-DOS.

Over de herkomst van de naam is geen duidelijkheid. Waarschijnlijk afkomstig van ***G**raphics **W**indows **BASIC***, maar het heeft niets te maken met het besturingssysteem Windows. Waarschijnlijk wordt met *Graphics Windows* in de naam verwezen naar de mogelijkheid om in grafische modus meerdere schermen te bewerken.

GW-BASIC is een afgeleide van BASICA, ook een op de PC onder MS-DOS draaiende interpreter. De GW-BASIC variant werd door Microsoft ter beschikking gesteld aan OEM's om in licentie te nemen en uit te brengen op eigen copyright en naam.

De interpreter vertaalt regel voor regel het programma met commando's naar machinetaal en voert die dan uit (runtime). Dat is beduidend trager dan een compiler, die eerst de hele broncode omzet naar machinetaal en uiteindelijk uitvoert. Microsoft leverde ook een BASIC compiler met de naam BASCOM.

Ondertussen kwam Microsoft met QuickBASIC, een betere BASIC compiler.

Om in bepaalde schermen een programma uit te kunnen voeren, moeten er grafische schermen ingesteld worden. Op oude IBM- en PC computers zijn de resoluties erg laag. Het statement `SCREEN` kent drie resoluties.

1. middenresolutie 320 bij 200 beeldpunten
2. hoge resolutie 640 bij 200 beeldpunten
3. super resolutie 640 bij 325 beeldpunten

Het statement `SCREEN` kent vier parameters:

```
SCREEN par1, par2, par3, par4
```

meestal is de tweede parameter niet nodig. De andere drie worden het meeste gebruikt.

par1 : beeldschermstand
0 = tekststand (40 of 80 kolommen)
1 = middenresolutie (320 x 200)
2 = hoge resolutie (640 x 200)
104 = superresolutie
105 = superresolutie met mogelijkheid voor het afdrukken van tekst

par3 en par4 : active en visuele graphics pagina
Voor 128K geheugen zijn deze beide parameters ingesteld op 3. Hebt u echt nog zo'n computer en gebruikt u GW-BASIC niet op Windows XP, probeer dan de gebruikershandleiding van uw computer te raadplegen. Hebt u meer dan 128K geheugen, dan kan het zijn dat u hiervoor 7 moet nemen. Als uw computer een kleurenadapter gebruikt dan kunnen deze parameters weggelaten worden.

Hier volgen enkele voorbeelden:

```
SCREEN 105,,3,3     : 128K micro (en sommige met meer dan 128K), superresolutie
SCREEN 105,,7,7     : sommige 256K micro's, superresolutie
SCREEN 1,0          : kleurenadapter, middenresolutie (par2=0)
SCREEN 1,,3,3       : 128 micro, middenresolutie
```

Voordat we het beeldschermresolutie instellen, moet er eerst geheugenruimte worden gereserveerd. Dit wordt gedaan met onderstaande statement:

```
CLEAR ,19202
```

dit is alleen het geval bij 128K micro's. Bij microcomputers met meer dan 128K geheugen is het statement helemaal niet nodig, sterker nog, het statement `CLEAR ,19202` zou zelfs een systeemcrash veroorzaken.

Werkt u met een kleurenadapter dan kunt u met het statement `COLOR` de achtergrondkleur en de afdrukkleur instellen.

```
COLOR par1, par2
```

par1 : achtergrondkleur
kleurcodes 0 t/m 15

par2 : 0 of 1
code 0 keuze uit de drie afdrukkleuren groen, rood of bruin
code 1 keuze uit de drie kleuren cyaanblauw, paars of wit. Deze drie kleuren hebben de codes 1, 2 en 3. Ze worden gebruikt in de `LINE`-, `CIRCLE`- en `PAINT`-statements.

U kunt de mooiste tekeningen maken in uw programma's als zowel horizontaal als verticaal dezelfde schaal gebruikt zou worden. In de Basic versies die we nu kennen is dat ook geen probleem, maar als u programmeert in GW-BASIC dan kan het tekenen nog vreemde situaties geven.
Horizontaal hebben we meer beeldpunten dan verticaal. Onderstaand programma zou bijvoorbeeld een vierkant moeten tekenen.

```
10 CLEAR ,19202 : SCREEN 105,,3,3    (of    10 SCREEN 105,,3,3)
20 LINE (50,50)-(250,250),1,B
30 A$=INPUT$(1)
40 END
```

Als u dit programma start, zal het een rechthoek tekenen en geen vierkant. De letter B in regel 20 vertelt het `LINE` statement dat het een Box (vierkant) moet tekenen. Het statement weet zelf niet dat de beeldpunten zowel horizontaal als verticaal niet gelijk zijn. Toch is er een mogelijkheid om in een juiste schaal te tekenen, zodat we wel een vierkant hebben. Er moet een functie worden gedefinieerd die de horizontale coördinaten (x-coördinaten) met 1,55 vermenigvuldigen. Er zal ook een nieuwe correctiefactor worden ingesteld voor de x-coördinaat. We krijgen dus uit die functie een nieuw scherm met de juiste schaal. Met mooie deelbare getallen zal het middelpunt telkens goed afgerond worden.

Voor superresolutie wordt onderstaande functie gedefinieerd:

```
DEF FNX(X)=INT(1.55*(50+X)+.5)
```

Hierdoor komt het midden (160,160) terecht op (326,160)
en (0,0) komt op (78, 0)
en (320,0) komt op (574,0)
Met (46+x) in plaats van (50+x) komt (160,160) terecht op (319,0); dat is iets dichter bij 320, maar 50 is mooier dan 46!

In de meeste programma's komt u daarom ook weer regel 10 tegen met daarbij de gedefinieerde functie.
Hebt u geen superresolutie, pas dan weer het `SCREEN` statement aan en verander in de functie het getal 50 in 5. Pas ook zonodig de factor 1.55 aan uw systeem aan. Denk er ook aan dat u met een vierkant beeld werkt, hetgeen tot gevolg heeft dat U=160 en V=160 veranderd moeten worden. De letters U en V zijn de oorsprong van het wiskundige coördinatensysteem; dikwijls is dit het midden van het beeldscherm (160,160), maar in lagere resolutie kan dit ook (100,100) zijn.

Marco Kurvers

Basic controls: de tabbladen. (Deel 2)

Nu u weet dat er twee soorten tabblad controls bestaan, de TabStrip en de SSTab, ga ik eens dieper op in wat de TabStrip control allemaal voor onderdelen heeft. Hoe kunnen we de tabs aansturen en de juiste frames activeren? Zoals uitgelegd in het eerste deel is de TabStrip veel kleiner en heeft veel minder mogelijkheden dan de SSTab. De frames, die de besturingselementen moeten vasthouden, zijn in de SSTab control niet nodig.

Binnenin de TabStrip.

De meerdere tabs, die u op de TabStrip ziet, hebben zelf geen eigen functie. Als u dubbelklikt op de control, ziet u onderstaande subroutine verschijnen:

```
Private Sub TabStrip1_Click()

End Sub
```

maar er is geen enkele event dat zich op een tabblad zou baseren. Alles moet u dus zelf doen om een goede tabbladen formulier voor elkaar te krijgen.

De control werkt met een tabs collectie. U kunt tijdens het ontwerpen tabs toevoegen en verwijderen bij gebruik van de eigenschappenpagina van de TabStrip control, maar u kunt dat ook doen tijdens de uitvoer door de methoden aan te roepen.

Zoals ik u ook in Deel 1 heb verteld is de TabStrip control geen container. Om de actieve tabbladen met de objecten bij elkaar te kunnen houden, moet u gebruik maken van Frame controls of andere containers die de grootte van het interne gebied op een tabblad kunnen vergelijken welke opgedeeld zijn bij alle Tab objecten in de control. Door gebruik te maken van een control array, bijvoorbeeld een Frame control array, kunt u elk item in de array specificeren met elk Tab object. Op de volgende pagina is een voorbeeld te zien.

```
Option Explicit

Private mintCurFrame As Integer 'Huidige Frame aanwezig

Private Sub TabStrip1_Click()
   If TabStrip1.SelectedItem.Index = mintCurFrame Then Exit Sub
   'Het frame wijzigen is niet nodig,
   'anders, verberg oude frame en laat nieuwe zien.
   Frame1(TabStrip1.SelectedItem.Index).Visible = True
   Frame1(mintCurFrame).Visible = False
   'Geef mintCurFrame een nieuwe waarde.
   mintCurFrame = TabStrip1.SelectedItem.Index
End Sub
```

Onthoud Wanneer u de controls in een container groepeert, moet u gebruik maken van de show/hide strategie, zoals hierboven, in plaats van de ZOrder methode om een frame naar voren te brengen. Anders, controls die toegang ondersteunen naar toetsen zoals de ALT + de toetsen, zullen reageren als toetsenbord commando's, evenzo als de container niet de opperste control is. Begrijp dus dat u ook de controls, als voorbeeld de OptionButton controls, in groepen bij elkaar moet houden in een container, anders zullen alle OptionButtons op het formulier als één groep werken.

Tip Gebruik een Frame control met de `BorderStyle` ingesteld als `None` in plaats van een PictureBox control als container. Een Frame control gebruikt minder overhead dan een PictureBox control.

De Tabs eigenschap van de TabStrip control is de collectie van alle Tab objecten. Elk Tab object heeft eigenschappen gekoppeld met zijn huidige status en vorm. Bijvoorbeeld, u kunt een ImageList control koppelen met de TabStrip control, en dan de images op de verschillende tabs gebruiken. U kunt ook een ToolTip met elk Tab object koppelen.

De Tabs collectie.

Een Tabs collectie is een container van meerdere Tab objecten.

Syntaxis (twee mogelijkheden)
tabstrip.Tabs**(***index***)**
tabstrip.Tabs.Item**(***index***)**

tabstrip	Een object expressie die evalueert als een TabStrip control.
index	Een integer of string die een member varieert van een object collectie. De integer is de waarde van de Index eigenschap van het gewenste Tab object; de string is de waarde van de Key eigenschap van het gewenste Tab object.

Tijdens de ontwerptijd, gebruik de Insert Tab en Remove Tab knoppen op de Tabs tabblad in de eigenschappen pagina (Properties Page) van de TabStrip control om Tab objecten van de Tabs collectie toe te kunnen voegen en te kunnen verwijderen.

De Tabs collectie heeft een `Count` eigenschap die de aantal Tab objecten teruggeeft. Om de Tabs in de collectie te kunnen bewerken, kunt u onderstaande methoden gebruiken in run time:

`Add`	Voegt Tab objecten toe aan de TabStrip control.
`Item`	Bepaalt het Tab object door de juiste sleutel of index van de collectie op te geven.
`Clear`	Leegt de hele Tab objecten collectie.
`Remove`	Verwijdert het Tab object door de juiste sleutel of index van de collectie op te geven.

tabstrip.Add**(***index, sleutel, tekst, image***)**

De `Add` methode heeft parameters, als volgt uitgelegd:

Parameter	**Omschrijving**
index	Optioneel. Een opgegeven integer die de positie aangeeft waar u de Tab wilt invoegen. Als u geen index opgeeft zal de Tab aan het eind van de Tabs collectie worden toegevoegd.
sleutel	Optioneel. Een unieke string die de Tab identificeert. Gebruik de sleutel om de juiste Tab te krijgen. Dit is hetzelfde als wanneer de `Key` eigenschap van het nieuwe Tab object wordt ingesteld nadat het nieuwe object is toegevoegd in de Tabs collectie.
tekst	Optioneel. Een string die verschijnt op een Tab. Dit is hetzelfde als wanneer de `Caption` eigenschap van het nieuwe Tab object wordt ingesteld nadat het nieuwe object is toegevoegd in de Tabs collectie.
image	Optioneel. De index van een gekoppelde ImageList control. De image wordt weergegeven op de Tab. Dit is hetzelfde als wanneer de `Image` eigenschap van het nieuwe Tab object wordt ingesteld nadat het nieuwe object is toegevoegd in de Tabs collectie.

Om tabs toe te voegen tijdens ontwerptijd, kunt u in de eigenschappenpagina (properties page) van de TabStrip control op de Insert Tab knop klikken. Die knop kunt u vinden op de tab die Tab heet. U kunt dan de geschikte velden invullen voor de nieuwe tab.

Om tabs toe te voegen tijdens uitvoertijd, gebruik dan de `Add` methode die de verwijzing naar het nieuwe ingevoegde of toegevoegde Tab object geeft. Bijvoorbeeld, de volgende code maakt een nieuwe tab met de tekst "Opties" welke de sleutel "MijnOpties" heet en als tweede tab, de index is 2, wordt ingevoegd:

```
Set X = TabStrip1.Tabs.Add(2, "MijnOpties", "Opties")
```

Marco Kurvers

BASIC cursus: Liberty BASIC (1).

Ik ga voor de Nieuwsbrieven een serie artikelen schrijven over het programmeren met Liberty BASIC. Natuurlijk zal dat via de Nieuwsbrieven tergend langzaam gaan, want een nieuwe Nieuwsbrief verschijnt om de drie tot vier maanden. Om de gang er wat in te houden kunt u daarom de listings die bij deze reeks artikelen behoren, ook op het Internet vinden. Ga daarvoor naar het forum van Liberty BASIC hier in Nederland (www.libertybasic.nl) en bekijk daar het subforum "Leren programmeren met Liberty BASIC 2009". Om de listings uit de artikelen te kunnen testen, heeft u Liberty BASIC of Just BASIC nodig. Ik ga ervan uit dat u Liberty BASIC vanaf het Internet www.libertbasic.com weet te downloaden en te installeren. Liberty BASIC is shareware, maar Just BASIC is freeware en te downloaden vanaf www.justbasic.com.
Vaak heb ik van cursusleden vernomen dat ik het "werken met bestanden" verwaarloos. U wordt op uw wenken bediend. Ik (we gaan samen) ga een LEDENADMINISTRATIE programma schrijven. Daarvoor ga ik eerst een leden-NAW bestand (naam adres woonplaats bestand) maken. Daarna maak ik uit dit NAW bestand een tweede aanvullend bestand waarin de NAW gegevens niet meer voorkomen, maar waarin wel de aanvullingen staan die bij de adressen uit het eerste bestand horen. Ik ga beide bestanden samengevoegd op het scherm tonen, maar ik ga beide bestanden apart op schijf bewaren.

Eerst maken we een NAW bestand.
Het wordt een Random Access File (een RAF bestand), hetgeen een bestand is dat uit records van een specifieke lengte bestaat. Het bestand zal LEDEN.BGG heten en de records uit mijn bestand zijn 163 tekens (karakters) groot. Elk record bestaat weer uit velden met de volgende vaste lengtes. Het lidnummer-veld is maximaal 7 posities lang. Voor de achternamen zijn per record 30 lege plekken gereserveerd, waardoor de langste achternaam maximaal 30 tekens lang kan zijn. Kijk goed naar Figuur 1 hieronder. Het is een invulformulier waarmee we records kunnen samenstellen.

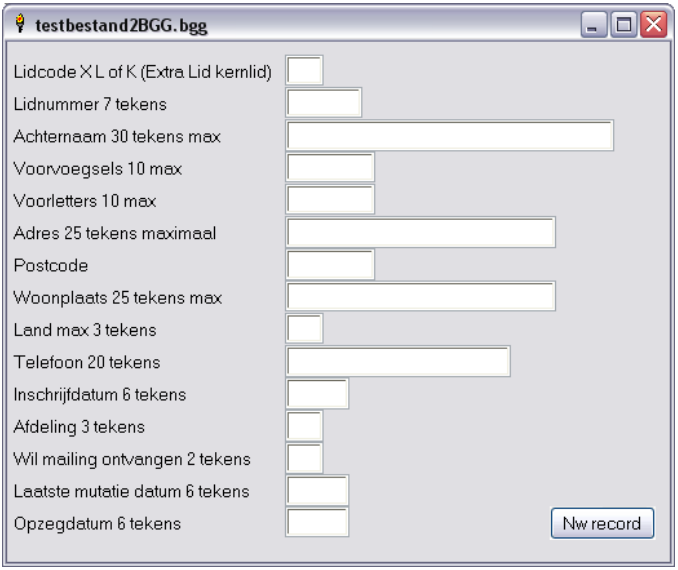
De figuur bestaat uit een standaard venster, zie zilvergrijze Windows opmaak, met rechtsboven de drie standaardknoppen voor het minimaliseren van het venster, herstellen van het vorig formaat en een afsluitknop waarmee het venster afgesloten kan worden. Het venster heeft als opschrift "testbestand2BGG.bgg", hetgeen een willekeurige tekst is.

```
Open "testbestand2BGG.bgg" for Window as #main
Wait
```

De "handle" van dit venster is main. Dat ziet u aan `#main`. Alle commando's voor dit venster beginnen daarom met `#main`. Rechtsonder staat een knop met het opschrift "Nw record". De "handle" van deze knop begint ook met `#main`, waardoor Liberty BASIC weet dat de knop in het venster `#main` hoort. Om de knop (in het jargon control geheten) ook een unieke identificatie te geven, geef ik de handle van de knop tevens een extensie **.knl**, hetgeen weer een willekeurige tekst is. De knop moet voor het openen van het venster waar het bij hoort, worden gedefinieerd.

```
button #main.knl,"Nw record",[nextRec],UL,420,360
```

Let op de verplichte komma's in de listingregel hierboven. Het deel tussen de twee rechte haken `[nextRec]` heet in het Engels een label. Het geeft in het computer programma de plek aan waar de uitvoering van het programma verder moet gaan, als de control - de knop in dit geval - bediend wordt. Als de knop bediend wordt – ingedrukt wordt – dan spreken we van een "event".



Figuur 1.

`UL,420,360` geeft de X, Y positie van de control (knop) aan ten opzichte van de linkerbovenrand (UpperLeft) van het venster `#main`.

Verder staan er 15 invulvelden (wit) en 15 tekstvelden waarin de teksten Lidcode X L of K enz. staan. De invulvelden zijn textbox controls en de tekstvelden zijn statictext controls.
De ENTER knop op uw toetsenbord is geen control en die veroorzaakt normaliter geen event. Daarom heb ik de NwRec knop geplaatst. De ENTER toets is een onderdeel (een toets) van het toetsenbord en de toetsen van het toetsenbord kunnen we wel scannen. Het aftasten van het toetsenbord (scannen) gebeurt dus niet automatisch door Windows.

U mag het venster zelf ontwerpen, maar u mag ook een hulp programma er voor gebruiken. Liberty BASIC kent vele hulp programma's om vensters te ontwerpen. Het programma

FREEFORM.BAS is zelf in Liberty BASIC geschreven en hoort bij het Liberty BASIC pakket.

Het maken van het ledenbestand is niet het hoofddoel van deze serie. Het ledenbestand maken wij als eenmalige opgave. Het bestand mag niet gewijzigd worden. Alle aanvullingen komen in een ander bestand terecht. Kenners onder u begrijpen direct dat we op den duur te maken zullen krijgen met INDEX bestanden, want de gegevens uit het ledenbestand zullen straks gesorteerd mogen worden, terwijl het ledenbestand zelf intact gelaten wordt. Gelukkig komen deze problemen pas in de laatste serie delen aan de orde. We gaan in de volgende delen eerst nadenken over hoe we de gegevens uit een bestand netjes op het scherm presenteren. In Figuur 1 valt op dat er geen e-mail adres wordt gevraagd. Dat komt omdat we wettelijk voor het vragen en opslaan van een e-mail adres een aparte toestemming van de eigenaar nodig hebben. We mogen eigenlijk ook geen "gegevens"-bestanden aan elkaar koppelen. Op de meeste logo's en plaatjes staat een copyright. Ook daar zullen we voorzichtig mee om moeten gaan.

Als dit uw eerste keer met Liberty BASIC of Just BASIC is, dan heeft u veel documentatie materiaal op te halen. De met Liberty BASIC meegeleverde tutorial is uit het Engels vertaald en te downloaden op
http://www.libertybasic.nl/viewtopic.php?f=17&t=329
De Nederlandse tutorial maakt deel uit van "Lessen1.zip". De tutorial (lb4tutorialNL.lsn) is met Liberty BASIC te runnen. In de tutorial wordt geprobeerd om u in 2x zes lessen een algemene basiskennis van programmeren te geven. In de Lessen1.zip file staan ook eenvoudige voorbeelden (bestanden0 en bestanden1.lsn) en lessen over de verschillende soorten, het gebruik en de aanmaak van bestanden. Ook dat is een aanrader om uw geheugen over het werken met bestanden wat op te frissen. Ik zal vaak voorbeelden uit deze Lessen1.zip aanhalen. Als u de Liberty BASIC topics (lijst met commando's en functies) wilt bestuderen, dan kunt u ze alle 281 in de ingebouwde help van LB vinden. Als u het liever in het Nederlands leest, dan volgt hier een link.
http://www.libertybasic.nl/index1.html
Voor een uitgebreide toelichting kunt u op de vetgedrukte topic klikken.

Hieronder volgt een klein deel van de listing die ik geschreven heb om een ledenbestand aan te kunnen maken. In dit listingdeel herken je de namen van de velden en de bijbehorende veldlengtes die we in onze listingen uit de serie artikelen zullen toepassen. Omdat het in deze serie artikelen echter niet gaat om het maken van dit "ledenbestand", heb ik de listing voor het maken van dit werkbestand op het forum geplaatst. Op het forum staat tevens een fictieve leden database met 25 records voor de lezers die niet eens enkele fictieve ledenrecords willen maken. Ga naar het forum en knip en plak de listing in de editor van Liberty BASIC. Maak daarmee uw eigen werk- ledenbestand of download gewoon mijn werkledenbestand. De listing op het forum vind je op
http://www.libertybasic.nl/viewtopic.php?f=15&t=428

Maak je eigen lijst (ledenbestand) of download het bestand dat ik speciaal voor deze serie artikelen aangemaakt heb om verder goed mee te kunnen doen met Programmeren met Liberty BASIC.

```
[openBestand]
open bestand$ for random as #r len = 163
bggBestIsOpen = 1
field #r, 3 as LIDCODE$, _
          7 as LIDNUMMER$, _
          30 as NAAM$, _
          10 as VOORVOEG$, _
          10 as VOORLETT$, _
          25 as ADRES$, _
          7 as POSTC$, _
          25 as PLAATS$, _
          3 as LAND$, _
          20 as TEL$, _
          6 as LIDSINDS$, _
          3 as AFD$, _
          2 as MAILING$, _
          6 as MUT$, _
          6 as OPZEG$
Return
```

In deel 2 van deze serie zullen we het ledenbestand op schijf opzoeken en openen. Daarna zullen we van elk record de NAW gegevens tonen in een listbox. Er komen knoppen om de getoonde gegevens te kunnen sorteren. We kunnen de gesorteerde gegevens in de listbox tonen.

In deel 3 van deze serie zullen we een geselecteerde record in een nieuw venster tonen. In hetzelfde venster zullen we ook extra items aan het gekozen record toe kunnen voegen. Die gegevens worden in een apart bestand bewaard. Het lidnummer wordt daarbij het gemeenschappelijk koppel-item tussen de twee bestanden. We zullen verder gebruik maken van radiobuttons, comboboxen, checkboxen en popupmenu's.
Oh ja, ik vergat bijna ook de texteditor te vermelden voor het memoveld dat we zullen programmeren.

Gordon Rahman

Cursussen

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

Software

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

Hoe te bestellen

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

ABN-AMRO nummer 49.57.40.314
HCC BASIC ig
Haarlem

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

Alle prijzen zijn inclusief verzendkosten voor Nederland en België.

Vraagbaken

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

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