

TNT Post
Port betaald
Port Payé
Pays-Bas

Nieuwsbrief

15de Jaargang feb 2008 Nummer 1



HCC - Postbus 6340 2001 HH Haarlem



Inhoud

Onderwerp

blz.

Onderdelen en bouwstenen	4
Wat doe jij met programmeren?	5
Wat doet een programmeur?	6
Software ontwikkelmethode	9
Puzzel: Gouden blokken	14
Visual Basic en de menu editor	14
Programma voor de valse stemcomputer	18
BASIC dialecten - spaghetticode	22
Oplossing: verf mengen	26
Visual Basic 2005 – andere veranderingen	27

Deze uitgave kwam tot stand met bijdragen van:

Naam	Blz
Marco Kurvers	4,6,9,14,22,27
Bart van Cleef	5
Henk van Weers	14
Fred Luchsinger	18
Piet Boere	26



Contacten

Functie	Naam	Telefoonnr.	E-mail
Voorzitter	Willem Gobel	0118-490168	voorz@basic-gg.hcc.nl
Secretaris a.i.	Willem Gobel Heemskerckplein 27 4384 BD Vlissingen	0118-490168	secre@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

Na twee mooie Kerstdagen en een prachtige jaarwisseling gaan we weer verder in een nieuw jaar. Het jaar 2008 waar we weer verder kunnen programmeren in BASIC. Misschien lukt het u wel dit jaar om een leuk programma voor elkaar te krijgen, of lukt het u misschien wel om iemand anders te kunnen helpen met een programmeer probleem. In ieder geval, we gaan er weer wat moois van maken.

Er zijn weer nieuwe bijdragen. Zo is er een deel 3 van “Wat doe jij met programmeren?” ingezonden.

Ook hoe je menu's maakt in Visual Basic laat ik u zien, en hoe we omgaan met verschillende soorten variabelen.

Marco Kurvers

Onderdelen en bouwstenen

In de programmeertalen gebruiken we onderdelen om het programma op te kunnen bouwen. Echter is dat niet het enige dat we doen; voordat we kunnen zeggen: ‘We hebben het programma opgebouwd’, moeten er eerst bouwstenen van worden gemaakt. Misschien vindt u dat vreemd en had u gedacht dat onderdelen al zelf bouwstenen zijn. Dat is echter niet zo. Bouwstenen zijn gewoon objecten die uit meerdere onderdelen bestaan. De onderdelen vinden we terug in de ‘control flow’, zie nieuwsbrief nr 4 van 2007.

Bouwstenen als projecten

Als u ziet hoe de map System32 van Windows eruit ziet dan zult u verbaasd zijn over het aantal bouwstenen die in de map staan. Allemaal DLL, OCX en EXE besturingsprogramma's die voor elke applicatie nodig zijn. Daarbij heeft Windows ze zelf ook nodig. Helaas is het gebruik van deze bouwstenen, in bijvoorbeeld Visual Basic 6, erg lastig. Hoewel de bouwstenen als projecten in elkaar zijn gezet kunnen ze in VB 6 niet meer als projecten in het hoofdproject bestuurd worden, tenzij men weet hoe de bouwstenen genoemd worden. Ze kunnen altijd nog in het menu ‘Referenties’ gekozen worden. Is er geen goede connectie; dan zit er niets anders op dan de onderdelen van de bouwstenen, zoals de methoden, aan te roepen via API.

Bouwstenen en Visual Studio .NET / 2005

Met Visual Studio .NET en Visual Basic 2005 Express Edition is bovenstaand probleem verdwenen. De versies .NET en 2005 zorgen ervoor dat de bouwstenen in de solution van het project terecht komen. Dat betekent dat alles wat de bouwstenen hebben ook direct te gebruiken zijn. Het hoeven niet speciaal bouwstenen te zijn van Windows, want ook het .NET Framework geeft de belangrijkste bouwstenen aan de solution.

De solution is de hele groep die helemaal uit projecten kan bestaan, zelfs uit andere programmeertalen is mogelijk. In de solution kunt u er voor kiezen om bijvoorbeeld een C++ klasse in te voegen terwijl u in de Basic taal bezig bent. Nu dat mogelijk is, is het werken met XML en met (web)servers dichterbij gekomen.

Om met meerdere soorten programmeertalen en bouwstenen, die normaal gesproken niet in Basic thuis horen, te kunnen gebruiken, is alleen Visual Studio .NET daar geschikt voor. Ook al is er in Visual Basic 2005 een solution aanwezig; deze versie geeft toch niet de mogelijkheden die met een Visual Studio .NET pakket wel gedaan kunnen worden.

Basic en de DataSet

Als laatste, over de bouwstenen, is er in .NET / 2005 een nieuwe objectlibrary ontwikkeld. De DataSet, een onderdeel van ADO.NET (ActiveX Data Objects) zie nieuwsbrief nr 4 van 2007. Dankzij de DataSet hoeven we niet meer de Access database te gebruiken. De DataSet is op zichzelf een database die gebruik maakt van XML.

Marco Kurvers

Wat doe jij met programmeren?

Het verhaal van Fred Luchsinger heeft mij zo aangesproken dat ikzelf maar eens een stukje ben gaan schrijven

In het nummer van mei 2007 vraagt Renee Bosveld zich af wat anderen zoal met programmeren doen. Bij mij is het begonnen in 1975 tijdens mijn HTS studie werktuigbouwkunde te Venlo. We begonnen met ponskaarten en ALGOL 60. Op de TU Eindhoven kwam hier voor de metaalbewerkingsmachines (draaien, boren, frezen) nog Fortran en enkele machinetalen bij.

Privé heb ik toen een Commodore 64 aangeschaft. Wat was dat mooi geprogrammeerd binnen de 64K, ongelooflijk! Ik programmeerde in BASIC en draaide het programma dan door een compiler, waarna het in machinetaal stond.

In de tussentijd heb ik een database gevuld voor een winkel in auto-onderdelen, die dit nog jaren heeft gebruikt.

Op deze computer heb ik in 1986 nog mijn afstudeerverslag, 40 karakters per regel, gemaakt. Na de Commodore 64 kwam de Commodore Amiga waarvan ik de GW-BASIC handleiding nog steeds heb en gebruik.

Vervolgens begon bij mij het PC-tijdperk en ben ik tot heden GW-BASIC trouw gebleven omdat ik geen zin heb om weer een nieuwe taal te leren, of zijn de opvolgers snel te leren?

Ik houd een administratie bij van mijn sigarenbandenverzameling maar mis nog een programma waar beelden aan toegevoegd kunnen worden. Stel het voor als een ledenlijst met pasfoto's (wie van de lezers heeft een oplossing?)

Het programmeren in GW-BASIC heeft inmiddels een leuke toepassing gekregen: het oplossen van logische puzzels. Momenteel ben ik bezig met het schrijven van een programma voor 'Bruggen bouwen' van Puzzelsport.

Mijn subroutines sla ik ook op, alle puzzel oplosprogramma's hebben dezelfde inlees-, wegschrijf-, wijzig- en parameters van de oplosroutines. De parameters en matrices hebben dezelfde grootte en namen, wat erg veel tijd spaart.

Graag zou ik van de lezers horen wie ook nog in GW-BASIC programmeert en samen dan puzzel oplosprogramma's te schrijven, want een ander kan mijn fouten er wel uit halen, waardoor ikzelf weer wat kan leren.

Doel is bij mij ook om hiervan zoveel te leren dat ikzelf een keer hoger scoor op het Nederlands kampioenschap 'Logisch puzzelen' van Puzzelsport.

Bart van Cleef, Delft

Wat doet een programmeur?

Softwareontwikkelaar

Een persoon die, of bedrijf dat zich bezighoudt met het programmeren van software is een softwareontwikkelaar. Ook wel beschreven als programmeur of computerprogrammeur. Het beroep wordt in het Engels (software) developer genoemd.

In de begintagen van de computer hield een programmeur zich nog bezig met de hardware en programmeerde zij (zelden hij) de computer door het maken van de nodige verbindingen, bijvoorbeeld door het plaatsen en verwijderen van stekkers en/of verbindingen. Tegenwoordig bestaan er programmeertalen die de programmeur veel hulpmiddelen aanreiken.

Het wetenschappelijk veld dat zich bezighoudt met softwareontwikkeling wordt software engineering genoemd. Dit is een vakgebied binnen de informatica of informatiekunde.

Ontwikkelproces

Het ontwikkelen van software is het totale proces van het schrijven van software. Dit bestaat uit een aantal stappen waarvan het eigenlijke programmeren stap twee, drie en vier beslaat. Niet voor ieder te ontwikkelen programma zijn alle stappen gebruikelijk, of zelfs maar relevant (lang niet ieder programma heeft een zo uitgebreide gebruikersinterface dat deze afzonderlijk geëvalueerd zal worden).

1. Het probleem vaststellen,
2. Het probleem opsplitsen in deelproblemen waar nodig/mogelijk,
3. Mogelijke oplossingen bedenken voor alle deelproblemen, nagaan welke deelproblemen met reeds eerder ontwikkelde programmatuur kunnen worden opgelost,
4. Gebruikersinterface vastleggen,
5. Een testversie maken met alleen de gebruikersinterface,
6. Gebruikersinterface testen op gebruiksvriendelijkheid,
7. Functionaliteit programmeren (soms met gebruik van verschillende programmeertalen),
8. Programmacode documenteren,
9. De door mensen leesbare code in uitvoerbare code omzetten, door middel van compileren,
10. Het deelprogramma testen om te kijken of het in alle toelaatbare omstandigheden werkt. Anders terug naar stap 7,
11. Deelprogramma's samenvoegen en weer (uitgebreid) testen om te kijken of het totale programma ook in alle toelaatbare omstandigheden werkt. Anders terug naar stap 7,
12. Handleiding schrijven,
13. Cursus ontwikkelen. Deze stappen worden in grotere projecten doorgaans niet door dezelfde personen genomen. Bovendien worden niet al deze stappen uitgevoerd voor elk stuk software. Voor typische server software geldt over het al-

gemeen dat er geen gebruikersinterface is in de betekenis zoals die in de stappen worden gebruikt.

De ervaring leert dat in dit traject het feitelijk programmeren, ook wel de implementatie genoemd, slechts 30 tot 35% van de tijd in beslag neemt. Een andere ervaring is dat waar 1 programmeur 1 maand voor nodig heeft, doen 2 programmeurs in 2 maanden; de stap naar teamwerk is een hele grote stap die in een niet al te groot project vaak averechts werkt. Een veelgebruikte benaming voor dit verschijnsel is de 'mythical man month'. Een vergelijking die vaak wordt gebruikt is dat 1 zwangerschap niet sneller kan verlopen door 9 vrouwen tegelijkertijd zwanger te laten zijn. Dit verschijnsel is een manifestatie van de onmogelijkheid om een bepaalde taak in kleinere taken onder te verdelen. Op een bepaald moment is de inspanning om te communiceren met de andere leden van een team zo groot dat de productiviteit van het totale team terugloopt als er meer mensen aan het team worden toegevoegd.

Berucht is ook de 80/20-regel: 80% van de applicatie wordt geschreven in 20% van de tijd. 80% van de tijd is nodig om de resterende 20% functionaliteit te ontwikkelen. In de praktijk betekent dit ook zowel als: software is nooit af.

Ontwikkelmethodes

Een software ontwikkelmethode wordt gebruikt om het ontwikkelproces systematisch aan te pakken. De op bladzijde 6 beschreven stappen hebben binnen de verschillende ontwikkelmethodes hun eigen plaats. Ontwikkelmethodes zijn in te delen in categorieën:

- watervalmethodes
- iteratieve methodes

Een watervalmethode bestaat uit een aantal stappen die na elkaar worden doorlopen. Het eindresultaat van een stap is het beginpunt voor de volgende stap. Dit is de klassieke manier om automatiseringsprojecten aan te pakken. Een nadeel van deze methode is de doorlooptijd die nodig is om tot het eindresultaat te komen. Bovendien wordt uitgegaan van de veronderstelling dat bij de start van een project alle eisen en wensen bekend zijn. In de praktijk is dit over het algemeen niet het geval.

Iteratieve methodes lijken op watervalmethodes, maar gebruiken iteraties om delen van de functionaliteit te bouwen. Zo kan bijvoorbeeld eerst de functionaliteit rond de invoer van gegevens worden gemaakt en getest, waarna verder wordt gegaan met een iteratie waarin de uitvoer wordt gebouwd. Door deze onderverdeling in kleinere delen is het gemakkelijker om in te spelen op wijzigende eisen en wensen. Daarbij is de planning beter te controleren.

In de praktijk zijn verschillende methodes te combineren. Niet elke stap uit een be-

paalde methode is verplicht. Wel wordt elke stap methodisch uitgevoerd, het mag nooit zoiets zijn als 'we rommelen maar wat aan, en als het werkt zijn we blij', wat bij amateur-programmeurs wel regelmatig voorkomt.

Voorbeelden van software ontwikkelmethodes zijn eXtreme Programming en Rapid Application Development.

De hulpmiddelen

Programmeren

Voor het maken van de software gebruikt de programmeur verschillende hulpmiddelen:

- een teksteditor om de bron-tekst te schrijven.
- een of meerdere codegenerators om vanuit een eenvoudige brontekst of databasetekst bron-tekst automatisch te genereren.
- een compiler om (delen van) het programma naar machinetaal te vertalen, eventueel een assembler, soms een interpreter.
- een archiver en een linker om alle delen samen te voegen zodat een uitvoerbaar programma of een programma bibliotheek ontstaat.
- een verscheidenheid aan test-tools om regressie testen mee uit te voeren, aan profiling te doen en meestal een debugger waarmee de programmeur gericht technische fouten op het spoor kan komen en verhelpen.
- een version control systeem voor het bijhouden van wijzigingen in - en verschillende versies van onderdelen van de software.
- een Defect-database om problemen te managen (zoals bijvoorbeeld Bugzilla).

Soms maakt een programmeur gebruik van een geïntegreerde ontwikkelomgeving (IDE) waarin teksteditor, compiler, debugger en vele andere software ontwikkelgereedschappen geïntegreerd zijn. Hierdoor is de ontwikkelcyclus 7 tot 11 soms drastisch te versnellen.

Veel IDE's zijn voor slechts 1 technologie, vaak van dezelfde leverancier, geschikt. Enkele IDE leveranciers bieden hiervoor een oplossing, waarmee in 1 IDE met verschillende compilers of debuggers gewerkt kan worden.

Projectmanagement

Om het hele software ontwikkelproces te beheersen wordt vaak gebruikgemaakt van algemene projectmanagementsoftware, zoals programma's om Gantt-charts te maken, kosten te beheersen en communicatie binnen een groep te faciliteren (Group Support Systems). Om bij te houden welke fouten zijn gevonden en opgelost wordt vaak een issue tracker gebruikt.

Software ontwikkelmethode

Een software ontwikkelmethode is een structuur die je op legt aan het ontwikkelen van softwareproducten. Synoniemen hiervoor zijn ook wel softwarelevenscyclus en softwareproces. Er zijn vele verschillende modellen voor dit soort processen, waarbij elke een bepaalde aanpak beschrijft tot een bepaald aantal taken of activiteiten die plaats vinden tijdens het proces.

Processen en meta-processen

Een groeiend aantal software ontwikkelingsbedrijven implementeren software ontwikkelmethodes. Veel van deze bedrijven bevinden zich in de defensiesector, met name in de Verenigde Staten, waar deze bedrijven een 'waardering' gebaseerd op dienen te hebben om contacten te verkrijgen. ISO 12207 is een standaard voor het omschrijven van een methode om een levenscyclus voor een project te selecteren, implementeren en te controleren.

Het Capability Maturity Model (CMM) is een van de meestgebruikte modellen. Door onafhankelijke toetsing kan bepaald worden hoe goed een bedrijf software produceert tot aan hoe ze hun processen definiëren en uitvoeren. ISO 9000 omschrijft standaarden om processen formeel te organiseren en documenteren.

ISO 15504, ook wel bekend als 'Software Process Improvement Capability Determination' (SPICE), is een raamwerk om softwareprocessen te toetsen. De levenscyclus van het software ontwikkelingsproces wordt wijdverspreid gebruikt. Deze standaard is gericht op het opstellen van een helder model voor het vergelijken van processen. SPICE wordt, net als CMM en CMMI, veel gebruikt. Het modelleert processen om softwareontwikkeling te beheren, beheersen, leiden en te controleren. Dit model wordt daarna gebruikt om te bepalen wat een ontwikkelingsorganisatie echt doet tijdens het ontwikkelen van software. Deze informatie wordt vervolgens geanalyseerd om zwakheden te identificeren en verbetering te bevorderen. Ook identificeert het sterke punten die bevorderd kunnen worden of kunnen worden geïntegreerd in vaste gewoontes voor de organisatie of een team.

Six Sigma is een projectmanagementmethode die data en statistische analyse gebruikt om de prestatie van een bedrijf te verbeteren. Het identificeert en elimineert 'defecten' binnen producerende- en service georiënteerde processen. Het maximaal toegestane aantal defecten is 3,4 per miljoen gelegenheden. Echter is Six Sigma een productiegeoriënteerde methode en geen software-georiënteerde methode en dient het verder ontwikkeld te worden om van toepassing te kunnen zijn op softwareontwikkeling.

Procesactiviteiten/stappen

Software-engineeringprocessen zijn opgemaakt uit vele verschillende activiteiten, voornamelijk de volgende. Ze worden gezien als sequentiële stappen in het waterval-

proces, maar andere processen kunnen ze opnieuw rangschikken of combineren op verschillende manieren.

Analyse van de vereisten

Het achterhalen van de vereisten van een benodigd softwareproduct is de eerste stap van het maken ervan. Hoewel klanten waarschijnlijk denken te weten wat de software moet doen, is er misschien vaardigheid en ervaring in software-engineering nodig om onvolledige, dubbelzinnige en tegenstellende vereisten te herkennen.

Specificatie

Specificatie is de taak van het in detail beschrijven van de te schrijven software, op een mathematisch rigoureuze wijze. In de praktijk, zullen de meest succesvolle specificaties geschreven worden voor het begrijpen en optimaliseren van applicaties die reeds redelijk doorontwikkeld waren, hoewel veiligheidskritieke softwaresystemen meestal zeer doordacht worden gespecificeerd voor het ontwikkelen van de applicatie.

Software Architectuur

Met de architectuur van een softwaresysteem wordt een abstracte representatie van dat systeem bedoeld. Architectuur houdt zich bezig met het vaststellen of een systeem de vereisten haalt van het op te leveren product, alsook het mogelijk maken dat toekomstige vereisten voldaan kunnen worden. De architectuur spreekt ook interfaces aan tussen het systeem en andere softwaresystemen, alsook de onderliggende hardware en het besturingssysteem.

Implementatie (of Programmeren)

Het reduceren van een ontwerp tot code mag misschien de meest voor de hand liggende onderdeel zijn van een baan als software-ingenieur, maar het hoeft niet noodzakelijk het grootste onderdeel te zijn.

Testen

Het testen van (onderdelen) software, met name wanneer twee of meer verschillende ontwikkelaars samen werken, is een taak van de software-ingenieur.

Documentatie

Een heel belangrijke (en vaak onderschatte en over het hoofd geziene) taak is het documenteren van het interne ontwerp van software met als doel bij te staan bij toekomstig onderhoud en verbeteringen.

Documentatie is het meest belangrijke bij externe interfaces.

Software Training en Support

Een groot percentage van de softwareprojecten gaat verkeerd omdat de ontwikkelaars niet inzien dat het niet uitmaakt hoeveel tijd en planning een team stopt in het maken van software, als niemand in de organisatie het uiteindelijk ook daadwerkelijk gaat gebruiken. Mensen zijn af en toe resistent tegen verandering en gaan het zich begeven op onbekend terrein uit de weg. Dus als onderdeel van de implementatiefase is het zeer belangrijk trainingen te geven aan de meest enthousiaste gebruikers (om opwinding en vertrouwen op te bouwen), en deze training te verschuiven naar de neutrale gebruikers,

gemengd met gretige aanhangers, en als laatste de rest van de organisatie mee te krijgen om de nieuwe software te accepteren. Gebruikers zullen vele vragen hebben en softwareproblemen tegenkomen wat leidt tot de volgende fase van software.

Onderhoud

Onderhouden en verbeteren van software om om te kunnen gaan met nieuwe problemen of nieuwe vereisten kan veel meer tijd nemen dan de initiële realisatie van de software. Het is niet alleen dat je misschien code moet toevoegen die niet helemaal past in het originele ontwerp, maar alleen al het bepalen hoe de software werkt en zich gedraagt nadat ze opgeleverd is en eventueel al veranderingen heeft ondergaan kan al veel moeite kosten voor een ontwikkelaar. Ongeveer tweederde van alle softwareontwikkelingswerk is onderhoud, maar deze statistiek kan misleidend zijn. Een klein deel hiervan is maar het repareren van bugs. Het meeste onderhoudswerk is het uitbreiden van systemen met nieuwe dingen, welke vaak gezien kunnen worden als nieuw werk. Ter vergelijking, ongeveer tweederde van alle werk in civiele techniek, architectuur en constructies is ook onderhoud op een vergelijkbare manier.

Procesmodellen

Al tientallen jaren probeert men voorspelbare processen te vinden voor het verbeteren van productiviteit en kwaliteit. Sommige modellen proberen de onhandelbare taak van het softwareproces te systematiseren en formaliseren. Anderen passen projectmanagementtechnieken toe op het schrijven van software. Zonder projectmanagement kunnen softwareprojecten makkelijk te laat of buiten budget opgeleverd worden. Het feit dat veel softwareprojecten niet voldoen aan de eisen op het gebied van functionaliteit, budget en/of planning bewijst dat projectmanagement erg lastig is.

Watervalmodellen

Het bekendste en oudste model is de watervalmethode, waarbij ontwikkelaars (ongeveer) de volgende stappen achter elkaar uitvoeren:

- Vereisten opstellen en analyseren
- Opstellen van een aanpak
- Opstellen van een raamwerk voor de oplossing
- Code schrijven
- Testen
- Uitrollen en onderhouden

Na elke stap wordt er doorgegaan met de volgende stap. Net als bij het bouwen van een huis wordt er na het bouwen van de muren niks meer gedaan aan de fundering. Als er geen iteratie in de planning is opgenomen kan er in principe niet meer teruggegaan worden naar een eerdere stap om fouten te verbeteren. Op deze manier worden fouten pas opgelost in de eindfasen van een project, waar het oplossen veel meer tijd, moeite en geld kost.

In oude procesmodellen (CMM), werd het ontwerpen voor het coderen gedaan, meestal door verschillende mensen per processtap.

Iteratieve processen

Iteratieve ontwikkeling omschrijft een methodiek om de stappen van een project op te delen in kleine stukken, om zo te voorkomen dat een project uitloopt in een ramp door problemen of foute aannames. Iteratieve processen hebben de voorkeur bij (commerciële) ontwikkelaars, omdat ze de mogelijkheid bieden de juiste beslissingen te maken voor een klant die goed kan omschrijven wat hij wil.

Agile software ontwikkelingsprocessen zijn gebaseerd op de grondbeginselen van iteratief ontwikkelen. Bovenop deze basis hebben ze een meer op mensen gerichte blik dan traditionele aanpakken. Agile processen gebruiken feedback in plaats van planning als basis. De feedback wordt geleverd door regelmatige tests en het evolueren van de software.

Agile processen lijken efficiënter te zijn dan oudere 'traditionele' aanpakken, ze hebben minder programmeertijd nodig om een product van hogere kwaliteit af te leveren, maar ze hebben als nadeel dat het niet goed mogelijk is een langetermijnplanning te maken met zulke procesmodellen. Het komt erop neer dat ze het meeste waar voor je geld bieden, maar dat je nooit weet wanneer je je waar krijgt...

Extreme Programming (XP) is de bekendste agile-methode. Bij XP worden de fases in extreem kleine stappen uitgevoerd in tegenstelling tot andere non-agile-modellen. De eerste keer dat de stappen doorlopen worden kan een dag of een week duren, in tegenstelling tot bijvoorbeeld het watervalmodel, waar een stap maanden of jaren kan duren. Als eerste worden er automatische tests geschreven, om zo concrete doelen te hebben om naar te ontwikkelen. Daarna wordt er door programmeurs in paren ontwikkeld totdat alle tests doorlopen zijn en de ontwikkelaars geen nieuwe tests meer kunnen bedenken. Ontwerp en architectuur komen voort uit refactoring en komen na het coderen. Het ontwerpen wordt gedaan door dezelfde mensen die ook ontwikkelen. Het werkende, maar incomplete systeem wordt uitgerold en gedemonstreerd aan de gebruiker(s). Pas als het deel is goedgekeurd door de gebruiker, wordt er doorgegaan met ontwikkelen aan het volgende stuk van de software.

Formele methoden

Formele methoden zijn wiskundige aanpakken om soft- en hardware problemen op te lossen op het niveau van requirements, specificaties en ontwerpen. Voorbeelden van enkele formele methodes zijn bijvoorbeeld de B-Methode, Petri nets, RAISE en VDM. Verschillende formele specificatieschriften zijn ook voorhanden zoals de Z-notatie. Algemeen gesteld kan dit gebruikt worden voor het bouwen en valideren van het gedrag van een applicatie door het ontwerpen van een aantal Eindige Toestandsautomaten.

Op Eindige Toestandsautomaten (FSM) gebaseerde Methodologieën,

maken uit te voeren softwarespecificaties en omzeilen van conventionele manier van programmeren mogelijk.

Recente aanpakken proberen specificaties en code in een activiteit te stoppen om te garanderen dat de specificaties en code goed op elkaar aansluiten. Terwijl agile-methoden specificaties van alle vereisten in code propageert, proberen methoden zoals Virtuele Eindige Toestandsautomaat (VFSM) uitvoerbare specificaties te ontwikkelen om zo het hele programmeren te ontwijken.

En Basic?

Het ontwijken van de code wordt steeds vaker gedaan. Zo is de programmering, dat ook in Basic gedaan wordt, maar één keer nodig. Maar dan moet men wel aan de Virtuele Eindige Toestandsautomaat houden.

Een voorbeeld van zo'n techniek is het automatiseren van machines en ook robotica. Om de techniek in de code toe te passen moet men aan de Control Flow houden zodat de machine goed bestuurd wordt.

Voor zulke machines werd vooral Basic gebruikt. BBC BASIC was één van de beste programmeertaal om robotarmen te besturen.

Als u van plan bent om in Basic een groot project te schrijven dan is de software ontwikkelmethode van groot belang. Ook al kan dit onderwerp best moeilijk zijn om te begrijpen, toch wordt het overal gebruikt.

Hier zal ik meermalen op terugkomen omdat het niet alleen in theorie, maar ook in de praktijk gebruikt wordt. Ik kan al een voorbeeld geven op de keuzestructuren (IF ... THEN constructies). Al deze structuren is in principe iteratieve code, omdat bepaald moet worden wat het moet doen.

Wilt u meer weten?

Zie de site <http://nl.wikipedia.org/wiki/Softwareontwikkelmethode>

Mocht deze niet werken, probeer dan: <http://www.wikipedia.nl>
Ga naar 'zoeken' en type het woord 'Softwareontwikkelmethode' in.

Onderaan de site, bij 'Zie ook', kunt u naar andere ontwikkelmethodes surfen die ik hier niet besproken heb.

Marco Kurvers

Puzzel: Gouden blokken

Twee inbrekers roofden in het donker een vitrine leeg bij de tentoonstelling:

“Kunst in goud”

Bij daglicht bleek het te gaan om 12 kubusjes zuiver goud met de afmetingen 1 x 1 x 1; 2 x 2 x 2 3 x 3 x 3 enz t/m de laatste, die de afmeting had van 12 x 12 x 12 cm. Bij elkaar bijna 117 ½ kilo goud.

Ze gunden elkaar geen grammetje meer dan de ander en tijd voor omsmelten was er niet bij. Een weegschaal hadden ze ook niet, maar wel een centimeter om de afmeting van elke kubus te meten.

Programmeer uitdaging:

Schrijf een programma om uit te rekenen hoe de boeven de buit in precies gelijke gewichten kunnen verdelen.

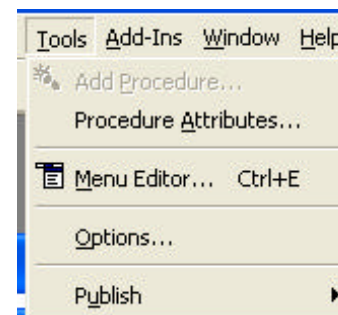
Henk van Weers

Visual Basic en de menu editor

Menu's maken was in oudere Basic versies koek en ei. Alleen de code was voldoende om horizontale of verticale menu's te maken. Om de gebruiker de keuze te laten maken, werd vaak gevraagd: 'Geef uw keuze:'

Visual Basic

Tegenwoordig, voor programmeurs die met de IDE werken, is er een andere manier om een menu te maken. De IDE heeft een geïntegreerd menu editor dat geopend kan worden door naar het menu 'Tools' te gaan en te klikken op 'Menu Editor...', of gebruik te maken van de sneltoets Ctrl + E.



Het nadeel van het menu editor is; het is een dialoog-scherm. Dat betekent dat tijdens het ontwerpen van het menu de IDE vergrendeld is. U kunt niet werken met uw project en werken met het menu editor tegelijk.

Hoe het menu editor werkt wordt op de volgende bladzijde uitgelegd.

Voordat u hieraan begint, bedenkt u eerst wat u wilt. Maak in het kladblok of in Word uw menu, en als u denkt dat u hem heeft dan kunt u uw menu hier invoeren. Zonder schets gelijk al het menu maken kan leiden tot een rommeltje en dat is toch niet de bedoeling. Hieronder worden de onderdelen van het menu editor omschreven.

Caption:

Type hier de omschrijving van het menu item. Dat item zal op het formulier, waar u mee bezig bent, verschijnen.

Name:

Geef een duidelijke naam voor het menu item. Die naam zal tevens het object worden waar u in de code mee kunt werken.

Index:

U mag meerdere menu items als een array gebruiken. Zorg er wel voor dat u steeds dezelfde naam inlaadt. U hebt veel code nodig om dynamisch dit menu item methode te gebruiken. Niet aanbevolen.

Shortcut:

Geef, indien u dat wenst, een sneltoets voor het menu item. Wanneer u op het pijltje klikt, zal een lijst met sneltoetsen verschijnen. Houd de sneltoetsen uniek met de andere sneltoetsen.

NegotiatePosition:

Indien het hoofdformulier een actief kindformulier bevat, dan wordt het menu van dit kindformulier getoond i.p.v. het menu van het hoofdformulier. Enkel menu items waar NegotiatePosition ? 0 blijven behouden.

Checked

Kies uit het menu welk menu item gevlagd mag worden. Als voorbeeld kunnen er menu items: 'Vet, Schuinschrift, Onderstrepen' zijn. Kiest u voor 'Vet', dan zal het hele programma daarop reageren, omdat dat item gevlagd is.

U moet zelf de eigenschap 'Checked' programmeren. Vergeet dus niet in de Click gebeurtenis van een menu item de 'Checked' eigenschap te controleren en op 'False' of 'True' te zetten.

Enabled

Maakt het menu item onzichtbaar of zichtbaar. Onzichtbaar wil niet zeggen dat u het menu item niet meer ziet. Het zal alleen buiten werking zijn.

Visible

Zelfde methode als bij 'Enabled', maar nu zal deze wel verdwijnen wanneer het menu item onzichtbaar wordt gemaakt.

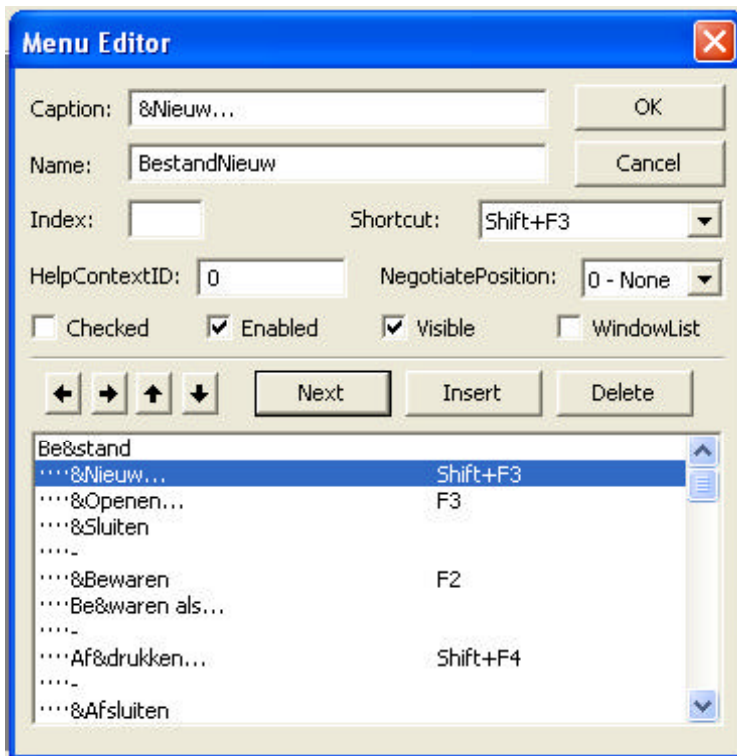
WindowList

Kies deze optie als u een lijst in het menu wilt gebruiken. Deze optie wordt gebruikt in de Multiple Document Interface.

U ziet dat de naam begint met de ampersand '&'. Het symbool zorgt ervoor dat de letter 'B' onderstreept zal worden en dat de gebruiker de letter samen met de ALT toets kan gebruiken.

Als u klaar bent met het menu item dan klikt u op de knop 'Next'. Klik alleen op de knop 'Insert' als u tussen het menu items in wilt voegen. Als u

een menu item wilt verwijderen, selecteer er dan een en klik op de knop 'Delete'.



Hier ziet u een compleet gevulde menu editor. Hoewel de naam niet met 'mni' of 'mnu' begint, is dit ook toegestaan. Als u maar een duidelijke naam gebruikt. Onderaan het figuur worden de besturingspijlen omschreven.

Pijl links

Klik op die pijl als u een submenu item lager wilt, bijvoorbeeld terug naar het hoofdmenu. Het hoofdmenu is het menu item zonder de punten.

Pijl rechts

Klik op die pijl als u voor een menu item één of meerdere submenu items wilt maken. Indien u alleen nog het hoofdmenu hebt, zal als eerste maar vier punten verschijnen, zoals u in bovenstaand figuur ziet.

Pijlen omhoog/omlaag

Klik op die pijlen om de aanwezige menu items te bewerken.

Streepje

Type in de tekst een streepje om een lijn in de menulijst te krijgen. Geef een naam op die duidelijk aan moet geven dat het om een lijn gaat. Gebruik geen ampersand bij het streepje.

Drie punten achter de tekst

Als er een dialoogscherm moet komen na een klik op het menu item, dan kunt u achter de tekst van het menu item drie punten plaatsen. In bovenstaand figuur kunt u zien wanneer ze nodig zijn en wanneer niet. Natuurlijk mag u helemaal zelf bepalen wat u met het menu doet, want ze zijn niet verplicht.

Hoe verder...

Dubbelklik op een menu item om de klik gebeurtenis (click event) code te openen. Het geraamte, dat automatisch wordt gemaakt, ziet er uit als onderstaande code. De naam die u ziet is als een voorbeeld, want elke naam van een menu item zal anders zijn.

```
Private Sub BestandNieuw_Click()  
    'type hier uw code  
    'deze code is de actie, zodat de klik wat doet  
    'bijvoorbeeld een nieuw scherm openen  
End Sub
```

Verander nooit de naam van de subroutine in code, en laat die subroutines altijd op Private staan. Als u later de namen toch verandert in het menu editor en u hebt al subroutines in code zoals het bovenstaande, dan zullen ze niet mee veranderen. Dat betekent dat de Click ongeldig wordt en niet meer als een event wordt gezien. Visual Basic zal dan de hele subroutine als gebruikers subroutine beschouwen.

Marco Kurvers

Programma voor de valse stemcomputer

De opgave was om een programma voor zo'n apparaat te ontwerpen.

Ik heb als bevooroordeelde partij "Partij C" genomen. Om zo min mogelijk op te vallen geef ik, in onderstaand voorbeeld "Partij A" per 100 stemmen 15 stemmen extra waarvan hij er dan weer 1 aan C moet afstaan. Aan J geef ik, per 100 stemmen, 5 stemmen extra. Dit gebeurt in de sub "Testspel" waarmee, na 5 ingaven, 1 stemronde van 100 aangemaakt wordt, want anders blijf je invoeren voordat je aan tenminste 100 stemmen komt. Die sub wordt dus in een "werkelijk" programma niet aangeroepen. Alle partijen staan per 100 beurten 1 stem aan C af m.u.v die, die geen één stem gekregen hebben, want negatieve stemmen bestaan niet. Wat verder zichtbaar moet zijn is, in welk stembureau je bent om te stemmen en ook de datum waarop je stemt. De Chef van het stembureau start vanaf het openingsscherm het scherm waarop gestemd wordt, en als er geen stemmers meer zijn sluit hij dit af door de vraag op welke partij hij stemt met een # te beantwoorden. Daarmee komt hij weer in het openingsscherm terecht en kan hij de stemresultaten bekijken, mits hij het goede wachtwoord kent hetgeen hier zijn eigen functie is. Maakt hij een typfout of is er een ander die dit bekijken wil dan kan men 3 pogingen wagen, na 3 foutieve ingaven komt het openingsscherm weer tevoorschijn.

Alleen de "fraudeur" kent ook het 2de wachtwoord (stiekem).

De schermen zien er zo uit:

Fig. 1: Het openingsschermb

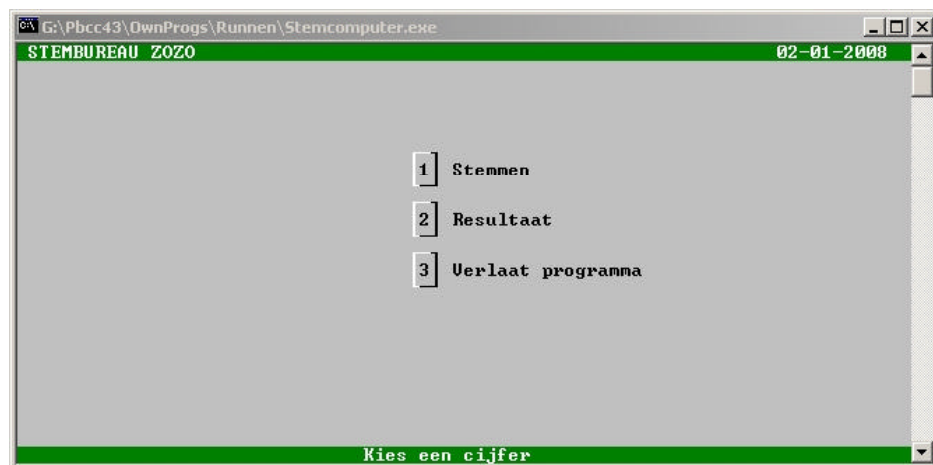


Fig. 2: De Chef van het stembureau heeft 1 gekozen, dit ziet degene die gaat stemmen

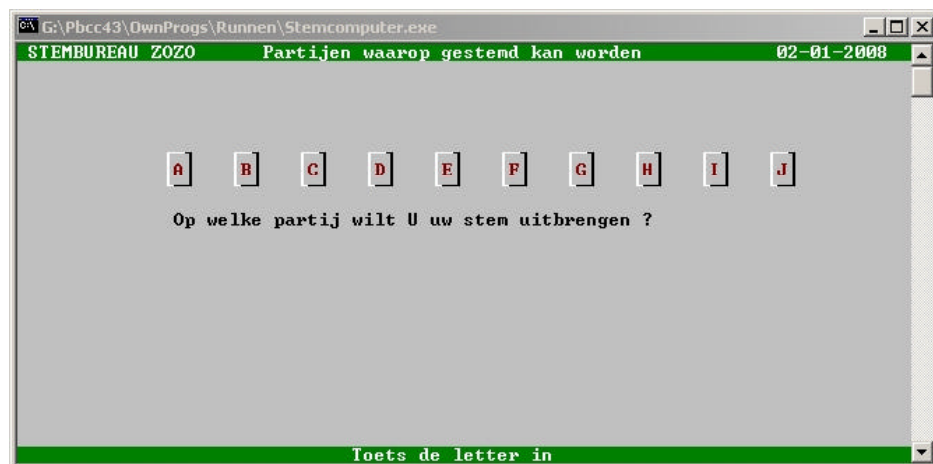


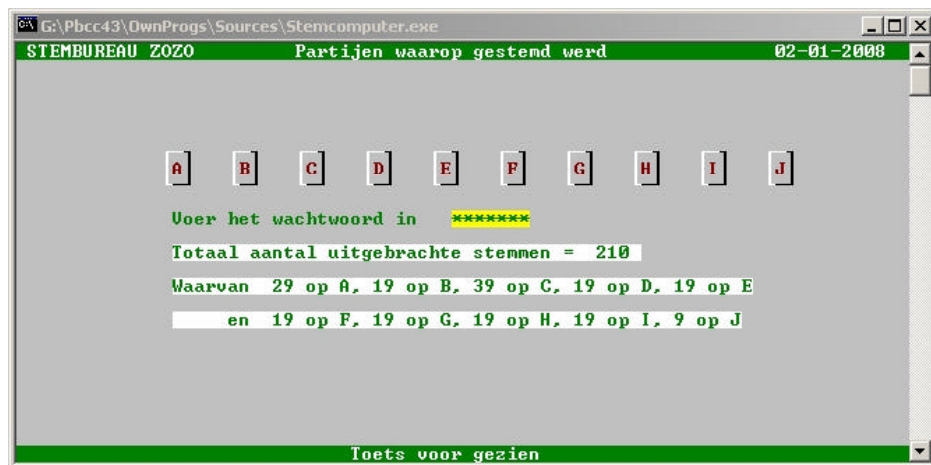
Fig. 3: Dit ziet degene die gestemd heeft



Als men de vraag met ja of nee beantwoord heeft dan komt bij "ja" eerst nog de mededeling "U heeft gestemd", daarna komt in beide gevallen het scherm van figuur 2 weer en kan de volgende persoon stemmen.

Fig. 4: Nadat De Chef van het stembureau de stemming afgesloten heeft, op het overzichtsschermb (zie fig.1) positie 2 gekozen heeft en het juiste wachtwoord ingevuld heeft, ziet hij het resultaat van de stemming. Maar hij ziet alleen de regel met het totaal aantal uitgebrachte stemmen.

Als hij ook het 2de wachtwoord kent en inbrengt dan ziet hij ook de details. Op de volgende bladzijde ziet u het scherm waar er op A 31 stemmen zijn uitgebracht, op B t/m I 21 en op J 11. Van A, B, D, E, F, G, H, I en J elk, zijn 2 stemmen naar C overgeheveld.



Bij invoer van het wachtwoord werd volgende routine gebruikt:

```
SUB Wachtwoord (MAXM&, Wachtwrđ$)
'Onzichtbaar intoetsen van een wachtwoord
REDIM Deel$(MAXM&)
VE& = 11: HO& = 40: Wachtwrđ$ = ""
LOCATE VE&,HO&: COLOR 2,14: PRINT STRING$(MAXM&,32);
DO
    HO& = 40 + LEN(Wachtwrđ$)
    IF LEN(Wachtwrđ$) > 0 THEN LOCATE VE&, HO&-1: CURSOR ON
    PRINT CHR$(42);
    LOCATE VE&, HO&: CURSOR OFF
    Deel$(I&) = INKEY$: Wachtwrđ$ = Wachtwrđ$ + Deel$(I&)
    LOOP UNTIL LEN(Wachtwrđ$) = MAXM&
    CURSOR ON: PRINT CHR$(42);: COLOR 2,15
END SUB
```

MAXM& is de lengte van het wachtwoord. Het wachtwoord dat ingegeven wordt, wordt met Wachtwrđ\$ aan het stemprogramma doorgegeven en daar op juistheid beoordeeld.

Om de "Windows-toetsen" op het scherm te krijgen werd van de volgende subroutine gebruik gemaakt:

```
SUB Knop3D(Raam&,VE&,HO&)
Count& = 1
SELECT CASE Raam&
CASE 1
    CURSOR OFF
    LOCATE VE&,HO& :COLOR 15,7
    PRINT CHR$(218) + STRING$(Count&,196);
```

```
COLOR 0,7 :PRINT CHR$(191)
LOCATE VE&+1,HO& :COLOR 15,7 :PRINT CHR$(179);
LOCATE ,HO&+2: COLOR 0,7 :PRINT CHR$(179)
LOCATE VE&+2,HO& :COLOR 15,7 :PRINT CHR$(192);
COLOR 0,7 :PRINT STRING$(Count&,196) + CHR$(217);
LOCATE VE&+1,HO&+3
CASE 2
    CURSOR OFF
    LOCATE VE&,HO& :COLOR 0,7
    PRINT CHR$(218) + STRING$(Count&,196);
    COLOR 15,7 :PRINT CHR$(191)
    LOCATE VE&+1,HO& :COLOR 0,7 :PRINT CHR$(179);
    LOCATE ,HO&+2: COLOR 15,7 :PRINT CHR$(179)
    LOCATE VE&+2,HO& :COLOR 0,7 :PRINT CHR$(192);
    COLOR 15,7 :PRINT STRING$(Count&,196) + CHR$(217);
    LOCATE VE&+1,HO&+3
END SELECT
END SUB
```

Dit zijn gewone tekstfiles die als "include"-files aan het programma zijn toegevoegd. Men kan ze ook zo overnemen in het programma als dat het commando #INCLUDE niet kent.

De overige routines staan op de kwartaaldiskette. Het programma is geschreven in Power Basic 4.3

Fred Luchsinger

BASIC dialecten - spaghetticode

Een hobbyist die denkt in een programmeertaal als BASIC te kunnen programmeren, kan inderdaad zijn programma prima voor elkaar hebben. Maar als een gevorderde programmeur de code bekijkt is het gewoon treurig als hij ziet hoe erg de code opgebouwd is. De meeste programmeurs noemen zo'n code ook wel spaghetticode. Maar waarom is dat erg? De hobbyist zou ook zeggen: "Ach, wat maakt de code nou uit!? Ik ben al blij als het programma werkt en inderdaad, het programma werkt zoals ik het wil!"

Spaghetticode heeft nadelen die vaak over het hoofd wordt gezien. Er worden programma's geschreven waarbij heel vaak gemiddeld 20% code ongebruikelijk is. Spaghetticode kan problemen veroorzaken dat geen compilerfouten zijn, zoals hieronder staat.

- springen uit of in de lussen zonder eindcontrole, later daar meer over.
- variabelen in procedures of functies gaan delen terwijl ze eigenlijk niet in an-

dere subroutines (procedures/functions) bekend mogen zijn. Gedeelde variabelen zijn openbaar en dat kan leiden tot de foutmelding "Stack overflow" als de subroutines te vaak aangeroepen worden.

- Subroutines die zichzelf aanroepen zonder eindcontrole, ook die kunnen de foutmelding "Stack overflow" veroorzaken.

Zo zijn er nog meer op te noemen.

De compiler foutmeldingen waar ik het eerst over had zijn niet dezelfde foutmeldingen als hierboven. De compiler kan namelijk geen spaghetti-code controleren. Fouten, die door zulke code worden veroorzaakt, kan pas tijdens het draaien van het programma worden ontdekt. Natuurlijk niet door de compiler, want als men het programma start dan is de Basic-code allang vertaald in binaire code.

In oudere BASIC dialecten was de kans op spaghetti-code programma's groter dan tegenwoordig. Dat komt doordat de versies gestroomlijnd werken. Alles kan worden opgebouwd, van groot naar klein gezien:

1. Projecten
 2. Klassen en modules
 3. Procedures en functies, in punt 2 noemen we ze ook: methoden en eigenschappen (eigenlijk alleen in klassen)
 4. Objectinstanties
 5. Variabelen en velden
- In de nieuwe Basic versies (.NET en 2005) kunnen we punt 4 en 5 als één punt beschouwen

Wanneer mag ik springen?

Sommige BASIC dialecten kennen nog steeds de onvoorwaardelijke sprong. Zonder reden in het programma springen kan een nadeel zijn, maar het kan ook weer zijn voordeel hebben.

Een aantal jaren geleden heb ik de bijdrage geschreven over: "GOTO statement, valstrik of hulpmiddel?"

Misschien kunt u die nog herinneren. In de bijdrage stond toen alles wat het statement voor problemen gaf, maar dat het ook weer een voordeel had. Ja, maar wat mag ik nou wel en wat niet? Bekijk eens onderstaand programma. Het is bedoeld voor alle BASIC dialecten die dit ondersteunen, dus ik heb de sleutelwoorden niet vet gemaakt.

```
DIM I AS INTEGER
```

```
FOR I = 1 TO eindwaarde
  PRINT "DE TELLER STAAT OP: " + STR$(I)
  IF I = controlewaarde THEN
    PRINT "CONTROLEWAARDE GEVONDEN!"
    GOTO labelnaam
  END IF
NEXT I
```

```
labelnaam:
REM VERDER CODE HIER
```

Onderstaand programma doet hetzelfde, maar nu zonder de sprong.

```
DIM I AS INTEGER
```

```
FOR I = 1 TO eindwaarde
  PRINT "DE TELLER STAAT NU OP: " + STR$(I)
  IF I = controlewaarde THEN
    PRINT "CONTROLEWAARDE GEVONDEN!"
    I = eindwaarde
  END IF
NEXT I
REM VERDER CODE HIER
```

Door de eindwaarde van de lus aan de lusvariabele toe te kennen, zal de lus eerder stoppen dan in de FOR regel aangegeven. U hoeft geen label te gebruiken en u hoeft niet uit de lus te springen. Hoewel er eerst een voorwaardelijke controle wordt uitgevoerd, zal toch de sprong met GOTO onvoorwaardelijk zijn. Het tweede voorbeeld beëindigt de lus beter door eerder de eindwaarde te geven.

Omgaan met variabelen

U zou het niet geloven, maar ook de variabelen verkeerd gebruiken kan een onleesbaar spaghetti-code programma veroorzaken. Net zoals de hobbyist zou zeggen: "Ja, maar mijn programma werkt goed", een rommel in de code moet het niet worden.

Wanneer veel procedures en/of functies de waarde van een variabele nodig hebben, gaat men al gauw die variabele globaal maken of delen.

Kijk eens naar onderstaand figuur. "start" is de beginmodule, de hoofdletters zijn de methoden (procedures en/of functies).

start	A	B	C	D
X = 10		X		X

Hier ziet u dat twee methoden, B en D, dezelfde variabele nodig hebben, of beter gezegd: u wilt dat B en D waarde 10 gebruiken van X. Al gauw wordt de variabele globaal gedeclareerd door deze bovenaan in de module te zetten, het declaratie-deel (start). Allemaal goed en wel, want het werkt prima, maar kijk nu eens naar onderstaand figuur.

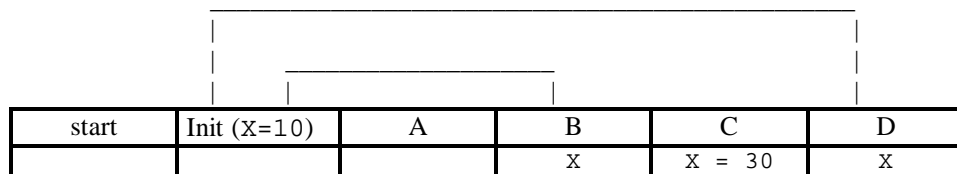
start	A	B	C	D
X = 10		X	X = 30	X

Wanneer methode C aangeroepen wordt, gebeurt er iets wat men vaak niet in de ga-

ten heeft. De waarde van X wordt in methode C veranderd, maar ook in B en D krijgt X de nieuwe waarde en dat is meestal niet de bedoeling. U zou dan overal in de methoden X de juiste waarde moeten geven wanneer u denkt dat het moet. Als u ook maar op één verkeerde regel een waarde geeft, kan het slecht aflopen, bijvoorbeeld voor:

```
??   de schermlocatie:LOCATE X, Y 'voor C128 BASIC 7.0
      LOCATE Y, X 'voor PowerBASIC en GW-BASIC
??   plaatjes zetten
```

U zou met bovenstaand probleem dus de oplossing vinden door overal variabele X te besturen, terwijl hij overal bekend is. Het is niet de juiste oplossing om van de spaghetti-code af te komen. Het is nog beter om onderstaand figuur proberen te begrijpen en het in de praktijk te gebruiken.



Door variabele X te initialiseren in een methode i.p.v het declaratiedeel, zal X niet meer overal gebruikt kunnen worden. Dat is een nadeel, zou u zeggen, maar het voordeel is nu dat de module veiliger wordt. Door B en D in methode Init aan te roepen en variabele X mee te geven, zal variabele X in C niet dezelfde zijn. Als X daar niet gedeclareerd is, zal dat een compilerfout veroorzaken. Wat voor waarde u ook geeft in C, in B en D zal X gewoon 10 blijven.

De figuren geven als voorbeeld dat de variabele de waarde 10 krijgt, maar hetzelfde geldt voor het inlezen van een gegevensstructuur (tekstbestand of een recordset). U mag aan de methoden ook deze gegevensstructuren meegeven. Lees geen tekstbestand of recordset in in het declaratiedeel (start).

Tip! U ziet dat u recordsets in de methoden inleest of bewerkt, dat is beter en veiliger. Doe dat echter niet met de gegevensstructuur die de hele database draagt. Van daaruit is het de bedoeling om de recordsets in te lezen en te bewerken. De database is daarom het enige object dat openbaar in de Main module van het programma gedeclareerd moet worden. Het heeft geen zin om de hele database aan alle methoden mee te geven. De database moet gewoon in het hele programma bekend zijn. Wanneer een recordset nodig is gebruikt u de methode van het openbare database object. Sommige BASIC dialecten ondersteunen dit wel, anders kunt u gebruik maken van tekstbestanden en de statements OPEN en CLOSE.

Samenvatting

Nog steeds wil ik zeggen, moeten is er niet bij. Bij elke programmeur ziet de code er weer anders uit. Het gaat er alleen om dat alles goed in elkaar zit en later nog steeds het programma goed gelezen en begrepen kan worden, zelfs als een andere programmeur het programma zou bekijken.

Ieder heeft zijn eigen techniek. Andere soort variabelen, wel of geen klassen gebruiken, zo veel mogelijk argumenten (dat zijn de waarden) meegeven aan de methoden of juist zeer weinig. Niemand hoeft aan u te vertellen hoe het programma in details eruit moet zien, ik ook niet. Het gaat er alleen om dat het programma goed kan draaien en niet teveel geheugen zal gebruiken, zodat de kans op de foutmelding "Stack overflow" nihil is.

Marco Kurvers

Oplossing: verf mengen

De oplossing is zeer eenvoudig, vooral als je alle niet ter zake doende elementen uit de opgave haalt en je richt op wat er nu eigenlijk gevraagd wordt, namelijk wat is de verhouding tussen de beide verfbussen na het heen en weer vullen.

Die verhouding is gelijk.

Het wel of niet doorroeren is helemaal niet belangrijk. Het percentage werd niet gevraagd. Als beide bussen voor en na het mengen even vol zitten betekend dat automatisch dat de overgebrachte hoeveelheid verf in beide bussen ook gelijk moet zijn. Goed de vraag lezen was dus al bijna voldoende om het antwoord zo te geven. Een computerprogramma is dus feitelijk overbodig. Hiermee heb ik willen aantonen dat het van belang is altijd naar de essentie van een vraagstelling te komen en de niet ter zake doende verpakking opzij te leggen.

Tot slot voor de ongelovige onder u; test het zelf even uit. Niet met verf maar bijvoorbeeld met witte en zwarte damstenen of gekleurde fiches van twee kleuren. Neem twee bakjes met gelijke hoeveelheden van elk een kleur. Neem uit een bakje een bepaald aantal stuks, bijvoorbeeld 5, en doe die in het andere bakje. Haal hetzelfde aantal er weer uit zonder te kijken welke kleur het is en doe die weer in het eerste bakje. En zie de verhouding is altijd gelijk, hoe vaak u het ook probeert. Op deze puzzel heb ik 3 antwoorden ontvangen die allemaal de tot gelijke verhouding kwamen, waarvan 2 voorzien van een eenvoudig Basic programma.

Piet Boere

Visual Basic 2005 – aantal veranderingen

De TextBox

Meestal wordt de TextBox gebruikt om eenregelige tekst in te voeren. Voor alinea's kan de RichTextBox worden gebruikt, maar u kunt ook de TextBox instellen op meerdere regels.

De TextBox heeft een eigenschap `Lines()`. Met die eigenschap kunt u per regel bewerken. Om een regel te kunnen lezen kunt u de index opgeven alsof de eigenschap een normale array is. Dat is echter niet het geval. De eigenschap is een alleen-lezen eigenschap, maar om een regel te kunnen bewerken kunt u helaas geen index opgeven. Onderstaande regel werkt goed.

```
strRegel = txtRegels.Lines(0)
```

Maar deze onderstaande regel niet.

```
txtRegels.Lines(0) = strRegel
```

Om het probleem op te lossen moet er een omweg worden gemaakt door meteen alle regels aan een array toe te kennen, een of meerdere regels van de array te bewerken en die als een hele array, dus niet de bewerkte elementen, terug te geven. Zie onderstaand voorbeeld.

```
Dim strArray() As String = txtRegels.Lines
strArray.SetValue(strRegel, 0)
txtRegels.Lines = strArray
```

De tweede regel mag trouwens ook zonder de methode `SetValue()`, door variabele `strRegel` direct aan het array-element toe te kennen:

```
strArray(0) = strRegel
```

Waar is de Caption eigenschap?

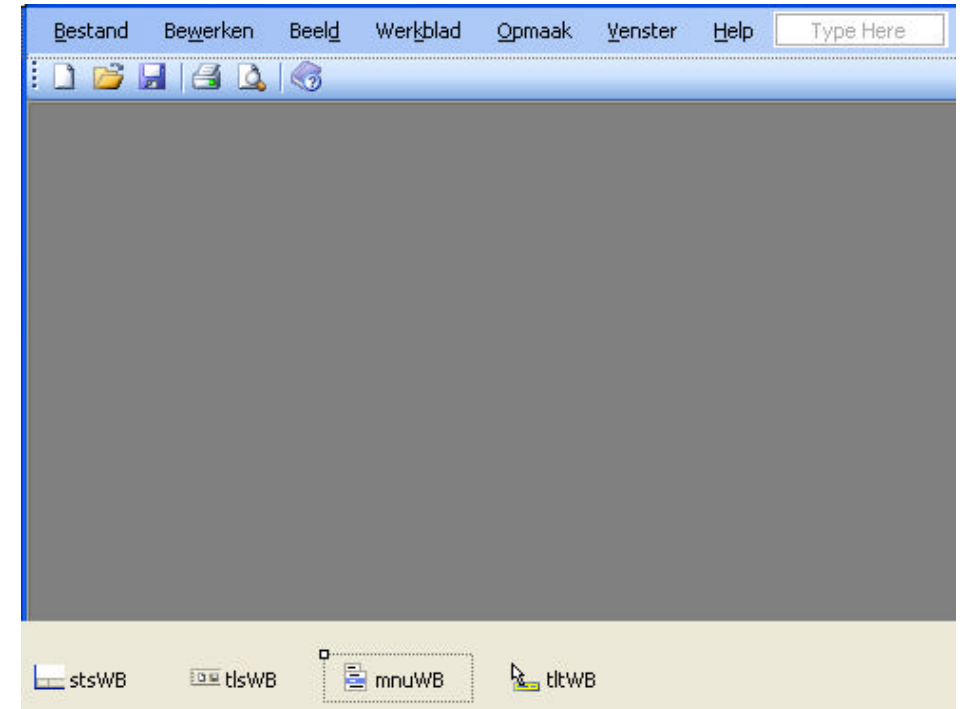
In VB 2005 en VB.NET wordt bij verschillende componenten niet meer de eigenschap `Caption()` ondersteund. Die componenten hebben nu ook de eigenschap `Text()`. Een voorbeeld: het besturingselement `Label` die in versie 6 nog de eigenschap `Caption()` heeft. Waarom wordt die niet meer ondersteund?

De hoofdzaak is dat alle componenten nu een vader hebben. Ze erven nu de eigenschappen en methoden van het hoofdcomponent `TControl`. In oudere Basic versies had elk besturingselement zijn eigen methoden en eigenschappen. Nu is dat dus niet meer het geval. Het voordeel is, dat het maken van eigen componenten, gemakkelijker gaat omdat u vanuit `TControl` de bestaande methoden en eigenschappen kunt overerven en in uw eigen component ook nieuwe methoden en eigenschappen toe

kunt voegen.

Waar is het menu editor?

In eerdere versies was er een ingebouwd menu editor die geopend kon worden via het menu `Tools`. Het nadeel was dat tijdens het bewerken van het menu de IDE niet toegankelijk was. Nu kan het wel. U kunt zowel het menu als het formulier of de code bewerken.

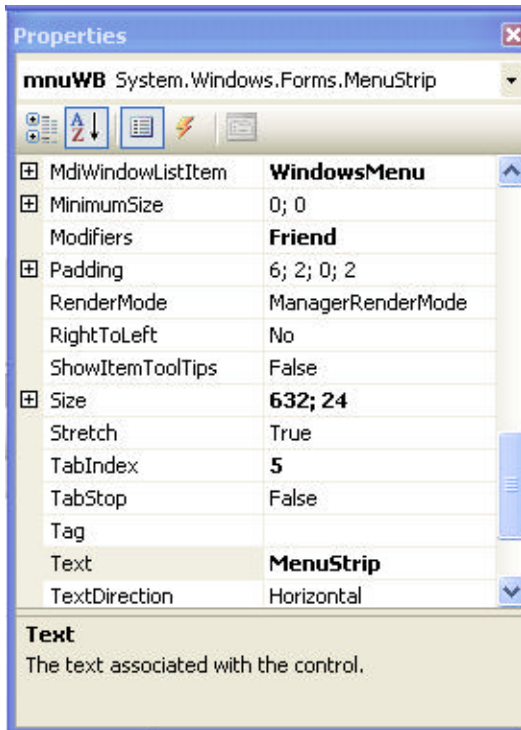


Het enige wat u moet doen is uit de toolbox de component `MenuStrip` halen en naar het formulier slepen. U kunt ook een kant en klare formulier kiezen met alles erop en eraan, zoals bovenstaand voorbeeld, door naar 'Add form' te gaan en daar een formulier te kiezen met menu. U krijgt dan wel alle menu-items in het Engels.

De andere componenten betekenen het volgende:

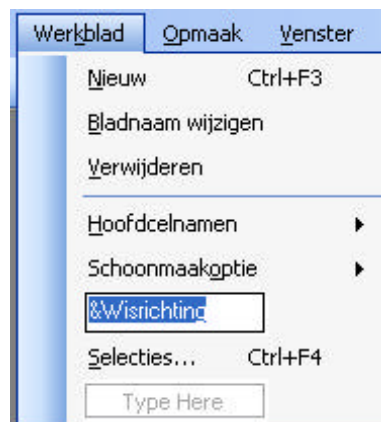
- `stsWB`:
Deze component zorgt voor een `StatusBar`.
- `tlsWB`:
Deze component zorgt voor een `ToolStrip`, anders genoemd: `Werkbalk`.
- `mnuWB`:
Deze component zorgt voor een `MenuStrip`, het menu editor is daardoor niet meer aanwezig.

- tltWB:
Deze component zorgt voor tooltips. Meestal hebben de componenten zelf een ToolTip eigenschap.



De MenuStrip heeft net als de andere componenten ook eigenschappen. U kunt zien dat de eigenschappenlijst ook totaal is veranderd en dat de componenten allemaal afstammen van System. Hoewel de menustrip een component is stamt het niet af van TControl. De MenuStrip hoort nog steeds bij de collection Forms. In eerdere versies was dat nooit te zien omdat het verborgen was in zogenaamde zwarte dozen, zoals men dat noemt in vliegtuigen. We konden nooit aan de eigenschappen van de menu-items komen. Dankzij het .NET Framework, dat is de hiërarchie van alle componenten waar System de basis van is, werkt alles veel gemakkelijker.

Zie eens onderstaand figuur als we een menu openen. Door te klikken op een menu-item kunt u het direct bewerken.



Zoals u de bewerkingsmethode hiernaast ziet, zo zal u natuurlijk denken: "Hadden ze dat niet eerder kunnen bedenken?"

Inderdaad, het is nu zeer eenvoudig geworden. Volg de tekst "Type Here" waar u een menu-item toe kunt voegen. Echter: invoegen kan niet, maar u kunt wel elk menu-item slepen naar een andere plek waar u maar wilt.

Ook de separator doet u door het min teken in te toetsen.

Wilt u een sneltoets erbij hebben, volg dan de eigenschappenlijst van het menu-item. De eigenschappenlijst is dezelfde lijst als bij voorgaande figuur, met eigenschappen speciaal voor het menu-item.

In de volgende nieuwsbrief leg ik de code uit.

Marco Kurvers

Inhoud kwartaaldiskette

2008 – 1

De kwartaaldiskette bestaat uit:

Bestandsnaam	Type
Onderdelen en bouwstenen	RTF
Wat doe jij met programmeren	RTF
Wat doet een programmeur	RTF
Software ontwikkelmethode	RTF
Puzzel Gouden blokken	RTF
Visual Basic en de menu editor	DOC
Programma voor de valse stemcomputer	BAS
BASIC dialecten spaghetticode	DOC
Oplossing verf mengen	RTF
Visual Basic 2005 aantal veranderingen	DOC

Indien niet alle bestanden op één diskette passen, kan het uit meerdere diskettes bestaan.

Zie aanwijzing op bladzijde 31 voor het bestellen van bovenstaande diskette



Basic Cursussen

De cursussen worden uitsluitend geleverd met cursusboek
Alle prijzen zijn inclusief verzendkosten voor Nederland en België:

Cursussen

Qbasic: Cursusboek en lesmateriaal op diskette, € 11,00 voor leden. Niet leden € 13,50

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 cursusboeken en/of diskettes of CD-ROM kunnen worden besteld door storting van het verschuldigde bedrag op :

Postbanknummer 7227036

HCC BASIC GG

Montfoort

Onder vermelding van de juiste cursus. Wanneer u elektronisch bankiert vermeld dan ook uw adres bij opmerkingen. Uw adres wordt namelijk niet automatisch meegezonden. Houdt u rekening met een levertijd van ongeveer twee weken.

Zoals reeds eerder vermeld kunt u vanaf uitgave 1 van 1998, de teksten en broncodes van de Nieuwsbrieven bestellen op diskette. Ze zijn ook te downloaden van onze website. De diskettes worden bij tijd en wijlen aangevuld met bruikbare hulpprogramma's, informatieve artikelen en voorbeeldprogramma's. Aanvullingen zijn altijd in subdirectories geplaatst.

Op de catalogusdiskette staat een korte, maar duidelijke beschrijving van elk programma. via penningmeester.



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
Assembler	Hans Lunsing	ma. t/m vr.	20-22	(071) 5232703	j.lunsing@hccnet.nl
Basic onder Linux	Hans Lunsing	ma. t/m vr.	20-22	(071) 5232703	j.lunsing@hccnet.nl
Basic op Acorn en Risc OS	Pieter Drost	ma. t/m vr.	18-22	(06) 41525307	pieter_drost@aconet.org
Commodore	Hans Kessels				j.w.m.kessels@kader.hcc.nl
GW-Basic	Hans Lunsing	ma. t/m vr.	20-22	(071) 5232703	j.lunsing@hccnet.nl
Hardware-aansturing	G. van der Sel	Schriftelijk via de secr. of e-mail			seccr@basic-gg.hcc.nl
Liberty Basic	Gordon Rahman	De hele week	19-23	(023) 5334881	grahman@planet.nl
MSX-Basic	Erwin Nicolai	vr. t/m zo.	18-22	(0516) 541680	

Waarover	Wie	Wanneer	Tijd	Telefoon	Email
Power Basic	Hans Lunsing	ma. t/m vr.	20-22	(071) 5232703	j.lunsing@hccnet.nl
Pure Basic	Frans Steur	ma. en do.	20-22	(073) 6447194	x.steur@hccnet.nl
QBasic	Hans Lunsing	ma. t/m vr.	20-22	(071) 5232703	j.lunsing@hccnet.nl
	Bart Vreken	woensdag	19-21	(020) 6122096	
QuickBasic	Hans Lunsing	ma. t/m vr.	20-22	(071) 5232703	j.lunsing@hccnet.nl
	Bart Vreken	woensdag	19-21	(020) 6122096	
	Martin Michels	donderdag	19-21	(036) 5319800	
VB voor DOS	Martin Michels	donderdag	19-21	(036) 5319800	
VB voor Windows	Jeroen v. Hezik	ma. t/m zo.	19-21	(0346) 214131	j.a.van.hezik@kader.hcc.nl
	Martin Michels	donderdag	19-21	(036) 5319800	
Basic algemeen, zoals VBA Office	Marco Kurvers	ma. t/m vr. za. t/m zo.	19-23 13-17	(0342) 424452	m.a.kurvers@hccnet.nl m.a.kurvers@tronicasoftware.nl