Visual Basic® Handboek

Microsoft® Excel

Versie 5

Automatiseren, aanpassen en programmeren in Microsoft Excel met het programmeersysteem Microsoft Visual Basic for Applications.

Microsoft Corporation

Art.nr. 43758
Printed in Ireland 10

Inhoudsopgave

Produktondersteuning

Als u een vraag hebt over een Microsoft®-produkt, raadpleeg dan eerst:

- de gedrukte produktdocumentatie.
- de online Help.
- de LEESMIJ-bestanden. Deze bestanden worden tijdens de installatie van het programma naar de vaste schijf gekopieerd. In deze bestanden vindt u informatie die pas na het drukken van de produktdocumentatie bekend is geworden.
- elektronische informatiebronnen zoals CompuServe of bulletin boards, indien beschikbaar.

Als u er niet in slaagt een antwoord te vinden op uw vraag, kunt u contact opnemen met de dichtstbijzijnde Microsoft-vestiging.

Het 'Microsoft Support Network'

Het 'Microsoft Support Network' is een wereldwijd support-programma dat in de Benelux in het eerste kwartaal van 1994 geïntroduceerd zal worden. Tot die datum zullen de huidige betaalde support-programma's, bekend onder de naam 'Online Priority Support Programs', beschikbaar blijven.

Daarnaast zijn er jaar-abonnementen op CD-ROM verkrijgbaar die u in staat stellen om snel een oplossing voor een technisch probleem te vinden:

- Microsoft Technet: uitgebreide database met bekende problemen en oplossingen voor Microsoft-toepassingen en -besturingssystemen.
- Microsoft Developer Network: uitgebreide informatie voor (Microsoft Windows) ontwikkelaars.

Informatie hierover kunt u aanvragen bij Customer Service, te bereiken onder de hieronder genoemde telefoonnummers.

Contact opnemen met uw Microsoft-vestiging

U kunt de Microsoft-vestigingen in de Benelux bereiken van maandag tot en met vrijdag, van 9.00 tot 12.00 uur en van 13.00 tot 17.00 uur (op vrijdag tot 16.00 uur), behalve op feestdagen.

Wanneer u Microsoft opbelt, dient u bij uw computer te zitten en de documentatie bij de hand te hebben. Zorg ervoor dat u antwoord kunt geven op de volgende vragen:

- Wat is het versienummer van het Microsoft-produkt waarmee u werkt?
- Welke apparatuur gebruikt u (inclusief, indien van toepassing, netwerkapparatuur)?
- Welk besturingssysteem gebruikt u?
- Zijn er berichten op het scherm verschenen en zo ja, wat is daarvan de exacte formulering?
- Wat gebeurde er en waarmee was u bezig toen het probleem zich voordeed?
- Hoe hebt u geprobeerd het probleem op te lossen?

De adressen van de Microsoft-vestigingen in Nederland, België en Luxemburg luiden als volgt:

Nederland	Microsoft BV	
	Telefoon:	02503-89189
	Customer Service:	02503-77700
	CompuServe:	020-6880085 (GO MSBEN)
	Bulletin Board Service:	02503-34221 (1200/2400/9600 baud, 8N1, ANSI)
	Technische ondersteuning:	02503-77877
België	Microsoft NV	
	Telefoon:	02-7303911
	Customer Service:	02-7303922
	CompuServe:	02-2150530 (GO MSBEN)
	Bulletin Board Service:	02-7350045 (1200/2400/9600 baud, 8N1, ANSI)
	Technische ondersteuning:	02-5133274
Luxemburg	Microsoft NV	
	Telefoon:	(32) 2-7303911
	Customer Service:	(32) 2-7303922
	CompuServe:	(32) 2-2150530 (GO MSBEN)
	Bulletin Board Service:	(32) 2-7350045 (1200/2400/9600 baud, 8N1, ANSI)
	Technische ondersteuning:	(32) 2-5133274

Gebruikte conventies

In dit handboek worden de typografische conventies gehanteerd die in de volgende tabel staan beschreven. Waarschijnlijk kent u de gebruikte termen of sleutelwoorden uit Visual Basic® nog niet. Deze worden echter verderop in dit handboek nader uitgelegd.

Voorbeeld van de conventie	Beschrijving
setup	Woorden of tekens die u moet typen worden vet weergegeven.
object	In de lopende tekst worden belangrijke nieuwe termen cursief weergegeven. Meestal gebeurt dit als ze voor het eerst worden gebruikt in het boek.
naamvaneigenschap	In de codesyntaxis worden de variabelen die zich op de plaats bevinden van informatie die u moet invoeren, cursief weergegeven.
EXCEL.EXE (Microsoft Windows™) MICROSOFT EXCEL (Apple® Macintosh®)	Bestandsnamen en de namen van directory's en folders worden in hoofdletters weergegeven.
ENTER	Namen van toetsen en toetscombinaties, zoals ENTER en CTRL+R, worden in kleine hoofdletters weergegeven.
CTRL+V	Een plusteken (+) tussen de toetsnamen geeft aan dat het om een toetscombinatie gaat. CTRL+V betekent bijvoorbeeld dat u CTRL ingedrukt moet houden terwijl u op V drukt.
PIJL-OMLAAG	Afzonderlijk pijltoetsen worden weergegeven door de richting waarin de pijl wijst met het voorvoegsel PIJL (PIJL-LINKS, -RECHTS, -OMHOOG of -OMLAAG). Het woord "pijltoetsen" wordt gebruikt voor al deze toetsen samen.
BACKSPACE, HOME	De overige stuurtoetsen worden aangeduid met de bijbehorende namen.
[*expressielijst*]	In de syntaxis worden de optionele onderdelen tussen vierkante haken weergegeven.
{**Terwijl** | **Totdat**}	In de syntaxis, wordt een verticale streep gebruikt om een keuzemogelijkheid tussen twee of meer onderdelen aan te geven. U kunt een van deze mogelijkheden kiezen. Soms worden accolades gebruikt om onderdelen te groeperen die worden gescheiden door een verticale streep.

Voorbeeld van de conventie	Beschrijving
`Syntaxisfout`	Dit lettertype wordt gebruikt voor door de gebruiker gedefinieerde variabelen en foutberichten.

`Sub AandelenVerkoop ()`
`.`
`.`
`.`
	Een kolom of een rij met drie puntjes geeft aan dat een deel van het voorbeeldprogramma met opzet is weggelaten.

Sub, **Indien**, **AndereDir**, **BerichtVenster**, **Waar**	Woorden die vet worden weergegeven met een beginhoofdletter geven aan dat het gaat om een sleutelwoord van Visual Basic, ofte wel een taalspecifieke term.
Toevoegen, **Hoogte**, Toepassing, Bereik, Rij	Namen van eigenschappen, methoden en objecten hebben ook een beginhoofdletter. **Hoogte** wordt ook vet weergegeven omdat dit een eigenschap is en **Toevoegen** ook, omdat het een methode is.

In dit handboek worden verder de algemene conventies gebruikt, alsmede de muis- en toetsenbordconventies, die ook in het *Microsoft Excel handboek* worden gehanteerd.

Programma-opmaak in dit handboek

De volgende richtlijnen worden gehanteerd in de programmeercode in dit handboek. Zie hoofdstuk 6, "Werken met Visual Basic-programmacode in procedures", voor meer informatie over Visual Basic-programmacode.

- Dit lettertype wordt voor programmacode gebruikt

```
Sub GoeiemorgenWereld
Cellen(1,1).Waarde = "Goeiemorgen, wereld!"
Einde Sub
```

- Een enkel aanhalingsteken (') geeft het begin van een commentaarregel aan.

```
' Dit is commentaar. Deze twee regels worden
' genegeerd als het programma wordt uitgevoerd.
```

- Namen van macro's, door de gebruiker gedefinieerde functies, argumenten en variabelen worden overal in deze handleiding weergegeven met een beginhoofdletter. Houd er rekening mee dat namen van macro's en functies geen spaties mogen bevatten. Als een naam uit meer dan één woord bestaat, krijgen de andere woorden in die naam ook een beginhoofdletter.

```
' De door de gebruiker gedefinieerde functie FinContr staat in de
' module Financieel.
Functie FinContr (LaatsteInkomsten, Uitgaven, Belastingen, Provisie)
```

Veel macro-ontwikkelaars geven er de voorkeur aan om een onderstrepingsteken (_) te gebruiken om de woorden in de namen van macro's of **Functie**-procedures te scheiden. Bovendien hebben sommige ontwikkelaars een voorkeur voor variabelen en argumenten die met een kleine letter beginnen om verwarring tussen variabelen en eigenschappen te vermijden.

```
' De macro Aandelen_Inkomsten staat in de module Financieel.
Sub Aandelen_Inkomsten (verkochteAandelen, prijsPerAandeel)

geluidOpnemenMogelijk = Toepassing.KanGeluidOpnemen
```

- Sleutelwoorden zijn te herkennen aan een beginhoofdletter. Ingebouwde constanten beginnen met "xl" in kleine letters.

```
' Sub is een sleutelwoord.
Sub Titel (TitelTekst)

' xlHandmatig is een ingebouwde constante.
Toepassing.Berekening = xlHandmatig
```

- Controlestructuren en instructies in procedures die met **Sub** of **Functie** beginnen, springen in ten opzichte van de omringende programmacode.

```
Sub ControleerOpnemenGeluid
    GeluidOpnemenMogelijk = Toepassing.KanGeluidOpnemen
    Indien GeluidOpnemenMogelijk Dan
        Cellen(1,1).GeluidNotitie.Opnemen
    Einde Indien
Einde Sub
```

- Het teken waarmee wordt aangegeven dat de regel doorloopt, een onderstrepingsteken (_), geeft aan dat de programmacode die op de volgende regel wordt voortgezet, in de Visual Basic-module deel uitmaakt van dezelfde logische regel. U kunt deze instructies allemaal op één regel in een Visual Basic-module typen. U kunt ook de logische regel in een aantal regels opdelen en zelf het onderstrepingsteken toevoegen.

```
ActiefBlad.Rechthoeken.Toevoegen _
    Breedte:=200; _
    Hoogte:=200; _
    Links:=50; _
    Boven:=50
```

Opmerking Als u programmacode schrijft of bewerkt kunt u in de keuzelijst "Land/Taal" van het tabblad Module (menu **Extra**, dialoogvenster **Opties**) de landinstellingen kiezen die worden gebruikt voor het decimaalteken, het lijstscheidingsteken, het valutasymbool en de datumnotatie. Het decimaalteken, het lijstscheidingsteken, het valutasymbool en de datumnotatie die gebruikers te zien krijgen wanneer ze uw programmacode uitvoeren, is afhankelijk van de landinstellingen van hun besturingssysteem.

In de programmacode-voorbeelden in dit boek wordt ervan uitgegaan dat u gebruik maakt van de Nederlandse instellingen. Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over het schrijven van toepassingen die in andere landen gebruikt moeten kunnen worden.

Een opmerking over terminologie

In de "Inleiding" en in hoofdstuk 1 tot en met 3 van dit handboek worden de woorden *macro* en *door de gebruiker gedefinieerde functie* gebruikt voor de twee belangrijkste hulpmiddelen in Microsoft Excel om het uitvoeren van handelingen te automatiseren en het programma verder aan te passen. Ofschoon ervaren gebruikers en andere werkblad-programmeurs wellicht bekend zijn met deze begrippen, zijn het geen typische Visual Basic-termen.

Hierom worden in hoofdstuk 4, "Inleiding in Visual Basic-procedures", de termen macro en door de gebruiker gedefinieerde functie vervangen door een standaard Visual Basic-term: *procedure*. Deze term is nagenoeg synoniem aan de voorgaande begrippen: macro's en door de gebruiker gedefinieerde functies zijn twee voorbeelden van procedures. Alle termen worden volledig uitgelegd in hoofdstuk 4.

Inleiding

Met Visual Basic beschikt u in Microsoft Excel over een krachtige en eenvoudig te gebruiken programmeertaal. De naam Visual Basic wordt hier gebruikt voor de implementatie van het Programmeersysteem Microsoft Visual Basic for Applications in Microsoft Excel. Met Visual Basic kunt u vaak terugkerende handelingen automatiseren, aangepaste voorzieningen en functies in het programma aanbrengen en zelfs complete toepassingen ontwikkelen.

In Microsoft Excel kunt u vaak terugkerende handelingen automatiseren met macro's. Een *macro* is een serie instructies waarmee u Microsoft Excel duidelijk maakt welke handelingen moeten worden uitgevoerd. Deze instructies worden geschreven in Visual Basic. Hoewel Visual Basic een programmeertaal is, hoeft u geen programmeur te zijn om macro's te schrijven. In feite hoeft u zelfs geen Visual Basic te kennen om met macro's te werken.

Microsoft Excel beschikt namelijk over een macrorecorder waarmee u macro's kunt opnemen. Met de macrorecorder kunt u de handelingen die u uitvoert en de opdrachten die u kiest, opslaan in een macro. Later kunt u deze macro afspelen, zodat de opgeslagen handelingen automatisch worden uitgevoerd. Hierdoor bespaart u tijd en moeite. Zie hoofdstuk 1, "Terugkerende taken automatiseren", voor meer informatie over de macrorecorder.

Als u hebt geleerd hoe u macro's opneemt en afspeelt, zult u ontdekken hoe handig ze zijn. Waarschijnlijk wilt u dan de mogelijkheden van uw macro's uitbreiden door er uw eigen Visual Basic-programmacode aan toe te voegen. In dit handboek wordt uitgelegd hoe u dit kunt doen. Visual Basic is eenvoudig onder de knie te krijgen. Of u nu een nieuwe gebruiker bent of al veel ervaring hebt met het ontwikkelen van macro's, u zult zien dat u met Visual Basic uw produktiviteit aanzienlijk kunt vergroten.

Met Visual Basic kunt u aangepaste opdrachten, menu's, berichten en knoppen maken. Bovendien kunt u voor al deze onderwerpen zelf schermhulp maken. Op deze manier kunt u Microsoft Excel ombouwen tot een geheel andere toepassing. Of u nu dagelijks terugkerende handelingen wilt automatiseren of krachtige toepassingen met veel toeters en bellen gaat ontwikkelen, met de hulpmiddelen van Visual Basic kunt u Microsoft Excel volledig aan uw behoeften aanpassen.

Aan de slag

In de rest van dit hoofdstuk vindt u een beschrijving van de gedrukte en de online documentatie die u kunt gebruiken om meer van Visual Basic te weten te komen. In dit handboek wordt ervan uitgegaan dat u al bekend bent met Microsoft Excel. Als u nog nooit met Microsoft Excel hebt gewerkt, kunt u dit handboek voorlopig beter aan de kant leggen. Zie het *Microsoft Excel handboek* voor meer algemene informatie over Microsoft Excel.

Waar kunt u informatie over Visual Basic vinden

In het Microsoft Excel-programmapakket vindt u de volgende vier informatiebronnen over Visual Basic.

- Het *Microsoft Excel Visual Basic Handboek* (dit boek). Hierin vindt u meer informatie over het opnemen, starten en bewerken van macro's, over het maken en toepassen van door de gebruiker gedefinieerde functies en over het programmeren in Visual Basic. Aan de hand van deze informatie kunt u uw eigen macro's schrijven of zelfs complete toepassingen maken.
- In het Help-bestand van Microsoft Excel vindt u meer algemene informatie over macro's (wat macro's zijn en hoe u ze kunt opnemen en gebruiken).
- Zelfstudie. Dit zijn lessen binnen de schermhulp van Microsoft Excel waarin de basisvaardigheden op het gebied van macro's worden behandeld.

  Zie hoofdstuk 5, "Informatie oproepen terwijl u bezig bent", in het *Microsoft Excel handboek* voor meer informatie over het gebruik van de schermhulp.
- De *Visual Basic Naslaggids*. Deze schermhulp bevat gedetailleerde informatie over functies, instructies, methoden, eigenschappen en objecten in Visual Basic.

Hoe is dit boek geordend?

De hoofdstukken en bijlagen van het *Microsoft Excel Visual Basic Handboek* zijn zo geordend, dat de moeilijkheidsgraad van de behandelde onderwerpen steeds toeneemt. Een aantal gebruikers zal waarschijnlijk alleen macro's willen opnemen en starten en aangepaste functies willen maken, maar verder geen kennis van Visual Basic nodig hebben. Andere gebruikers willen wellicht een aantal kenmerken van Visual Basic onder de knie krijgen, maar zijn niet van plan om volledige toepassingen te maken. Ten slotte zal een derde groep gebruikers alle mogelijkheden van Visual Basic volledig willen benutten. De hoofdstukken zijn zo gerangschikt, dat alle gebruikers de voor hen interessante informatie snel kunnen vinden.

- In hoofdstuk 1 tot en met 3 wordt uitgelegd wat macro's zijn en hoe deze worden opgenomen, uitgevoerd, weergegeven en bewerkt. Verder vindt u hier informatie over het maken en toepassen van door de gebruiker gedefinieerde functies. Deze hoofdstukken zijn bedoeld voor alle gebruikers die terugkerende werkzaamheden willen automatiseren, maar vooral voor gebruikers die nog weinig ervaring hebben met het ontwikkelen van macro's.
- In hoofdstuk 4 vindt u een inleiding in Visual Basic. De term macro wordt in dit hoofdstuk vervangen door een Visual Basic-term, namelijk *procedure*. Verder vindt u in dit hoofdstuk informatie over het werken met meerdere procedures, waarmee complexere taken kunnen worden uitgevoerd.
- In hoofdstuk 5 tot en met 10 vindt u een gedetailleerdere beschrijving van de Visual Basic-taal. In deze hoofdstukken wordt uitgelegd hoe programmacode wordt gebruikt in procedures, hoe argumenten worden doorgegeven aan en waarden worden verkregen uit procedures. Verder wordt beschreven hoe u de manier waarop de programmacode wordt uitgevoerd, kunt besturen, hoe u werkt met objecten en hun eigenschappen, hoe u fouten in de programmacode opspoort en hoe u Microsoft Excel kunt laten samenwerken met andere toepassingen. Deze hoofdstukken zijn met name bedoeld voor gebruikers die meer willen dan alleen eenvoudige taken automatiseren.
- In hoofdstuk 11 tot en met 13 wordt beschreven hoe u Microsoft Excel aan uw behoeften kunt aanpassen en hoe u zelf toepassingen kunt ontwikkelen. Hier wordt met name ingegaan op het zelf maken van dialoogvensters, opdrachten, menu's, knoppen en invoegmacro's. Deze hoofdstukken zijn speciaal geschreven voor de ervaren macro-ontwikkelaar, maar ook gebruikers die alleen menu's en werkbalken willen aanpassen, kunnen hier bruikbare informatie vinden.
- In de bijlagen A tot en met D wordt uitgelegd hoe u internationale toepassingen schrijft. Deze bijlagen bevatten verder informatie voor ervaren gebruikers van Microsoft Excel die overschakelen op Visual Basic. Tenslotte wordt uitgelegd hoe u aangepaste Help-bestanden kunt maken voor Visual Basic-toepassingen en treft u in deze bijlagen een lijst aan met alle werkbalkknoppen in Microsoft Excel versie 5.0.

Voorbeelden van programmacode

Het Microsoft Excel-programmapakket bevat voorbeelden van programmacode die u kunt openen en starten. Deze codevoorbeelden zijn niet alleen zeer geschikt als leermiddel, ze komen ook zeer goed van pas bij het ontwikkelen van uw eigen toepassingen. U kunt willekeurige onderdelen uit deze voorbeelden kopiëren en in uw eigen toepassingen plakken (indien nodig kunt u ze ook nog wijzigen). Deze voorbeelden van programmacode bevinden zich in de map of de directory VOORBLDN of in de map of de directory waarin u Microsoft Excel hebt geïnstalleerd.

De schermhulp gebruiken

Microsoft Excel beschikt over een uitgebreide schermhulp voor de Visual Basic-taal, voor de objecten die worden ondersteund door Microsoft Excel en voor de eigenschappen en methoden van die objecten.

U kunt de Visual Basic Naslaggids op een van de volgende vier manieren openen:

- Selecteer in een Visual Basic-module een eigenschap, een instructie of een ander sleutelwoord. Druk vervolgens op F1 in Microsoft Excel voor Windows of op COMMAND+SHIFT+VRAAGTEKEN in Microsoft Excel voor de Macintosh. Er verschijnt een venster met context-afhankelijke hulp.
- Kies in Microsoft Excel voor Windows de opdracht **Inhoudsopgave** in het menu **Help** (het menu **?**). Kies in Microsoft Excel voor de Macintosh de opdracht **Microsoft Excel Help** in het menu **Help** (het menu **?**) of in het menu **Venster**. Kies vervolgens het onderwerp "Programmeren met Microsoft Excel".
- Kies de knop Zoeken in het venster **Help** en zoek naar een onderwerp of een Visual Basic-term.
- Kies de opdracht **Objectenoverzicht** in het menu **Beeld** en kies vervolgens de knop Help bij objecten (deze wordt in de vorm van een vraagteken weergegeven) voor meer informatie over een object, een methode, een eigenschap of een functie.

De Visual Basic Naslaggids installeren

Wanneer u Microsoft Excel installeert, wordt de Visual Basic Naslaggids automatisch geïnstalleerd, tenzij u expliciet opgeeft dat dit niet hoeft. (De naam van dit bestand is VBA_XL.HLP in Microsoft Windows en VBA EXCEL HELP op de Apple Macintosh). Als u niet van plan bent gebruik te maken van Visual Basic, hoeft u ook de Visual Basic Naslaggids niet te installeren en kunt u de rest van dit gedeelte overslaan en met hoofdstuk 1 beginnen. In het standaard Help-bestand van Microsoft Excel (MAINXL.HLP in Windows of MAIN EXCEL HELP op de Macintosh) vindt u beperkte informatie over het opnemen en afspelen van macro's.

Als u verder wilt gaan dan het opnemen van macro's en de Visual Basic Naslaggids nog niet hebt geïnstalleerd, valt het aan te raden deze alsnog te installeren met behulp van het programma Microsoft Excel Setup. De informatie uit de Visual Basic Naslaggids zal u zeer goed van pas komen bij het ontwikkelen van toepassingen. Zie hoofdstuk 1, "Microsoft Excel installeren en starten", in het *Microsoft Excel handboek* voor meer informatie over het gebruik van het programma Setup en het installeren van optionele bestanden.

De Visual Basic Naslaggids gebruiken

De Visual Basic Naslaggids bevat naast informatie over alle sleutelwoorden in de Visual Basic-taal ook informatie over verschillende procedures en uiteenlopende informatie die geen betrekking heeft op sleutelwoorden. Als u wilt weten hoe deze informatie is geordend, kunt u het venster **Inhoudsopgave** van de Visual Basic Naslaggids oproepen.

▶ Het venster Inhoudsopgave van de Visual Basic Naslaggids oproepen

1. Kies de opdracht **Inhoudsopgave** in het menu **Help**.
2. Kies het onderwerp "Programmeren met Visual Basic".

In het venster **Inhoudsopgave** van de Visual Basic Naslaggids kunt u informatie opvragen over de volgende onderwerpen.

- De algemene taken en procedures in Visual Basic
- Visual Basic-functies
- Microsoft Excel-objecten in Visual Basic
- Eigenschappen die samenhangen met de objecten
- Methoden die samenhangen met de objecten
- Visual Basic-instructies
- Visual Basic-sleutelwoorden gerangschikt op programmeertaak
- Overige naslaginformatie

De meeste informatie in de Visual Basic Naslaggids heeft betrekking op sleutelwoorden. Een *sleutelwoord* is een woord of symbool dat wordt herkend als een onderdeel van de Visual Basic-programmeertaal. Voorbeelden van sleutelwoorden zijn instructies, functies, objecten, eigenschappen en methoden. Ook al kent u de betekenis van deze woorden nog niet en is u nog niet duidelijk hoe Visual Basic-programmacode precies werkt, zult u veel aan de Visual Basic Naslaggids hebben terwijl u Visual Basic leert.

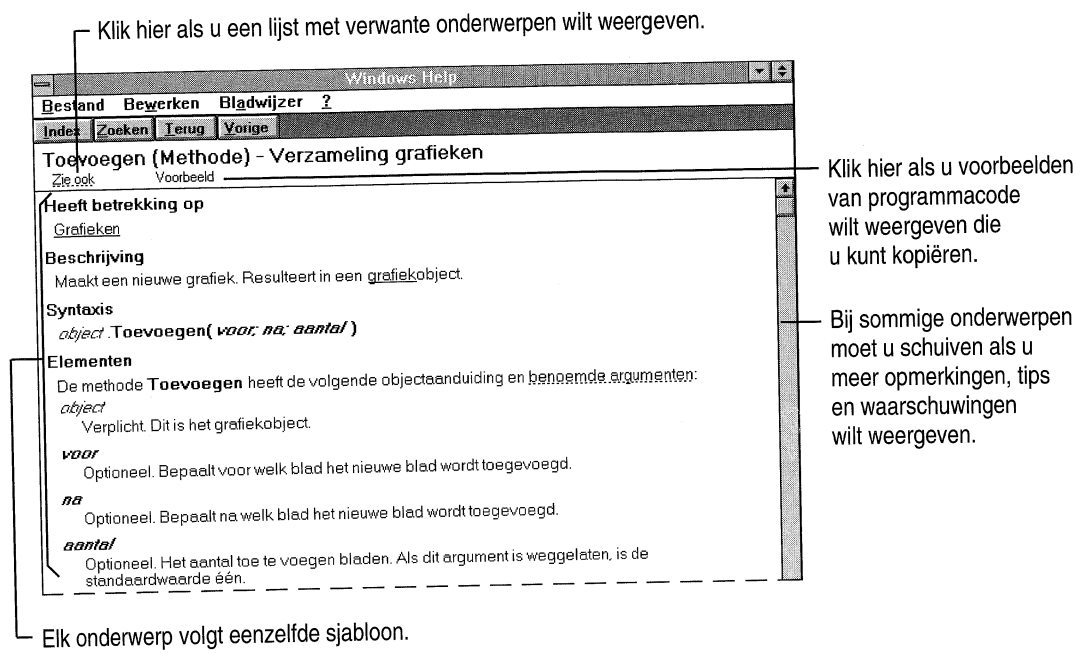
In de Visual Basic Naslaggids wordt gebruik gemaakt van een sjabloon voor de beschrijving van de sleutelwoorden. De onderdelen die deel uitmaken van de sjabloon, worden in de volgende tabel weergegeven. In een Help-onderwerp dat betrekking heeft op een bepaald sleutelwoord, komen zelden alle onderdelen aan de orde.

Kopje in de schermhulp	Informatie over het sleutelwoord
Titel (zonder label)	De naam van het sleutelwoord waarover u de informatie leest.
Zie ook	Een lijst met aanverwante onderwerpen.
Voorbeeld	Een voorbeeld van het sleutelwoord, zoals het in de programmacode wordt gebruikt. U kunt dit voorbeeld kopiëren en in uw macro plakken.
Heeft betrekking op	Een lijst met de objecten waarop een methode of een eigenschap betrekking heeft. Dit kopje verschijnt alleen in beschrijvingen van methoden en eigenschappen.
Beschrijving	Wat het sleutelwoord is of doet, of welk resultaat het geeft.
Syntaxis	De *syntaxis* van het sleutelwoord, waarin alle argumenten bij het sleutelwoord in de juiste volgorde worden vermeld.
Elementen	Een beschrijving van de elementen in de syntaxis. Verder wordt hier vermeld of deze elementen optioneel of verplicht zijn en welke waarden ze kunnen aannemen.
Opmerkingen	Overige informatie met betrekking tot het sleutelwoord, het gebruik ervan of de bijbehorende syntaxis.

In de secties Syntaxis en Elementen worden de onderdelen die u moet typen (de sleutelwoorden zelf bijvoorbeeld) vet weergegeven. Variabelen die de onderdelen vertegenwoordigen die u moet typen (de objecten bijvoorbeeld), worden cursief weergegeven. Andere variabelen (argumenten bijvoorbeeld) worden zowel cursief als vet weergegeven. Zie hoofdstuk 3, "Door de gebruiker gedefinieerde functies maken", voor meer informatie over argumenten.

De Visual Basic Naslaggids bevat voorbeelden voor nagenoeg elk sleutelwoord van Visual Basic. U kunt het onderwerp Voorbeeld kiezen om de voorbeeldcode in een ander venster weer te geven. U kunt deze programmacode kopiëren en in uw eigen macro plakken. Sommige voorbeelden zijn volledige procedures, de meeste voorbeelden bestaan echter uit één of twee regels. Zie hoofdstuk 2, "Opgenomen macro's bewerken", voor meer informatie over het plakken van voorbeeldcode in uw eigen macro's.

In de volgende figuur ziet u een voorbeeld van een doorsnee Help-onderwerp uit de Visual Basic Naslaggids.



Informatie voor ervaren ontwikkelaars van Microsoft Excel-macro's

Als u veel ervaring hebt met het ontwikkelen van Microsoft Excel-macro's, hebt u al de nodige tijd en moeite geïnvesteerd in het ontwikkelen van macro's in de macrotaal van Microsoft Excel versie 4.0. De macro's die in deze taal zijn geschreven, worden in Microsoft Excel versie 5.0 op de volgende wijze ondersteund.

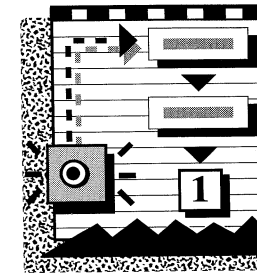
- Alle macro's die voor voorgaande versies zijn geschreven, kunnen in Microsoft Excel versie 5.0 worden gebruikt.
- Microsoft heeft de macrotaal van Microsoft Excel versie 4.0 uitgebreid en bijgewerkt, zodat ook de nieuwe voorzieningen van versie 5.0 worden ondersteund.
- U kunt nog steeds nieuwe macrobladen maken en macro's opnemen of schrijven met behulp van de vorige macrotaal.
- De schermhulp bevat de Macrofuncties Naslaggids, die volledig is bijgewerkt.

In toekomstige versies van Microsoft Excel zal Microsoft de macrotaal van Microsoft Excel 4.0 niet langer bijwerken. Het is dan ook raadzaam om nieuwe macro's te schrijven in Visual Basic. Als u wilt dat oude macro's ook kunnen beschikken over de nieuwe voorzieningen die worden toegevoegd aan toekomstige versies van Microsoft Excel, moet u deze macro's herschrijven als Visual Basic-procedures. U kunt deze macro's ook oproepen vanuit nieuwe Visual Basic-modulen. Microsoft Excel vertaalt de macro's die in de vorige macrotaal zijn geschreven niet in Visual Basic.

Het Microsoft Excel-programmapakket bevat informatie die de overgang van de macrotaal uit Microsoft Excel 4.0 naar Visual Basic vergemakkelijkt. De voorbeelden van programmacode die in het vorige gedeelte aan de orde zijn gekomen, worden bijvoorbeeld geleverd in twee versies. De ene versie maakt gebruik van Visual Basic en de andere van de vorige macrotaal. U kunt deze twee versies vergelijken om te zien waar de verschillen liggen. Zie Bijlage B, "Omschakelen van de macrotaal van Microsoft Excel 4.0", voor meer informatie over de omschakeling naar versie 5.0.

HOOFDSTUK 1

Terugkerende taken automatiseren



Tijdens het werken in Microsoft Excel, zult u waarschijnlijk bepaalde terugkerende taken uitvoeren. Het is bijvoorbeeld mogelijk dat u voor het bijwerken van verkoopcijfers, het uitzetten van gegevens in een grafiek of het toepassen van een bepaalde opmaak herhaaldelijk dezelfde reeks handelingen moet uitvoeren en dezelfde opdrachten moet kiezen. U kunt veel tijd en moeite besparen door zulke taken te automatiseren. Hiertoe beschikt u in Microsoft Excel over de krachtige programmeertaal Visual Basic.

Ook wanneer u geen ervaring hebt met programmeren, kunt u direct van Visual Basic gebruik maken. In Microsoft Excel beschikt u over een *macrorecorder*, een hulpprogramma dat Visual Basic-programmacode voor u maakt. In dit hoofdstuk vindt u eenvoudige, stapsgewijze instructies voor het automatiseren van simpele taken met de macrorecorder. Het opnemen van macro's is echter maar het begin. U zult opgenomen macro's doorgaans moeten bewerken en aanpassen aan uw specifieke behoeften. Zie hoofdstuk 2, "Opgenomen macro's bewerken", voor informatie over het bekijken en bewerken van opgenomen Visual Basic-programmacode.

Inhoud

- Hoe macro's taken vereenvoudigen
- Een macro opnemen
- Een macro starten
- Het gebruiksgemak van een macro verhogen
- Macro-opties instellen en wijzigen
- Tips voor het opnemen van macro's

Hoe macro's taken vereenvoudigen

In Microsoft Excel kunt u taken automatiseren met macro's. Een macro is een reeks opdrachten die automatisch wordt uitgevoerd. Als u bijvoorbeeld de opmaak van een cellenbereik wilt wijzigen, kiest u de opdracht **Celeigenschappen** in het menu **Opmaak**, kiest u de tab Lettertype, selecteert u vervolgens het lettertype, de lettertype-opmaak en de grootte (punten) van het gewenste lettertype en kiest u ten slotte OK. Met een macro kunt u deze handelingen combineren en ze in één keer uitvoeren.

Door uw eigen macro's op te nemen kunt u Microsoft Excel nog efficiënter op uw behoeften afstemmen. U kunt iedere reeks handelingen opnemen. Vervolgens kunt u de macro afspelen, oftewel *starten*, zodat de handelingen die u hebt opgenomen automatisch worden herhaald. Als u een macro eenmaal hebt opgenomen, kunt u deze aan een menu toevoegen of aan een knop toewijzen. Het starten van de macro is dan even simpel als het kiezen van een menu-opdracht of het klikken op een knop.

Wanneer neemt u een macro op?

Telkens wanneer u regelmatig dezelfde reeks toetsen aanslaat, dezelfde opdrachten kiest of dezelfde reeks handelingen uitvoert, is het opnemen van een macro het overwegen waard. Met macro's kunt u onder andere de volgende terugkerende taken automatiseren:

- Een groep werkmappen openen en gegevens uit deze werkmappen ophalen.
- Een aantal cellenbereiken afdrukken.
- Een database openen en sorteren, een rapport maken en de database sluiten.
- Een nieuw werkblad opzetten, inclusief het invoeren van titels, het aanpassen van kolombreedten en het toepassen van speciale opmaakkenmerken.

Voorbeeld

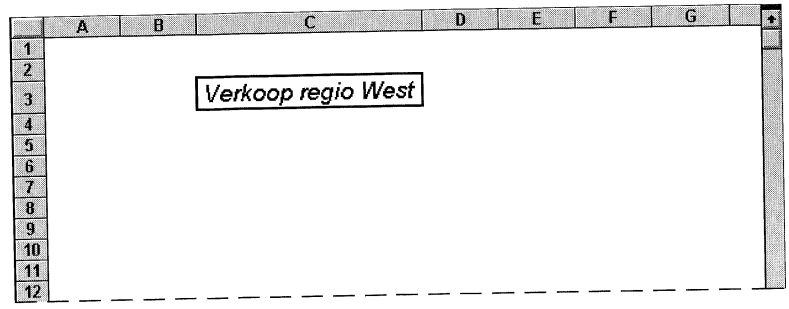
Stel dat u regelmatig een nieuw werkblad opzet waarin u verkoopgegevens kunt invoeren. Nadat u hebt overgeschakeld op een nieuw werkblad, voert u de volgende handelingen uit:

- De rasterlijnen uitschakelen.
- Cel C3 selecteren.
- De titel **Verkoop regio West** invoeren.

- De titel opmaken met het lettertype Times New Roman en hieraan de puntgrootte 18 toewijzen.
- De titel vet en cursief maken.
- Een donkerblauwe rand om de cel aanbrengen.
- Kolom C breder maken zodat de titel erin past.

U kunt het opzetten van dit werkblad aanzienlijk versnellen door de gehele taak op te nemen in een macro. Als u deze macro later start, wordt het werkblad automatisch opgezet volgens de reeks handelingen die u hebt opgenomen.

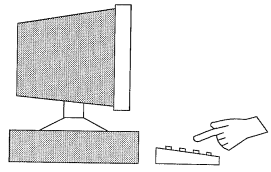
Het voltooide werkblad ziet u in de volgende figuur.



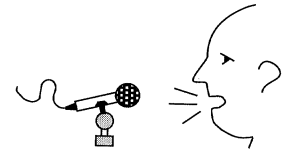
Opmerking U kunt werkbladen, inclusief titels en opmaak, ook opzetten door gebruik te maken van een sjabloon. Het voordeel van een macro is echter dat u deze later kunt uitbreiden en aanpassen om complexere taken te automatiseren. U kunt een macro bijvoorbeeld een berichtvenster laten weergeven, waarin u de tekst van de titel of de lokatie van de titel in het werkblad typt. In dit handboek wordt u geleidelijk vertrouwd gemaakt met het aanpassen en uitbreiden van macro's.

Het opnameproces

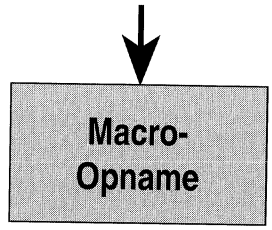
De macrorecorder van Microsoft Excel slaat handelingen en opdrachten op terwijl u deze uitvoert en kiest. De werking van de macrorecorder is vergelijkbaar met die van een bandrecorder. Met een bandrecorder kunt u opnemen wat u zegt en met de macrorecorder kunt u opnemen wat u doet.



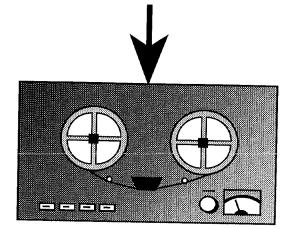
Acties die u onderneemt...



Door u gesproken woorden...



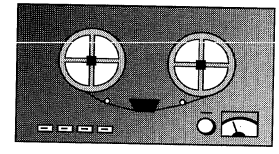
...worden opgenomen als macro's.



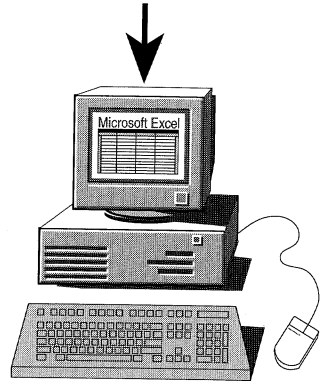
...worden op de band opgenomen.

Als u een macro start, volgt Microsoft Excel de opgenomen reeks instructies op. De macrorecorder herhaalt wat u hebt gedaan, net zoals een bandrecorder herhaalt wat u hebt gezegd.

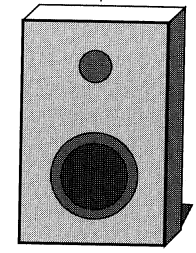
Wanneer u een opgenomen macro afspeelt...



Wanneer u een opgenomen band afspeelt...



...voert Microsoft Excel de acties automatisch uit.



...wordt het opgenomen geluid ten gehore gebracht.

Een macro opnemen

De eerste stap bij het automatiseren van taken in Microsoft Excel is het opnemen van een macro. In het volgende gedeelte vindt u een beschrijving van de algemene procedure die u volgt als u een macro wilt opnemen. Daarna volgen stapsgewijze instructies voor het opnemen van een voorbeeldmacro.

▶ Een macro opnemen

1. Kies de opdracht **Macro opnemen** in het menu **Extra** en kies vervolgens de opdracht **Nieuwe macro opnemen**.
2. Typ een naam voor de macro in het vak "Macronaam".

   De naam mag letters, cijfers en onderstrepingen bevatten en moet met een letter beginnen. In de naam mogen geen spaties of leestekens voorkomen.
3. Typ een beschrijving van de macro in het vak "Beschrijving".
4. Kies de knop Opties en selecteer de gewenste opties.

   Kies de knop Help voor een beschrijving van deze opties.
5. Kies de knop OK.

   Terwijl de macrorecorder opneemt, wordt de knop Opname stoppen in een aparte werkbalk op het scherm weergegeven.
6. Voer de handelingen uit die u wilt opnemen.
7. Kies de knop Opname stoppen.

   U kunt ook de opdracht **Macro opnemen** in het menu **Extra** en vervolgens de opdracht **Opname stoppen** kiezen.

Tip U kunt de procedure voor het opnemen van een macro verkorten door de stappen 2 en 3 in de hiervoor beschreven procedure over te slaan. Als u dit doet, wordt automatisch de naam "Macro*n*" aan de macro toegewezen, waarin *n* staat voor het laagste getal dat nog niet voor een andere macronaam in gebruik is.

Voorbeeld

In het volgende gedeelte wordt de procedure beschreven voor het opnemen van een macro die een werkblad maakt. Deze macro schakelt de weergave van de rasterlijnen uit, voert in cel C3 een titel in en voorziet de cel van een aangepaste opmaak.

U schakelt eerst de macrorecorder in en geeft de macro een naam en een beschrijving.

▶ **De opname van de voorbeeldmacro starten**

1. Kies **Macro opnemen** in het menu **Extra**, en kies vervolgens **Nieuwe macro opnemen**.
2. Typ in het vak "Macronaam" de tekst **TitelMaken**.
3. Typ in het vak "Beschrijving" de tekst **Maakt een werkblad met een titel in cel C3 en maakt de cel op**.
4. Kies de knop OK.

Vervolgens schakelt u de weergave van de rasterlijnen in het werkblad uit.

▶ **De rasterlijnen uitschakelen**

1. Kies de opdracht **Opties** in het menu **Extra**.
2. Selecteer de tab Weergave.
3. Schakel onder "Vensteropties" het aankruisvakje "Rasterlijnen" uit.
4. Kies de knop OK.

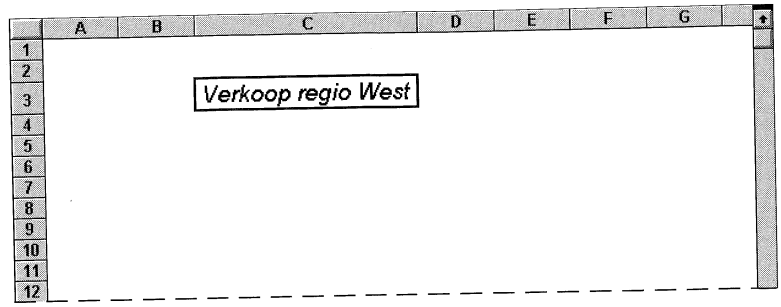
Ten slotte voert u een titel in de cel in, voorziet u de cel van een speciale opmaak, en sluit u de macro af. In dit voorbeeld wordt aangenomen dat de optie Selectie verplaatsen na Enter op de tab Bewerken van het dialoogvenster **Opties** is uitgeschakeld.

▶ **De titel invoeren, de cel opmaken en de macro afsluiten**

1. Selecteer cel C3, typ **Verkoop regio West** en druk op ENTER.
2. Kies de opdracht **Celeigenschappen** in het menu **Opmaak**.
3. Selecteer de tab Lettertype.
4. Selecteer "Times New Roman" in het vak "Lettertype".
5. Selecteer "Vet cursief" in het vak "Lettertype-opmaak".
6. Selecteer "18" in het vak "Punten".
7. Kies de tab Rand.
8. Selecteer "Omtrek" onder "Rand".
9. Selecteer onder "Type" de dikste ononderbroken lijn.
10. Kies de knop OK.
11. Kies de opdracht **Kolom** in het menu **Opmaak**, en kies vervolgens **Aan selectie aanpassen**.
12. Kies de knop Opname stoppen.

    U kunt ook de opdracht **Macro opnemen** in het menu **Extra** en vervolgens de opdracht **Opname stoppen** kiezen.

Het werkblad moet er nu uitzien als in de volgende figuur.



Een macro starten

Als u een macro hebt opgenomen, kunt u deze op ieder moment afspelen (starten). Microsoft Excel voert dan alle opdrachten uit die in de macro zijn opgeslagen.

▶ **Een macro starten**

1. Kies de opdracht **Macro** in het menu **Extra**.
2. Typ of selecteer een naam in het vak "Macronaam of verwijzing".
3. Kies de knop Starten.

Opmerking U kunt een macro tijdens het afspelen onderbreken door op ESC te drukken (of op COMMAND+PUNT op de Macintosh). Dit kan nodig zijn als u de verkeerde macro hebt gestart of een macro niet volledig hoeft uit te voeren. Als u een macro onderbreekt, verschijnt het dialoogvenster **Macrofout**. Kies de knop Help voor meer informatie over de opties in dit dialoogvenster. Als u het dialoogvenster wilt sluiten, drukt u nogmaals op ESC of kiest u de knop Einde.

Voorbeeld

U kunt de macro afspelen die u in het vorige voorbeeld hebt opgenomen.

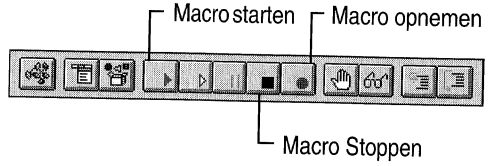
▶ **De macro TitelMaken starten**

1. Schakel over op een nieuw werkblad in de werkmap.
2. Selecteer cel A1.
3. Kies de opdracht **Macro** in het menu **Extra**.
4. Typ in het vak "Macronaam of verwijzing" de naam **TitelMaken**.
5. Kies de knop Starten.

Het werkblad moet er nu uitzien als in de figuur in het vorige gedeelte.

De werkbalk Visual Basic gebruiken

U geeft de werkbalk Visual Basic weer door de opdracht **Werkbalken** in het menu **Beeld** te kiezen. Met de knoppen op deze werkbalk kunt u macro's opnemen en starten en de opname van een macro stoppen.



De andere knoppen op de werkbalk Visual Basic worden elders in dit handboek en in de schermhulp beschreven.

Het gebruiksgemak van een macro verhogen

Als u een macro hebt opgenomen, kunt u deze aan een menu toevoegen of aan een knop of een ander grafisch object toewijzen. De macro is dan even snel en eenvoudig te gebruiken als de ingebouwde menu-opdrachten en knoppen van Microsoft Excel. U kunt hierdoor niet alleen sneller werken, maar zorgt ook dat anderen gemakkelijker van uw macro gebruik kunnen maken.

Een macro toevoegen aan het menu Extra

Als u een macro toevoegt aan het menu **Extra**, kunt u de macro kiezen zoals u alle andere menu-opdrachten kiest in Microsoft Excel.

▶ **Een macro toevoegen aan het menu Extra**

1. Kies **Macro** in het menu **Extra**.
2. Typ of selecteer een macronaam in het vak "Macronaam of verwijzing".
3. Kies de knop Opties.
4. Schakel onder "Toewijzen aan" het aankruisvakje "Opdracht in menu Extra" in en typ de opdrachtnaam zoals u deze wilt laten verschijnen in het menu **Extra**.
5. Kies de knop OK.
6. Kies de knop Sluiten.

Een macro toewijzen aan een knop in een werkblad

U kunt in een werkblad of grafiekblad een knop maken en hieraan een macro toewijzen. Door een macro aan een knop te verbinden, maakt u de macro zichtbaar en direct beschikbaar terwijl u aan het werk bent. Brengt u de knop bijvoorbeeld aan in een werkblad, dan is de macro altijd beschikbaar als u het werkblad opent.

▶ **Een knop maken in een werkblad en daaraan een macro toewijzen**

Voordat u deze procedure volgt, moet u de werkbalk Tekenen weergeven. Hiertoe kiest u de opdracht **Werkbalken** in het menu **Beeld**.

1. Kies de knop Nieuwe knop op de werkbalk Tekenen.
2. Wijs op de plaats waar u een van de hoeken van de knop wilt hebben.
3. Sleep met de muis totdat de knop de gewenste grootte en vorm heeft.

   Als u de muisknop loslaat, verschijnt het dialoogvenster **Toewijzen aan object**.
4. Als u een bestaande macro aan de knop wilt toewijzen, typt of selecteert u de naam van die macro in het vak "Macronaam of verwijzing" en kiest u vervolgens de knop OK.

   Als u een nieuwe macro aan de knop wilt toewijzen, kiest u de knop Opnemen, waarna u de procedure volgt voor het opnemen van een macro.

Opmerking Als u de opdracht **Objecteigenschappen** in het menu **Opmaak** kiest, verschijnt een dialoogvenster waarin u het lettertype en de uitlijning van de knoptekst kunt veranderen. Tevens kunt u in dit dialoogvenster de bescherming wijzigen, bepalen wat met de knop gebeurt als de grootte en plaats van aangrenzende cellen verandert en opgeven of de knop moet worden afgedrukt als het werkblad wordt afgedrukt.

U kunt ook de standaardnaam van een knop of een ander grafisch object vervangen door het object te selecteren en een nieuwe naam te typen in het naam/verwijzingsvak links van de formulebalk. Dit is tevens de naam waarmee u naar de knop verwijst in Visual Basic-programmacode.

Een macro toewijzen aan een werkbalkknop

Als u een macro toewijst aan een werkbalkknop, is de macro beschikbaar in alle bladen van de werkmap, mits de werkbalk wordt weergegeven. U kunt een macro het beste toewijzen aan een aangepaste knop die nog niet in gebruik is. *Aangepaste knoppen* zijn speciaal bedoeld voor macro's en andere aanpassingen. U vindt deze knoppen in de categorie Aangepast, zoals hierna wordt beschreven.

U kunt uw macro ook toewijzen aan een ingebouwde knop die al in gebruik is. Als u dit doet, wordt de normale functie van de knop vervangen door de functie van uw macro.

Als u een macro aan een ingebouwde knop of een aangepaste knop hebt toegewezen, kunt u de knop op een ingebouwde werkbalk plaatsen of een nieuwe werkbalk maken. Als u de knop op een werkbalk hebt geplaatst, functioneert de knop net zo als een ingebouwde werkbalkknop, bijvoorbeeld de knop Auto-som of de knop Afdrukken, met als enig verschil dat u met de knop een macro start die u eraan hebt toegewezen.

▶ **Een macro aan een ingebouwde werkbalkknop toewijzen**

1. Kies de opdracht **Werkbalken** in het menu **Beeld**.
2. Kies de knop Aanpassen.

   Het dialoogvenster **Aanpassen** verschijnt en de normale functie van de ingebouwde werkbalkknop wordt opgeheven.
3. Als de ingebouwde knop wordt weergeven op een werkbalk, kiest u de knop.

   Als de ingebouwde knop niet wordt weergegeven, selecteert u een categorie in het vak "Categorieën" en sleept u de knop uit het vak naar een zichtbare werkbalk.
4. Kies de opdracht **Toewijzen aan knop** in het menu **Extra**.
5. Als u een bestaande macro aan de ingebouwde knop wilt toewijzen, typt of selecteert u de naam van die macro in het vak "Macronaam of verwijzing" en kiest u vervolgens de knop OK.

   Wilt u een nieuwe macro aan de ingebouwde knop toewijzen, dan kiest u knop Opnemen en volgt u de procedure voor het opnemen van een macro.
6. Indien nodig kiest u de knop Sluiten, zodat het dialoogvenster **Aanpassen** wordt gesloten.

In plaats van de normale functie van een ingebouwde knop op te heffen, kunt u een van de aangepaste knoppen die niet in gebruik zijn naar een werkbalk slepen en de macro aan die knop toewijzen.

▶ **Een macro toewijzen aan een aangepaste knop en deze knop op een werkbalk plaatsen**

1. Kies **Werkbalken** in het menu **Beeld**.
2. Als u de aangepaste knop aan een nieuwe werkbalk wilt toevoegen, selecteert u de tekst in het vak "Werkbalknaam", typt u de naam van de nieuwe werkbalk, en kiest u vervolgens de knop Nieuw.

   Wilt u de aangepaste knop aan een bestaande werkbalk toewijzen, dan kiest u de knop Aanpassen.
3. Selecteer "Aangepast" in het vak "Categorieën".
4. Sleep de aangepaste knop uit het vak naar een zichtbare werkbalk.

   De knop wordt aan de werkbalk toegevoegd op de plaats die u aanwijst en het dialoogvenster **Toewijzen aan knop** verschijnt.
5. Als u een bestaande macro aan de aangepaste knop wilt toewijzen, typt of selecteert u de naam van die macro in het vak "Macronaam of verwijzing" en kiest u de knop OK.

   Wilt u een nieuwe macro aan de aangepaste knop toewijzen, dan kiest u de knop Opnemen en volgt u de procedure voor het opnemen van een macro.

Voorbeeld

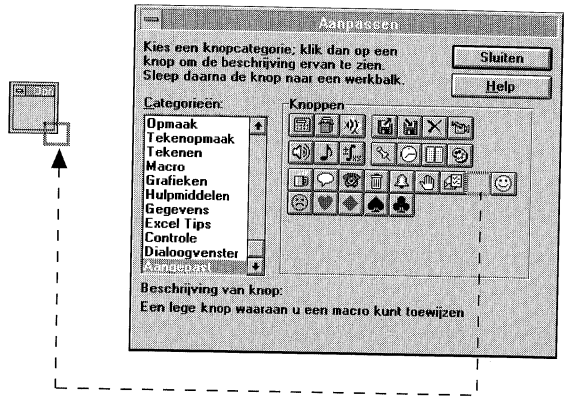
In de volgende instructies ziet u hoe u de macro **TitelMaken** aan een aangepaste knop toewijst en de knop op een nieuwe werkbalk plaatst. Als u de macro **TitelMaken** nog niet hebt gemaakt, volgt u eerst de instructies in het voorbeeld in "Een macro opnemen" eerder in dit hoofdstuk.

▶ **De macro TitelMaken toewijzen aan een aangepaste knop op een nieuwe werkbalk**

1. Kies de opdracht **Werkbalken** in het menu **Beeld**.
2. Selecteer de tekst in het vak "Werkbalknaam" en typ vervolgens **Speciale opmaak**.
3. Kies de knop Nieuw.

   De nieuwe werkbalk Speciale opmaak verschijnt. De werkbalk bevat nog geen knoppen.
4. Selecteer "Aangepast" in het vak "Categorieën".
5. Sleep de lege werkbalkknop naar de werkbalk Speciale opmaak, zoals is aangegeven in de volgende figuur.

   Het dialoogvenster **Toewijzen aan knop** verschijnt.
6. Typ of selecteer de tekst **TitelMaken** in het vak "Macronaam of verwijzing".
7. Kies de knop OK.
8. Kies de knop Sluiten.



Een macro toewijzen aan een grafisch object

Het kan voorkomen dat u een macro aan een grafisch object wilt toewijzen. Een *grafisch object* is een afbeelding die u in Microsoft Excel plakt vanuit een andere toepassing of een afbeelding die u maakt met de knoppen op de werkbalk Tekenen. Een grafisch object is een goede geheugensteun, vergemakkelijkt het gebruik van een macro en verfraait een werkblad. De procedure voor het toewijzen van een macro aan een grafisch object komt in grote lijnen overeen met de procedure voor het toewijzen van een macro aan een knop in een werkblad of grafiekblad.

U kunt bijvoorbeeld een macro opnemen die een maandrapport voor een afzetgebied in het buitenland maakt. U kunt die macro toewijzen aan een vlag die het betreffende afzetgebied aanduidt. Als u het rapport wilt maken, hoeft u alleen op de vlag te klikken.

▶ Een macro toewijzen aan een grafisch object

1. Selecteer het grafische object door erop te klikken.

   Als het object een werkbalkknop is of als aan het object al een macro is toegewezen, houdt u CTRL (in Microsoft Excel voor Windows) of COMMAND (in Microsoft Excel voor de Macintosh) ingedrukt en klikt u op het object.

2. Kies de opdracht **Macro Toewijzen aan object** in het menu **Extra**.

3. Als u een bestaande macro aan het grafische object wilt toewijzen, typt of selecteert u de naam van die macro in het vak "Macronaam of verwijzing" en kiest u de knop OK.

   Wilt u een nieuwe macro aan het grafische object toewijzen, dan kiest u de knop Opnemen en volgt u de procedure voor het opnemen van een macro.

Zie hoofdstuk 13, "Grafische objecten in werkbladen en grafieken maken", in het *Microsoft Excel Handboek* voor meer informatie over het maken van grafische objecten.

Een andere macro toewijzen aan een knop of grafisch object

Het kan voorkomen dat u een macro die u aan een knop of grafisch object hebt toegewezen wilt vervangen door een andere macro, dat u een nieuwe macro wilt opnemen voor een bestaand object, of dat u een bestaande macro wilt toewijzen aan een nieuw object. In Microsoft Excel kunt u deze wijzigingen eenvoudig aanbrengen. In de volgende procedure vindt u instructies voor het toewijzen van een andere macro aan een grafisch object of een knop in een werkblad of grafiekblad.

▶ **Een andere macro toewijzen aan een knop of een grafisch object in een werkblad**

1. In Microsoft Excel voor Windows houdt u CTRL ingedrukt en selecteert u de knop of het grafische object.

   In Microsoft Excel voor de Macintosh houdt u COMMAND ingedrukt en selecteert u de knop of het grafische object.

2. Kies de opdracht **Toewijzen aan object** in het menu **Extra**.

3. Als u aan de knop of het grafische object een bestaande macro wilt toewijzen, typt of selecteert u de naam van die macro in het vak "Macronaam of verwijzing" en kiest u de knop OK.

   Wilt u de aan de knop of het grafische object toegewezen macro wissen, dan wist u de naam van de macro in het vak "Macronaam of verwijzing" en kiest u de knop OK.

   Wilt u een nieuwe macro aan de knop of het grafische object toewijzen, dan kiest u de knop Opnemen en volgt u de stappen voor het opnemen van een macro.

Voor het wijzigen van een macro die aan een aangepaste knop op een werkbalk is toegewezen, gaat u op analoge wijze te werk, maar moet u eerst de opdracht **Werkbalken** in het menu **Beeld** kiezen en vervolgens de knop Aanpassen kiezen.

Macro-opties instellen en wijzigen

In het gedeelte "Een macro opnemen" eerder in dit hoofdstuk is een aantal van de opties behandeld die u kunt instellen bij het opnemen van een macro. Nadat u een macro hebt opgenomen, kunt u de volgende macro-opties wijzigen:

- De beschrijving van de macro
- De sneltoets waarmee u de macro start
- De naam van de macro, zoals deze wordt weergegeven in het menu **Extra**

Verder kunt u opties instellen waarmee de macro wordt toegewezen aan een onderwerp in een Helpbestand. Zie Bijlage C, "Aangepaste Help-onderwerpen weergeven", voor meer informatie over het gebruik van Helpbestanden. Zie hoofdstuk 2, "Opgenomen macro's bewerken", als u wilt weten hoe u de naam van een macro wijzigt.

▶ **Opties voor een bestaande macro instellen of wijzigen**

1. Kies de opdracht **Macro** in het menu **Extra**.
2. Typ of selecteer in het vak "Macronaam of verwijzing" de naam van de macro waarvan u de opties wilt wijzigen.
3. Kies de knop Opties.
4. Breng de gewenste wijzigingen aan in de opties.

   Kies de knop Help voor een beschrijving van deze opties.
5. Kies de knop OK.

Het verwijzingstype in een opname instellen

Een van de opties die u kunt instellen als u een macro opneemt, is het type verwijzing dat wordt opgenomen. Een *verwijzing* is het adres van een cel in Microsoft Excel, zoals A1 of B5. Verwijzingen kunnen absoluut of relatief zijn.

Als u het verwijzingstype instelt op *absoluut*, wordt de exacte positie opgenomen van iedere cel die u selecteert. Stelt u daarentegen het verwijzingstype in op *relatief*, dan wordt van iedere cel die u selecteert de positie ten opzichte van de vorige geselecteerde cel opgenomen.

Stel dat u cel A1 selecteert en vervolgens een macro opneemt die cel C4 selecteert en de inhoud van cel C4 vet maakt. Houd hierbij in gedachten dat cel C4 zich ten opzichte van cel A1 (de vorige selectie) twee cellen naar rechts en drie cellen naar beneden bevindt.

Als u het verwijzingstype instelt op absoluut voordat u cel C4 selecteert, maakt de macro altijd cel C4 vet. Stelt u daarentegen het verwijzingstype in op relatief voordat u cel C4 selecteert, dan selecteert de macro de cel die zich ten opzichte van de actieve cel twee cellen naar rechts en drie cellen naar beneden bevindt en wordt deze cel vet gemaakt. In dit geval selecteert de macro dus cel C4 als de actieve cel A1 is, D4 als de actieve cel B1 is, E5 als de actieve cel C2 is, enzovoort.

Het voordeel van relatieve verwijzingen is dat u de macro overal in het werkblad kunt gebruiken. Absolute verwijzingen bieden daarentegen de mogelijkheid een macro altijd op dezelfde cellen toe te passen, ongeacht uw positie in het werkblad. De voorbeeldmacro **TitelMaken**, die u volgens de instructies eerder in dit hoofdstuk hebt opgenomen, is bedoeld om te worden toegepast op cel C3 en kan daarom het beste met absolute verwijzingen worden opgenomen.

▶ **Het verwijzingstype instellen**

- Kies de opdracht **Macro opnemen** in het menu **Extra**. In het vervolgmenu ziet u de opdracht **Relatieve verwijzingen gebruiken**.

  Als bij de opdracht **Relatieve verwijzingen gebruiken** een vinkje staat, gebruikt Microsoft Excel relatieve verwijzingen. Als u absolute verwijzingen wilt gebruiken, kiest u de opdracht. Het vinkje verdwijnt dan, waaraan u kunt zien dat het verwijzingstype is gewijzigd.

  Als bij de opdracht **Relatieve verwijzingen gebruiken** geen vinkje staat, worden absolute verwijzingen gemaakt. Als u wilt overschakelen op relatieve verwijzingen, kiest u de opdracht. Het vinkje verschijnt dan naast de opdracht.

Zie hoofdstuk 10, "Formules en koppelingen maken", in het *Microsoft Excel Handboek* voor meer informatie over absolute en relatieve verwijzingen.

Opgenomen macro's permanent beschikbaar maken

Met een andere optie kunt u opgeven waar de macro die u opneemt, moet worden opgeslagen. Hiertoe kiest u achtereenvolgens de opdrachten **Macro opnemen** en **Nieuwe macro opnemen** in het menu **Extra**. In het dialoogvenster **Nieuwe macro opnemen** kiest u vervolgens de knop Opties. U kunt opgeven of u de macro wilt opslaan in de actieve werkmap, in een nieuwe werkmap of in de persoonlijke macrowerkmap.

De *persoonlijke macrowerkmap* is een verborgen werkmap die altijd is geopend, tenzij u opgeeft dat dit niet het geval moet zijn. Deze werkmap is een handige opslagplaats voor opgenomen macro's die u in uiteenlopende werkmappen gebruikt. Aangezien de persoonlijke macrowerkmap altijd is geopend, zijn de macro's die u erin opslaat altijd beschikbaar. De macronamen worden weergegeven in het dialoogvenster **Macro** (dat verschijnt als u de opdracht **Macro** kiest in het menu **Extra**), en de macro-sneltoetsen zijn beschikbaar (tenzij deze toetsen in gebruik zijn voor macro's in andere zichtbare werkmappen). De persoonlijke macrowerkmap wordt automatisch opgeslagen.

De persoonlijke macrowerkmap weergeven

De persoonlijke macrowerkmap is vergelijkbaar met een gewone werkmap. Het enige verschil is dat deze werkmap aanvankelijk slechts één blad bevat, waarin alle door u opgegeven macro's worden opgeslagen. Als u wat meer vertrouwd bent met Visual Basic, kunt u de persoonlijke macrowerkmap nog voor andere doeleinden gebruiken.

U geeft de persoonlijke macrowerkmap weer door de opdracht **Zichtbaar maken** in het menu **Venster** te kiezen.

De lokatie van de persoonlijke macrowerkmap

De persoonlijke macrowerkmap bestaat pas vanaf het moment dat u er een macro in opslaat. De werkmap wordt dan opgeslagen in de opstartdirectory of -map met de naam PERSNLK.XLS (in Microsoft Excel voor Windows) of PERSOONLIJKE MACROWERKMAP (in Microsoft Excel voor de Macintosh). Zie hoofdstuk 35, "Bepalen wat er gebeurt als u Microsoft Excel start", in het *Microsoft Excel Handboek* voor meer informatie over de opstartdirectory of -map.

Informatie over globale macro's voor gebruikers van Microsoft Excel 4.0

Als u hiervoor hebt gewerkt met Microsoft Excel versie 4.0, blijven globale macro's die u hebt opgeslagen in het Algemeen macroblad beschikbaar in Microsoft Excel versie 5.0. Het bestand ALGEMEEN.XLM (in Microsoft Excel voor Windows) of ALGEMEEN MACROBLAD (in Microsoft Excel voor de Macintosh) opent automatisch een verborgen document, net zoals in Microsoft Excel 4.0. De globale macro's die u opneemt in Microsoft Excel 5.0 worden echter opgeslagen in de persoonlijke macrowerkmap.

Tips voor het opnemen van macro's

Als u een opgenomen macro in een nieuw werkblad start, kan het gebeuren dat de macro een onverwachte uitwerking heeft. Aan de hand van de volgende tips kunt u macro's opnemen die beter aan uw behoeften voldoen:

- Maak een plan van wat u wilt doen, voordat u begint met opnemen.

  Vergeet niet dat de macrorecorder alles opneemt wat u doet, ook uw vergissingen.

- Selecteer eerst cellen of objecten en schakel dan pas de recorder in.

  Door deze werkwijze worden uw macro's overdraagbaar. Met andere woorden, u kunt de macro toepassen op een willekeurige actieve selectie. Een uitzondering hierop is een macro die u altijd op dezelfde cel wilt uitvoeren. (De voorbeeldmacro **TitelMaken**, die u volgens de instructies eerder in dit hoofdstuk hebt opgenomen, is een macro waarvoor u eerst de recorder inschakelt en vervolgens de cel selecteert).

- Schakel naar de werkmap en het blad waarvoor de macro bestemd is, voordat u de recorder inschakelt.

- Kijk tijdens het opnemen van macro's naar het venster met de Visual Basic-programmacode.

  Visual Basic-macro's worden opgeslagen in speciale bladen, zogenoemde *Visual Basic-modulen.* Als u de recorder hebt gestart en de opdracht **Alle vensters** in het menu **Venster** kiest, kunt u de module bekijken terwijl u in het werkblad werkt. U kunt dan zien hoe de macrorecorder Visual Basic-programmacode maakt terwijl u een macro opneemt. Zie hoofdstuk 4, "Inleiding in Visual Basic-procedures", voor meer informatie over Visual Basic-modulen.

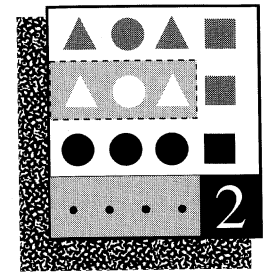
- Gebruik de macrorecorder als een leermiddel.

  Als u de Visual Basic-programmacode bestudeert die de recorder maakt, komt u meer te weten over de werking van deze programmacode. Houd er wel rekening mee dat de recorder niet altijd de meest aantrekkelijke of efficiënte programmacode maakt en dat u de opgenomen programmacode doorgaans zult willen bewerken. Hoe u dit doet, leest u in het volgende hoofdstuk.

Tot zover de inleiding over macro's en de macrorecorder. Hoewel u simpele taken zeer goed kunt automatiseren door macro's op te nemen, anticipeert de macrorecorder niet in iedere situatie precies wat u wilt doen. In de meeste gevallen zult u opgenomen macro's willen bewerken en aanpassen. In het volgende hoofdstuk wordt beschreven hoe u de Visual Basic-programmacode kunt bewerken die de macrorecorder heeft gemaakt. In latere hoofdstukken wordt beschreven hoe u zelf Visual Basic-programmacode kunt maken.

HOOFDSTUK 2

Opgenomen macro's bewerken



In hoofdstuk 1, "Terugkerende taken automatiseren", is behandeld hoe u een macro opneemt die automatisch de rasterlijnen uitschakelt, een rand om een cel aanbrengt en de cel opmaakt. Deze opgenomen macro voldoet voor deze taak, maar kan voor meer doeleinden bruikbaar worden gemaakt.

In dit hoofdstuk wordt behandeld hoe u de macro's die u hebt opgenomen kunt bekijken, bestuderen en aanpassen. Hierbij wordt de macro **TitelMaken**, die u volgens de instructies in hoofdstuk 1 hebt opgenomen, gebruikt als voorbeeld. U kunt deze macro zodanig aanpassen dat deze de rasterlijnen zowel in- als uitschakelt, een titel invoert in de geselecteerde cel in plaats van in een vaste cel en alleen specifieke opties voor het lettertype en de celrand instelt.

Inhoud

Een opgenomen macro bekijken en wijzigen

Macro's worden geschreven in de programmeertaal Visual Basic en worden opgeslagen in speciale bladen, zogenoemde Visual Basic-modulen. Als u meer wilt weten over de werking van een macro of als u een macro wilt bewerken, moet u deze eerst weergeven. Er zijn drie methoden om een macro weer te geven.

U kunt een macro weergeven met het dialoogvenster **Macro**.

▶ **Een macro weergeven met het dialoogvenster Macro**

1. Kies de opdracht **Macro** in het menu **Extra**.
2. Typ of selecteer in het vak "Macronaam of verwijzing" de naam van de macro die u wilt weergeven.
3. Kies de knop Bewerken.

   De macro verschijnt in een venster.

U kunt ook een macro weergeven die u hebt toegewezen aan een knop of een ander grafisch object.

▶ **Een macro weergeven die is toegewezen aan een knop in een werkblad of een ander grafisch object**

1. Klik in Microsoft Excel voor Windows met de rechtermuisknop op het object.

   Houd in Microsoft Excel voor de Macintosh COMMAND ingedrukt en klik op het object.
2. Kies de opdracht **Macro toewijzen** in het snelmenu.
3. Typ of selecteer in het vak "Macronaam of verwijzing" de naam van de macro die u wilt weergeven.
4. Kies de knop Bewerken.

De derde manier om een macro weer te geven is de Visual Basic-module te activeren waarin de macro is opgeslagen en de daarin aanwezige programmacode weer te geven. U moet dan wel de naam van die module kennen.

Voorbeeld

In dit voorbeeld ziet u hoe u de Visual Basic-programmacode kunt weergeven van de voorbeeldmacro **TitelMaken**, die u hebt opgenomen volgens de instructies in hoofdstuk 1.

▶ **De macro TitelMaken weergeven**

1. Kies de opdracht **Macro** in het menu **Extra**.
2. Typ of selecteer in het vak "Macronaam of verwijzing" de naam **TitelMaken**.
3. Kies de knop Bewerken.

Hierna wordt een Visual Basic-module weergegeven. (De naam van de module is Module1, tenzij u de naam van het blad hebt gewijzigd of een module hebt ingevoegd voordat u de macro opnam). De module bevat de volgende programmacode.

```
' TitelMaken
'
' Maakt een werkblad met een titel in cel C3 en maakt de cel op
'
```

```
Sub TitelMaken()
    ActiefVenster.RasterlijnenWeergeven = Onwaar
    Bereik("C3").Selecteren
    ActieveCel.R1K1Formule = "Verkoop regio West"
    Met Selectie.Lettertype
        .Naam = "Times New Roman"
        .TekstOpmaak = "Vet cursief"
        .Grootte = 18
        .Doorhalen = Onwaar
        .Superscript = Onwaar
        .Subscript = Onwaar
        .CountourLettertype = Onwaar
        .Schaduw = Onwaar
        .Onderstrepen = xlGeen
        .KleurIndex = xlAutomatisch
    Einde Met
    Selectie.Randen(xlLinks).Lijnopmaak = xlGeen
    Selectie.Randen(xlRechts).Lijnopmaak = xlGeen
    Selectie.Randen(xlBoven).Lijnopmaak = xlGeen
    Selectie.Randen(xlBeneden).Lijnopmaak = xlGeen
    Selectie.Omtrek dikte:=xlDik; kleurIndex:=xlAutomatisch
    Selectie.HeleKolom.AanSelectieAanpassen
Einde Sub
```

U hoeft in dit stadium niet helemaal te begrijpen hoe deze macro werkt. U kunt de stappen in de voorbeelden in dit hoofdstuk volgen om wat ervaring op te doen met het bewerken van macro's. Wilt u echter meer weten over een specifieke term, dan selecteert u deze en drukt u op F1 (in Windows) of op COMMAND+SHIFT+VRAAGTEKEN (op de Macintosh), zodat het Help-onderwerp over de term wordt weergegeven.

Visual Basic-programmacode lezen

Als u een macro opneemt, maakt Microsoft Excel een reeks instructies in Visual Basic-programmacode die corresponderen met de handelingen die u uitvoert. Als u de macro wilt aanpassen, kunt u deze instructies bewerken. U kunt ook opmerkingen toevoegen om het effect van de programmacode toe te lichten.

Instructies

Instructies zijn gecodeerde opdrachten aan Microsoft Excel om handelingen uit te voeren. Zo is de programmacode

```
ActieveCel.R1K1Formule = "Verkoopprognose 1994"
```

een Visual Basic-instructie om de tekst "Verkoopprognose 1994" in te voeren in de actieve cel. Als Microsoft Excel deze instructie opvolgt, gebeurt hetzelfde als wanneer u de tekst handmatig in de cel typt en op ENTER, RETURN of een pijltoets drukt om de tekst in te voeren.

Instructies kunnen bestaan uit sleutelwoorden, operatoren, variabelen en oproepingen van procedures. Zie hoofdstuk 6, "Werken met Visual Basic-programmacode in procedures", voor meer informatie over instructies en de onderdelen waaruit deze zijn opgebouwd.

Sleutelwoorden zijn termen die een speciale betekenis hebben in Visual Basic. Zo geven de sleutelwoorden **Sub** en **Einde Sub** het begin en het einde van een macro aan. Sleutelwoorden worden op kleurenmonitoren standaard als blauwe tekst weergegeven.

Tip Als u meer wilt weten over een sleutelwoord dat wordt gebruikt in een macro, hoeft u alleen het sleutelwoord te selecteren en op F1 (in Windows) of op COMMAND+SHIFT+VRAAGTEKEN (op de Macintosh) te drukken. Er verschijnt dan een onderwerp van de online Visual Basic Naslaggids waarin u informatie over de term aantreft. Is de term de naam van een variabele, een opmerking of de naam van een macro, dan geeft het Help-systeem een venster weer waarin onderwerpen met aanvullende informatie worden opgesomd. U kunt in de online Visual Basic Naslaggids ook zoeken naar een sleutelwoord. De *Microsoft Excel Visual Basic Naslaggids* is in gedrukte vorm verkrijgbaar bij Microsoft Press. Zie de "Inleiding" van deze handleiding als u wilt weten hoe u documentatie kunt bestellen.

Opmerkingen

Opmerkingen geven een beschrijving van een macro en lichten toe hoe de onderdelen van een macro samenwerken, maar hebben geen invloed op de werking van een macro. Door opmerkingen toe te voegen maakt u het veel eenvoudiger macro's aan te passen. U begrijpt immers sneller wat in een bepaalde macro gebeurt als u het doel en de werking van de macro in uw eigen woorden hebt omschreven. U geeft het begin van een opmerking aan met een enkel aanhalingsteken, zoals te zien is in de eerste regel van de macro in het vorige voorbeeld. Opmerkingen lopen altijd door tot het einde van de regel. De programmacode die op een opmerking volgt, moet op een nieuwe regel beginnen.

Aan het begin van een opgenomen macro wordt automatisch een opmerking toegevoegd. Deze opmerking bevat de naam en de beschrijving van de macro. Opmerkingen worden op kleurenmonitoren standaard groen weergegeven. U kunt de kleuren van de verschillende onderdelen van een macro wijzigen. Hoe u dit doet, leest u verderop in dit hoofdstuk in "Visual Basic aanpassen".

De schuifbalken gebruiken en de invoegpositie verplaatsen

U kunt naar een deel van de macro gaan dat niet zichtbaar is in het venster van de Visual Basic-module door op de schuifbalken aan de rechter- en onderrand van het venster te klikken. De positie van het schuifblokje geeft aan waar u zich in de module bevindt. Bij het gebruik van de schuifbalken wordt de invoegpositie niet verplaatst. Als u een macro wilt bewerken, moet u de invoegpositie handmatig verplaatsen.

Als u de invoegpositie wilt verplaatsen met de muis, gaat u zo nodig met de schuifbalken naar de gewenste plaats in de module, wijst u met de aanwijzer op de plaats waar u de invoegpositie wilt hebben en klikt u met de muis. Behalve met de muis kunt u de invoegpositie ook verplaatsen met toetsencombinaties. Als u dit doet, schuift het venster automatisch.

Om de invoegpositie als volgt te verplaatsen	Drukt u op deze toetsencombinatie
Een woord naar links	CTRL+PIJL-LINKS
Een woord naar rechts	CTRL+PIJL-RECHTS
Naar het begin van de module	CTRL+HOME
Naar het einde van de module	CTRL+END
Naar het volgende blad	CTRL+PGDN
Naar het vorige blad	CTRL+PGUP
Naar de volgende macro	CTRL+PIJL-OMLAAG
Naar de vorige macro	CTRL+PIJL-OMHOOG

Het venster splitsen in deelvensters

Soms zult u verschillende delen van een macro of twee verschillende macro's in dezelfde Visual Basic-module tegelijkertijd willen bekijken. In plaats van steeds heen en terug door de module te schuiven, kunt u de delen van het document die u wilt zien, weergeven in deelvensters. U kunt in ieder deelvenster schuiven zonder de positie van het andere venster te wijzigen. Hierdoor kunt u gemakkelijk verschillende delen van een macro met elkaar vergelijken. U kunt ook programmacode in het ene deelvenster kopiëren en in het andere deelvenster plakken.

▶ **Het venster in twee deelvensters splitsen met de muis**

- Dubbelklik op het splitsblokje in de verticale schuifbalk.

  U kunt het splitsblokje ook omlaag slepen totdat de deelvensters de gewenste hoogte hebben.

U kunt een venster ook met het toetsenbord in deelvensters splitsen.

▶ **Het venster in twee deelvensters splitsen met het toetsenbord**

- Kies de opdracht **Splitsen** in het menu **Venster**.

Programmacode bewerken

U bewerkt programmacode op dezelfde manier als u tekst bewerkt in de meeste tekstverwerkingsprogramma's. U selecteert en wist programmacode, typt over programmacode heen of voegt programmacode in door de invoegpositie te verplaatsen en nieuwe tekst te typen.

U geeft aan welke tekst u wilt bewerken door deze te selecteren. Met de muis of het toetsenbord kunt u tekst op verschillende manieren selecteren.

Een veel gebruikte techniek om tekst te selecteren is over de tekst te slepen met de muis. U zult echter vaak hele woorden of hele regels willen bewerken. U kunt de gewenste hoeveelheid programmacode snel selecteren door bepaalde toetsencombinaties te gebruiken of door met de muis te klikken.

▶ **Een onderdeel of tekstgedeelte selecteren**

- Plaats de muisaanwijzer op de positie waar de selectie moet beginnen, houd de muisknop ingedrukt en sleep de aanwijzer naar de plaats waar de selectie moet eindigen.

  U kunt ook de invoegpositie op de plaats zetten waar de selectie moet beginnen, vervolgens SHIFT ingedrukt houden en klikken op de plaats waar de selectie moet eindigen. Ten slotte kunt u nog SHIFT ingedrukt houden en met de pijltoetsen de invoegpositie op de plaats zetten waar de selectie moet eindigen.

Als u te veel of te weinig tekst hebt geselecteerd, kunt u de omvang van de selectie op de volgende manier wijzigen.

▶ **De omvang van een selectie wijzigen**

- Houd SHIFT ingedrukt en klik op de plaats waar de selectie moet eindigen.

  U kunt ook SHIFT ingedrukt houden en met de pijltoetsen naar de plaats gaan waar de selectie moet eindigen.

Als u een selectie wilt annuleren, klikt u op een willekeurige plaats in de Visual Basic-module of drukt u op een pijltoets.

Opmerkingen aan de programmacode toevoegen

Het toevoegen van opmerkingen aan programmacode is een uitstekende manier om het doel van de verschillende onderdelen van de macro toe te lichten. Opmerkingen maken het ook eenvoudiger een bepaald onderdeel te vinden als u de macro wilt aanpassen aan andere omstandigheden.

Voorbeeld

U kunt bij wijze van oefening een aantal opmerkingen toevoegen aan de macro **TitelMaken**, die u hebt opgenomen volgens de aanwijzingen in hoofdstuk 1. Raadpleeg hierbij de programmacode in het vorige voorbeeld. Verplaats de invoegpositie naar het einde van de instructie `Sub TitelMaken ( )`, druk tweemaal op ENTER, druk eenmaal op TAB, en typ de volgende opmerking.

```
'Opties instellen in dialoogvenster Weergave (Extra; Opties).
```

Verplaats het invoegpunt naar het einde van de instructie `Bereik("C3").Selecteren`, druk op ENTER en typ de volgende opmerking.

```
'De titel invoeren.
```

Verplaats het invoegpunt naar het einde van de instructie die begint met `ActieveCel`, druk tweemaal op ENTER en typ het volgende.

```
'Opmaakopties lettertype instellen.
```

Verplaats het invoegpunt naar het einde van de volgende instructie `Einde Met`, druk tweemaal op ENTER en typ het volgende.

```
'Opmaakopties rand instellen.
```

Verplaats het invoegpunt naar het einde van de tweede instructie die eindigt met `kleurIndex:=xlAutomatisch`, druk tweemaal op ENTER en typ de volgende opmerking.

```
'Kolombreedte instellen.
```

Overbodige programmacode wissen

De macrorecorder zet de meeste handelingen die u uitvoert om in programmacode. Vaak worden echter in de Visual Basic-module meer instructies opgenomen dan u echt in de macro wilt hebben en moet u de ongewenste programmacode wissen, zodat de macro precies doet wat u wilt.

Voorbeeld

Stel dat u de werkbladtitel niet altijd in cel C3 wilt invoeren als u deze macro uitvoert. In plaats daarvan wilt u de titel invoeren en opmaken in de cel die is geselecteerd op het moment dat u de macro start. Hiertoe wist u eerst de volgende instructie.

```
Bereik("C3").Selecteren
```

U kunt ook de opmaak van de titel wijzigen door het blok met de tekst van de titel te wijzigen. U wist alle niet-onderstreepte instructies in het volgende gedeelte.

```
'Opmaakopties lettertype instellen.
Met Selectie.Lettertype
    .Naam = "Times New Roman"
    .TekstOpmaak = "Vet cursief"
    .Grootte = 18
    .Doorhalen = Onwaar
    .Superscript = Onwaar
    .Subscript = Onwaar
    .ContourLettertype = Onwaar
    .Schaduw = Onwaar
    .Onderstrepen = xlGeen
    .KleurIndex = xlAutomatisch
Einde Met
```

Na deze laatste bewerking moet de macro **TitelMaken** er als volgt uitzien.

```
' TitelMaken
'
' Maakt een werkblad met een titel in cel C3 en maakt de cel op
'
Sub TitelMaken()
    'Opties instellen in dialoogvenster Weergave (Extra; Opties).
    ActiefVenster.RasterlijnenWeergeven = Onwaar

    'De titel invoeren.
    ActieveCel.R1K1Formule = "Verkoop regio West"

    'Opmaakopties lettertype instellen.
    Met Selectie.Lettertype
        .Naam = "Times New Roman"
        .TekstOpmaak = "Vet cursief"
        .Grootte = 18
    Einde Met
```

```
'Opmaakopties rand instellen.
    Selectie.Randen(xlLinks).Lijnopmaak = xlGeen
    Selectie.Randen(xlRechts).Lijnopmaak = xlGeen
    Selectie.Randen(xlBoven).Lijnopmaak = xlGeen
    Selectie.Randen(xlBeneden).Lijnopmaak = xlGeen
    Selectie.Omtrek dikte:=xlDik; kleurIndex:=xlAutomatisch

    'Kolombreedte instellen.
    Selectie.HeleKolom.AanSelectieAanpassen
Einde Sub
```

U moet ook de ook de regel commentaar wijzigen waarin specifiek wordt verwezen naar cel C3.

De bewerkte versie van de macro is aanzienlijk korter en voert alleen de gewenste handelingen uit. U kunt controleren of u geen instructies abusievelijk hebt gewist door de macro opnieuw uit te voeren. De macro moet de titel in de actieve cel invoeren en de cel op de gewenste manier opmaken.

Programmacode zoeken en vervangen

Voor het zoeken en wijzigen van tekst in een macro gebruikt u de opdrachten **Zoeken** en **Vervangen** in het menu **Bewerken**. Hebt u bijvoorbeeld de naam gewijzigd van een werkblad waarnaar in uw macro's wordt verwezen, dan moet u deze naam ook overal in de programmacode wijzigen. Met de opdracht **Zoeken** kunt u nagaan op welke plaatsen de door u opgegeven tekst voorkomt. Met de opdracht **Vervangen** kunt u de tekst zoeken en automatisch wijzigen.

▶ Zoeken naar tekst

1. Zorg dat een Visual Basic-module actief is en kies de opdracht **Zoeken** in het menu **Bewerken**.
2. Typ de tekst die u wilt zoeken in het vak "Zoeken naar".
3. Geef aan in welke richting u wilt zoeken door "Omhoog" of "Omlaag" te selecteren in het vak "Zoekrichting", of selecteer "Alles" als u een gehele selectie wilt doorzoeken.
4. Geef de omvang van de zoekopdracht aan door "Procedure", "Module", "Alle modulen" of "Geselecteerde tekst" in het vak "Zoeken in" te selecteren.
5. Selecteer een of meer van de opties die in de volgende tabel worden beschreven, als de tekst die u zoekt aan bepaalde criteria moet voldoen.
6. Kies de knop Volgende zoeken.

Maak bij het zoeken naar tekst gebruik van de volgende opties in het dialoogvenster **Zoeken**.

Als u wilt zoeken naar	Selecteert u
Hele woorden en geen tekens in woorden (bijvoorveeld "cel", maar niet "cellen")	Heel woord
Woorden waarin op een bepaalde manier gebruik is gemaakt van hoofdletters en kleine letters (bijvoorbeeld "Cel", maar niet "cel")	Identieke hoofdletters/kleine letters
Woorden met een bepaalde tekenreeks (bijvoorbeeld als u met **Cel*** wilt zoeken naar "Celtekst", "Celopmaak", enzovoort)	Jokertekens gebruiken

Als u de knop Volgende zoeken kiest, wordt de tekst geselecteerd op de plaats waar deze het eerst wordt aangetroffen of verschijnt een bericht dat de tekst niet is gevonden. U zoekt naar de volgende plaats waar de tekst voorkomt door de knop Volgende zoeken opnieuw te kiezen.

Als u de gevonden tekst wilt bewerken, kiest u de knop Annuleren. Als u verder wilt gaan met zoeken nadat u de gevonden tekst hebt bewerkt, kiest u opnieuw de opdracht **Zoeken** in het menu **Bewerken** en herhaalt u de procedure voor het zoeken naar tekst.

Als u begint met zoeken ergens midden in het zoekgebied dat u hebt opgegeven in stap 4, zoekt Microsoft Excel vanaf de plaats van de invoegpositie tot aan het begin of het einde van de macro. Vervolgens verschijnt een bericht waarin u wordt gevraagd of u wilt doorgaan met zoeken. Kies de knop OK als u de rest van het zoekgebied wilt doorzoeken, of kies de knop Annuleren als u wilt ophouden met zoeken.

▶ Tekst vervangen

1. Kies **Vervangen** in het menu **Bewerken**.
2. Typ de tekst waarnaar u wilt zoeken in het vak "Zoeken naar".
3. Typ de vervangende tekst in het vak "Vervangen door".
4. Geef aan in welke richting u wilt zoeken door "Omhoog" of "Omlaag" te selecteren in het vak "Zoeken", of selecteer "Alles" als u een hele selectie wilt doorzoeken.
5. Geef de omvang van de zoekopdracht aan door Procedure, Module, Alle modulen of Geselecteerde tekst te selecteren in het vak "Zoeken in".
6. Selecteer een of meer van de opties die in de vorige tabel worden beschreven, als de tekst die u zoekt aan bepaalde criteria moet voldoen.
7. Kies de knop Vervangen als u iedere vervanging wilt bevestigen.

   Wilt u de tekst overal vervangen zonder bevestiging, dan kiest u de knop Alles vervangen.

U kunt de laatste vervangbewerking annuleren of het resultaat ervan ongedaan maken.

▶ **Een vervangbewerking annuleren of ongedaan maken**

- Kies de opdracht **Ongedaan maken** in het menu **Bewerken**.

  Als u iedere vervanging afzonderlijk hebt bevestigd (met de knop Vervangen), wordt alleen de laatste vervanging ongedaan gemaakt. Als u alle vervangingen in één keer hebt doorgevoerd (met de knop Alles vervangen), worden deze ook alle ongedaan gemaakt.

Programmacode verplaatsen en kopiëren

De volgende procedure beschrijft hoe u programmacode verplaatst en kopieert met het Klembord. U gebruikt het Klembord om de programmacode die u wilt verplaatsen of kopiëren tijdelijk wilt opslaan.

▶ **Programmacode verplaatsen of kopiëren met het Klembord**

1. Selecteer de programmacode die u wilt verplaatsen.
2. Als u de programmacode wilt verplaatsen, kiest u de opdracht **Knippen** in het menu **Bewerken** of drukt u op CTRL+X.

   Wilt u de programmacode kopiëren, dan kiest u de opdracht **Kopiëren** in het menu **Bewerken** of drukt u op CTRL+C.
3. Plaats de invoegpositie daar waar u de tekst wilt invoegen.

   Als u de programmacode wilt kopiëren naar een andere Visual Basic-module, kunt u ook die module activeren en daarin de invoegpositie op de gewenste plaats zetten.
4. Kies **Plakken** in het menu **Bewerken** of druk op CTRL+V.

Voorbeeld

De taak die wordt uitgevoerd door de macro **TitelMaken** kan in twee verschillende taken worden verdeeld: enerzijds het instellen van opties voor de weergave van het actieve werkblad en anderzijds het invoeren van tekst in een cel van dat werkblad en het opmaken van die cel. U kunt programmacode van de ene naar de andere lokatie verplaatsen en kopiëren, zodat u twee macro's krijgt die u onafhankelijk van elkaar kunt gebruiken. Door een grote taak in kleinere taken te verdelen, krijgt u meer inzicht in een macro en kunt u deze gemakkelijker bewerken.

U kunt de macro **TitelMaken** verdelen in twee meer handelbare macro's door de volgende instructies te selecteren en vervolgens **Knippen** te kiezen in het menu **Bewerken**.

```
'Opties instellen in dialoogvenster Weergave (Extra; Opties).
ActiefVenster.RasterlijnenWeergeven = Onwaar
```

Plaats de invoegpositie achter de instructie `Einde Sub`, druk tweemaal op ENTER en typ het volgende.

```
Sub RasterlijnenInstellen()
```

Vergeet niet op ENTER te drukken aan het einde van de regel. Voeg de programmacode vervolgens op de nieuwe lokatie in door **Plakken** in het menu **Bewerken** te kiezen. Druk op ENTER en typ het volgende.

```
Einde Sub
```

U hebt nu twee macro's, **TitelMaken** en **RasterlijnenInstellen**. De programmacode van beide macro's moet er nu als volgt uitzien.

```
Sub TitelMaken()
    'De titel invoeren.
    ActieveCel.R1K1Formule = "Verkoop regio West"

    'Opmaakopties lettertype instellen.
    Met Selectie.Lettertype
        .Naam = "Times New Roman"
        .TekstOpmaak = "Vet cursief"
        .Grootte = 18
    Einde Met

    'Opmaakopties rand instellen.
    Selectie.Randen(xlLinks).Lijnopmaak = xlGeen
    Selectie.Randen(xlRechts).Lijnopmaak = xlGeen
    Selectie.Randen(xlBoven).Lijnopmaak = xlGeen
    Selectie.Randen(xlBeneden).Lijnopmaak = xlGeen
    Selectie.Omtrek dikte:=xlDik; kleurIndex:=xlAutomatisch

    'Kolombreedte instellen.
    Selectie.HeleKolom.AanSelectieAanpassen
Einde Sub

Sub RasterlijnenInstellen()
    'Opties instellen in dialoogvenster Weergave (Extra; Opties).
    ActiefVenster.RasterlijnenWeergeven = Onwaar
Einde Sub
```

Ten slotte wijst u de nieuwe macro **RasterlijnenInstellen** toe aan een toetsencombinatie, een knop of een grafisch object.

▶ **De macro RasterlijnenInstellen aan een toetsencombinatie toewijzen**

1. Kies de opdracht **Macro** in het menu **Extra**.
2. Typ of selecteer de naam **RasterlijnenInstellen** in het vak "Macronaam of verwijzing".

3. Kies de knop Opties.
4. Schakel onder "Toewijzen aan" het aankruisvakje "Sneltoets" in en typ de letter **R** in het vak.
5. Kies de knop OK en kies vervolgens de knop Sluiten.

Programmacode in een bestaande macro opnemen

U kunt aan een bestaande macro nieuwe programmacode toevoegen met de macrorecorder. Dit is handig als u een lange reeks stappen hebt opgenomen en bewerkt en daarna een van de taken wilt wijzigen of een nieuwe taak wilt toevoegen.

Als u programmacode wilt toevoegen aan een bestaande macro, kunt u deze programmacode handmatig typen in de Visual Basic-module met de macro. Een snelle manier om nieuwe taken aan een bestaande macro toe te voegen is de nieuwe programmacode op te nemen met de macrorecorder. De programmacode wordt ingevoegd op de plaats die u opgeeft.

▶ **Nieuwe programmacode opnemen en invoegen in een bestaande macro**

1. Plaats de invoegpositie op de lokatie in de macro waar u programmacode wilt toevoegen.
2. Kies de opdracht **Macro opnemen** in het menu **Extra** en kies vervolgens **Beginpositie opname bepalen**.
3. Activeer het blad van waaruit u de nieuwe taken wilt opnemen.
4. Kies de opdracht **Macro opnemen** in het menu **Extra** en kies vervolgens **Starten op beginpositie**.
5. Voer de taken uit die u wilt opnemen.
6. Kies de knop Macro-opname stoppen.

   U kunt ook de opdracht **Macro opnemen** in het menu **Extra** en vervolgens de opdracht **Opname stoppen** kiezen.

Tekst uit een bestand invoegen

Het is vaak handig programmacode in een ander bestand dan de werkmap op te slaan. U kunt de programmacode dan bewerken in een tekstverwerkingsprogramma zoals Microsoft Word of een reservekopie van uw programmacode bewaren als tekst. Het kan ook gebeuren dat u een tekstbestand krijgt toegeleverd van een informatiedienst en de programmacode wilt overbrengen naar een Visual Basic-module.

▶ **Tekst uit een bestand invoegen**

1. Activeer een Visual Basic-module en plaats de invoegpositie op de lokatie waar u tekst uit een bestand wilt invoegen.
2. Kies de opdracht **Bestand** in het menu **Invoegen**.
3. Typ of selecteer in het vak "Bestandsnaam" de naam van het bestand met de tekst die u wilt invoegen.
4. Kies de knop OK.

Een macro uitbreiden met Visual Basic-voorzieningen

Visual Basic biedt u veel meer mogelijkheden dan de macrorecorder. Met Visual Basic kunt u macro's testen onder speciale omstandigheden en een handeling alleen door de macro laten uitvoeren als aan een bepaalde voorwaarde wordt voldaan. U kunt een macro laten verzoeken om informatie van de gebruiker of de gebruiker op de hoogte houden van de voortgang van de macro terwijl deze wordt uitgevoerd.

In de volgende gedeelten wordt beschreven hoe u een aantal van de speciale voorzieningen van Visual Basic aan uw macro kunt toevoegen. Het is in dit stadium niet zo belangrijk dat u ieder onderdeel van de hier gegeven voorbeelden begrijpt. Het is belangrijker dat u ziet dat u met de voorzieningen van Visual Basic meer flexibele macro's kunt maken dan met de macrorecorder.

De instelling van een optie wijzigen

Voorbeeld

Met de eerder in dit hoofdstuk beschreven macro **RasterlijnenInstellen** kunt u de weergave van de rasterlijnen alleen uitschakelen. U kunt een andere macro schrijven die de rasterlijnen inschakelt, maar u kunt de macro **RasterlijnenInstellen** ook zodanig aanpassen dat deze werkt als een *aan/uit-schakelaar*. Als u op de toetsencombinatie drukt die u aan de macro hebt toegewezen, schakelt u de rasterlijnen uit. Drukt u opnieuw op deze toetsencombinatie, dan worden de rasterlijnen ingeschakeld.

U kunt de huidige instelling van de optie "Rasterlijnen" controleren in het dialoogvenster **Opties** (dat u weergeeft met de opdracht **Opties** in het menu **Extra**) en de instelling van de optie wijzigen met het sleutelwoord **Niet**. U bewerkt de macro **RasterlijnenInstellen** door een wijziging aan te brengen in de volgende regel uit het vorige codevoorbeeld:

```
ActiefVenster.RasterlijnenWeergeven = Onwaar
```

U wijzigt deze regel in de onderstreepte instructie in de volgende programmacode:

```
Sub ToewijzenRasterlijnen( )
    'Opties instellen in dialoogvenster Weergave (Extra; Opties).
    ActiefVenster   .RasterlijnenWeergeven = _
    Niet ActiefVenster.RasterlijnenWeergeven
Einde Sub
```

Macro's interactief maken

Voorbeeld

De macro **TitelMaken**, zoals u deze hebt aangepast volgens de instructies in het laatste voorbeeld, voert altijd de tekst "Verkoop regio West" in de actieve cel in. U kunt de macro **TitelMaken** breder toepasbaar maken door de gebruiker tijdens de uitvoering van de macro de titel te laten invoeren en deze informatie opslaan in een *variabele*, die tijdelijk de plaats van deze informatie inneemt. (Zie hoofdstuk 4, "Inleiding in Visual Basic-procedures", voor meer informatie over het gebruik van variabelen).

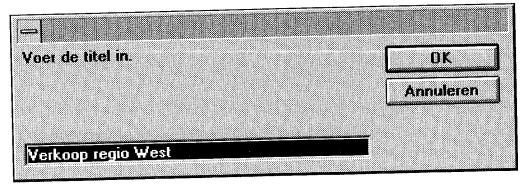
U maakt dit invoervenster, door de volgende regels:

```
'De titel invoeren.
ActieveCel.R1K1Formule = "Verkoop regio West"
```

te wijzigen in:

```
'De titel invoeren.
TitelTekst = Invoervenster( _
    aanwijzing := "Voer de titel in."; _
    standaard := "Verkoop regio West")
ActieveCel.R1K1Formule = TitelTekst
```

Tijdens de uitvoering van de macro **TitelMaken** wordt het volgende dialoogvenster weergegeven.



Controlestructuren toevoegen

U kunt een macro meer mogelijkheden geven door controlestructuren toe te voegen. Met controlestructuren kunt u Microsoft Excel instructies laten herhalen en laten beslissen welke instructies moeten worden uitgevoerd.

Stel bijvoorbeeld dat u alle cellen in het bereik A1:A10 met een getal groter dan 500 vet wilt maken. Dit lukt niet met een opgenomen macro, want in Microsoft Excel bestaat geen menu-opdracht waarmee u de waarde van een cel kunt vergelijken met 500 of waarmee u een handeling tien keer kunt herhalen.

U kunt dit resultaat wel bereiken met twee controlestructuren, een **Indien...dan**-instructie en een **Voor...volgende**-lus, zoals in het volgende voorbeeld.

```
'Maakt cellen met een getal groter dan 500 vet.
Sub CellenVet ()
    Voor I = 1 Tot 10                          'Voor elke cel in bereik,
        Indien Cellen(I; 1).Waarde > 500 Dan 'indien groter dan 500,
            Cellen(I; 1).Lettertype.Vet = Waar       'tekst vet maken.
        Einde Indien
    Volgende I
Einde Sub
```

Zie hoofdstuk 7, "De uitvoering van de programmacode besturen", voor meer informatie over het gebruik van controlestructuren in Visual Basic-programmacode.

Een codevoorbeeld kopiëren uit de schermhulp

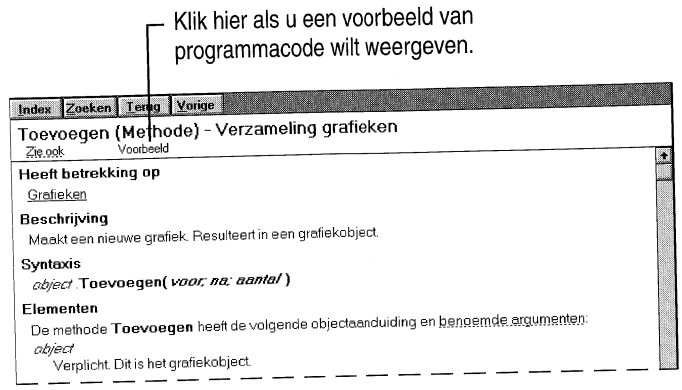
Veel van de onderwerpen in de online Visual Basic Naslaggids bevatten voorbeelden van programmacode, die u kunt kopiëren en in een Visual Basic-module kunt plakken. Een aantal van de voorbeelden zijn complete macro's; andere bestaan slechts uit een of twee regels programmacode.

▶ Een codevoorbeeld kopiëren uit de schermhulp

1. Activeer een Visual Basic-module en plaats de cursor op het Visual Basic sleutelwoord waarvan u een voorbeeld wilt zien..
2. Druk in Microsoft Excel voor Windows op F1. In Microsoft Excel voor de Apple Macintosh drukt u op COMMAND+/. Het Help-onderwerp voor het sleutelwoord verschijnt.
3. Klik op het woord "Voorbeeld", boven aan het Help-venster, zoals aangegeven in de volgende figuur.
4. Kies de knop Kopiëren in het venster met het voorbeeld.

   Het dialoogvenster **Kopiëren** verschijnt.
5. Selecteer het gedeelte van het voorbeeld dat u wilt kopiëren.
6. Kies de knop Kopiëren.
7. Activeer opnieuw de Visual Basic-module.
8. Plaats de invoegpositie op de lokatie waar u de programmacode wilt plakken.

9. Kies de opdracht **Plakken** in het menu **Bewerken**.



Visual Basic-modulen afdrukken

U kunt iedere macro in een actieve Visual Basic-module afdrukken. Dit is handig als u een systeem van macro's wilt controleren, herschrijven en op fouten wilt doorzoeken terwijl u de computer niet binnen bereik hebt.

▶ **Programmacode afdrukken**

1. Activeer een Visual Basic-module en kies de opdracht **Afdrukken** in het menu **Bestand**.
2. Stel de gewenste afdrukopties in.
3. Kies de knop OK.

Visual Basic aanpassen

U kunt in Visual Basic drie categorieën opties aanpassen, namelijk de algemene omgeving, de kleuren van de programmacode-onderdelen en de internationale instellingen.

▶ **Visual Basic-opties aanpassen**

1. Kies de opdracht **Opties** in het menu **Extra**.
2. Selecteer de tab Module Opmaak of de tab Module Algemeen.
3. Breng de gewenste wijzigingen aan in de weergegeven opties.
4. Kies de knop OK.

In de volgende tabel vindt u beschrijvingen van de opties die u kunt instellen in het tabblad Module Opmaak.

Optie	Beschrijving
Lettertype	Het lettertype van de tekst die wordt weergegeven in de module en het venster Foutopsporing
Punten	De puntgrootte van de tekst die wordt weergegeven in de module en het venster Foutopsporing
Kleuren	De achtergrond- en voorgrondkleuren voor verschillende programmacode-onderdelen

In de volgende tabel vindt u beschrijvingen van de opties die u kunt instellen in het tabblad Module Algemeen.

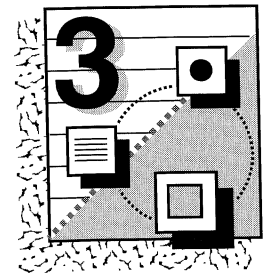
Optie	Beschrijving
Automatische inspringing	Hiermee geeft u aan of programmacode automatisch even ver moet inspringen als op de vorige regel.
Tabbreedte	Hier geeft u de ruimte tussen tabposities aan, uitgedrukt in spaties (van 1 tot 40).
Syntaxisfouten weergeven	Hiermee geeft u aan of na iedere coderegel die u typt onmiddellijk moet worden gecontroleerd of u de juiste syntaxis hebt gebruikt.
Macro onderbreken bij fouten	Hiermee bepaalt u of de onderbrekingsmodus moet worden geactiveerd als Microsoft Excel een fout in de programmacode aantreft. Als u deze optie niet kiest, roept Microsoft Excel een door de gebruiker gedefinieerde foutafhandelingsroutine op, indien voorhanden.
Definitie variabelen vereist	Met deze optie geeft u aan of iedere variabele expliciet moet worden gedeclareerd.

Ten slotte kunt u in het dialoogvenster **Opties** bepaalde internationale instellingen kiezen. Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over internationale instellingen.

Zie hoofdstuk 6, "Werken met Visual Basic-programmacode in procedures", voor meer informatie over variabelen en declaraties. Zie hoofdstuk 8, "Fouten in de programmacode opsporen en programmacode testen", voor meer informatie over de onderbrekingsmodus en over door de gebruiker gedefinieerde foutafhandelingsroutines.

HOOFDSTUK 3

Door de gebruiker gedefinieerde functies maken



In dit hoofdstuk wordt behandeld hoe u zelf functies definieert en deze functies in het werkblad invoert. Een functie die door een gebruiker is gedefinieerd, werkt net als elke ingebouwde Microsoft Excel-werkbladfunctie, bijvoorbeeld SOM of GEMIDDELDE. Maar aangezien de gebruiker zelf de functie definieert, kan deze precies bepalen wat de functie moet doen.

Eén enkele door de gebruiker gedefinieerde functie kan vaak een lange of geneste werkbladformule of zelfs een serie formules vervangen. Doordat verschillende werkbladformules worden vervangen door één formule, zijn dergelijke door de gebruiker gedefinieerde functies makkelijker te onthouden en efficiënter in het gebruik.

Inhoud

Alle functies die u zelf definieert, gebruiken Visual Basic-code, maar u hoeft niet veel van programmeren af te weten om te kunnen werken met dergelijke functies. Dit hoofdstuk is slechts een uitgangspunt. U wordt vertrouwd gemaakt met door de gebruiker gedefinieerde functies, maar veel termen, onderwerpen en informatie die van belang zijn in Visual Basic, worden vermeden.

Toch geldt dat hoe meer u afweet van Visual Basic, des te beter u in staat zult zijn functies te maken die op uw behoeften zijn toegespitst. Zie hoofdstuk 6, "Werken met Visual Basic-programmacode in procedures", en hoofdstuk 7, "De uitvoering van de programmacode besturen", voor meer informatie over door de gebruiker gedefinieerde functies.

Wat doet een door de gebruiker gedefinieerde functie?

U maakt een door de gebruiker gedefinieerde functie in een Visual Basic-module door rekenkundige expressies, ingebouwde Microsoft Excel-functies en Visual Basic-code te combineren. U voorziet de functie van een serie invoerwaarden, waarna de functie berekeningen uitvoert op basis van deze waarden en als uitkomst een nieuwe waarde geeft.

Opmerking Een door de gebruiker gedefinieerde functie kan ook werken met tekst, datums en waarden en dus niet alleen met cijfers en rekenkundige expressies. Hoofdstuk 6 bevat meer informatie over dit type door de gebruiker gedefinieerde functies.

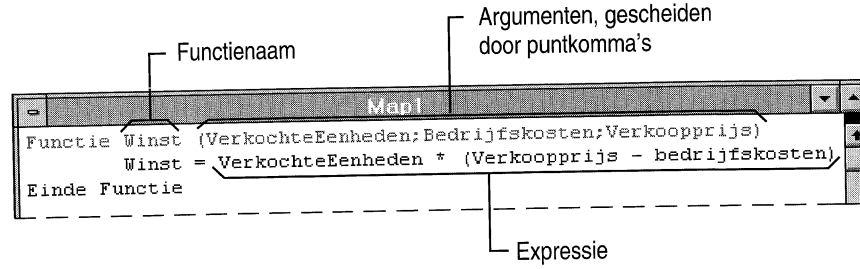
Een door de gebruiker gedefinieerde functie lijkt veel op een macro. Er zijn echter ook verschillen tussen een door de gebruiker gedefinieerde functie en het type macro dat u tot nu toe hebt leren opnemen en bewerken. Enkele van deze verschillen zijn in de volgende tabel vermeld.

Opgenomen macro's	Door de gebruiker gedefinieerde functies
Voeren een bewerking uit, bijvoorbeeld een grafiek maken of cellen verplaatsen	Resulteren in een waarde; kunnen geen bewerkingen uitvoeren
Kunnen worden opgenomen	Moeten in een Visual Basic-module worden gemaakt
Worden ingesloten door de sleutelwoorden **Sub** en **Einde Sub**	Worden ingesloten door de sleutelwoorden **Functie** en **Einde Functie**

Het belangrijkste verschil is, dat opgenomen macro's bewerkingen uitvoeren die een werkblad op een of andere manier wijzigen, terwijl door de gebruiker gedefinieerde functies resulteren in waarden.

De onderdelen van een door de gebruiker gedefinieerde functie

In de volgende figuur ziet u de onderdelen van een heel eenvoudige door de gebruiker gedefinieerde functie. De functie **Winst** berekent de brutowinst op basis van het aantal eenheden dat van een bepaald produkt is verkocht, de totale kosten om elk produkt te produceren en de prijs die elke eenheid heeft opgebracht. De functie heeft drie argumenten (VerkochteEenheden, Bedrijfskosten en Verkoopprijs) en een rekenkundige expressie.



Een door de gebruiker gedefinieerde functie accepteert waarden, maakt berekeningen en resulteert in een waarde. Daarom moet zo'n functie bestaan uit de volgende onderdelen, waarin de verschillende bewerkingen worden afgehandeld.

- De instructies **Functie** en **Einde Functie**. Deze Visual Basic-sleutelwoorden geven het begin en het einde van de functie aan.
- Een naam. Dit is de unieke aanduiding van de functie.
- Argumenten. Dit zijn de waarden die u opgeeft. Deze vormen het uitgangspunt voor de berekeningen die de functie uitvoert. De argumenten voor de functie definieert u door de argumentnamen tussen haakjes te typen achter de functienaam, waarbij u de argumenten van elkaar scheidt door een lijstscheidingsteken.

  Het gebruikte lijstscheidingsteken is afhankelijk van de instellingen voor uw landcode. In de Nederlandse versie wordt de puntkomma als lijstscheidingsteken gebruikt. Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over het schrijven van code voor internationaal gebruik.

- Visual Basic-code en expressies. Dit zijn de instructies die de functie vertellen welke berekeningen er moeten worden uitgevoerd. Een *expressie* is een combinatie van cijfers, variabelen en rekenkundige operatoren die een waarde oplevert.
- De resultaatwaarde. Dit is de waarde die de functie als resultaat van de berekeningen oplevert. U definieert een resultaatwaarde door achter de naam van de functie een expressie op te geven.

  NaamFunctie = expressie

Werken met de onderdelen van de door de gebruiker gedefinieerde functie

Sommige onderdelen van de door de gebruiker gedefinieerde functie lijken op onderdelen waarmee u al hebt gewerkt in werkbladen, terwijl andere onderdelen waarschijnlijk nieuw voor u zijn. Dit deel van het hoofdstuk gaat uit van uw huidige kennis van werkbladen en werkbladformules en geeft bij elk onderdeel een uitleg. Zie hoofdstuk 10, "Formules en koppelingen maken", in het *Microsoft Excel Handboek* voor achtergrondinformatie over functies.

Argumenten

Argumenten in door de gebruiker gedefinieerde functies lijken op de namen die u definieert voor cellen in een werkblad. Een argument is een bepaald type variabele. Een *variabele* is een naam die een waarde vertegenwoordigt. Het kan ook een verwijzing zijn naar een object, zoals een cel of een werkblad. Net als namen de werkbladformules makkelijker leesbaar maken, vormen argumenten en variabelen de basis voor doorzichtige expressies in door de gebruiker gedefinieerde functies.

Voorbeeld

De volgende functie berekent de winst na belasting (nettowinst). De argumenten zijn hier VerkochteEenheden, Bedrijfskosten, Verkoopprijs en PercentageInkomstenbelasting. De overige variabelen zijn BrutoWinst en NettoWinst. NettoWinst is een speciale variabele, omdat deze de resultaatwaarde bevat.

```
Functie NettoWinst(Verkochte
eenheden;Bedrijfskosten;Verkoopprijs;PercentageInkomstenbelasting)
    BrutoWinst=VerkochteEenheden*(Verkoopprijs-Bedrijfskosten)
    NettoWinst=BrutoWinst*(1-PercentageInkomstenbbelasting)
Einde Functie
```

In dit voorbeeld kunt u zien hoe argumenten en andere variabelen samenwerken in expressies. De expressie die de brutowinst berekent, gebruikt drie van de argumenten in de functie. Het resultaat van de expressie wordt opgeslagen in de variabele BrutoWinst. De expressie die de nettowinst berekent, gebruikt één argument en de variabele BrutoWinst. Het resultaat wordt opgeslagen in NettoWinst.

U kunt ook een functie definiëren waarvoor geen argumenten nodig zijn. De functie zou bijvoorbeeld de waarde in een cel of de actuele tijd als uitgangspunt voor berekeningen kunnen nemen, in plaats van waarden te gebruiken die u opgeeft in argumenten. ASELECT en NU zijn ingebouwde werkbladfuncties die geen argumenten vereisen.

Visual Basic-expressies

Expressies lijken op de formules die u in een werkblad invoert. De meeste functies en rekenkundige operatoren die u in werkbladformules gebruikt, zijn ook beschikbaar voor door de gebruiker gedefinieerde functies. Er zijn echter verschillen tussen werkbladformules en Visual Basic-expressies.

- Werkbladformules worden in cellen ingevoerd, terwijl u Visual Basic-expressies in een Visual Basic-module opgeeft.
- Werkbladformules beginnen met een "is gelijk"-teken en het resultaat wordt in de cel met de formule geplaatst. Aangezien Visual Basic moet weten waar het resultaat van een door de gebruiker gedefinieerde functie moet worden opgeslagen, worden Visual Basic-expressies voorafgegaan door een variabele en een "is gelijk"-teken. De variabele aan de linkerzijde van de vergelijking is de bestemmings- of opslagplaats van de waarde die rechts berekend wordt. In de volgende regel is BrutoWinst een variabele die het resultaat bevat van de rekenkundige expressie rechts ervan.

  ```
  BrutoWinst=VerkochteEenheden*(Verkoopprijs-Bedrijfskosten)
  ```

  De hele coderegel (een variabele gevolgd door een "is gelijk"-teken en een expressie) wordt een *toewijzingsinstructie* genoemd, omdat in deze regel een waarde aan een variabele wordt toegewezen.
- U kunt Visual Basic-instructies zoals **Indien**, **Voor** en **Doorlopen** opnemen in door de gebruiker gedefinieerde functies.

Voorbeeld

De functie NettoWinst in het voorgaande voorbeeld laat enkele van de verschillen tussen werkbladformules en Visual Basic-expressies zien. De volgende door een gebruiker gedefinieerde functie geeft deze verschillen ook en bevat tevens een **Indien...Dan...Anders**-instructie. De functie berekent de provisie die een effectenkantoor ontvangt voor de verkoop van effecten. Deze provisie is variabel en hangt af van de totale verkoopprijs.

Normaal gesproken bedraagt de provisie F 25 plus drie cent per aandeel. Maar voor transacties van meer dan F 15.000 wordt een quantumkorting van 10 procent verstrekt, zodat slechts 90 procent van de verkochte aandelen in de provisieberekening wordt opgenomen. De functie houdt rekening met deze voorwaarden door middel van de **Indien...Dan...Anders**-instructie.

```
Functie Provisie(AandelenVerkocht; PrijsPerAandeel)
    TotaleVerkPrijs = AandelenVerkocht * PrijsPerAandeel
    Indien TotaleVerkPrijs <= 15000 Dan
        Provisie = 25 + 0,3 * AandelenVerkocht
    Anders
        Provisie = 25 + 0,3 * (0,9 * AandelenVerkocht)
    Einde Indien
Einde Functie
```

De **Indien...Dan...Anders**-instructie wordt beschreven in hoofdstuk 7, maar in de functie **Provisie** kunt u zien hoe deze instructie werkt. Dit voorbeeld is iets complexer dan de voorgaande voorbeelden in dit hoofdstuk, maar het illustreert hoe u door de gebruiker gedefinieerde functies flexibeler kunt maken door er Visual Basic-code aan toe te voegen.

Resultaatwaarden

U geeft een resultaatwaarde op door achter de naam van de door de gebruiker gedefinieerde functie een waarde of een rekenkundige expressie te zetten. Een of meer regels in de functie moeten de volgende vorm hebben, *NaamFunctie* = *expressie*. In een door de gebruiker gedefinieerde functie is de naam van de functie vergelijkbaar met een variabele. De resultaatwaarde van de functie is de waarde die in *NaamFunctie* wordt opgeslagen, nadat Microsoft Excel de functie heeft berekend.

Zo bevat de functie Provisie twee verschillende toewijzingsinstructies die een resultaatwaarde berekenen. `Provisie = 25 + 0,03 * AandelenVerkocht` wordt gebruikt als de totale verkoopprijs minder dan of gelijk aan F 15,000 is; zo niet, dan wordt `Provisie = 25 + 0,03 * (0,9 * AandelenVerkocht)` gebruikt. De code is zo opgesteld dat slechts één van deze voorwaarden van toepassing kan zijn. Nadat de functie de berekeningen heeft uitgevoerd, wordt de resultaatwaarde opgeslagen in de variabele `Provisie`.

Over het algemeen geldt dat u een functie zo moet definiëren dat deze één waarde oplevert voor een bepaalde reeks argumenten. Een uitzondering hierop vormt een door de gebruiker gedefinieerde functie die als resultaat een matrix geeft. Meer bijzonderheden hierover vindt u in hoofdstuk 6.

Een door de gebruiker gedefinieerde functie maken

In hoofdstuk 2, "Opgenomen macro's bewerken", hebt u gezien hoe u reeds opgenomen macro's kunt bewerken in een Visual Basic-module. U kunt in een Visual Basic-module ook een nieuwe functie definiëren.

▶ Een door de gebruiker gedefinieerde functie maken

1. Als u naar een Visual Basic-module in de actieve werkmap wilt overschakelen, selecteert u de tab van een Visual Basic-module.

   –Of–

   Als u een nieuwe Visual Basic-module wilt maken, kiest u **Macro** in het menu **Invoegen** en vervolgens de opdracht **Module**.

2. Typ het sleutelwoord **Functie** gevolgd door de naam van de door de gebruiker gedefinieerde functie.
3. Typ hierna tussen haakjes de lijst met argumenten, waarbij u de argumenten van elkaar scheidt door een puntkomma of het desbetreffende scheidingsteken.
4. Druk op ENTER om naar de volgende regel te gaan.

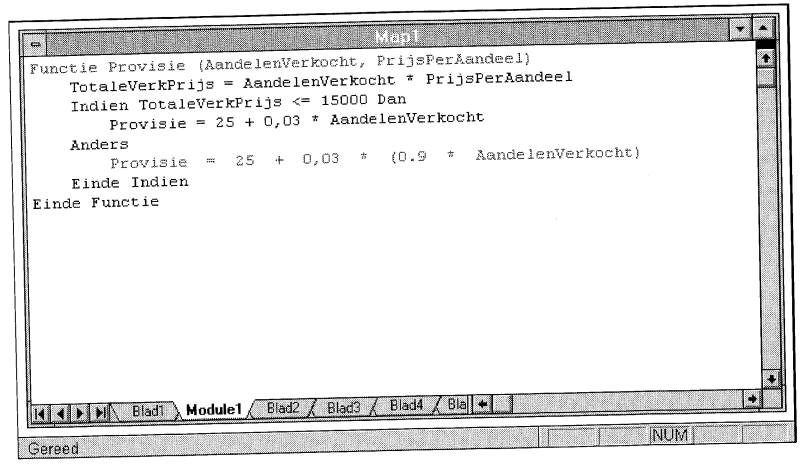
   Microsoft Excel controleert de syntaxis van de regel die u net hebt getypt. De Visual Basic-sleutelwoorden worden nu blauw weergegeven of in een andere kleur, als deze is opgegeven.

5. Druk op TAB, typ de eerste coderegel en druk vervolgens op ENTER.

   TAB laat de code inspringen, zodat deze makkelijker te lezen is.

6. Typ de overige coderegels.
7. Typ **Einde Functie** en druk op ENTER.

In de volgende figuur ziet u de functie Provisie uit het voorbeeld in een Visual Basic-module.



Een door de gebruiker gedefinieerde functie aan het werk zetten

Zodra u zelf een functie hebt gedefinieerd, kunt u deze functie in het werkblad invoeren. Een door de gebruiker gedefinieerde functie wordt op exact dezelfde wijze ingevoerd en gebruikt als een ingebouwde Microsoft Excel-functie. U voert de naam van de functie bijvoorbeeld in een werkbladcel in, op dezelfde manier als u de naam van een ingebouwde functie, zoals SOM of GEMIDDELDE, zou invoeren.

▶ **Een door de gebruiker gedefinieerde functie invoeren in een werkblad**

1. Typ een "is gelijk"-teken (=) in de cel waarin u de functie wilt invoeren, gevolgd door de functienaam en haakje openen.
2. Typ de waarden, gescheiden door een puntkomma of het desbetreffende lijstscheidingsteken, en sluit de lijst af met haakje sluiten.
3. Druk op ENTER.

Opmerking U kunt een functie in een geselecteerde cel invoeren door de opdracht **Functie** in het menu **Invoegen** te kiezen en verder de instructies te volgen in de Wizard Functies. Zie "Werken met de Wizard Functies", in hoofdstuk 10 in het *Microsoft Excel Handboek* voor meer informatie over de Wizard Functies.

In de volgende formule wordt de functie Provisie uit de vorige figuur gebruikt. De waarden voor AandelenVerkocht en PrijsPerAandeel zijn respectievelijk 100 en F 50. U zou de formule als volgt in een werkbladcel invoeren.

```
=Provisie(100;50)
```

De door de gebruiker gedefinieerde functies snel terugvinden

Als u in het werkblad de opdracht **Functie** in het menu **Invoegen** kiest, wordt in het dialoogvenster **Wizard Functies** een lijst weergegeven met de beschikbare functies en de bijbehorende categorieën. Alle rekenkundige functies, datum- en tijdfuncties, database-functies, enzovoort staan bij elkaar, zodat u deze makkelijk kunt terugvinden. Nadat u een functie hebt gedefinieerd, kan het handig zijn deze toe te wijzen aan een categorie. U kunt een van de ingebouwde categorieën gebruiken of de categorie Door de gebruiker gedefinieerd.

U wijst een functie toe aan een categorie met de opdracht **Objectenoverzicht** in het menu **Beeld**. Met sommige termen in het dialoogvenster **Objectenoverzicht** bent u waarschijnlijk niet vertrouwd. In hoofdstuk 4, "Inleiding in Visual Basic-procedures", leert u meer over het Objectenoverzicht.

▶ **Een door de gebruiker gedefinieerde functie toewijzen aan een categorie**

1. Schakel over naar de Visual Basic-module in uw werkmap waarin de door de gebruiker gedefinieerde functie zich bevindt.
2. Kies **Objectenoverzicht** in het menu **Beeld**.
3. Typ of selecteer de naam van de werkmap waarin uw module zich bevindt in het vak "Bibliotheken/Werkmappen".
4. Selecteer de module met de gewenste functie in het vak "Objecten/Modulen".
5. Selecteer de naam van de gewenste functie in het vak "Methoden/Eigenschappen".
6. Kies de knop Opties.
7. Selecteer de categorienaam in het vak "Functiecategorie".
8. Kies de knop OK.
9. Kies de knop Weergeven.

Meer leren over door de gebruiker gedefinieerde functies

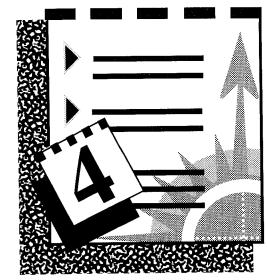
U bent nu aan het einde gekomen van de inleiding over door de gebruiker gedefinieerde functies. U hebt gezien hoe Visual Basic-instructies de door de gebruiker gedefinieerde functies kunnen uitbreiden (vergelijk de eenvoudige functie Winst eerder in dit hoofdstuk met de gedetailleerde beslissingen die verderop gemaakt worden in de functie Provisie). Over Visual Basic zelf is echter nog heel wat meer te vertellen.

Als u verwacht regelmatig zelf functies te definiëren, is het raadzaam hoofdstuk 6 en 7 door te nemen. Daar kunt u lezen hoe u informatie doorgeeft aan of ophaalt uit Visual Basic-procedures (door de gebruiker gedefinieerde functies zijn in feite eveneens procedures). U vindt daar tevens informatie over de programmataal Visual Basic. Daarnaast leert u in die hoofdstukken hoe u argumenten, variabelen en resultaatwaarden nauwkeurig kunt besturen. Dit kan van pas komen bij tekstfuncties, datum- en tijdfuncties, enzovoort.

In het volgende hoofdstuk zult u zien hoe verschillende macro's en door de gebruiker gedefinieerde functies kunnen samenwerken om gecompliceerde taken te automatiseren.

HOOFDSTUK 4

Inleiding in Visual Basic-procedures



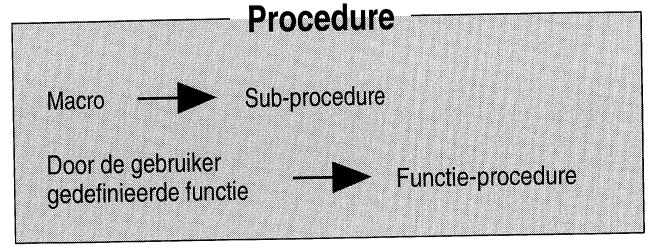
Dit hoofdstuk bevat een inleiding in Visual Basic-procedures en geeft een inzicht in wat deze procedures zijn, wat u ermee kunt doen, hoe ze samenwerken en wat de inhoud ervan is. In feite hebt u in de voorgaande hoofdstukken al kennisgemaakt met procedures, alleen werden deze toen geen procedures genoemd. De macro's die u hebt opgenomen in hoofdstuk 1, "Terugkerende taken automatiseren", zijn **Sub**-procedures, terwijl de door de gebruiker gedefinieerde functies die u hebt gemaakt in hoofdstuk 3, "Door de gebruiker gedefinieerde functies maken", een bepaald type **Functie**-procedure zijn. In dit hoofdstuk wordt nader ingegaan op deze typen procedures. In de volgende hoofdstukken maakt u uitgebreider kennis met de Visual Basic-programmacode die de functies in staat stelt de gewenste taken uit te voeren.

Inhoud

- Wat is een Visual Basic-procedure?
- Hoe procedures samenwerken om een complexe taak te vervullen
- Procedures oproepen
- Procedures en modulen ordenen in werkmappen

Wat is een Visual Basic-procedure?

Nu u vertrouwd bent met macro's en door de gebruiker gedefinieerde functies, kunt u voortaan de standaardterm van Visual Basic hiervoor gebruiken, te weten "procedures".



Een *procedure* is een blok Visual Basic-programmacode dat in een Visual Basic-module wordt opgenomen en als één geheel wordt uitgevoerd. Elke procedure omvat een reeks Visual Basic-instructies die tot doel heeft één bepaalde taak af te handelen. Visual Basic kent twee hoofdtypen procedures: **Sub**-procedures en **Functie**-procedures.

Een ***Sub**-procedure* voert een actie uit maar resulteert niet in een waarde. Deze procedure staat tussen de instructies **Sub** en **Einde Sub**.

```
'Geeft het aantal pieptonen dat is opgegeven in AantalTonen
Sub Pieptonen(AantalTonen)
    Voor Teller = 1 Tot AantalTonen
        Pieptoon                              'Geeft een pieptoon
    Volgende Teller
Einde Sub
```

Als een **Sub**-procedure wordt uitgevoerd vanuit een Microsoft Excel-werkblad of -grafiekblad (bijvoorbeeld door de opdracht **Macro** in het menu **Extra** te kiezen of op een macroknop te klikken), kan deze ook een macro worden genoemd, maar het blijft een **Sub**-procedure. Als een **Sub**-procedure uitsluitend wordt uitgevoerd door andere procedures in een Visual Basic-module, wordt de term "macro" niet gebruikt. Verderop in dit hoofdstuk zult u zien hoe een procedure kan worden gestart vanuit een andere procedure.

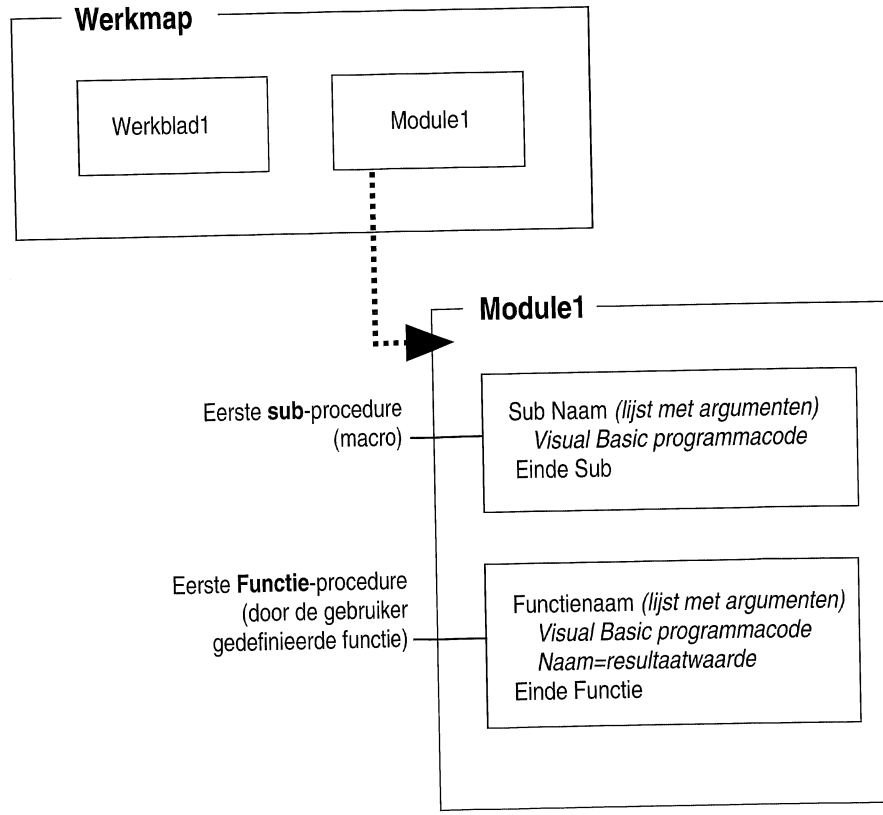
Een ***Functie**-procedure* lijkt op een **Sub**-procedure, maar resulteert in een waarde. Deze procedure staat tussen de instructies **Functie** en **Einde Functie**.

```
'Geeft de vierkantswortel van een getal, zelfs als dit negatief is
Functie AbsWort(Num)
    TijdWrd = ABS(Num)                        'Maakt Num positief
    AbsWort = Wortel(TijdWrd)                 'Geeft de wortel
Einde Functie
```

Als u de **Functie**-procedure zodanig schrijft dat deze geen bewerkingen uitvoert die de Microsoft Excel-omgeving wijzigen, kunt u de naam van de **Functie**-procedure in een werkbladcel invoeren, zodat de procedure resulteert in een waarde voor de desbetreffende cel. In dit geval wordt de **Functie**-procedure ook wel een door de gebruiker gedefinieerde functie genoemd, zoals u in hoofdstuk 3, "Door de gebruiker gedefinieerde functies maken", hebt gezien. Niettemin blijft het een gewone **Functie**-procedure met een aantal beperkingen. Als een **Functie**-procedure uitsluitend wordt uitgevoerd door andere procedures in een Visual Basic-module, wordt de term "door de gebruiker gedefinieerde functie" niet gebruikt.

Als u een **Functie**-procedure wilt gebruiken als een door de gebruiker gedefinieerde functie, kan de procedure geen bewerkingen uitvoeren die de Microsoft Excel-omgeving wijzigen. Zulke bewerkingen zijn bijvoorbeeld invoegen, verwijderen of opmaken van cellen of wijzigen van celwaarden; bladen of werkmappen verplaatsen, verwijderen of toevoegen of de naam ervan wijzigen; berekeningsmodus of schermweergave wijzigen. Het is ook niet toegestaan om in dergelijke **Functie**-procedures eigenschappen in te stellen of de meeste methoden uit te voeren, hoewel de procedure wel eigenschappen kan ophalen. Over het algemeen kunt u de meeste problemen voorkomen als de door de gebruiker gedefinieerde functies die u toepast, uitsluitend berekeningen uitvoeren op basis van gegevens die u verstrekt of die kunnen worden opgehaald uit Microsoft Excel.

De volgende figuur laat zien hoe **Sub**-procedures (macro's) en **Functie**-procedures (waarvan sommige door de gebruiker gedefinieerde functies zijn) deel uitmaken van modulen, die op hun beurt weer deel uitmaken van werkmappen.



Er is nog een derde, minder vaak gebruikte procedure, te weten de eigenschapsprocedure. Deze kan de instructies **Eigenschap Halen**, **Eigenschap Bepalen** of **Eigenschap Toewijzen** bevatten. In het gedeelte "Meer leren over programmeren met Visual Basic" in hoofdstuk 6 wordt nader ingegaan op eigenschapsprocedures.

Onderdelen van een procedure

De onderdelen van een procedure komen overeen met de onderdelen van een macro zoals beschreven in hoofdstuk 2, "Opgenomen macro's bewerken", en van een door de gebruiker gedefinieerde functie zoals beschreven in hoofdstuk 3, "Door de gebruiker gedefinieerde functies maken". De onderdelen van een procedure zijn:

- De instructies **Functie** en **Einde Functie** of **Sub** en **Einde Sub**. Deze Visual Basic-sleutelwoorden geven het begin en het einde van de procedure aan.
- Een naam. Dit is de unieke aanduiding van de procedure.
- Argumenten. Dit zijn de waarden die u opgeeft aan de procedure. Deze vormen het uitgangspunt voor de berekeningen die een **Functie**-procedure uitvoert, of de informatie op basis waarvan een **Sub**-procedure bepaalt wat er gedaan moet worden en hoe. De argumenten voor een **Sub-** of **Functie**-procedure definieert u net als bij een door de gebruiker gedefinieerde functie, namelijk door de namen van de argumenten tussen haakjes achter de procedurenaam te typen, gescheiden door het toepasselijke lijstscheidingsteken.

```
Sub CellenWissel (CelVerw1; CelVerw2)
    TijdWrd = CelVerw1.Waarde
    CelVerw1.Waarde = CelVerw2.Waarde
    CelVerw2.Waarde = TijdWrd
Einde Sub
```

  Welk teken als lijstscheidingsteken wordt gebruikt, is afhankelijk van de instellingen voor elk land. Voor het Nederlands is de puntkomma het scheidingsteken. Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over het lijstscheidingsteken.

- Visual Basic-programmacode. Deze programmacode bestaat uit de instructies die aangeven wat de procedure precies moet doen, de stappen of bewerkingen die achtereenvolgens moeten worden uitgevoerd.
- De resultaatwaarde. Een **Functie**-procedure heeft een resultaatwaarde, een **Sub**-procedure niet.

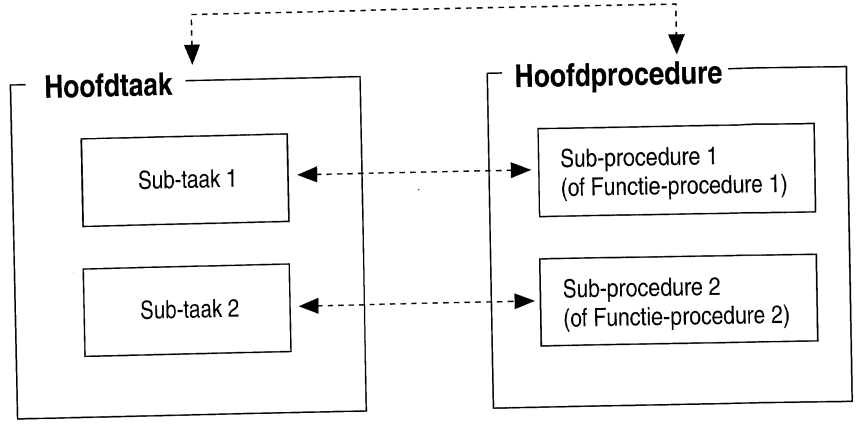
In latere hoofdstukken wordt uitgebreider ingegaan op Visual Basic-programmacode in procedures.

Visual Basic-programmacode bestaat uit een reeks sleutelwoorden die op een zinvolle manier zijn geordend. Als u informatie wilt hebben over een bepaald sleutelwoord, kunt u in de online Visual Basic Naslaggids zoeken naar de naam van het desbetreffende sleutelwoord. U kunt ook het gewenste sleutelwoord in een Visual Basic-module selecteren en vervolgens op F1 (in Windows) of op COMMAND+SHIFT+VRAAGTEKEN (op de Macintosh) drukken. Van de *Microsoft Excel Visual Basic Naslaggids* is ook een gedrukte uitgave verkrijgbaar bij Microsoft Press. Raadpleeg het hoofdstuk "Inleiding" van dit boek voor bestelinformatie.

Hoe procedures samenwerken om een complexe taak te vervullen

In hoofdstuk 1, "Terugkerende taken automatiseren", hebt u met de macrorecorder een aparte **Sub**-procedure gemaakt, die een eenvoudige taak automatisch uitvoerde. Naarmate u beter vertrouwd raakt met Visual Basic-procedures, zullen de taken die u door deze procedures wilt laten uitvoeren al gauw zo complex worden dat deze niet meer efficiënt door één enkele procedure kunnen worden afgehandeld.

Stel bijvoorbeeld dat u een procedure wilt maken die de winst na verkoop van een aandelenfonds berekent. Dit lijkt misschien een zelfstandige taak die door één enkele procedure kan worden afgehandeld, maar misschien komt u tot de conclusie dat deze taak uit een aantal deeltaken bestaat, zoals het opzetten van een nieuw werkblad, informatie opvragen van de gebruiker omtrent de kosten en het aantal betrokken aandelen en het invoeren van berekende waarden in het werkblad. Door een taak te analyseren, kunt u deze vaak onderverdelen in deeltaken, die stuk voor stuk kunnen worden afgehandeld door een aparte procedure.



Het probleem

De oplossing van Visual Basic

In het ideale geval wordt elke deeltaak door een aparte procedure afgehandeld. De programmeur moet zich tot doel stellen korte, eenvoudig te onderhouden procedures te ontwerpen die meerdere keren kunnen worden gebruikt en samenwerken met andere procedures. Korte procedures zijn gemakkelijker te ontwerpen, te implementeren en bij te werken, aangezien deze veel minder programmacode omvatten dan één enkele grote procedure. Tevens kunnen fouten makkelijker worden opgespoord. Door meerdere procedures te gebruiken, voorkomt u ook dat u herhaaldelijk dezelfde programmacode moet schrijven. U brengt gewoon de veelgebruikte programmacode onder in een aparte procedure, die vanuit andere procedures kan worden opgeroepen wanneer dat nodig is.

Een procedure kan een andere procedure oproepen of erdoor worden opgeroepen. *Oproepen* van een procedure betekent een procedure starten. Dankzij het feit dat een procedure andere procedures kan oproepen en dat meerdere procedures één en dezelfde procedure kunnen oproepen, zijn procedures in staat samen te werken.

De volgende procedure, genaamd VerkoopAandelen, roept bijvoorbeeld drie andere procedures op. De eerste opgeroepen procedure, BladInstellen, stelt een werkblad in volgens een bepaalde opmaak en met vaste koppen. De tweede procedure, AandelenInfo geeft dialoogvensters weer waarin de gebruiker informatie over aandelentransacties kan invoeren. De derde procedure, WaardenInvoegen, berekent provisies, belastingen en winst op basis van de informatie die de gebruiker heeft verstrekt en voegt deze waarden in het werkblad in.

```
Sub VerkoopAandelen()
    BladInstellen
    AandelenInfo
    WaardenInvoegen
Einde Sub
```

Zo nodig kunt u deeltaken weer verder onderverdelen in kleinere, makkelijker hanteerbare sub-deeltaken. Of misschien komt u wel tot de conclusie dat de oorspronkelijke hoofdtaak een deeltaak is van een nog grotere overkoepelende taak. De **Sub**-procedure VerkoopAandelen roept bijvoorbeeld drie andere **Sub**-procedures op, die op hun beurt weer andere procedures kunnen oproepen (of hierdoor worden opgeroepen), enzovoort. Het is zelfs mogelijk dat VerkoopAandelen wordt opgeroepen door een nog grotere procedure.

Hierna volgt een overzicht van alle **Sub**-procedures die door VerkoopAandelen worden opgeroepen. Misschien begrijpt u nog niet alle programmacode, maar u kunt in ieder geval zien hoe elke procedure met een afgebakende hoeveelheid werk bijdraagt aan de uitvoering van de overkoepelende taak. De procedures zijn ondergebracht in een module. Verderop in dit hoofdstuk wordt nader ingegaan op de ordening van procedures in een module.

```
'Begin module

'Constanten opgeven
Const TARWINSTBEL = 0,28
Const HOGEWINST = 1000000
Const OPSLAGTAR = 0,1

'Variabelen op module-niveau declareren
Dim OorsprPrijs
Dim AantalAandelen
Dim VerkoopPrijs

'Begin procedures
Sub VerkoopAandelen()
    BladInstellen           'Roept de procedure BladInstellen op
    AandelenInfo     'Roept de procedure AandelenInfo op
    WaardenInvoegen      'Roept de procedure WaardenInvoegen op
Einde Sub

'Maakt een nieuw werkblad gereed door formules in te voeren,
'kolombreedten aan te passen en rasterlijnen uit te schakelen.
Sub BladInstellen()
    Bereik("A2").Formule = "Oorspronkelijke prijs"
    Bereik("A3").Formule = "Aantal aandelen"
    Bereik("A4").Formule = "Verkoopprijs"
    Bereik("A5").Formule = "Provisie"
    Bereik("A6").Formule = "Belasting"
    Bereik("A7").Formule = "Nettowinst"
    Bereik ("A7").HeleKolom.AanSelectieAanpassen
    Bereik("B2:B7").GetalOpmaak = "F #.##0,00_);(F #.##0,00)"
    Bereik("A7:B7").Randen(xlBoven).Lijnopmaak = xlEffen
    Bereik ("B3").GetalOpmaak = "0"
    ActiefVenster.RasterlijnenWeergeven = Onwaar
Einde Sub

'Vraagt de gebruiker informatie over een transactie in te voeren.
'De antwoorden worden in variabelen op module-niveau opgeslagen.
Sub AandelenInfo()
    OorsprPrijs = InvoerVenster("Wat was de aankoopprijs?")
    AantalAandelen = InvoerVenster("Hoeveel aandelen hebt u gekocht?")
    VerkoopPrijs = InvoerVenster("Wat was de verkoopprijs?")
Einde Sub

'Informatie van gebruiker wordt in werkbladcellen ingevoegd
Sub WaardenInvoegen()
    Bereik("B2").Waarde = OorsprPrijs
    Bereik("B3").Waarde = AantalAandelen
    Bereik("B4").Waarde = VerkoopPrijs
```

```
    MijnProvisie = Provisie(AantalAandelen; VerkoopPrijs)
        Bereik("B5").Waarde = MijnProvisie
        'Procedure voor het opzoeken van belasting in tabel
        MijnBelast = Belast(AantalAandelen; OorsprPrijs; VerkoopPrijs)
        Bereik("B6").Waarde = MijnBelast
        BrutoWinst = AantalAandelen * (VerkoopPrijs - OorsprPrijs)
        NettoWinst = BrutoWinst - MijnProvisie - MijnBelast
        Bereik("B7").Waarde = NettoWinst
    Einde Sub

    'Makelaarsprovisie voor de transactie berekenen
    Functie Provisie(AandelenVerkocht; PrijsPerAandeel)
        TotaleVerkPrijs = AandelenVerkocht * PrijsPerAandeel
        Indien TotaleVerkPrijs <= 15000 Dan
            Provisie = 25 + 0,03 * AandelenVerkocht
        Anders
            Provisie = 25 + 0,03 * (0,9 * AandelenVerkocht)
        Einde Indien
    Einde Functie

    'Winstbelasting op kapitaal berekenen
    Functie Belast(Aandelen; PrijsInkoop; PrijsVerkoop)
        Winst = Aandelen * (PrijsVerkoop - PrijsInkoop)
        Indien Winst <= 0 Dan
            Belast = 0
        AndersIndien Winst > HOGEWINST Dan
            Belast = (TARWINSTBEL * Winst) + (Winst - HOGEWINST) * OPSLAGTAR
        Anders
            Belast = Winst * TARWINSTBEL
        Einde Indien
    Einde Functie

    'Einde module
```

In het voorgaande voorbeeld kunt u zien hoe een aantal **Sub**-procedures kan samenwerken door elkaar op te roepen. Op dezelfde manier kunnen ook **Functie**-procedures elkaar oproepen. Daarnaast is het mogelijk dat **Sub**-procedures **Functie**-procedures oproepen en andersom.

Bekijkt u bijvoorbeeld eens de volgende variant van de **Sub**-procedure VerkoopAandelen, die is geschreven als een **Functie**-procedure. Deze procedure roept twee andere **Functie**-procedures op om het eindresultaat te berekenen.

```
Functie VerkoopAandelen(OorsprPrijs; AantalAandelen; VerkoopPrijs)
    'BrutoWinst is een variabele die de brutowinst bevat
    BrutoWinst = AantalAandelen * (VerkoopPrijs - OorsprPrijs)
    'Roept Functie-procedure Provisie op om de provisie te berekenen
    MijnProvisie = Provisie(AantalAandelen; VerkoopPrijs)
    'Roept Functie-procedure Belast op om de belasting te berekenen
    MijnBelast = Belast(AantalAandelen; OorsprPrijs; VerkoopPrijs)
```

```
    'Geeft als resultaat de winst na aftrek van de onkosten
    VerkoopAandelen = BrutoWinst - MijnProvisie - MijnBelast
Einde Functie
.
.
.
```

Aansluitend zou dan de programmacode voor de **Functie**-procedures Provisie en Belast volgen, zoals deze in het vorige voorbeeld is opgenomen.

Opmerking Aangezien geen van deze **Functie**-procedures bewerkingen uitvoert die ongeoorloofde wijzigingen in een werkblad aanbrengen, kunt u de namen van deze functies ook in een werkbladcel invoeren als door de gebruiker gedefinieerde functie.

Procedures oproepen

De voorgaande gedeelten van dit hoofdstuk bevatten voorbeelden van procedures die andere procedures oproepen. Het oproepen van een procedure lijkt op het starten van een macro (zoals dit is besproken in hoofdstuk 1, "Terugkerende taken automatiseren"). U start procedures echter niet door op een knop te klikken of een opdracht in een menu te kiezen, maar de oproep vindt plaats binnen de Visual Basic-programmacode.

Telkens wanneer een procedure wordt opgeroepen, voert Visual Basic de instructies uit die tussen de instructies **Sub** en **Einde Sub** of **Functie** en **Einde Functie** voor de desbetreffende procedure staan. Als een procedure wordt opgeroepen waarvoor argumenten zijn gedefinieerd, vervangt Visual Basic de argumentnamen in de opgeroepen procedure door de overeenkomstige waarden die u in de argumentenlijst opgeeft. Stel dat u de procedure Pieptonen als volgt hebt gedefinieerd:

```
Sub Pieptonen(AantalTonen)
    Voor Teller = 1 Tot AantalTonen
        Pieptoon
    Volgende Teller
Einde Sub
```

In dat geval roept de volgende instructie (die u in een Visual Basic-module kunt opnemen) de procedure Pieptonen op en geeft de instructie het getal drie als argument door aan de procedure. *Doorgeven* van een waarde houdt in dat deze naar de procedure wordt gezonden.

```
Pieptonen 3
```

De procedure vervangt het argument AantalTonen in de argumentenlijst door het getal 3. Het uiteindelijke resultaat is dat uw computer drie pieptonen genereert.

Het gebruik van haakjes in een procedure

In de definitie van een **Sub**- of **Functie**-procedure wordt de procedurenaam altijd gevolgd door haakjes. Dit geldt ook als de procedure geen argumenten heeft, zoals in het volgende voorbeeld. (Visual Basic voegt de haakjes automatisch toe, zelfs als u deze niet typt).

```
Sub VerkoopAandelen()
.
.
.
Einde Sub
```

Wanneer u echter een **Sub**-procedure oproept, gebruikt u gewoonlijk geen haakjes. Stel dat de procedure BladInstellen zodanig is gewijzigd dat er drie argumenten nodig zijn om het werkblad op te maken, respectievelijk voor de koptekst, het aantal transacties dat u wilt invoeren en een datum. De volgende versie van de procedure VerkoopAandelen bevat een oproep van de herziene procedure BladInstellen.

```
Sub VerkoopAandelen()
    BladInstellen "Samenvatting"; 5; #21 dec 1994#
.
.
.
Einde Sub
```

Wanneer u een **Functie**-procedure oproept die argumenten heeft en de bijbehorende resultaatwaarde gebruikt, moeten de argumenten tussen haakjes staan, zoals is gebeurd in de zesde regel van de volgende procedure (`Winst = VerkoopAandelen(...)`). Als u geen gebruik maakt van de resultaatwaarde, kunt u de haakjes weglaten. In dit voorbeeld wordt gebruik gemaakt van de variant van de **Functie**-procedure VerkoopAandelen, die eerder in dit hoofdstuk is beschreven.

```
Sub WinstWeergeven()
    OudePrijs = InvoerVenster("Wat was de inkoopprijs?")
    Aandelen = InvoerVenster("Hoeveel aandelen hebt u gekocht?")
    NieuwePrijs = InvoerVenster("Wat was de verkoopprijs?")
    'Roept de Functie-procedure VerkoopAandelen op
    Winst = VerkoopAandelen(OudePrijs; Aandelen; NieuwePrijs)
    BerichtVenster "Uw winst bedraagt " & Winst
Einde Sub
```

Opmerking De regels ten aanzien van haakjes voor **Functie**-procedures gelden eveneens voor methoden. In hoofdstuk 5, "Werken met objecten in Visual Basic", wordt nader ingegaan op methoden.

U kunt de namen van ingebouwde functies of **Functie**-procedures direct in de argumentenlijst opnemen wanneer u procedures oproept. De volgende coderegel kan bijvoorbeeld gebruikt worden in plaats van de reeks coderegels in het vorige voorbeeld. (Het onderstrepingsteken is een vervolgteken, dat aangeeft dat de programmacode in zijn geheel op één regel moet worden getypt).

```
Winst = VerkoopAandelen(InvoerVenster("Wat was de inkoopprijs?"); _
  InvoerVenster("Hoeveel aandelen hebt u gekocht?"); _
  InvoerVenster("Wat was de verkoopprijs?"))
```

U kunt het oproepen van sommige procedures ook vereenvoudigen door benoemde argumenten te gebruiken. Hiermee kunt u specifieke argumenten aan een procedure doorgeven in elke gewenste volgorde. Zie "Instructies vereenvoudigen met benoemde argumenten" in hoofdstuk 6 voor meer informatie.

Procedures en modulen ordenen in werkmappen

Tot nu toe in dit hoofdstuk hebben procedures andere procedures alleen opgeroepen als deze zich alle binnen dezelfde Visual Basic-module bevonden. Dat is echter niet noodzakelijk. Een Visual Basic-module is gewoon een blad in een werkmap en u kunt een groot aantal modulen aan een werkmap toevoegen. En in elke module kunt u een groot aantal procedures opnemen.

Als een procedure een andere procedure oproept, zoekt Visual Basic de tweede procedure in eerste instantie in de module waarin de eerste procedure zich bevindt. Als de procedure niet in dezelfde module wordt gevonden, wordt in alle andere Visual Basic-modulen in de werkmap gezocht en vervolgens in andere werkmappen waarnaar een verwijzing is aangebracht. Visual Basic voert de tweede procedure uit zodra deze is gevonden, ongeacht in welke module de procedure is opgeslagen. (Zie "Procedures in een andere werkmap oproepen" verderop in dit hoofdstuk voor meer informatie over het aanbrengen van verwijzingen naar andere werkmappen).

Toch is het aan te bevelen om procedures te ordenen door soortgelijke procedures bij elkaar in dezelfde module op te slaan. U kunt bijvoorbeeld alle foutafhandelingsroutines samen in een bepaalde module onderbrengen, alle rekenkundige **Functie**-procedures in een andere module en alle automatische procedures in weer een andere module. (In hoofdstuk 9, "Foutafhandeling en foutwaarden", wordt nader ingegaan op foutafhandeling. Het maken van automatische procedures komt aan de orde in hoofdstuk 13, "Automatische procedures en invoegmacro's maken".) De volgende werkwijze geeft aan hoe u een nieuwe module in een werkmap invoegt.

▶ Een nieuwe Visual Basic-module in een werkmap invoegen

- Kies de opdracht **Macro** in het menu **Invoegen** en kies vervolgens **Module**.

Nadat u een module hebt ingevoegd, kunt u de nieuwe module een passende beschrijvende naam geven met de opdracht **Blad** in het menu **Bewerken**.

Ordening van een module

Een Visual Basic-module bevat gewoonlijk meerdere procedures. De volgorde van de procedures binnen een module is niet van belang. Aan het begin van de module, vóór de eerste procedure, kunt u een aantal regels met algemene gegevens opnemen. Dit gebied heet de *declaratie-sectie* van een module, aangezien u hier een aantal gebruikte variabelen en constanten opgeeft (declareert). Tevens kunt u hier een aantal Visual Basic-opties en andere specifieke informatie opgeven. (In hoofdstuk 6, "Werken met Visual Basic-programmacode in procedures", wordt nader ingegaan op het declareren van variabelen en constanten en het opgeven van Visual Basic-opties).

U kunt de elementen in de declaratie-sectie in elke gewenste volgorde opnemen, maar onder programmeurs is het gebruikelijk om eerst de opties op te geven, daarna variabelen te declareren en ten slotte constanten op te geven, zoals in het volgende voorbeeld.

```
'Begin module

'Declaraties

'Visual Basic-opties declareren
Optie Expliciet
Optie Vergelijken Binair

'Variabelen declareren
Dim OorsprPrijs
Dim AantalAandelen
Dim VerkoopPrijs

'Constanten declareren
Openbaar Const DOWJONES = 3500
Openbaar Const S_P500 = 290

'Einde Declaratie-sectie. Hierna volgen procedures.

Sub VerkoopAandelen()
    BladInstellen
    AandelenInfo
    WaardenInvoegen
Einde Sub

Sub BladInstellen()
.
.
.
```

Procedures in specifieke modulen oproepen

In de regel is het aan te raden elke procedure een unieke naam te geven, zodat er geen verwarring kan bestaan wanneer u de procedure oproept. Als het beslist nodig is twee procedures in twee verschillende modulen dezelfde naam te geven, moet u bij het oproepen van deze procedures tevens de modulenaam opgeven. Twee procedures binnen dezelfde module kunnen nooit dezelfde naam hebben.

Als u bij het oproepen van een procedure de modulenaam wilt opgeven, typt u de naam van de module tussen vierkante haken, gevolgd door een punt en de naam van de procedure. (De haakjes zijn niet verplicht als de modulenaam uit één woord bestaat en alleen letters en cijfers bevat). De volgende **Sub**-procedure roept bijvoorbeeld de **Sub**-procedure BladInstellen op die zich in de module Aangepaste Opmaak bevindt.

```
Sub VerkoopAandelen()
    [Aangepaste Opmaak].BladInstellen
    AandelenInfo
    WaardenInvoegen
Einde Sub
```

Als u een module-aanduiding opneemt in de Visual Basic-programmacode, wordt deze niet automatisch aangepast wanneer u een module een andere naam geeft. Als u de naam van een module wijzigt, moet u eveneens alle overeenkomstige module-aanduidingen wijzigen die voorafgaan aan procedurenamen. Anders kan Visual Basic deze procedures niet vinden.

Opmerking Als u de naam van een Visual Basic-module wilt wijzigen terwijl u in de module aan het werk bent, kiest u de opdracht **Blad** in het menu **Bewerken** en vervolgens de opdracht **Naam wijzigen**. Als u in een Microsoft Excel-werkblad werkt, bevindt de opdracht **Blad** zich in het menu **Opmaak** in plaats van het menu **Bewerken**.

Procedures beveiligen tegen oproepen van procedures uit andere modulen

U kunt voorkomen dat een procedure wordt opgeroepen vanuit een andere module. Hiertoe geeft u op dat het een persoonlijke procedure betreft. Dit is nuttig als een procedure niet bedoeld is voor toepassing buiten de eigen module of als de procedure dezelfde naam heeft als een procedure die vaak wordt opgeroepen vanuit andere modulen.

U geeft op dat een procedure persoonlijk is door de instructie **Functie** of **Sub** op de eerste regel van de procedure te laten voorafgaan door het sleutelwoord **Persnl**, zoals in het volgende voorbeeld:

```
Persnl Sub VerkoopAandelen()
```

Als u meer informatie over persoonlijke procedures wilt hebben, kunt u in de online Visual Basic Naslaggids zoeken naar *Functie* of *Sub* en de beschrijving van het sleutelwoord **Persnl** doorlezen.

Procedures in een andere werkmap oproepen

U kunt procedures in een andere werkmap alleen oproepen als u een verwijzing van Visual Basic naar de desbetreffende werkmap opgeeft. Als u eenmaal een verwijzing hebt opgegeven, zijn alle modulen in de werkmap (en derhalve ook alle procedures in deze modulen) beschikbaar voor Visual Basic. Vanaf dat moment kunt u de procedures in de andere werkmap net zo oproepen als de procedures in uw actieve werkmap.

De mogelijkheid om procedures in een andere werkmap op te roepen kan van pas komen wanneer u een werkmap of een andere invoegmacro krijgt met een reeks procedures die u wilt oproepen, maar liever niet naar uw eigen werkmap kopieert. U kunt bijvoorbeeld een verwijzing opgeven naar de invoegmacro Analysis ToolPak om toegang te krijgen tot de statistische, financiële en technische functies van deze toepassing. (Zie hoofdstuk 31, "Statistische analyse van gegevens", in het *Microsoft Excel Handboek* voor meer informatie over Analysis ToolPak).

▶ Een verwijzing naar een andere werkmap aanbrengen

1. Als u in een werkmap aan het werk bent en de werkmap waarnaar u een verwijzing wilt aanbrengen (een tweede werkmap) nog niet is opgeslagen, schakelt u over naar de tweede werkmap en slaat u deze op.
2. Activeer de eerste werkmap opnieuw, zorg dat een Visual Basic-module actief is en kies de opdracht **Verwijzingen** in het menu **Extra**.
3. Selecteer in het vak "Beschikbare verwijzingen" de naam van de tweede werkmap.

   Als de naam van de werkmap niet in de lijst voorkomt, kiest u de knop Bladeren om ernaar te zoeken op de schijf.
4. Schakel het aankruisvakje naast de naam van de tweede werkmap in.
5. Kies de knop OK.

U hoeft de module-aanduiding niet op te geven wanneer u een procedure oproept, tenzij een andere procedure in uw eigen werkmap of in de werkmap waarnaar u verwijst, dezelfde naam heeft.

U kunt de voorgaande werkwijze eveneens gebruiken om te verwijzen naar andere bureaublad-toepassingen, zoals Microsoft Word of Microsoft Project. Hierdoor krijgen uw procedures toegang tot de objecten, eigenschappen en methoden, enzovoort van deze toepassingen. Zie hoofdstuk 10, " Samenwerken met andere toepassingen", voor meer informatie over het werken met andere toepassingen. Zie hoofdstuk 5, "Werken met objecten in Visual Basic", voor meer informatie over objecten, eigenschappen en methoden.

Procedures in een bepaalde module in een bepaalde werkmap oproepen

Zodra er een verwijzing naar een werkmap is toegevoegd, kunt u de namen van een Visual Basic-module en een werkmap opgeven, als u precies weet waar de procedure zich bevindt. Hiervoor typt u de naam van de werkmap tussen vierkante haken, gevolgd door een punt, vervolgens de naam van de module tussen vierkante haken, gevolgd door een punt, en ten slotte de naam van de procedure. (De haakjes zijn niet verplicht als de naam van de werkmap of module uit één woord bestaat en alleen letters en cijfers bevat).

In het volgende voorbeeld voor Microsoft Excel voor Windows wordt de **Sub**-procedure BladInstellen opgeroepen, die zich bevindt in de module Aangepaste Opmaak in de werkmap VISUEEL.XLS.

```
Sub VerkoopAandelen()
    [VISUEEL.XLS].[Aangepaste Opmaak].BladInstellen
    AandelenInfo
    WaardenInvoegen
Einde Sub
```

In Microsoft Excel voor de Macintosh zou de oproep er als volgt kunnen uitzien:

```
Sub VerkoopAandelen()
    [Visuele effecten].[Aangepaste Opmaak].BladInstellen
    AandelenInfo
    WaardenInvoegen
Einde Sub
```

Als u de naam van de werkmap wijzigt, wordt de Visual Basic-programmacode niet automatisch bijgewerkt. Dat is een nadeel van het opgeven van namen van werkmappen. U hoeft de werkmap meestal alleen op te geven als er meerdere procedures met dezelfde naam bestaan.

Modulen beveiligen tegen oproepen door procedures in andere werkmappen

U kunt voorkomen dat de procedures in een Visual Basic-module worden opgeroepen door procedures in een andere werkmap. Hiervoor geeft u op dat het een persoonlijke module betreft. Dit kan nuttig zijn wanneer u een werkmap wilt doorgeven aan een collega, maar niet wilt dat deze procedures in bepaalde modulen oproept. Het is bijvoorbeeld mogelijk dat u een module hebt met procedures die van belang zijn voor een bepaalde werkmap, terwijl deze module niet bruikbaar is voor procedures in andere werkmappen.

U geeft op dat een procedure persoonlijk is door de instructie **Optie Persnl Module** op een willekeurige plaats in de declaratie-sectie van de module op te nemen. Voor meer informatie over de instructie **Optie Persnl Module** kunt u in de online Visual Basic Naslaggids zoeken naar *Optie Persnl Module*.

Tip Gebruikers van uw werkmap hoeven de persoonlijke modulen meestal helemaal niet te zien. Daarom is het misschien een goed idee deze modulen te verbergen. U verbergt een module door de opdracht **Blad Verbergen** in het menu **Bewerken** te kiezen.

Een verbroken koppeling tussen een knop of een grafisch object en een Visual Basic-module herstellen

In hoofdstuk 1, "Terugkerende taken automatiseren", hebt u gezien hoe u procedures kunt toewijzen aan knoppen en andere grafische objecten. Wanneer u een procedure toewijst aan een knop of een ander grafisch object, slaat Microsoft Excel een koppeling tussen het object en de toegewezen procedure op. Aangezien deze koppeling de naam bevat van de werkmap waarin de procedure is opgeslagen, kunnen koppelingen echter verloren gaan als u de naam van de werkmap buiten Microsoft Excel wijzigt (bijvoorbeeld met de MS-DOS®-opdracht **Rename**). Wanneer u op een object met een ongeldige koppeling klikt, verschijnt er een bericht op het scherm dat de procedure niet is gevonden.

Als u de naam wilt wijzigen van een werkmap die koppelingen heeft met objecten in andere bladen in andere werkmappen, opent u alle betrokken werkmappen en geeft u vervolgens een nieuwe naam op voor de eerste werkmap met de opdracht **Opslaan als** in het menu **Bestand**. Als de koppelingen met de werkmappen al verloren zijn gegaan, kunt u deze op de volgende manier herstellen.

▶ De koppeling tussen een grafisch object en een procedure herstellen

1. Schakel naar de werkmap die het object met de ongeldige koppeling bevat.
2. Kies de opdracht **Koppelingen** in het menu **Bewerken**.
3. Selecteer in het vak "Bronbestand" de vorige naam van de werkmap.
4. Kies de knop Wijzigen.
5. Het dialoogvenster **Koppelingen wijzigen** verschijnt. Typ of selecteer in het vak "Bestandsnaam" de nieuwe naam van de werkmap.
6. Kies in Microsoft Excel voor Windows de knop OK. Kies in Microsoft Excel voor de Macintosh de knop Wijzigen.
7. Kies de knop OK.

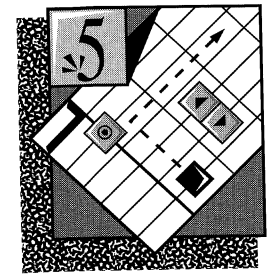
Alle koppelingen naar de vorige naam van de werkmap (niet alleen de koppeling die door het object wordt gebruikt) worden aangepast aan de nieuwe naam van de werkmap. Hierdoor worden echter niet de werkmapnamen gewijzigd die rechtstreeks in de programmacode zijn getypt, zoals `[MIJNMAP.XLS]` of `[Mijn werkmap]`. U moet de oude namen zelf aanpassen in de programmacode.

Opmerking Als u de naam wijzigt van een procedure of een Visual Basic-module met procedures die aan objecten zijn toegewezen, moet u de opdracht **Macro Toewijzen** in het menu **Extra** kiezen en handmatig de toewijzing aanpassen van elke procedure die u aan een object wilt toewijzen. Met andere woorden: de opdracht **Koppelingen** past gewijzigde werkmapnamen automatisch aan, maar gewijzigde module- of procedurenamen niet.

In het volgende hoofdstuk maakt u kennis met de objecten in Visual Basic en de Visual Basic-programmacode die wordt gebruikt om met deze objecten te werken.

HOOFDSTUK 5

Werken met objecten in Visual-Basic



Procedures die gebruik maken van Visual Basic kunnen meer dan alleen de bewerkingen herhalen die u hebt opgenomen. Met Visual Basic kunt u bijna alles besturen wat u maakt met Microsoft Excel - zelfs Microsoft Excel zelf.

Visual Basic zorgt voor deze besturing door objecten te definiëren waarop u in procedures bewerkingen kunt uitvoeren. In dit hoofdstuk maakt u kennis met objecten en leert u hoe u deze moet gebruiken om bewerkingen in Microsoft Excel automatisch te laten uitvoeren.

Inhoud

Inleiding in objecten

Een *object* is een element dat met Visual Basic wordt bestuurd. In Microsoft Excel gebruikt u objecten om bepaalde taken uit te voeren, net als in het gewone leven, waar u bijvoorbeeld een radio gebruikt om informatie en ontspanning te ontvangen. Zo is een Microsoft Excel-werkblad een object dat u gebruikt om gegevens te berekenen en weer te geven.

Net als radio's en andere objecten, heeft elk object in Microsoft Excel kenmerken die dat object bruikbaar maken. Deze kenmerken of *eigenschappen* bepalen de vorm of het gedrag van een object. Zo heeft een radio een eigenschap die bepaalt welk station u hoort: de eigenschap "frequentie".

In Microsoft Excel hebben werkbladen de eigenschap **Zichtbaar**, waarmee u kunt bepalen of een werkblad zichtbaar is. **Zichtbaar** is slechts één van de eigenschappen van een werkblad-object. U kunt de waarden van de eigenschappen van de objecten in Visual Basic-programmacode instellen of opvragen, zodat het gedrag van de objecten verandert. Zie "Objecten besturen met eigenschappen" verderop in dit hoofdstuk voor meer informatie over eigenschappen.

Behalve eigenschappen hebben objecten ook *methoden*: acties die objecten kunnen uitvoeren. Zo zou je kunnen zeggen dat een radio een methode "LuiderZetten" heeft, die het geluidsvolume doet toenemen. Werkbladen hebben bijvoorbeeld een methode **Berekenen**, waarmee u het gehele werkblad opnieuw kunt laten berekenen. **Berekenen** is slechts één van de methoden van een werkblad-object. U gebruikt methoden in Visual Basic-programmacode om objecten de gewenste acties te laten uitvoeren. Zie "Acties uitvoeren met methoden" verderop in dit hoofdstuk voor meer informatie over methoden.

Objecten gebruiken

Als u Visual Basic-programmacode schrijft, moet u eerst bepalen welke objecten u nodig hebt om de gewenste taak uit te voeren. Deze objecten gebruikt u dan in de programmacode. De programmacode die u schrijft doet meestal een of meer van de volgende drie dingen:

- De programmacode wijzigt de toestand van een object door de waarde van een van de eigenschappen van het object in te stellen.
- De programmacode onderzoekt de toestand van een object door de waarde van een van de eigenschappen van het object op te vragen.
- De programmacode laat het object een van zijn mogelijke taken uitvoeren door gebruik te maken van een van de methoden van het object.

Stel dat u wilt bepalen of een bereik in een werkblad leeg is. Als het bereik niet leeg is, wilt u de inhoud ervan wissen. Zodra het bereik leeg is, ofwel omdat het al leeg was ofwel omdat u de inhoud ervan hebt gewist, wilt u een formule aan het bereik toewijzen. De Visual Basic-procedure die u schrijft moet de volgende dingen doen:

- Gebruik maken van een bereikobject om het bereik aan te duiden waarop u een bewerking wilt uitvoeren
- De waarde van de eigenschap **Waarde** van het bereik opvragen om te bepalen of het bereik leeg is
- De methode **Wissen** van het bereik gebruiken om de inhoud van alle cellen te wissen als het bereik niet leeg is

- De eigenschap **Formule** van het bereik instellen om een formule aan het bereik toe te wijzen

Naarmate u meer te weten komt over Visual Basic, zult u merken dat Microsoft Excel veel objecten kent die u in programmacode kunt gebruiken om uw werk automatisch te laten uitvoeren, zoals:

- Werkmappen
- Werkbladen
- Bereiken
- Grafieken

Hoewel u zou kunnen verwachten dat een cel in een werkblad een object is, is dat in Microsoft Excel niet het geval. Als u met Visual Basic een bewerking wilt uitvoeren op een of meer cellen, moet u het bereikobject gebruiken. Voor het bewerken van een enkele cel gebruikt u een bereikobject dat alleen die bewuste cel bevat. Zie "Objecten, eigenschappen en methoden gebruiken in programmacode" verderop in dit hoofdstuk voor meer informatie over het bereikobject.

Tip Als u meer wilt weten over de manier waarop objecten kunnen helpen om taken te automatiseren, kunt u met de macrorecorder een taak opnemen. Aan de Visual Basic-programmacode die de macrorecorder maakt kunt u zien hoe de opgenomen macro objecten gebruikt.

Zoek in de online Visual Basic Naslaggids naar *Objecten, een beter begrip* voor meer inleidende informatie over object-gericht programmeren en objecten in Microsoft Excel.

Objecten besturen met eigenschappen

Procedures in Visual Basic worden geschreven om de besturing van objecten in Microsoft Excel te automatiseren. Stel dat u de breedte van een werkbladkolom wilt wijzigen, dan moet uw procedure het bereikobject wijzigen. In Visual Basic hebben de meeste objecten eigenschappen die hun vorm en gedrag bepalen.

U wijzigt objecten door hun eigenschappen te wijzigen. Neem bijvoorbeeld weer de radio. Een eigenschap van een radio die verandert als u deze gebruikt, is het volume. In Visual Basic zou men zeggen dat een radio een eigenschap "Volume" heeft, die gewijzigd wordt door de waarde ervan te wijzigen. Stel dat u het volume van de radio kunt instellen van 0 tot 10. Als u de radio kon besturen met Visual Basic, zou u een procedure kunnen schrijven die de waarde van de eigenschap "Volume" wijzigt van 3 naar 5 om de radio luider te laten spelen.

Alle objecten in Microsoft Excel hebben eigenschappen. Zo beschrijft de volgende tabel in het kort enkele eigenschappen van het bereikobject.

Eigenschap	Beschrijving
Kolom	Geeft de eerste kolom van het eerste gebied in het bereik
Formule	Geeft de formule van het bereik, genoteerd in de A1-verwijzingsstijl
Hoogte	Geeft de hoogte van een bereik, in punten (1/72 inch)
TekstTerugloop	Bepaalt of tekst binnen het bereik terugloopt

U kunt de waarden van deze eigenschappen wijzigen om de vorm of het gedrag van een bereikobject te wijzigen. Stel dat u alle kolommen in een bereik breder wilt maken, zodat de informatie in het bereik beter zichtbaar is. Een deel van de procedure kan de waarde van de eigenschap **Breedte** van het bereik overeenkomstig wijzigen. Als sommige cellen in het bereik een iets langere tekst bevatten, kan een ander deel van de procedure de waarde van de eigenschap **TekstTerugloop** wijzigen, zodat in de cellen meerdere regels tekst worden weergegeven.

Bij het gebruik van objecten in procedures bestaat een deel van het werk dus uit het instellen van de waarden van hun eigenschappen. U kunt ook de waarde van een eigenschap als resultaat laten geven. U wilt bijvoorbeeld een berekende waarde in een cel controleren om te beslissen of er nog aanvullende programmacode moet worden gestart. Uw procedure kan de eigenschap **Waarde** van de cel opvragen en het resultaat gebruiken als basis voor de beslissing of de aanvullende programmacode moet worden gestart.

Naar eigenschappen verwijzen

Voordat u eigenschappen kunt instellen of gebruiken, moet u weten hoe u naar een eigenschap verwijst in Visual Basic-programmacode. Als u naar een eigenschap verwijst, noemt u eerst het object waarvan u de eigenschap wilt instellen of opvragen, gevolgd door een punt en vervolgens de naam van de eigenschap. Hier volgt een voorbeeld van een verwijzing naar de eigenschap **Kolombreedte** van een bereikobject. De methode **Cellen** geeft een bereikobject als resultaat.

```
Cellen.KolomBreedte = 16
```

Het object en de eigenschap worden gescheiden door een punt. Door deze punt weet Visual Basic waar de naam van het object eindigt en de naam van de eigenschap begint. (Zie "Objecten, eigenschappen en methoden gebruiken in programmacode" verderop in dit hoofdstuk voor meer informatie over de methode **Cellen** en het bereikobject).

Waarden van eigenschappen gebruiken in procedures

Als u eigenschappen gebruikt in Visual Basic, kunt u:

- De waarde van een eigenschap instellen
- De waarde van een eigenschap opvragen

Bij de meeste eigenschappen kunt u de waarden ofwel instellen ofwel opvragen, afhankelijk van wat nodig is voor de procedure. Eigenschappen waarvan u de waarde zowel kunt instellen als kunt opvragen, worden lezen-schrijven-eigenschappen genoemd. Bij andere eigenschappen kunt u alleen de waarde opvragen. Deze eigenschappen worden alleen-lezen-eigenschappen genoemd. Als u wilt weten of een bepaalde eigenschap Lezen-Schrijven of Alleen-lezen is, kunt u in de online Visual Basic Naslaggids zoeken naar de naam van de eigenschap.

Soorten waarden van eigenschappen

In de meeste gevallen behoort de waarde van een eigenschap tot een van de volgende drie typen:

- Een numerieke waarde, bijvoorbeeld **14** voor de instelling van de eigenschap **Rijhoogte** van een cel
- Een tekenreeks, bijvoorbeeld "Jaartotalen" voor de instelling van de eigenschap **Waarde** van een cel
- Een waarde **Waar** of **Onwaar**, bijvoorbeeld **Waar** voor de instelling van de eigenschap **Vet** van een lettertype-object

Het is nuttig om te weten welk soort waarde van toepassing is voor een bepaalde eigenschap, zodat u niet per ongeluk probeert een eigenschap in te stellen op een waarde die niet geaccepteerd wordt door die eigenschap. Als u bijvoorbeeld de eigenschappen van een lettertype-object instelt, moet u weten dat voor de eigenschap **Naam** een tekenreeks nodig is om het lettertype aan te duiden, terwijl de eigenschap **Vet** ofwel de waarde **Waar** ofwel de waarde **Onwaar** nodig heeft.

Er zijn nog andere soorten waarden van eigenschappen. Als u meer wilt weten over de soorten waarden die u voor een bepaalde eigenschap kunt opgeven, kunt u in de online Visual Basic Naslaggids zoeken naar de naam van die eigenschap.

De waarde van een eigenschap instellen

U stelt de waarde van een eigenschap in als u de vorm of het gedrag van een object wilt wijzigen. U wijzigt bijvoorbeeld de waarde van de eigenschap **Waarde** van een cel om de inhoud van die cel in het werkblad te wijzigen. In dat geval wijzigt u zowel de manier waarop de cel verschijnt in het werkblad als de informatie die erin staat. Een ander voorbeeld is het instellen van de waarde van de eigenschap **TekstTerugloop** van een cel. In dat geval schakelt het gedrag van de cel heen en weer tussen wel en niet teruglopen van de tekst.

Gebruik de volgende syntaxis om de waarde van een eigenschap in te stellen:

object.eigenschap = expressie

waarbij *object* een verwijzing is naar een object (mogelijk inclusief zijn containers), *eigenschap* de naam van de eigenschap is, en *expressie* de waarde is waarop u de eigenschap wilt instellen. Zie "Objecten die andere objecten bevatten" voor meer informatie over containers.

De volgende instructies laten zien hoe eigenschappen worden ingesteld.

```
ActieveCel.Rijhoogte = 14
ActieveCel.Waarde = "Jaartotalen"
ActieveCel.Geblokkeerd = Onwaar
```

De waarde van een eigenschap opvragen

U vraagt de waarde van een eigenschap op als u de toestand van een object wilt bepalen voordat een procedure verdere acties uitvoert (zoals het toewijzen van de waarde van de eigenschap aan een ander object). U kunt bijvoorbeeld de eigenschap **Waarde** van een cel opvragen om de inhoud van de cel te bepalen voordat u programmacode start die deze waarde zou kunnen wijzigen.

Meestal gebruikt u de volgende syntaxis om de waarde van een eigenschap op te vragen:

variabele = object.eigenschap

waarbij *variabele* verwijst naar een variabele of een andere eigenschap waarin de gegeven waarde van de eigenschap wordt opgeslagen, *object* verwijst naar een object (mogelijk inclusief zijn containers), en *eigenschap* de naam is van een eigenschap van het object.

U kunt ook de waarde van een eigenschap opvragen als onderdeel van een meer complexe expressie, zonder die waarde aan een variabele toe te wijzen. U kunt bijvoorbeeld de positie van de rechterrand van een knopobject berekenen door de waarde van de eigenschap **Links** op te tellen bij de waarde van de eigenschap **Breedte**.

```
Indien MijnKnop.Links + MijnKnop.Breedte > 350 Dan
    MijnKnop.Breedte = MijnKnop.Breedte / 2
Einde Indien
```

In dit voorbeeld vraagt de eerste instructie de eigenschappen **Links** en **Breedte** op van een knopobject genaamd MijnKnop. De instructie telt de opgehaalde waarden van de eigenschappen op en gebruikt het resultaat om te bepalen of de tweede instructie moet worden uitgevoerd.

U kunt de waarde van een eigenschap ook opvragen om de waarde van een andere eigenschap in te stellen. U wilt bijvoorbeeld de eigenschap **Formule** van een bepaalde cel laten overeenkomen met de eigenschap **Formule** van een andere cel.

```
ActiefBlad.Cellen(1; 1).Formule = ActiefBlad.Cellen(1; 3).Formule
```

Deze instructie geeft als resultaat de waarde van de eigenschap **Formule** van cel C1 om vervolgens de eigenschap **Formule** van cel A1 op dezelfde waarde in te stellen.

Algemene eigenschappen

Sommige eigenschappen komen veel voor of worden door een groot aantal objecten in Microsoft Excel gemeenschappelijk gebruikt. Het is belangrijk om deze eigenschappen te kennen, als u Visual Basic wilt leren gebruiken. De volgende tabel toont een aantal van deze eigenschappen en beschrijft kort hoe ze objecten beïnvloeden in Microsoft Excel.

Eigenschap	Beschrijving
ActieveCel	De actieve cel van het werkvenster
ActiefBlad	Het actieve blad van de actieve werkmap
ActieveWerkmap	De actieve werkmap in Microsoft Excel
Vet of **Cursief**	De opmaakkenmerken van de tekst in een lettertype-object, resp. vet of cursief

Eigenschap	Beschrijving
Kolom of **Rij**	Een getal dat de eerste kolom of rij van het eerste gebied van een bereik aangeeft
KolomBreedte	De breedte van alle kolommen in het aangegeven bereik
Hoogte of **Breedte**	De hoogte of breedte van een object, in punten. Deze eigenschap geldt voor veel verschillende objecten in Microsoft Excel
RijHoogte	De hoogte van alle rijen in het aangegeven bereik, in punten
Selectie	Het huidige geselecteerde object in het werkvenster van het toepassingsobject, of de huidige selectie in het opgegeven vensterobject
Waarde	De waarde van een cel

Acties uitvoeren met methoden

Behalve eigenschappen hebben objecten ook methoden. Een telefoontoestel gaat bijvoorbeeld rinkelen als er opgebeld wordt; men zou kunnen zeggen dat telefoontoestellen een methode "Bellen" hebben. In Microsoft Excel is het mogelijk de inhoud van een cellenbereik te wissen. Bereikobjecten hebben dus een methode **Wissen**. Met sommige methoden kunt u objecten aanduiden die u kunt gebruiken in programmacode. U kunt bijvoorbeeld de methode **Bereik** gebruiken om een bereik aan te duiden dat u in een procedure wilt gebruiken.

Methoden zijn een onderdeel van objecten, net als eigenschappen. Het verschil tussen methoden en eigenschappen is dat eigenschappen waarden hebben die u kunt instellen of opvragen, terwijl methoden bewerkingen zijn die u een object laat uitvoeren. Bovendien kunt u aan de meeste eigenschappen één enkele waarde toewijzen, terwijl methoden één of meer argumenten kunnen hebben.

Vergelijk de tabel met eigenschappen van bereikobjecten in "Objecten besturen met eigenschappen" eerder in dit hoofdstuk met de volgende tabel, die in het kort enkele methoden beschrijft van een bereikobject.

Methode	Beschrijving
Berekenen	Berekent een opgegeven cellenbereik in een blad
Wissen	Wist de inhoud van het gehele bereik
Kopiëren	Kopieert het bereik naar het Klembord
Uitvullen	Verdeelt de tekst gelijkmatig over het bereik
Tabel	Maakt een gegevenstabel gebaseerd op invoerwaarden en formules die u in een werkblad vastlegt

Methoden kunnen de waarden van eigenschappen beïnvloeden. In het voorbeeld van de radio verandert de methode "LuiderZetten" de eigenschap "Volume". Op dezelfde manier heeft in Microsoft Excel elke cel van een bereik een eigenschap **Waarde**. Het bereikobject dat die cel bevat heeft een methode **Wissen**. Als in een procedure de methode **Wissen** wordt gebruikt, verandert de eigenschap **Waarde** van alle cellen van het bereik.

Methoden in programmacode gebruiken

Bij het gebruik van de methoden van een object in een procedure is de manier waarop de programmacode wordt geschreven afhankelijk van de vraag of de methode argumenten kan hebben. Als de methode geen argumenten kan hebben, gebruikt u de volgende syntaxis bij het schrijven van de programmacode:

object.methode

waarbij *object* de naam van het object is (mogelijk inclusief de namen van zijn containers) en *methode* de naam van de methode is. De methode **Uitvullen** van een bereikobject heeft bijvoorbeeld geen argumenten. Om de cellen in een bereikobject genaamd Prijs uit te vullen, schrijft u:

```
Prijs.Uitvullen
```

Als een methode wel argumenten heeft, is de manier waarop de programmacode wordt geschreven afhankelijk van de vraag of u de waarde waarin de methode resulteert, wilt opslaan. Als u de resultaatwaarde van een methode niet wilt opslaan, worden de argumenten zonder haakjes geschreven:

object.methode argumenten

waarbij *object* en *methode* de namen voorstellen van het object en de bijbehorende methode. Als de methode meer dan één argument heeft, worden de argumenten gescheiden door een lijstscheidingsteken en een spatie. De methode **Tabel** van een bereikobject kan bijvoorbeeld twee argumenten hebben. Als u een gegevenstabel wilt maken van een bereik genaamd Prijs, schrijft u:

```
Bereik("Prijs").Tabel Cellen(2; 1); Cellen(3; 1)
```

De resultaatwaarde van de methode **Tabel** wordt niet opgeslagen, dus worden de twee argumenten van de methode **Tabel** zonder haakjes geschreven.

Opmerking Het decimaalscheidingsteken, het lijstscheidingsteken, het valutasymbool en de datumnotaties die u gebruikt bij het schrijven of bewerken van programmacode worden bepaald door de instelling van de optie "Land/Taal" in het tabblad Module van het dialoogvenster **Opties** in het menu **Extra**. Het decimaalscheidingsteken, het lijstscheidingsteken, het valutasymbool en de datumnotaties die de gebruikers zien als de programmacode wordt uitgevoerd, worden bepaald door de landinstelling van het besturingssysteem.

De voorbeelden van programmacode in dit boek gaan uit van de landinstelling voor Nederland. Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over het schrijven van toepassingen voor gebruik in andere landen.

Als u de resultaatwaarde van een methode echter wel wilt opslaan, moet u de argumenten tussen haakjes zetten. De methode **SpellingControleren** van een werkblad resulteert bijvoorbeeld in een waarde (**Waar** of **Onwaar**) die aangeeft of de tekst correct is gespeld.

```
DitWoord = ActiefBlad.SpellingControleren(HoofdlettersNegeren:=Waar)
```

De resultaatwaarde van de methode **SpellingControleren** wordt opgeslagen in de variabele `DitWoord`. Daarom moet het argument van de methode tussen haakjes staan.

Benoemde argumenten gebruiken

Bij de meeste methoden zijn de volgende twee vormen van argumenten mogelijk:

- Conventioneel
- Benoemd

In het eerder gebruikte voorbeeld met de methode **Tabel** worden de argumenten als conventionele argumenten doorgegeven aan de methode. *Conventionele argumenten* staan in een bepaalde volgorde, die bepaalt hoe Visual Basic waarden van de argumenten interpreteert. U kunt in plaats daarvan echter ook *benoemde argumenten* gebruiken. Benoemde argumenten worden herkend aan hun naam in plaats van aan hun positie in de syntaxis. Als u bijvoorbeeld de methode **Tabel** wilt gebruiken met benoemde argumenten, kunt u schrijven:

```
Prijs.Tabel KolomInvoer := Cellen(3; 1); RijInvoer := Cellen(2; 1)
```

Benoemde argumenten zijn vaak beter te begrijpen bij het lezen van programmacode, en ook gemakkelijker te gebruiken. Zie "Instructies vereenvoudigen met benoemde argumenten" in hoofdstuk 6 voor meer informatie over het verschil tussen conventionele en benoemde argumenten.

Werken met een verzameling objecten

In Microsoft Excel is een *verzameling* een reeks verwante objecten. De verzameling Werkbladen bevat bijvoorbeeld alle werkbladen in een bepaalde werkmap. De verzameling Werkbalken bevat alle werkbalken van het toepassingsobject Microsoft Excel. Elk object in een verzameling wordt een *element* van die verzameling genoemd.

Verzamelingen zijn zelf ook objecten. Omdat verzamelingen objecten zijn, hebben ze hun eigen eigenschappen en methoden. U kunt deze eigenschappen en methoden gebruiken om zowel de afzonderlijke elementen van de verzameling als alle objecten in de verzameling te besturen.

Methoden die objecten als resultaat geven

Sommige objecten bezitten speciale methoden die resulteren in een verwijzing naar een ander object, gewoonlijk een verzameling. Een werkmapobject heeft bijvoorbeeld een methode **Werkbladen**, waarmee u een verwijzing kunt ophalen naar de volledige verzameling werkbladen in de werkmap, of naar één bepaald werkblad in de verzameling Werkbladen.

Opmerking De methode **Cellen**, die resulteert in een bereikobject, is een voorbeeld van een methode die als resultaat een object geeft dat geen verzameling is. Zie "Objecten, eigenschappen en methoden gebruiken in programmacode" verderop in dit hoofdstuk voor meer informatie over de methode **Cellen**.

Een afzonderlijk object uit een verzameling opvragen

Wanneer u wilt werken met één element uit een verzameling, moet u opgeven welk element u bedoelt. Als u bijvoorbeeld een afzonderlijk werkblad in een werkmap wilt aanduiden, kunt u de methode **Werkbladen** gebruiken om een bepaald blad uit de verzameling op te geven.

```
Werkbladen("MijnBlad")
```

Zoals u ziet, heeft de methode **Werkbladen** een argument nodig dat verwijst naar een bepaald werkblad. In dit geval is het argument de naam van het werkblad MijnBlad. U kunt ook een cijfer gebruiken om een element uit een verzameling aan te duiden.

```
Werkbladen(3)
```

Een cijfer dat wordt gebruikt om een element uit een verzameling aan te duiden, wordt een *index* genoemd. De eerste index in een verzameling is 1. Meestal bestaat de index uit één cijfer.

Alle objecten uit een verzameling opvragen

Het kan zijn dat u in een procedure wijzigingen wilt aanbrengen die alle objecten in de verzameling beïnvloeden. Als u bijvoorbeeld alle figuurobjecten van Blad1 wilt verwijderen, kunt u de volgende instructie gebruiken.

```
Werkbladen("Blad1").ObjectenTekenen.Verwijderen
```

Deze instructie gebruikt de methode **Werkbladen** om een verwijzing op te halen naar één enkel werkblad. Omdat er echter geen index is opgegeven voor de verzameling Figuren, worden alle elementen uit de verzameling verwijderd. (Als u de methode **Werkbladen** gebruikt zonder een index op te geven, resulteert dit in een verwijzing naar de gehele verzameling in plaats van naar een enkel element).

U kunt ook een lus schrijven met programmacode die wordt uitgevoerd op elk object in de verzameling. Als u bijvoorbeeld alle werkmappen in de verzameling Werkmappen wilt sluiten, kunt u de volgende programmacode gebruiken.

```
Voor Elke Map In Werkmappen
    Map.Sluiten
Volgende
```

In deze programmacode is Map een variabele die verwijst naar elk van de werkmappen in de verzameling Werkmappen. De programmacode binnen de lus past de methode **Sluiten** toe op het object aangeduid door de variabele, om elke werkmap in de verzameling te sluiten.

Zie "Dezelfde programmacode meerdere keren uitvoeren (lussen)" in hoofdstuk 7 voor meer informatie over het schrijven van lussen.

Een nieuw object maken in een verzameling

Als u met verzamelingen werkt, werkt u meestal met de objecten die reeds in de verzameling aanwezig zijn. Soms wilt u echter nieuwe objecten aan een verzameling toevoegen. Dit kunt u in de meeste verzamelingen in Microsoft Excel doen met de methode **Toevoegen**.

De methode **Toevoegen** kan de begininstellingen van de eigenschappen van het nieuwe object als argumenten hebben. Als u bijvoorbeeld een nieuw scenario genaamd Waarschijnlijk wilt toevoegen aan het actieve werkblad, gebruikt u de volgende instructie.

```
ActiefBlad.Scenarios.Toevoegen _
    naam = "Waarschijnlijk" _
    VeranderendeCellen = Bereik("A1:C5") _
Opmerking:= "Vermoedelijk scenario voor dit jaar"
```

Deze instructie stelt de beginwaarde van de eigenschappen **Naam** en **Opmerkingen** van het nieuwe werkblad in, alsook de scenariowaarden.

Zie het onderwerp *Scenario's* in de online Visual Basic Naslaggids voor meer informatie over scenario's.

Algemene verzamelingen

In Microsoft Excel zult u sommige verzamelingen vaker tegenkomen en gebruiken dan andere. De volgende tabel geeft een overzicht van de meest gebruikte verzamelingen van Microsoft Excel en beschrijft deze in het kort.

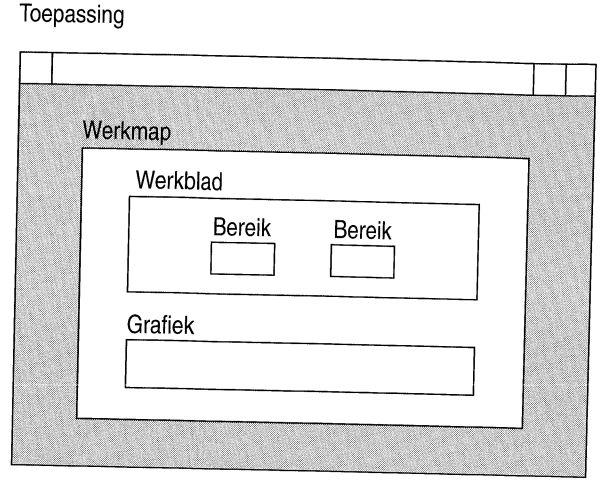
Verzameling	Beschrijving
Bladen	Bevat alle bladobjecten van een werkmap, waaronder objecten van het type werkblad, grafiek, Visual Basic-module en dialoogblad
Werkbladen	Bevat alle werkbladobjecten van een werkmap
Grafieken	Bevat alle grafiekbladen van een werkmap (Ingesloten grafiekobjecten worden omvat door de verzameling Grafiekobjecten)
Werkmappen	Bevat alle geopende werkmappen in Microsoft Excel

Zie het onderwerp *Objecten* in de online Visual Basic Naslaggids voor meer informatie over deze en andere verzamelingen in Microsoft Excel.

Objecten die andere objecten bevatten

Sommige objecten in Microsoft Excel bevatten andere objecten. Zo bevat een werkmap één of meer werkbladen. Een werkblad bevat rijen, kolommen en bereiken. De buitenste "container" van objecten in Microsoft Excel is de toepassing zelf (toepassingsobject genaamd). Alle andere objecten die u gebruikt, worden omvat door de toepassing. De toepassing bevat algemene objecten, zoals de menubalk, naast afzonderlijke werkmappen.

De verhouding van objecten tot hun container wordt in de volgende figuur geïllustreerd. De toepassing vormt een container van alle geopende werkmappen. De werkmap is een container die een werkblad en een grafiek bevat.



Werkbladen zijn containers die de afzonderlijke bereiken, rijen en kolommen bevatten. Bij werkbladen kunt u trouwens goed zien hoe objecten in meer dan één container tegelijk aanwezig kunnen zijn. Neemt u bijvoorbeeld een cellenbereik in een werkblad. Het werkblad waarin dat bereik zich bevindt, bevat eveneens de cellen van het bereik, en het bereik kan in een rij of in een kolom staan, die op zich ook containers zijn.

Containers gebruiken in procedures

Het feit dat objecten andere objecten kunnen bevatten, heeft als voordeel dat u in een procedure kunt verwijzen naar de container om aan te duiden welk object u wilt gebruiken. Stel dat een werkmap twee verschillende werkbladen bevat die allebei een cel bevatten die wordt aangeduid als Cellen(1; 1). U wilt een procedure schrijven die informatie plaatst in de eerste cel van het ene werkblad, maar niet in die van het andere blad. U kunt precies aangeven welke cel u wilt gebruiken door te verwijzen naar het werkblad dat het bereik bevat.

```
ActiefBlad.Cellen(1; 1).Waarde = 42
```

In dit codevoorbeeld bevat de verwijzing naar het object (een bereik) een verwijzing naar het werkblad dat het bereik bevat. Deze verwijzing naar de container geeft aan dat u de waarde van de eerste cel van het actieve werkblad wilt wijzigen, en niet die van een ander werkblad in de werkmap.

Als een verwijzing naar een object verwijzingen bevat naar een of meer containers van het object, moet Visual Basic een onderscheid kunnen maken tussen de verschillende delen van de informatie. Zo moet u in het vorige codevoorbeeld duidelijk maken dat `ActiefBlad` een verwijzing is naar een werkblad en dat `Cellen(1; 1)` een verwijzing is naar een bereikobject op het werkblad.

Om aan te geven waar de ene verwijzing eindigt en de volgende begint, scheidt u deze door middel van een punt. Als u verwijzingen naar objecten niet onderscheidt van verwijzingen naar hun containers, kan Visual Basic de programmacode niet correct lezen. Als er bijvoorbeeld geen punt staat tussen `ActiefBlad` en `Cellen(1; 1)`, ziet het vorige voorbeeld er zo uit:

```
ActiefBladCellen(1; 1).Waarde = 42
```

Visual Basic zal `ActiefBladCellen(1; 1)` verkeerd interpreteren als een element uit een variabelenmatrix. Omdat variabelenmatrices geen eigenschap **Waarde** hebben, veroorzaakt deze instructie een fout.

Het Objectenoverzicht gebruiken

Stel dat u tijdens het schrijven van Visual Basic-procedures wilt zien welke objecten beschikbaar zijn in de actieve werkmap, inclusief de namen van de procedures die u al hebt geschreven. U kunt deze informatie verkrijgen door gebruik te maken van het Objectenoverzicht.

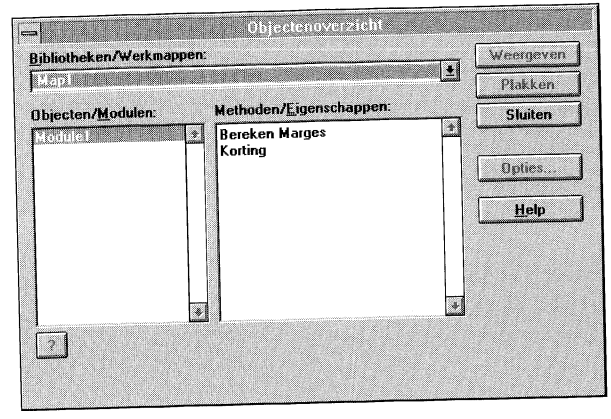
Het Objectenoverzicht heeft drie belangrijke functies:

- Het biedt een snelle manier om uw weg te vinden binnen de programmacode.
- Het laat zien welke objecten beschikbaar zijn voor uw procedures, waarbij u ook de eigenschappen en methoden van die objecten kunt bekijken.
- Het biedt een manier om codesjablonen in een module te plakken. U kunt de geplakte programmacode bewerken, hetgeen het schrijven van procedures versnelt en de kans op typefouten verkleint.

▶ Het Objectenoverzicht weergeven

- Kies de opdracht **Objectenoverzicht** in het menu **Beeld**.

  U kunt ook klikken op de knop Objectenoverzicht op de Visual Basic-werkbalk.



De informatie in het Objectenoverzicht

Het dialoogvenster **Objectenoverzicht** toont één van de volgende twee soorten informatie.

- De namen van alle objecten in een bibliotheek, alsook alle eigenschappen en methoden van elk object
- De namen van alle Visual Basic-modulen in een werkmap, alsook de namen van alle procedures in elke module

Het Objectenoverzicht bevat een vak met de namen van alle bibliotheken en werkmappen waarnaar wordt verwezen in de werkmap met de module die u aan het maken bent. Een *bibliotheek* is een bestand of een deel van een toepassing waar Visual Basic informatie kan vinden over objecten. Bibliotheken en werkmappen geven aan welke objecten u kunt gebruiken in uw procedures. Zie "Verwijzingen naar toepassingen en werkmappen toevoegen" in hoofdstuk 10 voor meer informatie over verwijzingen naar bibliotheken en werkmappen.

De lijst van bibliotheken/werkmappen bevat:

- Microsoft Excel, dat een bibliotheek bevat van objecten die specifiek zijn voor Microsoft Excel
- VBA (Visual Basic for Applications), dat functies en andere taalvoorzieningen bevat die u kunt gebruiken in Microsoft Excel
- De geopende werkmap, die uw programmacode bevat
- Alle andere bibliotheken waarnaar wordt verwezen door de programmacode in uw werkmap

Onder het vak "Bibliotheken/Werkmappen" bevindt zich het vak "Objecten/Modulen". De inhoud van het vak "Objecten/Modulen" is afhankelijk van de vraag of het vak "Bibliotheken/Werkmappen" de naam weergeeft van een objectenbibliotheek of van een werkmap, zoals beschreven in de volgende tabel.

Als het vak "Bibliotheken/Werkmappen" dit bevat	**Dan bevat het vak "Objecten/Modulen" het volgende**
De naam van een objectenbibliotheek	De beschikbare objecten in de bibliotheek, of categorieën van beschikbare functies
De naam van een werkmap	De namen van de modulen in de werkmap

Rechts van het vak "Objecten/Modulen" bevindt zich het vak "Methoden/Eigenschappen". De inhoud van het vak "Methoden/Eigenschappen" is afhankelijk van de vraag of het vak "Objecten/Modulen" de objecten in een bibliotheek of de modulen van een werkmap weergeeft, zoals beschreven in de volgende tabel.

Als het vak "Objecten/Modulen" dit bevat	Dan bevat het vak "Methoden/Eigenschappen" het volgende
De objecten in een objectenbibliotheek	De eigenschappen en methoden van elk object in de bibliotheek
De modulen in een werkmap	De namen van de procedures in elke module, inclusief eigenschapsprocedures

Zie "Objecten besturen met eigenschappen" en "Acties uitvoeren met methoden" eerder in dit hoofdstuk voor meer informatie over eigenschappen en methoden. Zie "Meer leren over programmeren met Visual Basic" in hoofdstuk 6 voor meer informatie over eigenschapsprocedures.

Naar een bepaalde procedure gaan

U kunt het Objectenoverzicht gebruiken om snel van de ene naar de andere procedure in een werkmap te gaan.

▶ **Naar een procedure in een werkmap gaan**

1. Schakel over naar een Visual Basic-module.
2. Kies de opdracht **Objectenoverzicht** in het menu **Beeld**.
3. Selecteer in het vak "Objecten/Modulen" de naam van de module die de gewenste procedure bevat.
4. Selecteer in het vak "Methoden/Eigenschappen" de naam van de procedure.
5. Kies de knop Weergeven.

Het Objectenoverzicht brengt u naar de module die de geselecteerde procedure bevat. De invoegpositie verschijnt naast de regel die volgt op de regel met de **Sub-** of de **Functie**-instructie van de procedure.

Bladeren in objectenoverzichten

U kunt het Objectenoverzicht ook gebruiken om het volgende te weten te komen:

- Welke objecten beschikbaar zijn in de bibliotheken waarnaar wordt verwezen in een werkmap
- Wat de eigenschappen en methoden van die objecten zijn

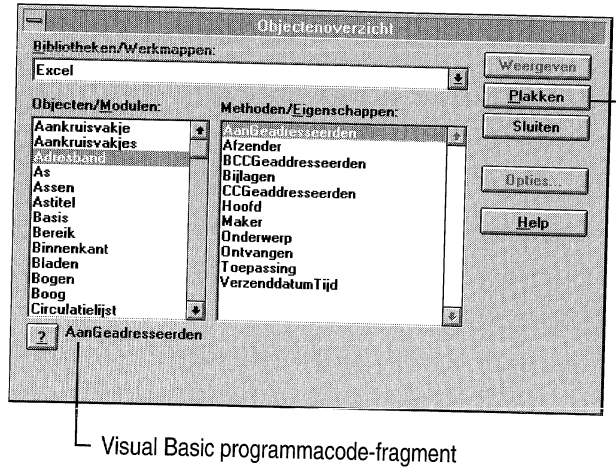
▶ **De eigenschappen en methoden van objecten nagaan**

1. Kies **Objectenoverzicht** in het menu **Beeld**.
2. Selecteer in het vak "Bibliotheken/Werkmappen" de naam van de objectenbibliotheek waarin u wilt bladeren of waarin het object dat u wilt gebruiken, zich bevindt.

3. Selecteer in het vak "Objecten/Modulen" de naam van het object waarvan u de eigenschappen en methoden wilt weten of dat u wilt gebruiken.

Programmacode plakken

Als u het Objectenoverzicht gebruikt om een object te vinden met een eigenschap of methode die u wilt gebruiken, kunt u een fragment Visual Basic-programmacode van de desbetreffende eigenschap of methode in uw procedure plakken. Vervolgens kunt u het codefragment bewerken zodat het in uw procedure past. Het codefragment wordt in de actieve module geplakt op de plaats van de invoegpositie.



Het codefragment bevat waar nodig syntaxis voor benoemde argumenten. Zie "Instructies vereenvoudigen met benoemde argumenten" in hoofdstuk 6 voor meer informatie over benoemde argumenten. Nadat u het codefragment in uw procedure hebt geplakt, kunt u de argumenten die u niet nodig hebt wissen en de gewenste waarden invoegen voor de argumenten die u wilt gebruiken.

▶ Een codefragment in een procedure plakken

1. In een Visual Basic-module plaatst u de invoegpositie op de plaats in de programmacode waar u het codefragment wilt plakken.
2. Kies de opdracht **Objectenoverzicht** in het menu **Beeld**.
3. Selecteer in de betreffende vakken de bibliotheek en het object waarvan u een eigenschap of methode wilt gebruiken.
4. Selecteer in het vak "Methoden/Eigenschappen" de methode of de eigenschap die u in de procedure wilt plakken.
5. Kies de knop Plakken.

Meerdere acties op een object uitvoeren

Procedures voeren vaak meerdere verschillende bewerkingen uit op hetzelfde object. Stel dat u meerdere eigenschappen van dezelfde cel wilt instellen. Eén manier om dat te doen is het gebruik van meerdere instructies.

```
ActiefBlad.Cellen(1; 1).Formule = "=SIN(180)"
ActiefBlad.Cellen(1; 1).Lettertype.Naam = "Arial"
ActiefBlad.Cellen(1; 1).Lettertype.Vet = Waar
ActiefBlad.Cellen(1; 1).Lettertype.Grootte = 8
```

Zoals u ziet, gebruiken al deze instructies dezelfde objectverwijzing. De instructie **Met** maakt deze programmacode gemakkelijker om in te voeren en te lezen, en bovendien efficiënter.

```
Met ActiefBlad.Cellen(1; 1)
    .Formule = "=SIN(180)"
    .Lettertype.Naam = "Arial"
    .Lettertype.Vet = Waar
    .Lettertype.Grootte = 8
Einde Met
```

U kunt **Met**-instructies ook nesten. U kunt bijvoorbeeld het voorgaande codevoorbeeld ook schrijven met de ene **Met**-instructie genest in de andere.

```
Met ActiefBlad.Cellen(1; 1)
    .Formule = "=SIN(180)"
    Met .Lettertype
        .Naam = "Arial"
        .Vet = Waar
        .Grootte = 8
    Einde Met
Einde Met
```

Zie "Een controlestructuur binnen een andere plaatsen (nesten)" in hoofdstuk 7 voor meer informatie over het nesten van Visual Basic-programmacode.

De objectverwijzing die volgt op de **Met**-instructie kan een verwijzing zijn naar een enkel object of naar een verzameling. Hier volgt een voorbeeld voor het instellen van de lettertype-eigenschappen van alle tekstvakken in het actieve blad.

```
Met ActiefBlad.Tekstvakken
    .Lettertype.Naam = "Arial"
    .Lettertype.Grootte = 8
Einde Met
```

Objecten, eigenschappen en methoden gebruiken in programmacode

In dit onderdeel wordt uitgelegd hoe objecten, eigenschappen en methoden precies samenwerken in Visual Basic-programmacode, te beginnen met een van de meestgebruikte objecten, het bereikobject. De codevoorbeelden die volgen tonen u enkele manieren om in Visual Basic bewerkingen uit te voeren op objecten en de besturing van uw toepassing nauwkeurig in te stellen.

Het bereikobject

Een bereikobject kan bestaan uit:

- Een cel
- Een rij of kolom
- Eén of meer celselecties
- Een 3D-bereik

Een bereikobject aanduiden met de methode Cellen

De methode **Cellen** is de meest gebruikelijke manier om een bereikobject aan te duiden. De methode **Cellen** geeft als resultaat een verzameling cellen; een bereikobject. Omdat cellen in rijen en kolommen op een werkblad staan gegroepeerd, is het vaak handig een cel aan te duiden met behulp van twee indexen. Op die manier kunt u de coördinaten van een cel opgeven door één index op te geven voor de rij en een tweede index voor de kolom waarin de cel zich bevindt.

Met de volgende programmacode kunt u bijvoorbeeld cel A1 op het actieve werkblad aanduiden.

```
Cellen(1; 1).Waarde = 24     'Stelt de Waarde van cel A1 in op 24
```

Het eerste cijfer duidt de rij van de cel aan, het tweede de kolom. Bij de methode **Cellen** kan de index voor de kolom ook de letter zijn die boven aan de kolom staat in plaats van het nummer van de kolom. Om cel A1 aan te duiden met een letter als kolom-index schrijft u de volgende coderegel.

```
Cellen(1; "A").Waarde = 24          'Stelt de Waarde van cel A1 in op
24
```

Hoewel u bij de methode **Cellen** een letter kunt gebruiken om een kolom-index aan te duiden, geeft dit niet altijd de meest bruikbare programmacode. U kunt bijvoorbeeld de volgende procedure schrijven door uitsluitend numerieke rij- en kolom-indexen te gebruiken.

```
Sub TabelOpmaken()
    Voor HetJaar = 1 Tot 5
        Cellen(1; HetJaar + 1).Waarde = 1990 + HetJaar
    Volgende HetJaar
    Voor HetKwartaal = 1 Tot 4
        Cellen(HetKwartaal + 1; 1).Waarde = "K" & HetKwartaal
    Volgende HetKwartaal
Einde Sub
```

Hoewel u in deze procedure tekenreeksfuncties van Visual Basic kunt gebruiken om te werken met verwijzingen in A1-stijl, is het veel gemakkelijker (en programmatechnisch beter) om numerieke rij- en kolom-indexen te gebruiken.

Voor eenvoudige procedures kent de Visual Basic-syntaxis een snelle methode om naar cellen te verwijzen: plaats de absolute verwijzing in A1-stijl tussen vierkante haken. Zo stelt bijvoorbeeld `[A1].Waarde = 24` de waarde van cel A1 in op 24. Zie het onderwerp *Cellen* in de online Visual Basic Naslaggids voor meer informatie over de methode **Cellen**.

De methode Bereik gebruiken

U kunt de methode **Bereik** gebruiken om een rechthoekig cellenbereik op te vragen. Het volgende voorbeeld vult het bereik A1:H8 met de tekenreeks "Test".

```
Sub BereikVullen()
    Bereik(Cellen(1; 1); Cellen(8; 8)).Waarde = "Test"
Einde Sub
```

U kunt de methode **Bereik** ook gebruiken om een benoemd bereik op te vragen, zoals in het volgende voorbeeld wordt getoond.

```
Sub DatabaseCriteriaWissen()
    Bereik("Criteria").InhoudWissen
Einde Sub
```

De methode **Bereik** kent eveneens snelle methoden om naar cellenbereiken te verwijzen in eenvoudige procedures. Zo stelt `Bereik("A1") = 24` de waarde van cel A1 in op 24. Zie *Bereik (methode)* in de online Visual Basic Naslaggids voor meer informatie over de methode **Bereik**.

Meervoudige selecties

Door gebruik te maken van de methode **Vereniging** en de methode **Bereik** kunt u meervoudige selecties ophalen. In het volgende voorbeeld wordt eerst een object genaamd MijnMeervoudigeSelectie gemaakt. Dit object wordt daarna gedefinieerd als de meervoudige selectie van A1:B2 en C3:D4 en vervolgens geselecteerd.

```
Sub MeervoudigeSelectieTonen()
    Dim r1; r2; MijnMeervoudigeSelectie Als Bereik
    Toewijzen r1 = Bereik(Cellen(1; 1); Cellen(2; 2))
    Toewijzen r2 = Bereik(Cellen(3; 3); Cellen(4; 4))
    Toewijzen MijnMeervoudigeSelectie = Vereniging(r1; r2)
    MijnMeervoudigeSelectie.Selecteren
Einde Sub
```

Als u met meervoudige selecties werkt, is de methode **Vlakken** heel nuttig. Deze methode verdeelt een meervoudige selectie in bereiken en resulteert in een verzameling van deze bereiken. U kunt de eigenschap **Aantal** voor deze verzameling gebruiken om te controleren of u met een meervoudige selectie te maken hebt. Zie hiervoor het volgende voorbeeld.

```
Sub GeenMeervoudigeSelecties()
    AantalGeselecteerdeVlakken = Selectie.Vlakken.Aantal
    Indien AantalGeselecteerdeVlakken > 1 Dan
        BerichtVenster "Opdracht niet voor meervoudige selecties "
    Einde Indien
Einde Sub
```

Voorbeelden van objecten, eigenschappen en methoden in programmacode

In dit onderdeel worden vier procedures beschreven die laten zien hoe objecten, eigenschappen en methoden samenwerken in Visual Basic-programmacode. U zult misschien nog niet alle programmacode begrijpen, maar als u de voorbeelden leest, krijgt u een indruk hoe op objecten in een reële context bewerkingen worden uitgevoerd.

In deze voorbeelden worden vele van de objecten, eigenschappen en methoden gebruikt die eerder in dit hoofdstuk zijn beschreven, en er worden ook enkele nieuwe geïntroduceerd. Raadpleeg de online Visual Basic Naslaggids voor een beschrijving van de Visual Basic-sleutelwoorden die u interesseren. (Als u deze voorbeelden in een Visual Basic-module typt, kunt u ook het desbetreffende sleutelwoord selecteren en op F1 drukken in Microsoft Excel voor Windows of op COMMAND+SHIFT+VRAAGTEKEN op de Macintosh).

Speciale opmaak toepassen op een bereik

In deze procedure wordt een vierkante celselectie gemaakt en wordt vervolgens een regelmatig patroon van rijen en kolommen in het bereik vet gemaakt. Let op de volgende elementen van de Visual Basic-programmeertaal:

- De eigenschap **ActieveCel**
- De eigenschap **HuidigBereik** (het huidige bereik is een vlak dat is omgeven door lege rijen en kolommen)

- De methode **GrootteWijzigen** (deze wijzigt het formaat van de huidige selectie)
- Het lettertype-object
- De eigenschap **Kleur**
- De functie **RGB** (deze geeft als resultaat een cijfer dat overeenkomt met een kleur)

```
Sub MooieOpmaak()
    Blauw = RGB(0; 0; 255)                    'De kleur blauw
    Zwart = RGB(0; 0; 0)                      'De kleur zwart
    'Maakt afwisselende rijen en kolommen vet.
    Toewijzen VierkantBereik = ActieveCel.HuidigBereik
    RijNr = VierkantBereik.Rijen.Aantal
    'Voert een lus uit voor alle rijen en kolommen waarbij
    'afwisselend een vette opmaak wordt toegepast in een
    'speciaal patroon.
    MaakVet = Waar
    Voor I = RijNr Tot 1 Stappen -1
        Toewijzen VierkantBereik = VierkantBereik.GrootteWijzigen(I; I)
        VierkantBereik.Lettertype.Vet = MaakVet
        Indien MaakVet = Onwaar Dan
            VierkantBereik.Lettertype.Kleur = Blauw
        Anders
            VierkantBereik.Lettertype.Kleur = Zwart
        Einde Indien
        MaakVet = Niet (MaakVet)
    Volgende I
Einde Sub
```

De volgende figuur toont het geselecteerde bereik met de opmaak Vet.

	A	B	C	D	E
1	**Mooi**	Mooi	**Mooi**	Mooi	**Mooi**
2	Mooi	Mooi	**Mooi**	Mooi	**Mooi**
3	**Mooi**	**Mooi**	**Mooi**	Mooi	**Mooi**
4	Mooi	Mooi	Mooi	Mooi	**Mooi**
5	**Mooi**	**Mooi**	**Mooi**	**Mooi**	**Mooi**

Een grafiek maken op basis van gegevens in het werkblad

In deze procedure wordt een 3D-kolomdiagram gemaakt, op basis van de gegevens in de cellen A1:D6. Zie de volgende figuur.

	A	B	C	D
1		Nippels	Nokken	Radertjes
2	1989	10	12	15
3	1990	11	14	18
4	1991	12	16	21
5	1992	13	18	24
6	1993	14	20	27

Let in de procedure op de volgende elementen van de Visual Basic-programmeertaal:

- De methode **Bereik**
- De methode **Selecteren**
- De methode **Toevoegen**
- De methode **Grafieken**
- De instructie **Toewijzen** (deze wijst een object toe aan een variabele)

```
Sub GrafiekDemo()
    'Declareer variabelen.
    Dim GrafiekBereik Als Object
    Dim NieuweGrafiek Als Object
    'Bepaal het bereik met de waarden voor de grafiek.
    Toewijzen GrafiekBereik = Werkbladen(1).Bereik("A1:D6")
    GrafiekBereik.Selecteren
    'Voeg een grafiekblad toe (gebaseerd op het grafiekbereik).
    'Wijs de nieuwe grafiek toe aan een variabele.
    Toewijzen NieuweGrafiek = Grafieken.Toevoegen
    NieuweGrafiek.Activeren
    'Selecteer het grafiektype en subtype.
    Met NieuweGrafiek
        .Type = xl3DKolom
        .Subtype = xlNormaal
    Einde Met
Einde Sub
```

Vinden en afdrukken van werkmapbestanden

In deze procedure wordt een eenvoudig venster weergegeven waarin de gebruiker wordt gevraagd een station en een directory op te geven, zoals C:\TEMP\. Vervolgens wordt in de directory gezocht naar werkmapbestanden (*.XLS) en wordt elk bestand geopend en afgedrukt. Let in de programmacode op de volgende elementen van de Visual Basic-programmeertaal:

- De functie **InvoerVenster**
- De functie **Dir** (deze geeft als resultaat een lijst van bestandsnamen)
- De constructie **Doorlopen...Lus** (deze herhaalt de programmacode meerdere malen zolang een bepaalde voorwaarde **Waar** is)
- De methode **Openen**
- De methode **Afdrukken**

```
Sub MappenAfdrukken()
    'Vraag de gebruiker in welke directory de    werkmappen
    'moeten worden gezocht en druk deze vervolgens af.
    NaamDir = InvoerVenster("Zoeken in directory"; "Mappen afdrukken")
    VolgendeMap = Dir(NaamDir & "*.xls")
        'Voorwaarde is waar zolang een werkmap wordt gevonden
        'in de opgegeven directory.
    Doorlopen Terwijl VolgendeMap <> ""
        BerichtVenster VolgendeMap & " wordt geopend."
        'Open de werkmap.
        Werkmappen.Openen NaamDir & VolgendeMap
        'Druk de werkmap af.
        ActieveWerkmap.Afdrukken
        'Sluit de werkmap.
        BerichtVenster VolgendeMap & " wordt gesloten"
        ActieveWerkmap.Sluiten
        'Roep Dir zonder argumenten op om eventuele volgende
        'bestandsnamen in het pad op te halen.
        VolgendeMap = Dir()
    Lus
Einde Sub
```

Gegevens van een werkblad controleren

In de volgende procedure worden gegevens gecontroleerd die zijn ingevoerd in een werkblad dat een voorraadlijst met onderdelen bevat. De eerste kolom van het werkblad bevat een hoeveelheid, die numeriek moet zijn. De tweede kolom bevat een waarde **Waar** of **Onwaar**, die aangeeft of het onderdeel besteld is. De derde kolom bevat een onderdeelnummer, waarvan de eerste vijf tekens cijfers moeten zijn en de overige tekens zowel cijfers als letters kunnen zijn. De voorbeeldgegevens zouden er ongeveer als volgt kunnen uitzien.

	A	B	C
1	Hoeveelheid	Besteld?	Onderdeelnummer
2	100	WAAR	12345-3XCD
3	200	WAAR	14365-75HD
4	250	WAAR	18675-67GR
5	350	ONWAAR	53985RY57
6	800	ONWAAR	43785-T6H6
7	210	WAAR	54980-3D5T
8	440	ONWAAR	56789-9KJ7

Let in het codevoorbeeld op de volgende elementen van de Visual Basic-programmeertaal:

- De instructie **Toewijzen**
- De eigenschap **HuidigBereik**
- De methode **Bereik**
- De methode **Rijen**

- De eigenschap **Aantal**
- De constructie **Voor Elke...Volgende** (deze herhaalt de programmacode éénmaal voor elk element van een verzameling)
- De functie **IsNumeriek** (deze resulteert in de waarde **Waar** als het argument numeriek is)
- De instructie **BerichtVenster** (deze geeft een bericht weer op het scherm)

In deze procedure wordt aangenomen dat er een titelrij is. Er wordt gecontroleerd of in elke kolom alle waarden van hetzelfde type zijn. De drie kolommen moeten achtereenvolgens bevatten: cijfers, Boole-waarden en serienummers die beginnen met 5 cijfers.

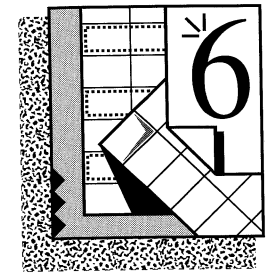
```
Sub TypeBereikControleren()
    'Stel de bereikvariabele in op het huidige bereikgebied.
    Toewijzen TeControlerenBereik = ActieveCel.HuidigBereik
    'Stel nieuw bereik in onder de titelrij.
    Toewijzen NieuwBereik = Bereik(TeControlerenBereik.Cellen(2; 1); _
    TeControlerenBereik.Cellen(TeControlerenBereik.Rijen.Aantal; _
    TeControlerenBereik.Kolommen.Aantal))
    'Controleer elke cel in de kolom Hoeveelheid.
    Voor Elke Cel In NieuwBereik.Kolommen(1).Cellen
        Indien Niet (IsNumeriek(Cel.Waarde)) Dan
            BerichtVenster "U moet een cijfer opgeven."
        Einde Indien
    Volgende
    'Controleer elke cel in de kolom Besteld.
    Voor Elke Cel In NieuwBereik.Kolommen(2).Cellen
        Indien Niet (Cel.Waarde = Waar Of Cel.Waarde = Onwaar) Dan
            BerichtVenster "U moet een Boole-waarde opgeven."
        Einde Indien
    Volgende
    'Controleer elke cel in de kolom Onderdeelnummer.
    Voor Elke Cel In NieuwBereik.Kolommen(3).Cellen
        Indien Niet (IsNumeriek(Links(cel.Waarde; 5))) Dan
            BerichtVenster "De eerste 5 tekens moeten numeriek zijn."
        Einde Indien
    Volgende
Einde Sub
```

Belangrijk Bij veel van de voorbeelden in dit hoofdstuk wordt aangenomen dat u deze pas gebruikt als er een werkblad actief is. Wanneer u problemen ondervindt bij het afspelen van uw programmacode vanuit een Visual Basic-module, kunt u een werkblad activeren en dan de programmacode afspelen, ofwel boven de regel die de fout veroorzaakt een regel invoegen als de volgende:

```
Werkbladen("Blad1").Activeer
```

HOOFDSTUK 6

Werken met Visual Basic-programmacode in procedures



In de vorige hoofdstukken hebt u kunnen kennismaken met procedures, objecten, eigenschappen en methoden. In dit hoofdstuk en in hoofdstuk 7, "De uitvoering van de programmacode besturen", zult u kennismaken met de belangrijkste termen en concepten van Visual Basic. Verder zal worden ingegaan op de werking van door u geschreven programmacode en zal worden uitgelegd hoe u de programmacode kunt gebruiken om de volgorde te bepalen waarin de procedures worden uitgevoerd.

Inhoud

- Visual Basic-programmacode begrijpen
- Variabelen en argumenten vooraf definiëren
- Met het gegevenstype Variant verschillende typen gegevens opslaan
- Het gegevenstype opgeven voor een variabele
- Het gegevenstype opgeven voor een argument
- Het gegevenstype opgeven voor het resultaat van een Functie-procedure
- Een object toewijzen aan een variabele
- Operatoren in een expressie gebruiken
- Constanten gebruiken die waarden vertegenwoordigen
- Instructies vereenvoudigen met benoemde argumenten
- Meer leren over programmeren met Visual Basic

De onderwerpen in dit hoofdstuk zijn geordend op basis van toenemende complexiteit. De meest eenvoudige informatie over de argumenten, variabelen en gegevenstypen in Visual Basic komt in het begin aan de orde. In de rest van het hoofdstuk wordt dieper ingegaan op deze onderwerpen en wordt bovendien aandacht geschonken aan andere geavanceerde voorzieningen.

Visual Basic-programmacode begrijpen

Procedures bestaan uit Visual Basic-programmacode. Hier volgt een lijst met een aantal termen en concepten die betrekking hebben op Visual Basic-programmacode. Een goed begrip van deze termen is van belang voor de verdere uitleg van Visual Basic in dit hoofdstuk.

- De *Visual Basic-programmacode* in een procedure geeft aan welke actie door de procedure moet worden uitgevoerd. De basiseenheid van Visual Basic-programmacode is de instructie.
- Een *instructie* is een syntactisch volledig stukje programmacode waardoor één actie, declaratie of definitie wordt opgegeven. De volgende regels met Visual Basic-programmacode bevatten allemaal één instructie.

```
Cellen(1; 1).Waarde = 4 'Geeft 4 op als waarde van cel A1
BerichtVenster "U kunt dit onderdeel niet verwijderen."
'Geeft berichtvenster weer
Winst = Inkomsten - Kosten 'Toewijzingsinstructie
```

  Soms bestaat een instructie uit andere instructies of functies. De instructie **Indien...Dan...Anders** bevat bijvoorbeeld altijd andere instructies.

```
Indien TotaleVerkPrijs <= 15000 Dan
    Provisie = 25 + 0,03 * AandelenVerkocht
Anders
    Provisie = 25 + 0,03 * (0,9 * AandelenVerkocht)
Einde Indien
```

  Omdat instructies zoals **Indien...Dan...Anders** verschillende regels met programmacode bevatten, worden ze ook wel *instructieblokken* genoemd.
- Een *functie* resulteert in een waarde. De ingebouwde functie **Uur** bijvoorbeeld geeft het uur van de dag als resultaat. En de functie **Hoofdletters** resulteert in een tekst in hoofdletters. Een ***Functie**-procedure* is een type functie dat u zelf kunt maken.
- Sommige sleutelwoorden in Visual Basic kunnen zowel functies als instructies zijn, afhankelijk van de manier waarop ze worden gebruikt. Zo bestaat er bijvoorbeeld een instructie **Datum** en een functie **Datum**.

```
Datum = #28-02-1994#
'Met de instructie Datum stelt u de systeemdatum in.
HuidigeDatum = Datum
'De functie Datum resulteert in de huidige datum.
```

Tip Zie *Definities* in de online *Visual Basic Naslaggids* als u vergeten bent wat een Visual Basic-term betekent.

Veelgebruikte Visual Basic-instructies

Veel instructies in de Visual Basic-programmacode worden gebruikt voor de volgende doeleinden.

- Waarden toewijzen aan variabelen (een zogeheten toewijzingsinstructie).

  variabele = *expressie*

  ```
  LeeftijdGebruiker = InvoerVenster("Wat is uw leeftijd?")
  ```

- Ingebouwde instructies uitvoeren (de syntaxis varieert van instructie tot instructie).

  instructie argumentenlijst

  ```
  Hernoemen "OUDBEST.TXT" Als "NWBEST.TXT" 'Nieuwe naam voor bestand
  ```

- De uitvoering van de programmacode besturen aan de hand van voorwaarden.

  ```
  Doorlopen
      EenGetal = InvoerVenster("Voer een getal in")
  Lus Totdat IsNumeriek(EenGetal)
  BerichtVenster "De wortel is: " & Wortel(EenGetal)
  ```

- De eigenschappen van objecten instellen.

  object.Eigenschap = *expressie*

  ```
  ActieveCel.Lettertype.vet = Waar
  ```

- De methoden van objecten uitvoeren.

  object.Methode argumentenlijst

  ```
  Toepassing.SpellingControleren Woord:="onmiddellijk"
  ```

Het gebruik van ronde haken bij argumenten

Sub-procedures, ingebouwde instructies en een aantal methoden geven geen resultaatwaarde. Als u dergelijke procedures, instructies of methoden oproept, hoeft u de argumenten dus niet tussen ronde haken te plaatsen. **Functie**-procedures, ingebouwde functies en een aantal andere methoden geven wel in een resultaatwaarde. Als u deze procedures, functies of methoden oproept, moet u de argumenten wel tussen ronde haken plaatsen. Wanneer u de resultaatwaarde negeert, of als u geen argumenten doorgeeft, gebruikt u helemaal geen ronde haken. Ook wanneer u benoemde argumenten gebruikt, gelden deze richtlijnen. Zie "Instructies vereenvoudigen met benoemde argumenten" verderop in dit hoofdstuk voor meer informatie over het gebruik van benoemde argumenten.

```
MaakMijnBladOp 15; 25
' Roept een Sub-procedure op.

MaakMijnBladOp AantalRijen:=15; RijGrootte:=25
' Roept een Sub-procedure op met benoemde argumenten

X = BerekenWinst(100; 50; 1000)
' Roept een Functie-procedure op. Slaat de resultaatwaarde op.

BerekenWinst 100; 50; 1000
' Roept een Functie-procedure op. Negeert de resultaatwaarde.

JuistGespeld = Toepassing.SpellingControleren woord:="onmiddellijk"
' Voert een methode uit. Slaat de resultaatwaarde op.

Toepassing.SpellingControleren woord:="onmiddellijk"
' Voert een methode uit. Negeert de resultaatwaarde.

HuidigeDatum = Datum
' Roept de ingebouwde functie Datum op en slaat de resultaatwaarde op,
' maar geeft geen argumenten door.
```

Variabelen en argumenten vooraf definiëren

Bij het schrijven van programmacode in Visual Basic is het niet nodig om variabelen en argumenten te definiëren voordat u deze kunt gebruiken. In de volgende **Functie**-procedure bijvoorbeeld is `getal` een argument, `TijdWrd` een variabele en `AbsWort` een andere variabele die ook de resultaatwaarde van de functie is. Visual Basic herkent en verwerkt deze onderdelen automatisch. Hierdoor is het maken van procedures in Visual Basic zo eenvoudig.

```
' Zorgt ervoor dat het argument positief
' is, voordat de wortel wordt berekend.
Functie AbsWort(getal)
    TijdWrd = Abs(getal)
    AbsWort = Wortel(TijdWrd)
Einde Functie
```

Er zijn echter bepaalde voordelen verbonden aan het vooraf declareren van variabelen en argumenten. Door een *declaratie* voor een variabele of een ander onderdeel op te geven, maakt u Visual Basic attent op het bestaan van dit onderdeel.

Waarom variabelen of argumenten vooraf definiëren?

Hoewel Visual Basic de meeste declaraties en gegevenstyperingen automatisch verzorgt, kunt u de programmacode sneller laten uitvoeren, ruimte besparen en fouten vermijden door zelf declaraties voor variabelen en argumenten op te geven.

In het vorige voorbeeld herkent Visual Basic de variabele `TijdWrd` automatisch. U kunt de variabele dus gebruiken alsof deze expliciet is gedeclareerd. Ofschoon dit erg handig is, kan deze handelwijze leiden tot subtiele fouten in uw programmacode. Stel dat u in de op een na laatste regel `TijWrd` typt in plaats van de juiste naam voor de variabele. De waarde van de nieuwe variabele `TijWrd` is nul en de wortel van nul is ook nul. Het resultaat van de functie is dus altijd nul in plaats van de echte waarde. In dit geval genereert Visual Basic echter geen foutbericht.

Als Visual Basic een nieuwe naam tegenkomt, kan de taal niet bepalen of u een nieuwe variabele wilt invoeren of dat u de naam van een bestaande variabele verkeerd hebt ingevoerd. Er wordt dus een nieuwe variabele met de nieuwe naam gemaakt. U kunt dit probleem omzeilen door Visual Basic een foutbericht te laten geven wanneer u een naam invoert die nog niet expliciet als variabele is gedeclareerd. U doet dit door de volgende instructie boven aan de Visual Basic-module te plaatsen.

```
Optie Expliciet
```

Nu moet u elke variabele expliciet declareren. Binnen een procedure declareert u een variabele expliciet met de instructie **Dim** gevolgd door de naam van de variabele.

Dim *variabelenaam*

De functie AbsWort krijgt dan de volgende vorm:

```
Optie Expliciet

Functie AbsWort(getal)
    Dim TijdWrd
    TijdWrd = Abs(getal)
    AbsWort = Wortel(TijdWrd)
Einde Functie
```

Wanneer u nu de variabele `TijdWrd` verkeerd typt als `TijWrd`, verschijnt er een foutbericht als u de programmacode start (of als u de functie in een cel invoert). Ook wanneer u de instructie **Optie Expliciet** niet gebruikt boven aan uw module, kunt u een aantal variabelen expliciet declareren. De overige variabelen worden dan automatisch impliciet gedeclareerd.

Wanneer u de instructie **Optie Expliciet** niet gebruikt en u variabelen impliciet laat declareren door Visual Basic, is het toch verstandig om sommige variabelen expliciet te declareren met de instructie **Dim**. Hierdoor kunt u gemakkelijker onthouden welke variabelen u hebt gebruikt en bovendien wordt de programmacode op deze manier eenvoudiger te lezen.

Fouten opsporen met expliciete declaraties

Omdat u met de instructie **Optie Expliciet** fouten kunt opsporen, wilt u wellicht deze instructie in alle procedures gebruiken. U kunt **Optie Expliciet** automatisch toevoegen aan elke module die u maakt door deze instructie als een omgevingsoptie in te voeren.

▶ **De Optie Expliciet automatisch in alle nieuwe modulen plaatsen**

1. Kies de opdracht **Opties** in het menu **Extra**.
2. Selecteer de tab Module.
3. Schakel het aankruisvakje "Definitie variabelen vereist" in.
4. Kies de knop OK.

Opmerking De instructie **Optie Expliciet** moet in de declaratie-sectie (boven aan de module) worden geplaatst, in elke module waarin u gebruik wilt maken van expliciete declaraties voor variabelen en argumenten. Als u het aankruisvakje "Definitie variabelen vereist" inschakelt, wordt **Optie Expliciet** aan alle nieuwe modulen toegevoegd, maar niet aan bestaande programmacode. U moet deze instructie handmatig toevoegen aan de bestaande modulen.

Met het gegevenstype Variant verschillende typen gegevens opslaan

Alle variabelen beschikken over een gegevenstype. Met het *gegevenstype* wordt aangegeven welk soort gegevens een variabele of een ander onderdeel kan bevatten (bijvoorbeeld een geheel getal, een tekenreeks of een datum).

Als u niet zelf een gegevenstype definieert, wordt standaard aan variabelen het gegevenstype **Variant** toegekend. Het gegevenstype **Variant** kan voor verschillende soorten gegevens worden gebruikt: voor getallen, tekenreeksen of datum en tijd. Door het gegevenstype **Variant** hoeft u variabelen niet te declareren en kunnen variabelen verschillende soorten gegevens bevatten tijdens de uitvoering van de programmacode.

Nagenoeg alle variabelen waarmee u tot nu toe hebt gewerkt, behoorden tot het gegevenstype **Variant** (beter bekend onder de kortere naam "**Variant**-variabelen").

Variant-variabelen zijn te vergelijken met cellen in een werkblad: het soort gegevens dat deze variabelen bevatten is afhankelijk van de waarde die u er als laatste in hebt geplaatst. In een cel kunt u bijvoorbeeld achtereenvolgens getallen, tekst of een datum plaatsen. Hetzelfde kunt u doen met **Variant**-variabelen.

In de volgende **Functie**-procedure, bevat de variabele X achtereenvolgens verschillende soorten gegevens.

```
Functie GeefGegevensType(EenWaarde)
    X = EenWaarde              ' X is een Variant die een waarde bevat.
    X = "Hallo Wereld"         ' X is een Variant die een tekenreeks bevat.
    X = #24-12-1994#           ' X is een Variant die een datum bevat.
    X = 10                     ' X is een Variant die een getal bevat.
    GeefGegevensType = TypeNaam(X    )   ' De resulterende naam is
Integer,
                                         ' variabele is nog steeds Variant.

Einde Functie
```

Visual Basic voert automatisch wiskundige bewerkingen uit op **Variant**-variabelen.

```
Dim EenWaarde
' Standaard een Variant.
EenWaarde = "17"
' EenWaarde bevat "17" (tekenreeks van twee tekens).
' Wijzig de numerieke waarde van EenWaarde in 2.
EenWaarde = EenWaarde - 15
' Wijzig EenWaarde in "U2" (een tekenreeks van twee tekens).
EenWaarde = "U" & EenWaarde
```

U kunt bewerkingen uitvoeren op **Variant**-variabelen zonder veel aandacht te besteden aan het soort gegevens dat ze bevatten. Als u echter wiskundige bewerkingen of functies op een **Variant**-variabele uitvoert, moet deze een getal of een waarde bevatten die als een getal kan worden geïnterpreteerd. U kunt bijvoorbeeld geen wiskundige bewerkingen uitvoeren op de waarde "U2", ook al bevat deze waarde een numeriek teken. Ook op de waarde "1040EZ" kunt u geen wiskundige bewerking uitvoeren. De waarden "+10" of "–,7E6" kunnen echter wel worden gebruikt voor berekeningen, omdat voor deze waarden gebruik is gemaakt van geldige getalnotaties.

Zie "Meer informatie over het gegevenstype Variant en andere gegevenstypen" verderop in dit hoofdstuk voor meer informatie of zie *Variant* in de online *Visual Basic Naslaggids*.

Het gegevenstype opgeven voor een variabele

Variant-variabelen zijn een erg nuttige voorziening in Visual Basic. Hierdoor kunt u zich concentreren op het schrijven van de programmacode, zonder u verder druk te maken over het gegevenstype dat een variabele zou moeten bevatten. Soms zult u echter variabelen van een bepaald gegevenstype willen declareren. Als u bijvoorbeeld weet dat een bepaalde variabele altijd een geheel getal zal bevatten, kunt u voorkomen dat voor deze variabele per ongeluk een tekenreeks wordt ingevoerd. Bovendien wordt de programmacode sneller en efficiënter uitgevoerd, en is de programmacode gemakkelijker te ontdoen van fouten wanneer u een specifiek gegevenstype voor de variabelen definieert.

Als u een specifiek gegevenstype wilt opgeven voor een variabele, maakt u gebruik van de instructie **Dim**. U moet dan de naam van de variabele en de naam van het gegevenstype opgeven.

Dim *variabelenaam* **Als** *type*

Het gedeelte **Als** *type* is optioneel. (U kunt een expliciete declaratie voor een variabele declareren zonder een gegevenstype te definiëren. De variabele krijgt dan automatisch het gegevenstype **Variant**). Meestal worden de declaraties voor de variabelen onmiddellijk na de eerste regel van een procedure (de definitie-regel) geplaatst. Hier volgt een herziene versie van de functie AbsWort die het argument `getal` omzet in een geheel getal en ervan uitgaat dat de variabele `TijdWrd` ook een geheel getal bevat.

```
Functie AbsWort(getal)
    ' Geeft op dat TijdWrd alleen een geheel getal mag bevatten
    Dim TijdWrd Als Integer
    TijdWrd = Abs(CInteger(getal))
    AbsWort = Wortel(TijdWrd)
Einde Functie
```

In het voorgaande voorbeeld wordt het gegevenstype **Integer** gebruikt, maar u kunt ook een aantal andere specifieke gegevenstypen gebruiken. In de volgende tabel worden de in Visual Basic ondersteunde, specifieke gegevenstypen vermeld. Bovendien wordt vermeld hoeveel geheugenruimte door het gegevenstype in beslag wordt genomen en wat het bereik is van elk gegevenstype.

Specifiek gegevenstype	Geheugen	Bereik
Boole	2 bytes	**Waar** of **Onwaar**
Integer	2 bytes	–32.768 tot en met 32.767
Lang (lang geheel getal)	4 bytes	–2.147.483.648 tot en met 2.147.483.647
Enkel (enkele nauwkeurigheid met zwevende komma)	4 bytes	–3,402823E38 tot en met –1,401298E–45 voor negatieve waarden; 1,401298E–45 tot en met 3,402823E38 voor positieve waarden

Specifiek gegevenstype	Geheugen	Bereik
Dubbel (dubbele nauwkeurigheid met zwevende komma)	8 bytes	–1,79769313486232E308 tot en met –4,94065645841247E–324 voor negatieve waarden; 4,94065645841247E–324 tot en met 1,79769313486232E308 voor positieve waarden
Valuta (getal met maximaal vier decimalen)	8 bytes	–922.337.203.685.477,5808 tot en met 922.337.203.685.477,5807
Datum	8 bytes	1 januari 0100 tot en met 31 december 9999
Tekenreeks	1 byte per teken	0 tot en met 65.535 tekens (en meer op bepaalde systemen)

Omdat Visual Basic het gebruik van specifieke gegevenstypen ondersteund, kunt u declaraties met specifieke gegevenstypen opnemen in uw programmacode, zoals u ziet in de volgende voorbeelden.

```
Dim Kosten Als Valuta
Dim IsFoutOpgetreden Als Boole
Dim NaamKlant Als Tekenreeks
Dim VervalDatum Als Datum
Dim AantalGasten Als Integer
Dim AantalZonnestelsels Als Lang
Dim GewichtSteekproef Als Enkel
Dim MassaVanHetHeelal Als Dubbel
```

Naast de voorgaande gegevenstypen ondersteunt Visual Basic ook gegevenstypen met speciale kenmerken. Een lijst met deze gegevenstypen ziet u in de volgende tabel.

Gegevenstype	Geheugen	Bereik
Object	4 bytes	Een willekeurige verwijzing naar een **Object**.
Variant	Naar behoefte	Een willekeurige numerieke waarde met maximaal hetzelfde bereik als het gegevenstype **Dubbel** of een willekeurige tekenreeks.
Door de gebruiker gedefinieerd (**Type**)	Benodigd geheugen bepaald door samenstellende elementen	Het bereik van elk element is hetzelfde als het bereik van het gegevenstype waarop het element is gebaseerd, zoals weergegeven in de vorige tabel.

Gegevenstype	Geheugen	Bereik
Matrix	Naar behoefte	Een verzameling opeenvolgend geïndexeerde elementen die allemaal hetzelfde intrinsieke gegevenstype hebben. Het aantal elementen is niet begrensd.

U hebt al kennisgemaakt met het gegevenstype **Variant**. Dit gegevenstype en de overige gegevenstypen worden verderop in dit hoofdstuk uitvoeriger behandeld. Daarnaast komt het gegevenstype **Object** uitgebreid aan de orde in hoofdstuk 5, "Werken met objecten in Visual Basic". Zie ook de online *Visual Basic Naslaggids* voor meer informatie over specifieke gegevenstypen.

Meerdere variabelen in een regel declareren

De instructie **Dim** kan meerdere declaraties per regel bevatten.

```
Dim I Als Integer; Bedr Als Dubbel
Dim UwNaam Als Tekenreeks; BetRek Als Valuta
Dim Test; J Als Integer; Bedrag
```

Opmerking In het voorgaande voorbeeld is aan de variabelen `Test` en `Bedrag` het gegevenstype **Variant** toegekend. Programmeurs met ervaring in andere programmeertalen hadden misschien verwacht dat alle variabelen in één regel hetzelfde gegevenstype krijgen (in dit geval **Integer**).

Het gegevenstype van een variabele controleren

Door het gegevenstype van een variabele te controleren kunnen fouten worden voorkomen. Als u wilt weten of een variabele een bepaald gegevenstype heeft, kunt u de functie **TypeNaam** gebruiken. Deze functie geeft de naam van het gegevenstype van de variabele als resultaat.

```
Indien TypeNaam(AantalEenheden) = "Integer" Dan
    'Gebruik de integer bij een berekening
Einde Indien
```

Zie *TypeNaam* in de online *Visual Basic Naslaggids* voor een lijst met de namen waarin de functie **TypeNaam** kan resulteren.

Er treedt een fout op als u probeert een wiskundige bewerking uit te voeren op een **Variant**-variabele die geen getal bevat en die ook geen waarde bevat die kan worden omgezet in een getal (zoals een datum- of tijdwaarde of een tekenreeks die kan worden omgezet in een getal). Het is daarom raadzaam te controleren of een **Variant**-variabele een waarde bevat die met succes kan worden omgezet in een getal. De functie **IsNumeriek** voert deze taak uit.

```
Doorlopen
    EenGetal = InvoerVenster("Voer een getal in")
Lus Totdat IsNumeriek(EenGetal)
BerichtVenster "De wortel is: " & Wortel(EenGetal)
```

IsNumeriek resulteert in de waarde **Waar** als het argument van de functie een getal is (integer, lang, enkel of dubbel). Zie *IsNumeriek* in de online *Visual Basic Naslaggids* voor meer informatie over de functie **IsNumeriek**.

Het gegevenstype opgeven voor een argument

Net zoals de overige variabelen krijgen de argumenten voor de door u geschreven procedures standaard het gegevenstype **Variant**. U kunt echter een ander gegevenstype voor deze argumenten kiezen door **Als** *type* na elk argument in de eerste regel van de procedure op te nemen (de definitie-regel). In de volgende **Functie**-procedure wordt bijvoorbeeld geëist dat voor het eerste argument een **Tekenreeks** wordt opgegeven en voor het tweede argument een **Integer**.

```
Functie Omkeren(T Als Tekenreeks; n Als Integer)
    ' Keert de eerste n tekens in T om.
    Dim TijdWrd Als Tekenreeks; I Als Integer
    Indien n > Lengte(T) Dan n = Lengte(T)
    Voor I = n Tot 1 Stappen -1
        TijdWrd = TijdWrd & Deel(T; I; 1)
    Volgende
    Omkeren = TijdWrd & Rechts(T; Lengte(T) - n)
Einde Functie
```

Misschien kunt u de programmacode in dit voorbeeld nog niet volledig doorgronden. Deze procedure keert de volgorde van de eerste *n* tekens in een willekeurige tekenreeks om. Desgewenst kunt u dit voorbeeld in een Visual Basic-module typen. Als u vervolgens de formule `=Omkeren("Even testen";4)` in een cel van een werkblad invult, zal deze formule "nevE testen" als resultaat opleveren.

Opmerking Ervaren programmeurs zijn vaak gewend onderscheid te maken tussen argumenten (waarden die aan een procedure worden doorgegeven als deze wordt opgeroepen) en parameters (namen van de variabelen die in de declaratie van de procedure worden gebruikt). In deze handleiding wordt de term *argument* gebruikt voor beide soorten waarden.

Het gegevenstype opgeven voor het resultaat van een Functie-procedure

Omdat alle Visual Basic-functies in waarden resulteren, hebben deze net zoals variabelen een bepaald gegevenstype. **Functie**-procedures resulteren dan ook net zoals variabelen standaard in het gegevenstype **Variant**. Als u geen resultaatwaarde voor de **Functie**-procedure opgeeft, is het resultaat een **Variant** met de waarde **Leeg**. U kunt echter ook in de declaratie van een **Functie**-procedure opgeven dat deze moet resulteren in waarden van een bepaald gegevenstype. In het volgende voorbeeld wordt de functie Omkeren gedwongen in een tekenreeks te resulteren.

```
Functie Omkeren(T Als Tekenreeks; n Als Integer) Als Tekenreeks
```

Als u in dit voorbeeld geen resultaatwaarde toewijst, resulteert de functie in een tekenreeks met de lengte nul (""). Als u in de declaratie van een functie hebt opgegeven dat deze resulteert in een numeriek gegevenstype (zoals **Integer** of **Dubbel**), resulteert deze functie in nul, tenzij u expliciet een resultaatwaarde hebt opgegeven.

Net zoals bij variabelen hoeft u voor een **Functie**-procedure geen gegevenstype op te geven, omdat het standaard gegevenstype **Variant** de meeste soorten gegevens automatisch afhandelt. Visual Basic werkt echter veel efficiënter als u expliciet een gegevenstype voor de functies declareert.

Een object toewijzen aan een variabele

Behalve voor het opslaan van waarden kunt u variabelen gebruiken om te verwijzen naar objecten. Hiervoor zijn dezelfde redenen te bedenken als voor het toewijzen van een waarde aan een variabele: de namen van variabelen zijn vaak korter en gemakkelijker te onthouden dan de waarden die ze bevatten (of in dit geval de objecten waarnaar ze verwijzen) en variabelen kunnen tijdens de uitvoering van de programmacode steeds naar andere objecten verwijzen.

U gebruikt de instructie **Toewijzen** om een willekeurig object dat door Visual Basic als zodanig wordt herkend, toe te wijzen aan een variabele. Alleen **Variant**-variabelen of variabelen die als **Object** of als een specifiek type object zijn gedeclareerd, kunnen verwijzen naar objecten. Verderop in dit gedeelte wordt dieper ingegaan op de mogelijke objecten.

```
Dim MijnRechthoek                ' Een Variant-variabele
Dim EenRechthoek Als Object      ' Een Object-variabele (een type object)

Toewijzen MijnRechthoek = ActiefBlad.ObjectenTekenen("Rechthoek 1")
Toewijzen EenRechthoek = ActiefBlad.ObjectenTekenen("Rechthoek 2")
```

Cellen en cellenbereiken (als objecten) toewijzen aan variabelen

Een van de meest voor de hand liggende objecten die u kunt toewijzen aan een variabele is een cel of een cellenbereik. Cellen maken deel uit van het objecttype Bereik.

Wanneer u een variabele gelijkstelt met een celverwijzing zonder gebruik te maken van de instructie **Toewijzen**, wordt de waarde uit die cel aan de variabele toegewezen en niet de celverwijzing.

```
' Slaat de waarde van cel A1 op in Verw
Verw = Werkbladen(1).Cellen(1; 1)
```

Als u de instructie **Toewijzen** gebruikt, bevat dezelfde variabele de celverwijzing in plaats van de waarde van de cel.

```
' Verw verwijst nu naar de cel zelf
Toewijzen Verw = Werkbladen(1).Cellen(1; 1)
```

In het voorgaande voorbeeld kunt u de variabele `Verw` net zo behandelen als de cel A1. U kunt bijvoorbeeld een aantal eigenschappen voor deze cel opgeven en zelfs een waarde in de cel opslaan.

```
Toewijzen Verw = Werkbladen(1).Cellen(1; 1)
Verw.Lettertype.Cursief = Waar
Verw.Lettertype.Vet = Waar
Verw.Lettertype.Naam = "Arial"
Verw.Lettertype.Grootte = 14
Verw.Waarde = Verw.Verschuiving(1; 0).Waarde
```

U kunt echter ook gebruik maken van de handzamere instructie **Met**.

```
Met Verw
    .Lettertype.Cursief = Waar
    .Lettertype.Vet = Waar
    .Lettertype.Naam = "Arial"
    .Lettertype.Grootte = 14
    .Waarde = Verw.Verschuiving(1; 0).Waarde
Einde Met
```

De complete syntaxis van de instructie **Toewijzen** bevat het sleutelwoord **Niets**.

Toewijzen *objectVar* = {*objectExpressie* | **Niets**}

De accoladen en het rechte streepje geven aan dat u een van de twee mogelijkheden *objectExpressie* of het sleutelwoord **Niets** moet kiezen. Als u een **Object**-variabele gelijk stelt aan **Niets** wordt elke relatie tussen *objectVar* en een object verbroken. Hierdoor komt de geheugenruimte vrij die voorheen was gereserveerd voor het object waarnaar werd verwezen. Bovendien wordt hierdoor voorkomen dat u per ongeluk wijzigingen aanbrengt in het object doordat u de variabele wijzigt. Het gebruik van **Niets** is meestal optioneel omdat Visual Basic zo nodig automatisch variabelen vrijmaakt.

```
Toewijzen Verw = Niets
```

Zie *Niets* in de online *Visual Basic Naslaggids* voor meer informatie over het sleutelwoord **Niets**.

Algemene en specifieke objecten toewijzen aan variabelen en argumenten

U kunt een variabele zo declareren dat deze verwijst naar een object in het algemeen, naar een specifiek object van Microsoft Excel, zoals een cellenbereik, een knop, een rechthoek of een werkblad, of naar een willekeurig OLE-object (Object Linking and Embedding). Zie hoofdstuk 10, " Samenwerken met andere toepassingen", voor meer informatie over OLE-objecten.

Het is aan te raden om waar mogelijk een specifiek objecttype op te geven in plaats van het algemene type **Object**. Hierdoor worden objecten automatisch op het type gecontroleerd, wordt de programmacode sneller uitgevoerd en wordt de leesbaarheid bevorderd. Hier volgt een aantal voorbeelden van specifieke objecttypen.

```
Dim EenObject Als Object
' Deze variabele kan naar een willekeurig object verwijzen
Dim KostenBereik Als Bereik
Dim MijnKnop Als Knop
Dim SamenvattingsBlad Als Werkblad

Toewijzen EenObject = ActiefBlad.ObjectenTekenen("Rechthoek 1")
Toewijzen KostenBereik = _
Werkbladen("Kostenbereik").Bereik(Cellen(1; 1); Cellen(3; 4))
Toewijzen MijnKnop = ActiefBlad.ObjectenTekenen("Knop4")
Toewijzen SamenvattingsBlad = Werkbladen("Samenvatting")
```

Zie *Objecten* in de online *Visual Basic Naslaggids* voor een volledige lijst met specifieke objecten waarnaar u kunt verwijzen in Microsoft Excel. Zoals u ziet worden veel objecten in het meervoud weergegeven (verzamelingen) en worden andere in het enkelvoud weergegeven (afzonderlijke objecten). U kunt bijvoorbeeld de variabele SamenvattingsBladen declareren als Werkbladen en de variabele SamenvattingsBlad als Werkblad.

```
Dim SamenvattingsBladen Als Werkbladen
' Gedeclareerd als verzameling werkbladen
Dim SamenvattingsBlad Als Werkblad
' Gedeclareerd als een werkblad-object
```

Nu kunt u een complete verzameling werkbladen in een bepaalde werkmap toewijzen aan SamenvattingsBladen en één blad aan SamenvattingsBlad.

```
Toewijzen SamenvattingsBladen = Werkmappen("SAMMAP.XLS").Werkbladen
Toewijzen SamenvattingsBlad = _
Werkmappen("SAMMAP.XLS").Werkbladen("Samenvatting")
```

Zie hoofdstuk 5, "Werken met objecten in Visual Basic", voor meer informatie over objecten.

In procedures kunt u ook de voorwaarde stellen dat een argument een object is. In dit geval gelden voor het argument dezelfde regels als voor een variabele. In het volgende voorbeeld wordt de verwijzing `Verw` uit het vorige voorbeeld gebruikt in een **Sub**-procedure.

```
Sub OpmaakEnKopie(Verw Als Bereik)
    Met Verw
        .Lettertype.Cursief = Waar
        .Lettertype.Vet = Waar
        .Lettertype.Naam = "Arial"
        .Lettertype.Grootte = 14
        .Waarde = Verw.Verschuiving(1; 0).Waarde
    Einde Met
Einde Sub
```

Opmerking Verwijzingen die u in variabelen opslaat, worden automatisch aangepast als de desbetreffende waarde wordt gewijzigd. Als u bijvoorbeeld de verwijzing `Toewijzen RijenVerw = Rijen(2)` invoegt, verwijst `RijenVerw` naar rij 2. Wanneer u vervolgens een nieuwe rij invoegt boven aan het werkblad, verwijst `RijenVerw` naar rij 3.

Operatoren in een expressie gebruiken

Een expressie is een combinatie van functies, getallen, variabelen en operatoren die gezamenlijk een waarde opleveren. De functies omvatten alle ingebouwde Visual Basic-functies, alle werkbladfuncties, alle invoegfuncties en alle door de gebruiker gedefinieerde functies. (Voordat u een invoegfunctie in Visual Basic kunt gebruiken, moet u een verwijzing naar de werkmap opnemen die deze invoegfunctie bevat. Zie "Een invoegmacro maken" in hoofdstuk 13 van dit boek voor meer informatie over invoegfuncties of hoofdstuk 37, "Invoegmacro's" in het *Microsoft Excel Handboek*). Er bestaan vier soorten operatoren:

- Wiskundige operatoren (+, –, /, *, ^, \ en **Rest**).
- Samenvoegingsoperatoren (**&**, ook wel de tekstoperator genoemd).

- Logische operatoren (**En**, **Eqv**, **Imp**, **Niet**, **Of** en **XOf**).
- Vergelijkingsoperatoren (=, >, <, >=, <= en < >).

  Twee andere operatoren vallen ook in de categorie vergelijkingsoperatoren: de equivalentie-operator (**Is**) en de gelijkenisoperator (**Zoals**).

De wiskundige, vergelijkings- en samenvoegingsoperatoren worden in expressies van Visual Basic op dezelfde wijze gebruikt als in werkbladformules. De logische operatoren worden beschreven in "De (on)waarheid van expressies vaststellen met logische operatoren". Alle wiskundige en vergelijkingsoperatoren (met uitzondering van \ en **Rest**) worden beschreven in hoofdstuk 10, "Formules en koppelingen maken", in het *Microsoft Excel Handboek*.

Met de operator \ kunt u twee getallen op elkaar delen. Het resultaat wordt weergegeven als een geheel getal. Voordat de deling plaatsvindt, worden de twee getallen afgerond op gehele getallen. In de volgende toewijzingsinstructie wordt aan `Resultaat` het getal 3 toegewezen.

```
Resultaat = 100,4 \ 32,3
```

De operator **Rest** geeft als resultaat de restwaarde van een deling, weergegeven als geheel getal. In de volgende toewijzingsinstructie wordt aan `Resultaat` het getal 4 toegewezen.

```
Resultaat = 100,4 Rest 32,3
```

Zie *Operatoren* in de online *Visual Basic Naslaggids* voor uitgebreide informatie over alle operatoren. Zie *Operatoren* in de online *Visual Basic Naslaggids* en kies vervolgens het onderwerp *Volgorde van de operatoren* voor meer informatie over de volgorde waarin de logische operaties worden getest.

De (on)waarheid van expressies vaststellen met logische operatoren

Logische operatoren vergelijken meestal twee expressies die **Waar** of **Onwaar** kunnen zijn. De vergelijking zelf vormt ook weer een expressie die resulteert in **Waar** of **Onwaar** volgens de regels die voor deze logische operator gelden. In de volgende tabel worden deze regels behandeld.

Syntaxis	Krijgt de waarde
expressie1 **En** *expressie2*	**Waar** als beide expressies **Waar** zijn, in alle andere gevallen **Onwaar**.
expressie1 **Eqv** *expressie2*	**Waar** als beide expressies hetzelfde zijn (allebei **Waar** of allebei **Onwaar**), in alle andere gevallen **Onwaar**.
expressie1 **Imp** *expressie2*	**Waar** tenzij *expressie1* **Waar** is en *expressie2* **Onwaar** is.
Niet *expressie*	**Waar** als *expressie* **Onwaar** is en **Onwaar** als deze **Waar** is.
expressie1 **Of** *expressie2*	**Waar** als een of beide expressies **Waar** zijn.
expressie1 **Xof** *expressie2*	**Onwaar** als beide expressies hetzelfde zijn, in alle andere gevallen **Waar**.

In het volgende voorbeeld wordt een bericht weergegeven dat afhankelijk is van de waarden van de variabelen A, B en C. Als A = 10, B = 8 en C = 6, worden beide uitdrukkingen **Waar**. Omdat beide expressies **Waar** zijn, is de **En**-expressie ook **Waar**.

```
A = 10: B = 8: C = 6                    ' Waarden toewijzen.
Indien (A > B) En (B > C) Dan           ' Expressies evalueren.
    Bericht = "Beide expressies zijn Waar."
Anders
    Bericht = "Ten minste een expressie is Onwaar."
Einde Indien
BerichtVenster Bericht                  ' Geef resultaten weer.
```

In het volgende voorbeeld controleert de operator **En** of de waarde van de variabele `Kosten` zich tussen twee waarden bevindt. De waarden worden door constanten gedefinieerd.

```
Const MAXKOSTEN = 1000; MINKOSTEN = 250
Dim Kosten Als Valuta; Inkomsten Als Valuta
Dim WinstVerw Als Bereik

Sub ContrKosten()
Toewijzen WinstVerw = Bereik("Winst")
Kosten = Bereik("Kosten").Waarde
Inkomsten = Bereik("Inkomsten").Waarde
    Indien Kosten < MAXKOSTEN En Kosten > MINKOSTEN Dan
        WinstVerw.Waarde = Inkomsten - Kosten
    Anders
        BerichtVenster "Kosten buiten het bereik, geen winst berekend."
        WinstVerw.Waarde = ""
    Einde Indien
Einde Sub
```

De operatoren **En**, **Niet** en **Of** zijn vergelijkbaar met de ingebouwde werkbladfuncties EN, NIET en OF, maar de syntaxis die wordt gehanteerd verschilt. **Eqv**, **Imp** en **Xof** zijn alleen beschikbaar in Visual Basic. Deze operatoren zijn niet direct beschikbaar in de werkbladen van Microsoft Excel.

Tip Als u de operatoren **Eqv**, **Imp** en **Xof** vaak gebruikt, kunt u ze beschikbaar maken in werkbladen door er door de gebruiker gedefinieerde functies van te maken. Hier volgt een voorbeeld van een dergelijke functie.

```
Functie XofFunctie (argument1; argument2)
    XofFunctie = argument1 Xof argument2
Einde Functie
```

De logische operatoren kunnen zowel gebruik maken van de waarde **Nulwaarde** als van de waarden **Waar** en **Onwaar**. Zie *Logische operatoren* in de online *Visual Basic Naslaggids* voor meer informatie over logische operatoren. Zie ook "Andere soorten informatie die in Variant-variabelen kunnen worden opgeslagen" verderop in dit hoofdstuk.

Objecten vergelijken met de equivalentie-operator (Is)

De operator **Is** wordt gebruikt om te bepalen of twee variabelen naar hetzelfde object verwijzen.

resultaat = *object1* **Is** *object2*

Als deze twee variabelen naar hetzelfde object verwijzen, levert de expressie de waarde **Waar** op. De objecten worden dan logisch *equivalent* genoemd. Het maakt geen verschil of de variabelen zijn gedeclareerd als **Variant**, **Object** of als een specifiek objecttype.

Tekenreeksen vergelijken met de gelijkenisoperator (Zoals)

De operator **Zoals** vergelijkt twee tekenreeksen.

resultaat = *tekenreeks* **Zoals** *patroon*

In deze syntaxis heeft het woord *patroon* betrekking op een tekstpatroon dat zich binnen in *tekenreeks* kan bevinden en dat jokertekens kan bevatten. De volgende instructie levert bijvoorbeeld de waarde **Waar** op.

```
Test = "Mysterie" Zoals "M*r"
```

Als *tekenreeks* overeenkomt met *patroon*, is *resultaat* **Waar**. Als er geen overeenstemming is, is *resultaat* **Onwaar**. En als *tekenreeks* of *patroon* de **Nulwaarde** bevat, krijgt *resultaat* ook de **Nulwaarde**. (*Tekenreeks* kan elke willekeurige tekenreeks-expressie zijn). Of er onderscheid wordt gemaakt tussen hoofdletters en kleine letters en welke sorteervolgorde voor de operator **Zoals** wordt gehanteerd, is afhankelijk van de instelling van de instructie **Optie Vergelijken**. Tenzij anders is vermeld, wordt **Optie Vergelijken Binair** gebruikt als de standaard vergelijkingsmethode voor elke module. Dat wil zeggen dat er onderscheid wordt gemaakt tussen hoofdletters en kleine letters. Zie *Optie Vergelijken* in de online *Visual Basic Naslaggids* voor meer informatie over dit onderwerp.

Ingebouwde jokertekens vormen een veelzijdig hulpmiddel bij het vergelijken van tekenreeksen. De jokertekens die u ter beschikking staan worden weergegeven in de volgende tabel.

Teken(s) in patroon	Komt in expressie overeen met
?	Eén willekeurig teken
*	Nul of meer tekens
#	Eén willekeurig cijfer (0 tot en met 9)
[*tekens*]	Eén willekeurig teken in *tekens*
[!*tekens*]	Eén willekeurig teken dat niet in *tekens* voorkomt

Als u een groep van een of meer tekens tussen vierkante haken ([ ]) plaatst, kunt u naar één teken in een expressie zoeken. U kunt hiervoor bijna alle tekens uit de tekenset van het American National Standards Institute (ANSI) gebruiken met inbegrip van cijfers. U kunt zelfs alleen naar de speciale tekens vierkante haak openen ([), vraagteken (?), hekje (#) en sterretje (*) zoeken door deze tussen vierkante haken te plaatsen. Het is niet mogelijk om naar de vierkante haak sluiten (]) in een groep te zoeken. Buiten een groep kunt u echter wel naar dit teken zoeken.

Zie *Zoals* in de online *Visual Basic Naslaggids* voor meer informatie over de operator **Zoals**.

Constanten gebruiken die waarden vertegenwoordigen

U zult vaak merken dat uw programmacode constante waarden bevat die telkens weer terugkomen. Of u bemerkt dat de programmacode afhankelijk is van bepaalde getallen die moeilijk te onthouden zijn: getallen die in zichzelf geen duidelijke betekenis hebben.

In dit soort gevallen kunt u de leesbaarheid van de programmacode aanzienlijk verbeteren door constanten te gebruiken. Bovendien kunt u uw programmacode dan gemakkelijker onderhouden. Een *constante* is een eenduidige naam die is toegekend aan een waarde. Ofschoon constanten veel weg hebben van variabelen, kunt u constanten niet wijzigen en kunt u geen nieuwe waarden aan constanten toewijzen, zoals u bij variabelen gewoon bent. Veel constanten zijn standaard beschikbaar in Visual Basic. U kunt echter ook uw eigen constanten definiëren.

Ingebouwde constanten gebruiken

Visual Basic bevat een groot aantal ingebouwde constanten die worden herkend als sleutelwoorden. De constanten die worden gebruikt door Microsoft Excel-objecten, worden voorafgegaan door de letters "xl". Constanten die door andere instructies en functies in Visual Basic worden gebruikt, worden meestal voorafgegaan door de letters "vb".

In de volgende instructies kunt u zien hoe u ingebouwde constanten kunt gebruiken.

```
' Stelt de berekeningsmodus in op handmatig
Toepassing.Berekening = xlHandmatig

' Haalt het valutasymbool op uit de matrix Internationale Eigenschappen
ValutaSymbool = Toepassing.Internationaal(xlValutaCode)

Antwoord = BerichtVenster("Wilt u doorgaan?"; vbJaNee)
' vbJaNee maakt een berichtvenster met de knoppen Ja en Nee.
Indien Antwoord = vbJa Dan
    Bericht = "U koos Ja."       ' vbJa is de programmacode voor de knop
Ja.
Anders
    Bericht = "U koos Nee."
Einde Indien
BerichtVenster Bericht
```

Ingebouwde constanten vertegenwoordigen meestal getallen. Welke getallen dit zijn, doet verder niet ter zake omdat Visual Basic deze informatie voor u bijhoudt. (xlHandmatig is 2; xlValutaCode is 25; vbJaNee is 4; vbJa is 6.) Het gebruik van constanten valt te verkiezen boven het gebruik van getallen, omdat de constanten gemakkelijker te onthouden zijn en duidelijker zijn in de programmacode.

In de afzonderlijke onderwerpen in de online *Visual Basic Naslaggids* worden de relevante constanten behandeld. Als u bijvoorbeeld *Berekening (eigenschap)* opzoekt in de *Visual Basic Naslaggids* ziet u dat de constante xlHandmatig en de overige verwante constanten in dit onderdeel worden besproken. Het onderwerp over het **BerichtVenster** bevat een tabel met de constanten die worden gebruikt voor het maken van verschillende soorten berichtvensters en de constanten die betrekking hebben op verschillende knoppen.

Uw eigen constanten maken

U kunt uw eigen constanten definiëren met de instructie **Const**.

Const *NAAMCONSTANTE* = *expressie*

Het argument *NAAMCONSTANTE* kan elke willekeurige geldige naam zijn (de regels voor dergelijke namen zijn hetzelfde als de regels voor de namen van variabelen. Zie "Meer informatie over procedures" verderop in dit hoofdstuk voor een beschrijving van deze regels). Het argument *expressie* is samengesteld uit verschillende numerieke constanten of tekenreeksconstanten en ingebouwde functies en operatoren. Het is een traditie onder programmeurs om hoofdletters te gebruiken voor door de gebruiker gedefinieerde constanten.

U kunt een constante ook declareren als een bepaald gegevenstype. Hoewel dit meestal niet nodig is, kan het de leesbaarheid van de programmacode en de nauwkeurigheid van bepaalde wiskundige bewerkingen verbeteren. (Een constante van het gegevenstype **Dubbel** geeft doorgaans een nauwkeuriger resultaat dan een constante van het gegevenstype **Enkel**).

Const *NAAMCONSTANTE* **Als** *type* = *expressie*

Omdat constanten meestal worden gebruikt door meerdere procedures in uw module, kunt u het beste alle **Const**-instructies samen boven aan de module boven de eerste regel van de procedure (de definitie-regel) plaatsen. (Als u ze in een procedure plaatst, zijn deze constanten alleen beschikbaar in die procedure en als u ze als openbare constanten opgeeft, zijn ze beschikbaar in alle procedures van alle modulen. Zie "Variabelen beschikbaar maken in een procedure of module, of openbare variabelen maken" verderop in dit hoofdstuk waar ook meer informatie te vinden is over de beschikbaarheid van constanten).

```
' Begin module
'
' Constanten definiëren voor deze module
' Deze constanten hebben betrekking op getallen of datum/tijd:
Const PI = 3,14159265
Const MAX_PLANETEN = 9
Const UITGIFTE_DATUM = #1-01-94#
'
' Deze constanten hebben betrekking op tekenreeksen:
Const VERSIE = "07.10.A"
Const PROGRAMMACODE_NAAM = "Enigma"

' Begin procedure-definities
Sub EersteSub ()        ' Eerste procedure in de module
    .
    .
    .
```

U kunt in de declaratie van constanten meer dan één constante op een regel plaatsen. U moet deze dan wel door puntkomma's of door andere lijstscheidingstekens van elkaar scheiden.

```
Const PI = 3,14; MAX_PLANETEN = 9; AVOGADRO = 6.0225E+2
```

De expressie aan de rechterzijde van het "is gelijk"-teken (=) is vaak een getal of een tekenreeks met letters, maar het kan ook een expressie zijn met operatoren die in een getal of een tekenreeks resulteren (die expressie kan echter geen procedures oproepen).

U kunt zelfs constanten definiëren met eerder gedefinieerde constanten.

```
Const PI2 = PI * 2
```

Instructies vereenvoudigen met benoemde argumenten

Voor de meeste ingebouwde functies, instructies en methoden van Visual Basic kunt u direct benoemde argumenten gebruiken als een alternatief voor het invoeren van waarden en puntkomma's (of andere lijstscheidingstekens). Een *benoemd argument* is een argumentnaam die Visual Basic kan herkennen. In plaats van voor elk argument een aparte waarde, gescheiden door komma's, op te geven in de volgorde die is vastgelegd in de syntaxis, kunt u waarden toewijzen aan specifieke benoemde argumenten. De methode **NamenGebruiken** heeft bijvoorbeeld de volgende zeven argumenten.

object.**NamenGebruiken**(***Namen; RelatiefAbsoluutNegeren; RijKolomNamenGebruiken; KolomWeglaten; RijWeglaten; Volgorde; ToevoegenLaatste***)

Zonder benoemde argumenten zou u op de volgende wijze de naam `Omzet` moeten toewijzen aan het bereik dat door `NaamVerw` wordt gedefinieerd.

```
Toewijzen NaamVerw = Werkbladen("Blad1").Bereik("A1:C3")
NaamVerw.NamenGebruiken "Omzet";;;;;;Waar
```

Met benoemde argumenten kunt u een of meer argumenten in een willekeurige volgorde invoeren door een waarde aan het benoemde argument toe te wijzen met een dubbele punt en een "is gelijk"-teken (:=) en tussen de verschillende argumenten met de bijbehorende waarden puntkomma's te plaatsen. Het vorige voorbeeld ziet er als volgt uit wanneer u gebruik maakt van benoemde argumenten:

```
Toewijzen NaamVerw = Werkbladen("Blad1").Bereik("A1:C3")
NaamVerw.NamenGebruiken ToevoegenLaatste:=Waar; Namen:="Omzet"
```

Zoals u ziet staan de argumenten in het tweede voorbeeld in omgekeerde volgorde.

Ook in de door u gemaakte procedures kunt u benoemde argumenten gebruiken. Visual Basic koppelt automatisch de namen van de argumenten aan de bijbehorende procedures. Dit is vooral handig als uw procedures verschillende optionele argumenten bevatten die u niet altijd hoeft op te geven.

Belangrijk U kunt geen gebruik maken van benoemde argumenten om het invoeren van vereiste argumenten te omzeilen. Met benoemde argumenten kunt u bepaalde argumenten in een willekeurige volgorde identificeren, maar u kunt alleen de optionele argumenten uit een procedure weglaten. Zie de gewenste functie, instructie of methode in de online *Visual Basic Naslaggids* als u wilt weten welke argumenten vereist en welke optioneel zijn.

Meer leren over programmeren met Visual Basic

Visual Basic is een krachtige programmeertaal met vele voorzieningen. U hoeft deze echter niet allemaal te beheersen om Visual Basic te kunnen gebruiken. In de volgende gedeelten van dit hoofdstuk worden de gegevenstypen, de procedures en de matrices uitvoeriger behandeld.

- Meer informatie over het gegevenstype **Variant** en andere gegevenstypen
- Andere soorten informatie die in **Variant**-variabelen kunnen worden opgeslagen
- Een gegevenstype converteren naar een ander gegevenstype
- Uw eigen gegevenstypen maken
- Variabelen beschikbaar maken in een procedure of module, of openbare variabelen maken
- Hoe lang behouden variabelen hun waarden?
- Constanten beschikbaar maken in een procedure of module, of openbare constanten maken
- Meer informatie over procedures
- Waarden opslaan in Visual Basic-matrices
- Eigenschapsprocedures gebruiken

Als u geen behoefte denkt te hebben aan deze informatie kunt u verder gaan met hoofdstuk 7, "De uitvoering van de programmacode besturen".

Meer informatie over het gegevenstype Variant en andere gegevenstypen

Het gegevenstype **Variant**, dat eerder in dit hoofdstuk is beschreven in "Met het gegevenstype Variant verschillende typen gegevens opslaan", kan verschillende soorten gegevenstypen afhandelen en automatisch converteren. Als u echter een beknopte, snelle programmacode wilt schrijven, doet u er goed aan om specifieke gegevenstypen te gebruiken. De kenmerken en de voordelen van het gegevenstype **Variant** en een aantal specifieke gegevenstypen worden in de volgende gedeelten van dit hoofdstuk uitgelegd.

Numerieke gegevenstypen

Als u er zeker van bent dat in een variabele alleen maar gehele getallen worden opgeslagen (zoals 12) in plaats van reële getallen (getallen met een breuk, zoals 3,57) kunt u deze variabele declareren als het gegevenstype **Integer** of **Lang**. Bewerkingen verlopen sneller wanneer u het gegevenstype **Integer** hebt gebruikt en bovendien neemt dit gegevenstype minder geheugenruimte in beslag dan het gegevenstype **Variant**. Het gegevenstype **Integer** is met name handig voor telvariabelen in **Voor...Volgende**-lussen.

Als de variabele een breuk bevat, kunt u deze variabele declareren als **Enkel**, **Dubbel** of **Valuta**. In het gegevenstype **Valuta** kunnen maximaal vier cijfers rechts van de decimale komma worden geplaatst en maximaal vijftien cijfers links ervan. Het is een snel en nauwkeurig gegevenstype met een vaste komma dat met name geschikt is voor de berekening van geldbedragen. Getallen met een zwevende komma (**Enkel** en **Dubbel**) hebben een veel groter bereik dan getallen van het gegevenstype **Valuta** maar zijn onderhevig aan kleine afrondingsfouten.

Alle numerieke variabelen kunnen toegewezen worden aan elkaar en aan variabelen van het gegevenstype **Variant**. In Visual Basic wordt een breuk afgerond (in plaats van afgekapt) wanneer een getal met een zwevende komma wordt toegewezen aan een geheel getal.

Numerieke waarden die zijn opgeslagen in Variant-variabelen

Als u getallen opslaat in een **Variant**-variabele, gebruikt Visual Basic de meest compacte representatie die mogelijk is. Als u dus een klein getal zonder decimale breuk opslaat, wordt deze waarde opgeslagen in dezelfde representatie als een **Integer**. Als u vervolgens een langer getal aan deze variabele toewijst, zal Visual Basic dezelfde representatie als **Lang** of (als het getal zeer lang is of over een decimale component beschikt) **Dubbel** gebruiken.

Als u een **Variant**-variabele die een getal bevat, toewijst aan een tekenreeksvariabele of een eigenschap, converteert Visual Basic automatisch de representatie van het getal naar een tekenreeks. U kunt een getal expliciet converteren naar een tekenreeks met de functie **CTreeks**. U kunt ook gebruik maken van de functie **Notatie**, waarmee u notatiekenmerken zoals een valutasymbool, een scheidingsteken voor duizendtallen en een decimaalscheidingsteken kunt instellen. (De functies **CTreeks** en **Notatie** maken beide gebruik van de instellingen voor het decimaalscheidingsteken en de andere symbolen zoals die in de landinstellingen van uw besturingssysteem zijn vastgelegd).

Zie *Conversie* in de online *Visual Basic Naslaggids* voor meer informatie over het converteren van gegevenstypen. Zie *Notatie* in de online *Visual Basic Naslaggids* voor meer informatie over de functie **Notatie**.

Tekenreekstypen

Standaard maken variabelen of argumenten met een tekenreeks (of tekst) gebruik van een *tekenreeks met variabele lengte*. De tekenreeks wordt groter of kleiner naarmate u meer of minder tekens aan de variabele toewijst. U kunt echter ook een vaste lengte voor een tekenreeks opgeven. Voor een *tekenreeks met vaste lengte* gebruikt u de volgende syntaxis.

Tekenreeks * *lengte*

In het volgende voorbeeld ziet u een declaratie voor een tekenreeks die altijd 50 tekens lang is.

```
Dim VoorNaam Als Tekenreeks * 50
```

Als u een tekenreeks met minder dan 50 tekens aan deze variabele toewijst, wordt `VoorNaam` opgevuld met voldoende spaties om de lengte van 50 tekens vol te maken. Als u een tekenreeks opgeeft die langer is dan 50 tekens, worden de overtollige tekens afgekapt.

```
Dim Treeks4 Als Tekenreeks * 4; T Als Tekenreeks
T = "Leverworst"
Foutopsp.Afbeelden T
Treeks4 = T
Foutopsp.Afbeelden Treeks4
```

De voorgaande programmacode levert het volgende resultaat op in het deelvenster Direct van het venster Foutopsporing. Zie "Het venster Foutopsporing gebruiken" in hoofdstuk 8 voor meer informatie over dit venster.

```
Leverworst
Leve
```

Omdat tekenreeksen met een vaste lengte vaak gedeeltelijk aangevuld zijn met spaties, valt het aan te raden om gebruik te maken van de functies **SpatiesWissen** en **SpatiesRWissen**. Zie *SpatiesWissen* en *SpatiesRWissen* in de online *Visual Basic Naslaggids* voor meer informatie over de functies **SpatiesWissen** en **SpatiesRWissen**.

Opmerking De instructie **Optie Vergelijken** die u gebruikt in een Visual Basic-module (**Optie Vergelijken Binair** of **Optie Vergelijken Tekst**) bepaalt wat in een gelijkheidsinstructie het resultaat is van de vergelijking van afzonderlijke tekens tussen verschillende tekenreeksen, tussen tekenreeksen en **Variant**-variabelen, en tussen **Variant**-variabelen die tekenreeksen bevatten onderling.

Standaard kiest Visual Basic **Optie Vergelijken Binair**, zodat bij vergelijkingen tussen tekenreeksen onderscheid wordt gemaakt tussen hoofdletters en kleine letters. Maar wanneer u de standaard ANSI-tekens gebruikt (0 tot en met 127), kunt u gebruik maken van de instructie **Optie Vergelijken Tekst**, zodat Visual Basic bij vergelijkingen geen onderscheid maakt tussen hoofdletters en kleine letters. Zie *Optie Vergelijken* in de online *Visual Basic Naslaggids* voor meer informatie over vergelijkingen van binaire gegevens en tekstgegevens en met name over het gebruik van dergelijke vergelijkingen in andere talen en met andere ANSI-tekens.

Tekenreeksen die zijn opgeslagen in Variant-variabelen

De werking van de operator + kan bij verschillende **Variant**-waarden variëren. Als beide **Variant**-variabelen getallen bevatten, voert de operator + een optelling uit. Als beide **Variant**-variabelen tekenreeksen bevatten, voegt de operator + deze tekenreeksen samen. Als één van de variabelen een getal bevat en de andere een tekenreeks, wordt eerst geprobeerd deze tekenreeks om te zetten in een getal. Als deze conversie lukt, voert de operator + een optelling uit en als de conversie mislukt, voegt de operator + de twee waarden samen, zodat een nieuwe tekenreeks ontstaat.

Wanneer u tekst wilt samenvoegen, kunt u beter de operator **&** gebruiken in plaats van de operator +. Bij het invoeren van programmacode moet u erop letten dat u een spatie typt tussen de naam van een variabele en de operator **&**. Als u deze spatie niet typt, interpreteert Visual Basic het teken **&** als het declaratieteken voor het gegevenstype (lange gehele getallen) van de variabele. Zie *Declaratieteken voor het gegevenstype* in de online *Visual Basic Naslaggids* voor meer informatie over het declaratieteken voor het gegevenstype.

Datums en tijden (het Datumtype)

In dit gedeelte wordt beschreven hoe in Visual Basic datums en tijden worden afgehandeld. Dit verschilt namelijk van de manier waarop in Microsoft Excel-werkbladen datums en tijden worden afgehandeld.

In Visual Basic worden **Datum**-variabelen opgeslagen als getallen van 64 bits (8 bytes) die betrekking hebben op de datums van 1 januari 0100 tot en met 31 december 9999. Tijden worden opgeslagen als 0:00:00 tot en met 23:59:59. Aan **Datum**-variabelen kunnen alle datumwaarden met tekst worden toegewezen die als zodanig herkenbaar zijn. Datums met tekst moeten tussen hekjes (#) worden geplaatst: #1 januari 1993# of #1 jan 93# zijn voorbeelden van datums met tekst. U kunt een **Variant**-variabele die een datum of tijdwaarde bevat, vergelijken met een datum met tekst.

```
Indien EenDatum > #3-06-93# Dan
Indien EenDatum > #3-06-93 13:20# Dan
```

Visual Basic kan een grote verscheidenheid aan datum- en tijdnotaties in tekst verwerken. De volgende voorbeelden zijn allemaal geldige datum- en tijdwaarden.

```
EenDatum = #3-06-93 13:20#
EenDatum = #27 maart 1993 1:20#
EenDatum = #2-apr-93#
EenDatum = #4 april 1993#
```

Datum-variabelen geven datums weer in de korte datumnotatie die wordt herkend door uw computer. Tijden worden weergegeven in de tijdnotatie die door uw computer wordt herkend (de 12- of de 24-uursnotatie).

Wanneer andere numerieke gegevenstypen worden geconverteerd naar het gegevenstype **Datum**, wordt de waarde links van het decimaalteken als datum geïnterpreteerd en de waarde rechts van het decimaalteken als tijd. Middernacht is 0 en middag is 0,5. Negatieve gehele getallen worden geïnterpreteerd als de datums voor 30 december 1899.

Variant-variabelen kunnen ook datum- en tijdwaarden bevatten. Voor deze variabelen gelden dezelfde algemene kenmerken als voor variabelen van het gegevenstype **Datum**. Houd echter rekening met het volgende.

Belangrijk Probeer altijd het gegevenstype **Datum** op te geven voor een variabele of een argument met een datum of een door de gebruiker gedefinieerde functie die in een datum resulteert.

```
Dim HuidigeDatum Als Datum
```

Als u dit niet doet, herkent Visual Basic de datums in de cellen van een werkblad niet, tenzij u deze hebt opgemaakt in een ingebouwde datumnotatie. Wanneer u een variabele in een door de gebruiker gedefinieerde functie hebt gedeclareerd met het gegevenstype **Datum**, past Microsoft Excel bovendien automatisch de datumnotatie toe op de cel waarin het resultaat van deze functie wordt geplaatst.

Verschillende functies kunnen resulteren in datum- en tijdwaarden.

```
Sub DatumTijdDemo ()
    Dim NuDirect; DagenOver; UrenOver; MinutenOver

    NuDirect = Nu                    ' Nu levert huidige datum/tijd op.
    DagenOver = Int(DatumSerieel(Jaar(NuDirect) + 1; 1; 1) - NuDirect)
    UrenOver = 24 - Uur(NuDirect)
    MinutenOver = 60 - Minuut(NuDirect)
    BerichtVenster DagenOver & " dagen over in dit jaar."
    BerichtVenster UrenOver & " uren over in deze dag."
    BerichtVenster MinutenOver & " minuten over in dit uur."
Einde Sub
```

U kunt op datum- en tijdwaarden ook berekeningen uitvoeren. Wanneer u gehele getallen bij de waarde optelt of van de waarde aftrekt, worden dagen bij de datum opgeteld of van de datum afgetrokken. Door een breuk op te tellen of af te trekken, wordt een bepaald gedeelte van de dag bij de tijd opgeteld of van de tijd afgetrokken. Als u bijvoorbeeld 20 optelt bij de datumwaarde, worden 20 dagen bij de datum opgeteld en als u 1/24 bij een tijdwaarde optelt, wordt één uur bij de tijd opgeteld. De waarde voor één minuut is 1/1440 en de waarde voor één seconde is 1/86400.

Zie *Datum* in de online *Visual Basic Naslaggids* voor meer informatie over verschillende onderwerpen die betrekking hebben op datums en tijden.

Opmerking Als u programmacode schrijft of bewerkt kunt u in de keuzelijst "Land/Taal" van het tabblad Module van het dialoogvenster **Opties** (menu **Extra**) de landinstellingen kiezen die worden gebruikt voor het decimaalteken, het lijstscheidingsteken, het valutasymbool en de datumnotatie. Het decimaalteken, het lijstscheidingsteken, het valutasymbool en de datumnotatie die gebruikers te zien krijgen wanneer ze uw programmacode uitvoeren, zijn afhankelijk van de landinstellingen van hun besturingssysteem.

In de codevoorbeelden in dit boek wordt ervan uitgegaan dat u gebruik maakt van de Nederlandse standaardinstellingen. Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over het schrijven van toepassingen die in andere landen gebruikt moeten kunnen worden.

Boole-variabelen

Boole-variabelen worden opgeslagen als getallen van 16 bits (2 bytes). Deze variabelen kunnen echter alleen de waarden **Waar** of **Onwaar** bevatten.

Wanneer u vergelijkingen uitvoert in Visual Basic, worden de waarden **Waar** en **Onwaar** vaak impliciet gebruikt. De volgende twee regels programmacode zijn bijvoorbeeld gelijkwaardig.

```
Indien (Inkomsten > Kosten) Dan BerichtVenster "Goed nieuws!"

Indien (Inkomsten > Kosten) = Waar Dan BerichtVenster "Goed nieuws!"
```

Wanneer andere numerieke gegevenstypen worden geconverteerd naar **Boole**-waarden, wordt nul geconverteerd naar **Onwaar** en alle andere waarden in **Waar**. Wanneer **Boole**-waarden worden geconverteerd naar andere gegevenstypen, wordt **Onwaar** omgezet in nul en **Waar** in –1. Bij het vergelijken van **Boole**-waarden met getallen, wordt de **Boole**-waarde eerst geconverteerd. `1 = Waar`, resulteert bijvoorbeeld in **Onwaar**, omdat **Waar** wordt geconverteerd naar -1 en `1 = -1` **Onwaar** is.

Object-variabelen

Object-variabelen worden opgeslagen als adressen van 32 bits (4 bytes). Deze adressen verwijzen naar objecten binnen een toepassing. Een variabele die als **Object** is gedeclareerd, kan vervolgens (met de instructie **Toewijzen**) worden toegewezen aan elk object dat als zodanig door de toepassing wordt herkend.

Andere soorten informatie die in Variant-variabelen kunnen worden opgeslagen

Zoals al eerder is gezegd, kunnen variabelen van het gegevenstype **Variant** veel verschillende soorten gegevens bevatten: getallen, tekenreeksen met tekst, Boole-waarden, objecten, datums en tijden, enzovoorts. Voor al deze gegevens kunt u ook een specifiek gegevenstype gebruiken. **Variant**-variabelen kunnen echter nog andere soorten informatie bevatten. In de volgende gedeelten komen deze aan de orde.

De lege waarde

In bepaalde omstandigheden is het van belang om te weten of een variabele ooit een waarde heeft bevat of om te bepalen of een cel leeg is. Een **Variant**-variabele heeft de waarde **Leeg** voordat er een waarde aan wordt toegewezen. De waarde **Leeg** is een speciale waarde die verschilt van nul, een lege tekenreeks met de lengte nul ("") of de Nulwaarde. Met de functie **IsLeeg** kunt u testen of een variabele de waarde **Leeg** bevat.

```
Indien IsLeeg(Z) Dan Z = 0

Indien IsLeeg(Cellen(1; 1).Waarde) Dan Cellen(1; 1).Waarde = 0
```

Wanneer een **Variant**-variabele de waarde **Leeg** bevat, kunt u deze variabele in expressies gebruiken. Afhankelijk van de expressie wordt deze variabele gezien als nul of als een tekenreeks met de lengte nul.

De waarde **Leeg** verdwijnt zodra er een waarde aan een **Variant**-variabele wordt toegekend (inclusief de waarde nul, een tekenreeks met de lengte nul of de waarde **Nulwaarde**). U kunt opnieuw de waarde **Leeg** instellen voor een **Variant**-variabele door een andere **Variant**-variabele met de waarde **Leeg** of het sleutelwoord **Leeg** aan deze variabele toe te wijzen.

De Nulwaarde

Het gegevenstype **Variant** kan een andere speciale waarde bevatten: de **Nulwaarde**. De **Nulwaarde** wordt algemeen gebruikt in database-toepassingen om onbekende of ontbrekende gegevens aan te duiden. Vanwege de manier waarop deze waarde wordt gebruikt in databases, beschikt de **Nulwaarde** over de volgende unieke kenmerken.

- Expressies waarin de **Nulwaarde** wordt gebruikt, geven altijd de **Nulwaarde** als resultaat. De **Nulwaarde** kan zich dus als het ware *voortplanten* door de expressies. Als een onderdeel van een expressie de **Nulwaarde** krijgt, krijgt de volledige expressie de **Nulwaarde**.
- Wanneer u een **Nulwaarde**, een **Variant**-variabele met een **Nulwaarde** of een expressie die een **Nulwaarde** krijgt, als een argument invoert, resulteren de meeste functies in de **Nulwaarde**.
- **Nulwaarden** kunnen zich voortplanten door intrinsieke functies die resulteren in waarden van het gegevenstype **Variant**.

U kunt ook een **Nulwaarde** toewijzen met het sleutelwoord **Nulwaarde**.

```
Z = Nulwaarde
```

Met de functie **IsNulwaarde** kunt u bepalen of een **Variant**-variabele de **Nulwaarde** bevat.

```
Indien IsNulwaarde(X) En IsNulwaarde(Y) Dan
    Z = Nulwaarde
Anders
    Z = 0
Einde Indien
```

Als u de **Nulwaarde** toewijst aan een variabele van een ander gegevenstype dan **Variant**, levert dit een fout op die kan worden opgespoord. Zie hoofdstuk 9, "Foutafhandeling en foutwaarden", voor meer informatie over het opsporen van fouten.

Het toewijzen van de **Nulwaarde** aan een **Variant**-variabele leidt niet tot een fout. De **Nulwaarde** plant zich voort door expressies die gebruik maken van **Variant**-variabelen (er bestaan echter functies die de **Nulwaarde** niet doorgeven). Elke **Functie**-procedure met een resultaatwaarde die gebruik maakt van het gegevenstype **Variant** kan resulteren in de **Nulwaarde**.

De **Nulwaarde** wordt niet automatisch toegewezen aan variabelen, tenzij u expliciet **Nulwaarde** voor een variabele opgeeft. Als u de waarde **Nulwaarde** niet gebruikt in uw toepassing, hoeft u geen programmacode te schrijven die op deze waarde test en deze verder afhandelt.

Zie *Is Nulwaarde* in de online *Visual Basic Naslaggids* voor meer informatie over **Nulwaarde** en over het testen op de waarde **Nuiwaarde**.

Het subtype Fout

Als u uw eigen foutwaarden maakt met de functie **CVarFout** en u deze wilt opslaan in variabelen, moeten deze variabelen gebruik maken van het gegevenstype **Variant**. Wanneer u foutwaarden hebt toegewezen aan deze **Variant**-variabelen, worden ze ingedeeld bij het subtype **Fout**. In het volgende voorbeeld is `MijnFout` een **Variant**-variabele van het subtype **Fout**.

```
Dim MijnFout
MijnFout = CVarFout(2010)
```

Ook al is `MijnFout` een **Variant**-variabele, wanneer u deze variabele gebruikt als argument van de functie **TypeNaam**, resulteert de functie **TypeNaam** in "Fout" en niet in "Variant". In dit geval kunt u met de functie **TypeNaam** vaststellen of een variabele een door de gebruiker gedefinieerde fout bevat. U kunt ook de functie **IsFout** gebruiken om foutwaarden op te sporen.

Zie "Foutwaarden maken die programmacode niet onderbreken" in hoofdstuk 9 voor meer informatie over door de gebruiker gedefinieerde fouten. Zie ook *CVarFout* of *IsFout* in de online *Visual Basic Naslaggids*.

Overige belangrijke kenmerken van Variant-variabelen

Waar nodig zullen in de rest van dit hoofdstuk de **Variant**-variabelen worden behandeld. Voorlopig is het voldoende om rekening te houden met de volgende extra kenmerken van **Variant**-variabelen.

- **Variant**-variabelen kunnen verwijzen naar objecten.
- Eén afzonderlijke **Variant**-variabele kan een volledige matrix bevatten. Deze matrix kan op zijn beurt **Variant**-variabelen bevatten of variabelen die gebruik maken van andere gegevenstypen. Zie "Waarden opslaan in Visual Basic-matrices" verderop in dit hoofdstuk voor meer informatie over matrices.

- **Variant**-variabelen kunnen geen door de gebruiker gedefinieerde gegevenstypen (records) of matrices met door de gebruiker gedefinieerde gegevenstypen bevatten. Zie "Uw eigen gegevenstypen maken" verderop in dit hoofdstuk voor meer informatie over dit onderwerp.

Een gegevenstype converteren naar een ander gegevenstype

Soms zult u een getal op een bepaalde manier willen weergeven. U kunt het gegevenstype van een variabele converteren naar een specifiek gegevenstype met een van de functies C*type*. Zo kunt u bijvoorbeeld een **Variant**-variabele opslaan als een **Valuta**-variabele om afrondingsfouten in latere berekeningen te voorkomen. Visual Basic beschikt over een aantal conversiefuncties die u kunt gebruiken om waarden naar een bepaald gegevenstype te converteren. Om een waarde te converteren naar het gegevenstype **Valuta** gebruikt u bijvoorbeeld de functie **CValuta**.

```
LoonPerWeek = CValuta(uren * uurLoon)
```

U kunt de functie **CDatum** gebruiken om een tekstwaarde om te zetten in een datum- en tijdwaarde. In de volgende programmacode wordt de eigenschap Waarde van een cel getest met de functie **IsDatum**. Als de eigenschap een tekst bevat die als een datum of tijd beschouwd kan worden, converteert Visual Basic de tekst in die cel naar een datum en worden de dagen tot het einde van het jaar geteld.

```
Dim EenDatum; DagenOver
Dim CelVerw; CelVerw2 Als Bereik
Toewijzen CelVerw = Werkbladen(1).Cellen(1; 1)
Toewijzen CelVerw2 = Werkbladen(2).Cellen(2; 1)

Indien IsDatum(CelVerw.Waarde) Dan
    EenDatum = CDatum(CelVerw.Waarde)
    Dagenover = DatumSerieel(Jaar(EenDatum) + 1; 1; 1) - EenDatum
    CelVerw2.Waarde = DagenOver & " dagen over dit jaar."
Anders
    BerichtVenster CelVerw.Waarde & " is geen geldige datum."
Einde Indien
```

Visual Basic beschikt over conversiefuncties voor de gegevenstypen **Boole**, **Valuta**, **Dubbel**, **Enkel**, **Lang**, **Integer**, **Tekenreeks**, **Datum** en **Variant**. Zie *Conversie* in de online *Visual Basic Naslaggids* voor meer informatie over dit onderwerp. Zie *Datum*, *IsDatum* en *DatumSerieel* in de online *Visual Basic Naslaggids* voor meer informatie over datums.

Uw eigen gegevenstypen maken

U kunt verschillende gegevenstypen combineren en zo uw eigen gegevenstypen maken. Door de gebruiker gedefinieerde gegevenstypen zijn vooral handig wanneer u een afzonderlijke variabele wilt maken waarin verschillende soorten informatie kunnen samenkomen.

Opmerking Ervaren programmeurs zijn wellicht beter bekend met de term *structuren*, ook bekend onder de naam *structs* in de programmeertaal C en *records* in Pascal. *Door de gebruiker gedefinieerd gegevenstype* is de term die in deze handleiding wordt gebruikt voor hetzelfde begrip.

U maakt een door de gebruiker gedefinieerd gegevenstype met de instructie **Type** die boven aan een Visual Basic-module moet worden geplaatst. Door de gebruiker gedefinieerde gegevenstypen zijn altijd openbaar. (Openbare variabelen komen aan de orde in het volgende gedeelte, "Variabelen beschikbaar maken in een procedure of module, of openbare variabelen maken"). Maar een variabele waarvoor u een door de gebruiker gedefinieerd gegevenstype in de declaratie opgeeft, kan lokaal of openbaar zijn of het kan een variabele op module-niveau zijn. Zo kunt u bijvoorbeeld het volgende gegevenstype definiëren, waarin de gegevens over een computersysteem zijn opgeslagen.

```
' Declaraties
Type SysteemInfo
    CVE Als Variant
    Geheugen Als Lang
    MonitorKleuren Als Integer
    Kosten Als Valuta
    AankoopDatum Als Variant
Einde Type
```

Voor dit door de gebruiker gedefinieerde gegevenstype kunt u variabelen lokaal, openbaar of op module-niveau declareren.

```
Dim MijnSysteem Als SysteemInfo; UwSysteem Als SysteemInfo
```

De waarden van de elementen van deze variabele worden net zo toegewezen en verkregen als eigenschappen worden ingesteld en opgehaald.

```
MijnSysteem.CVE = "486"
Indien MijnSysteem.AankoopDatum > #1-01-94# Dan
```

U kunt ook de ene variabele aan de andere toewijzen als ze beide hetzelfde door de gebruiker gedefinieerde gegevenstype hebben. Hierdoor worden alle elementen van de ene variabele aan dezelfde elementen van de andere variabele toegewezen.

```
UwSysteem = MijnSysteem
```

Een door de gebruiker gedefinieerd gegevenstype kan een gewone matrix (met een vaste grootte) bevatten.

```
Type SysteemInfo
    CVE Als Variant
    Geheugen Als Lang
    DisketteStations(25) Als Tekenreeks
    ' Matrix met een vaste grootte
    MonitorKleuren Als Integer
    Kosten Als Valuta
    AankoopDatum Als Variant
Einde Type
```

Een door de gebruiker gedefinieerd gegevenstype kan ook een matrix met een variabele grootte bevatten. (Zie *Redim* in de online *Visual Basic Naslaggids* voor meer informatie over matrices met variabele grootte.)

```
Type SysteemInfo
    CVE Als Variant
    Geheugen Als Lang
    DisketteStations() Als Tekenreeks
    ' Matrix met een variabele grootte
    MonitorKleuren Als Integer
    Kosten Als Valuta
    AankoopDatum Als Variant
Einde Type
```

U kunt de waarden in een matrix van een door de gebruiker gedefinieerd gegevenstype op dezelfde wijze benaderen als een eigenschap van een object.

```
Dim MijnSysteem Als SysteemInfo
MijnSysteem.DisketteStations(1) = "1,44 MB"
```

U kunt ook een matrix met door de gebruiker gedefinieerde gegevenstypen declareren.

```
Dim AlleSystemen(100) Als SysteemInfo
```

Volg dezelfde regels om de componenten van de gegevensstructuur te benaderen.

```
AlleSystemen(5).CVE = "486D50"
AlleSystemen(X).DisketteStations(2) = "2,88 MB"
```

Het nesten van gegevensstructuren kan zo complex worden als u zelf wilt. (Nesten betekent een gegevensstructuur binnen een andere gegevensstructuur opnemen). Zo kunnen door de gebruiker gedefinieerde gegevenstypen andere door de gebruiker gedefinieerde gegevenstypen bevatten.

```
Type StationInfo
    Type Als Tekenreeks
    Grootte Als Lang
Einde Type

Type SysteemInfo
    CVE Als Variant
    Geheugen Als Lang
    DisketteStations(26) Als StationInfo
    Kosten Als Valuta
    AankoopDatum Als Variant
Einde Type

Dim AlleSystemen(100) Als SysteemInfo
AlleSystemen(1).DisketteStations(0).Type = "Floppy"
```

U kunt een procedure-argument declareren als een door de gebruiker gedefinieerd gegevenstype.

```
Sub VulSysteemIn (EenSysteem Als SysteemInfo)
    EenSysteem.CVE = lstCVE.Tekst
    EenSysteem.Geheugen = txtGeheugen.Tekst
    EenSysteem.Kosten = txtKosten.Tekst
    EenSysteem.AankoopDatum = Nu
Einde Sub
```

U kunt een door de gebruiker gedefinieerd gegevenstype gebruiken voor de resultaatwaarde van een functie. Daarnaast kunt u een door de gebruiker gedefinieerd gegevenstype gebruiken voor een variabele die als argument in een procedure wordt ingevoerd. Door de gebruiker gedefinieerde gegevenstypen worden altijd op verwijzingen doorgegeven, zodat de procedure het argument kan wijzigen en vervolgens kan retourneren aan de oproepende procedure, zoals u in het vorige voorbeeld hebt kunnen zien. Zie "Meer informatie over procedures" verderop in dit hoofdstuk voor meer informatie over het doorgeven van waarden op verwijzingen.

Opmerking De maximale grootte van een record in een door de gebruiker gedefinieerd gegevenstype mag niet meer dan 65.535 bytes bedragen.

Variabelen beschikbaar maken in een procedure of module of openbare variabelen maken

Wanneer u een variabele declareert binnen een procedure, kan alleen de programmacode binnen die procedure de waarde van de variabele benaderen of wijzigen. Soms zult u echter een variabele met een groter bereik willen gebruiken, bijvoorbeeld een variabele in alle procedures in dezelfde Visual Basic-module of zelfs in alle procedures in een werkmap. Het *bereik* van een variabele is de beschikbaarheid van die variabele in de modulen. In Visual Basic kunt u het bereik van een variabele in de declaratie opnemen.

Een variabele kan een van de onderstaande drie bereiken hebben, afhankelijk van de manier waarop de variabele is gedeclareerd.

Bereik	Declaratie
Lokaal	**Dim** of **Statisch** binnen de procedure
Module	**Dim** of **Persnl** boven aan de module
Openbaar	**Openbaar** boven aan de module

Variabelen die u kunt gebruiken binnen een procedure (lokale variabelen)

Een *lokale variabele* is een variabele die alleen binnen de procedure waarin deze voorkomt, kan worden gebruikt. Dit type is bij uitstek geschikt voor variabelen die maar tijdelijk worden gebruikt. Het is mogelijk dat verschillende procedures bijvoorbeeld allemaal over een variabele met de naam `TijdWrd` beschikken, maar zolang elke `TijdWrd` lokaal is gedeclareerd, herkent elke procedure alleen de eigen variabele. Binnen een procedure kunt u de waarde van de lokale variabele `TijdWrd` rustig wijzigen, zonder dat dit consequenties heeft voor de variabelen in andere procedures.

```
Sub Procedure1()
    Dim TijdWrd Als Integer
    ' Lokale variabele, niet hetzelfde als TijdWrd hieronder
.
.
.
Einde Sub

Sub Procedure2()
    Dim TijdWrd Als Integer
    ' Lokale variabele, niet hetzelfde als TijdWrd hierboven
.
.
.
Einde Sub
```

Een procedure kan zelfs zichzelf oproepen (een techniek die bekend staat onder de naam *recursie*). Elke keer dat deze procedure wordt opgeroepen, wordt aan de procedure een eigen versie van de lokale variabelen toegekend. Lokale variabelen die met het sleutelwoord **Dim** zijn gedeclareerd, blijven slechts bestaan zolang de procedure wordt uitgevoerd. Lokale variabelen die met het sleutelwoord **Statisch** zijn gedeclareerd (zoals beschreven in "Hoe lang behouden variabelen hun waarden?" verderop in dit hoofdstuk), blijven bestaan totdat Visual Basic wordt afgesloten. Ze worden pas opnieuw ingesteld wanneer "run-time"-fouten niet worden afgehandeld, wanneer u Visual Basic onderbreekt of Microsoft Excel afsluit of wanneer u de module met de variabele wijzigt. Zie hoofdstuk 9, "Foutafhandeling en foutwaarden", voor meer informatie over het afhandelen van fouten.

Opmerking Impliciet gedeclareerde variabelen (die alleen zijn toegestaan als de instructie **Optie Expliciet** niet actief is) hebben altijd een lokaal bereik. Zie "Fouten opsporen met expliciete declaraties" eerder in dit hoofdstuk voor meer informatie over de instructie **Optie Expliciet**.

Variabelen die u kunt gebruiken binnen een module (variabelen op module-niveau)

Een *variabele op module-niveau* deelt de informatie met alle procedures binnen een module. Dergelijke variabelen zijn beschikbaar voor alle procedures binnen een module, maar niet voor de programmacode uit andere modulen. U maakt een variabele op module-niveau door de instructie **Dim** of **Persnl** boven aan de module te plaatsen, boven de definitie van de eerste procedure. (Op module-niveau is er geen verschil tussen de instructies **Persnl** en **Dim**, maar het valt aan te bevelen de instructie **Persnl** te gebruiken omdat deze de tegenstelling tussen **Persnl** en **Openbaar** duidelijker weergeeft en daardoor uw programmacode beter leesbaar maakt). Variabelen op module-niveau blijven bestaan totdat de Visual Basic wordt afgesloten of totdat de module wordt gewijzigd waarin ze zijn gedefinieerd.

```
Dim AantalWerknemers Als Integer          ' Variabele op module-niveau
Persnl MassaVanHeelal Als Dubbel          ' Variabele op module-niveau

Sub Procedure1()
    Dim TijdWrd Als Integer
    ' Variabele op procedure-niveau(lokaal)
.
.
.
```

Variabelen die door alle modulen kunnen worden gebruikt (openbare variabelen)

Een *openbare variabele* heeft het grootste bereik. De waarden in de openbare variabelen zijn beschikbaar voor alle procedures in elke module in alle werkmappen (als die werkmappen tenminste zijn geselecteerd in het vak "Beschikbare verwijzingen" van het dialoogvenster **Verwijzingen** in het menu **Extra**). Zie "OLE gebruiken met toepassingen" in hoofdstuk 10 voor meer informatie over het opnemen van verwijzingen in werkmappen.

Opmerking Ervaren programmeurs zijn waarschijnlijk beter bekend met de naam *globaal* als aanduiding voor variabelen, constanten en procedures die algemeen beschikbaar zijn. In deze context hebben globaal en openbaar dezelfde betekenis. Beide begrippen hebben betrekking op elementen die algemeen beschikbaar zijn.

Net zoals variabelen op module-niveau worden openbare variabelen opgegeven in de declaratie-sectie van een module (de bovenste sectie, boven de definities van de procedures). Het is niet mogelijke om openbare variabelen of variabelen op module-niveau in een procedure te declareren. De declaratie van openbare variabelen bevat echter de instructie **Openbaar** in plaats van de instructies **Dim** of **Persnl** die voor variabelen op module-niveau worden gebruikt.

```
Openbaar AantalWerknemers Als Integer              ' Openbare variabele
Openbaar MassaVanHeelal Als Dubbel                 ' Openbare variabele

Sub Procedure1()
    Dim TijdWrd Als Integer
    ' Variabele op procedure-niveau (lokaal)
.
.
.
```

U kunt openbare variabelen in elke willekeurige module declareren. Met het oog op de leesbaarheid van uw programmacode is het echter een goed idee om de naam van de variabele te laten voorafgaan door de naam van de module die de declaratie voor deze variabele bevat. Als u bijvoorbeeld in de module Kosten de variabele `KledingVerk` uit de module Inkomsten wilt oproepen, kunt u gebruik maken van de volgende programmacode.

```
Totaal = TotaalVerk + Inkomsten.KledingVerk
```

Hierdoor voorkomt u dat u openbare variabelen met dezelfde naam gebruikt die in twee verschillende modulen zijn gedeclareerd. Dit veroorzaakt namelijk een fout bij het uitvoeren van de programmacode.

Een openbare variabele blijft beschikbaar, en behoudt zijn waarde, vanaf het moment waarop een waarde aan de variabele wordt toegekend totdat de uitvoering van de programmacode is voltooid.

Opmerking Als u variabelen of procedures wilt maken die beschikbaar zijn voor alle modulen in de werkmap maar niet voor modulen uit andere werkmappen, plaatst u de instructie **Optie Persnl Module** in de module en declareert u vervolgens de variabele met het sleutelwoord **Openbaar** (procedures zijn standaard **Openbaar**). Als u de instructie **Optie Persnl Module** niet gebruikt, kunnen andere werkmappen ook gebruik maken van de openbare variabelen en procedures in de module, mits er een verwijzing bestaat tussen de twee werkmappen. Zie "OLE gebruiken met toepassingen" in hoofdstuk 10 voor meer informatie over het opgeven van verwijzingen naar werkmappen.

Twee variabelen met dezelfde naam gebruiken

Het bereik van een variabele kan tijdens de uitvoering van de programmacode niet veranderen. U kunt echter een andere variabele met dezelfde naam in een ander bereik definiëren. Zo kunt u bijvoorbeeld een openbare variabele met de naam `TijdWrd` declareren en vervolgens binnen een procedure een lokale variabele met de naam `TijdWrd` declareren. Verwijzingen naar `TijdWrd` binnen de procedure hebben betrekking op de lokale variabele, terwijl verwijzingen naar `TijdWrd` buiten de procedure betrekking hebben op de openbare variabele.

Algemeen geldt dat wanneer variabelen dezelfde naam maar een verschillend bereik hebben, er eerder toegang wordt gezocht tot lokale dan tot andere variabelen.

Het gebruik van verschillende variabelen met dezelfde naam kan aanleiding geven tot vergissingen en kan subtiele fouten veroorzaken in uw programmacode. Het valt dan ook aan te raden om gebruik te maken van unieke namen of, als u toch gebruik maakt van dezelfde namen, om de namen van modulen bij de variabelen te plaatsen.

Variabelen en procedures met dezelfde naam gebruiken

De namen van variabelen op module-niveau en van openbare variabelen kunnen ook in conflict zijn met de namen van procedures. Variabelen in een module mogen niet dezelfde naam hebben als de procedures of de gegevenstypen die in de module worden gedefinieerd. Ze mogen echter wel dezelfde naam hebben als openbare procedures, typen of variabelen die in een andere module zijn gedefinieerd. Als een variabele wordt benaderd vanuit een andere module, moet u bij de naam ook de modulenaam opgeven (bijvoorbeeld, `KostWijz.TotaleKosten`).

Als een variabele wordt gebruikt in de module die de declaratie voor deze variabele bevat, hoeft u echter niet de naam van de module op te geven. Procedures, variabelen en door de gebruiker gedefinieerde gegevenstypen waarvoor de instructie **Persnl** is opgegeven, hebben een bereik op module-niveau en mogen daarom niet dezelfde naam hebben als een andere procedure, variabele of door de gebruiker gedefinieerd gegevenstype in dezelfde module.

Hoe lang behouden variabelen hun waarden?

Behalve een bereik hebben variabelen ook een levensduur. De waarden in openbare variabelen en variabelen op het module-niveau blijven behouden totdat Visual Basic wordt afgesloten. Lokale variabelen die met de instructie **Dim** zijn gedeclareerd, bestaan echter alleen zolang de procedure wordt uitgevoerd waarin ze zijn gedeclareerd. Meestal blijven de waarden van lokale variabelen niet behouden wanneer een procedure is afgelopen, en wordt de geheugenruimte vrijgemaakt die door deze variabelen in beslag werd genomen. Wanneer de procedure later nogmaals wordt uitgevoerd, worden alle lokale variabelen opnieuw ingesteld.

U kunt echter de waarde van een lokale variabele bewaren door deze variabele statisch te maken. Met behulp van het sleutelwoord **Statisch** kunt u een of meer variabelen binnen een procedure declareren, net zoals u zou doen met de instructie **Dim**.

```
Statisch Diepte
```

In de volgende functie wordt bijvoorbeeld een totaal berekend door een nieuwe waarde toe te voegen aan het totaal van de voorgaande waarden dat is opgeslagen in de statische variabele `Accumuleren`.

```
Functie Totaal(getal)
    Statisch Accumuleren
    Accumuleren = Accumuleren + getal
    Totaal = Accumuleren
Einde Functie
```

Als `Accumuleren` zou zijn gedeclareerd met de instructie **Dim** in plaats van **Statisch**, zouden de geaccumuleerde waarden niet bewaard zijn gebleven gedurende de verschillende oproepen van de functie, en zou deze functie steeds dezelfde waarde opleveren.

U kunt hetzelfde resultaat bereiken door de variabele `Accumuleren` te declareren in de declaratie-sectie boven aan de module en er zo een variabele op module-niveau van te maken. Als u het bereik van een variabele echter op deze manier wijzigt, kunnen ook andere procedures van deze variabele gebruik maken. Omdat andere procedures de waarde van de variabele dan kunnen wijzigen, is het resultaat van de functie niet meer betrouwbaar. De programmacode is dan moeilijker te onderhouden.

De waarden van alle lokale variabelen in een procedure handhaven

Als u alle lokale variabelen in een procedure statisch wilt maken, plaatst u het sleutelwoord **Statisch** aan het begin van de procedure.

```
Statisch Functie Totaal(getal)
```

Hierdoor worden alle lokale variabelen in de procedure statisch, ongeacht of hun declaratie het sleutelwoord **Statisch** of **Dim** bevat, of dat ze impliciet zijn gedeclareerd. U kunt het sleutelwoord **Statisch** voor het eerste woord van de procedure plaatsen, zelfs voor procedures die ook als **Persnl** zijn gedeclareerd.

Constanten beschikbaar maken in een procedure of module, of openbare constanten maken

De instructie **Const** heeft net zoals de declaratie van een variabele een bepaald bereik. Op variabelen en constanten zijn de volgende regels van toepassing.

- Als u een constante wilt maken die alleen geldt binnen een procedure, moet u de constante binnen die procedure declareren.
- Als u een constante beschikbaar wilt maken voor alle procedures binnen een module, maar niet voor de programmacode buiten deze module, kunt u die constante in de declaratie-sectie van de module declareren of het sleutelwoord **Persnl** in de declaratie opnemen.
- Als u een constante beschikbaar wilt maken voor alle werkmappen, declareert u die constante in de declaratie-sectie van de module en plaatst u het sleutelwoord **Openbaar** voor **Const**.

Bij het gebruik van openbare constanten verdient het volgende probleem extra aandacht. Omdat constanten kunnen worden gedefinieerd op basis van andere constanten, moet u ervoor oppassen dat u geen cirkelverwijzing maakt tussen twee of meer constanten. Een *cirkelverwijzing* ontstaat wanneer er twee of meer openbare constanten naar elkaar verwijzen.

```
' In Module 1:
Openbaar Const A = B * 2

' In Module 2:
Openbaar Const B = A / 2
```

Als een toepassing een cirkelverwijzing bevat, genereert Visual Basic een foutmelding als u deze toepassing probeert te starten. U kunt de programmacode niet uitvoeren voordat u de cirkelverwijzing hebt opgelost.

Meer informatie over procedures

In dit gedeelte wordt dieper ingegaan op de manier waarop u namen kunt toekennen aan procedures, hoe u met argumenten informatie doorgeeft aan procedures en op welke wijze Visual Basic resultaatwaarden van **Functie**-procedures overdraagt.

Regels voor het benoemen van procedures, argumenten, variabelen en constanten

Neem de volgende regels in acht bij het benoemen van procedures, argumenten, variabelen en constanten.

- Visual Basic maakt geen onderscheid tussen hoofdletters en kleine letters maar gebruikt de hoofdletters of kleine letters die u als laatste hebt ingevoerd, als de naam van een procedure, argument, variabele of constante.
- Het eerste teken van een naam moet een letter zijn.
- Voor **Functie**-procedures mag u geen namen gebruiken die geïnterpreteerd kunnen worden als celverwijzingen (van het type A1 of R1K1) of als een gereserveerd sleutelwoord in Visual Basic. Zie *Gereserveerde sleutelwoorden* in de online *Visual Basic Naslaggids* voor een beschrijving van de gereserveerde en de overige sleutelwoorden in Visual Basic.

Waarde van een variabele wijzigen wanneer deze wordt doorgegeven aan een procedure (op waarde i.t.t. op verwijzing)

Als u een procedure oproept en u variabelen opgeeft als argumenten voor deze procedure, kunt u de waarde van deze variabelen door de procedure laten wijzigen.

De waarde van een variabele kan worden gewijzigd door een procedure wanneer deze waarde als een verwijzing aan de procedure wordt doorgegeven. Een waarde doorgeven *op verwijzing* is een manier om een argument aan een procedure door te geven, waardoor de procedure toegang heeft tot de werkelijke variabele die in het geheugen is opgeslagen. Hierdoor kan de waarde van de variabele blijvend worden gewijzigd door de procedure waaraan de variabele is doorgegeven. Doorgeven op verwijzing is de standaardmethode in Visual Basic.

Wanneer daarentegen een variabele wordt doorgegeven *op waarde*, wordt alleen een kopie van de variabele doorgegeven aan de procedure. Als deze waarde vervolgens in de procedure wordt gewijzigd, heeft dit alleen consequenties voor de kopie en niet voor de oorspronkelijke variabele. Dit is van belang in de procedure Omkeren, die eerder in dit hoofdstuk is beschreven in "Het gegevenstype opgeven voor een argument". Als het tweede argument namelijk niet is gedeclareerd met **ViaWaarde**, kunnen fouten optreden in de programmacode.

```
Functie Omkeren (T Als Tekenreeks; ViaWaarde n Als Integer)
    ' Keert de eerste n tekens in T om.
    Dim TijdWrd Als Tekenreeks; I Als Integer
    Indien n > Lengte(T) Dan n = Lengte(T)
    ' Door het wijzigen van de waarde van n kan
    ' een waarde in de oproep van de functie wijzigen.
    Voor I = n Tot 1 Stappen -1
        TijdWrd = TijdWrd & Deel(T; I; 1)
    Volgende
    Omkeren = TijdWrd & Rechts(T; Lengte(T) - n)
Einde Functie
```

De voorgaande functie kan dus de waarde van `I` in de volgende functie-oproep wijzigen, omdat `I` (het tweede argument in de oproep) overeenstemt met `n` (het tweede argument in de **Functie**-procedure).

```
R = Omkeren(T; I)
```

Stel nu dat het tweede argument in Omkeren niet was gedeclareerd met **ViaWaarde** en u dit argument had opgeroepen.

```
Dim I Als Integer; T Als Tekenreeks
I = 10
T = "Testen"
R = Omkeren(T; I)
' Nu is I = 6 (de lengte van T).
```

De functie past de argumenten aan, iets wat u doorgaans niet zult verwachten. Om dit soort neveneffecten te voorkomen in functies die de argumenten wijzigen, kunt u het sleutelwoord **ViaWaarde** in de declaratie van deze argumenten opnemen.

Argumenten doorgeven op verwijzing is een manier om **Sub**-procedures op dezelfde wijze te laten verlopen als **Functie**-procedures, maar u wijzigt dan de variabelen in plaats van de resultaatwaarden. De volgende twee regels programmacode kunnen bijvoorbeeld allebei de waarde van `MijnVariabele` wijzigen als deze op verwijzing wordt doorgegeven aan de **Sub**-procedure MijnSub.

```
MijnVariabele = MijnFunctie()
' MijnFunctie resulteert in een waarde
MijnSub MijnVariabele
' MijnSub wijzigt de variabele die op verwijzing is doorgegeven
```

Zie *Argumenten doorgeven*, *ViaWaarde* en *OpVerwijzing* in de online *Visual Basic Naslaggids* voor meer informatie over het doorgeven van argumenten op waarde en op verwijzing.

Microsoft Excel-matrices doorgeven aan een procedure

Als u een matrix van Microsoft Excel als een argument doorgeeft aan een Visual Basic-procedure, wordt deze matrix automatisch omgezet in een Visual Basic-matrix. In een werkblad van Microsoft Excel kunt u bijvoorbeeld een door de gebruiker gedefinieerde functie oproepen met de naam VerwerkVoedsel en een **Variant**-argument `Voedsel`.

```
Functie VerwerkVoedsel(Voedsel)
```

`Voedsel` wordt een Visual Basic-matrix (met **Variant**-variabelen) wanneer u een Microsoft Excel-matrix doorgeeft aan de functie via de volgende formule die u in een cel in een werkblad kunt invoeren.

```
=VerwerkVoedsel({"Appels";7;19})
```

Opmerking Als u programmacode schrijft of bewerkt, kunt u in de keuzelijst "Land/Taal" van het tabblad Module van het dialoogvenster **Opties** (menu **Extra**) de landinstellingen kiezen die worden gebruikt voor het decimaalteken, het lijstscheidingsteken, het valutasymbool en de datumnotatie. Het decimaalteken, het lijstscheidingsteken, het valutasymbool en de datumnotatie die gebruikers te zien krijgen wanneer ze uw programmacode uitvoeren, zijn afhankelijk van de landinstellingen van hun besturingssysteem.

In de codevoorbeelden in dit boek wordt ervan uitgegaan dat u gebruik maakt van de Nederlandse standaardinstellingen. Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over het schrijven van toepassingen die in andere landen gebruikt moeten kunnen worden.

Bij de conversie van een Microsoft Excel-matrix naar een Visual Basic-matrix worden de volgende regels gehanteerd.

- Een horizontale Microsoft Excel-matrix wordt geconverteerd naar een eendimensionale Visual Basic-matrix.

  De formule `=VerwerkVoedsel({"Appels";7;19})` wordt bijvoorbeeld omgezet in een matrix in de door de gebruiker gedefinieerde functie waarin Voedsel(1) = "Appels", Voedsel(2) = 7 en Voedsel(3) = 19.

- Een verticale Microsoft Excel-matrix wordt geconverteerd naar een tweedimensionale Visual Basic-matrix.

  De formule `=VerwerkVoedsel({"Appels"\7\19})` wordt bijvoorbeeld omgezet in een matrix in de door de gebruiker gedefinieerde functie waarin Voedsel(1) = "Appels", Voedsel(2) = 7 en Voedsel(3) = 19.

- Een tweedimensionale Microsoft Excel-matrix wordt omgezet in een tweedimensionale Visual Basic-matrix, en beide matrix-indices (voor rijen en kolommen) worden gebruikt.

  De formule =VerwerkVoedsel({"Appels";7\19;"Perziken"}) wordt bijvoorbeeld omgezet in een matrix in de door de gebruiker gedefinieerde functie waarin Voedsel(1;1) = "Appels", Voedsel(1;2) = 7, Voedsel(2;1) = 19 en Voedsel(2;2) = "Perziken".

Wanneer matrices worden teruggegeven aan Microsoft Excel, wordt een eendimensionale Visual Basic-matrix geconverteerd naar een horizontale Microsoft Excel-matrix en wordt een tweedimensionale Visual Basic-matrix geconverteerd naar een tweedimensionale Microsoft Excel-matrix.

Zie "Waarden opslaan in Visual Basic-matrices" verderop in dit hoofdstuk voor meer informatie over Visual Basic-matrices. Zie "Werken met matrices" in hoofdstuk 10 van het *Microsoft Excel Handboek* voor meer informatie over het gebruik van Microsoft Excel-matrices.

Een procedure met optionele argumenten maken

U kunt opgeven dat een argument in een procedure optioneel is door het sleutelwoord **Optioneel** in de argumentenlijst te plaatsen. Als u een optioneel argument opgeeft, worden alle volgende argumenten in de argumentenlijst ook optioneel en moeten deze gedeclareerd worden met het sleutelwoord **Optioneel**. De optionele argumenten moeten van het gegevenstype **Variant** zijn.

Als u de hier volgende versie van de functie Omkeren gebruikt, hoeft u het laatste argument van de functie niet te gebruiken. Zoals u ziet, kunt u naar het ontbrekende argument zoeken met de functie **IsZoek**.

```
Functie Omkeren (T Als Tekenreeks; Optioneel ViaWaarde n Als Integer)
    ' Keert de eerste n tekens in T om.
    ' Als n ontbreekt, worden alle tekens omgekeerd.
    Dim TijdWrd Als Tekenreeks; I Als Integer

    Indien IsZoek(n) Dan n = Lengte(T)

    Indien n > Lengte(T) Dan n = Lengte(T)
    Voor I = n Tot 1 Stappen -1
        TijdWrd = TijdWrd & Deel(T; I; 1)
    Volgende
    Omkeren = TijdWrd & Rechts(T; Lengte(T) - n)
Einde Functie
```

Een procedure maken met een onbepaald aantal argumenten

U kunt **Functie**-procedures maken die elk willekeurig aantal argumenten met het gegevenstype **Variant** accepteren. De standaardfuncties SOM, GEMIDDELDE en MEDIAAN zijn voorbeelden van functies die een variabel aantal argumenten accepteren. Om aan te geven dat een argument een onbeperkt aantal waarden kan bevatten, declareert u dit argument met het sleutelwoord **MatrixParam**. Het argument is dan een matrix. Als u afzonderlijke waarden uit deze matrix wilt ophalen, kunt u dit doen met een index of met de instructie **Voor Elke** zoals u bij elke andere matrix zou doen.

Opmerking De ondergrens van de matrix **MatrixParam** (nul of een) wordt bepaald door de ondergrens die in de Visual Basic-module wordt gebruikt. Visual Basic hanteert standaard de ondergrens nul, tenzij u een andere ondergrens hebt opgegeven door de declaratie **Optie Basis 1** in de declaratie-sectie van de module op te nemen. Zie *Optie Basis* in de online *Visual Basic Naslaggids* voor meer informatie over **Optie Basis**.

Als u bijvoorbeeld gebruik wilt maken van de werkbladfunctie MULTINOMIAAL zonder de volledige Analysis ToolPak te installeren, kunt u uw eigen versie van de functie maken. (De Analysis ToolPak is een invoegmacro die financiële, statistische en technische functies bevat. Zie hoofdstuk 31, "Statistische analyse van gegevens", in het *Microsoft Excel Handboek* voor meer informatie over deze invoegmacro).

```
' Resulteert in de verhouding tussen de faculteit van de som en het
' produkt van de faculteiten: (n1 + n2 + ...)! / (n1! * n2! * ...)

Functie MijnMultinomiaal (MatrixParam Getallen())
    Voor Elke x in Getallen
        s = s + x
    Volgende x
    Teller = Toepassing.Faculteit(s)
    Noemer = 1
    Voor Elke x in Getallen
        Noemer = Noemer * Toepassing.Faculteit(x)
    Volgende x
    MijnMultinomiaal = Teller / Noemer
Einde Functie
```

Meer informatie over resultaatwaarden voor een door de gebruiker gedefinieerde functie

Als de resultaatwaarden van een door de gebruiker gedefinieerde functie bestaat uit bereiken, matrices of tekst, moet u rekening houden met de volgende beperkingen.

- Een functie kan alleen een matrix als resultaat opleveren als het gegevenstype van de resultaatwaarde **Variant** is.
- De tekst die een door de gebruiker gedefinieerde functie doorgeeft aan een Microsoft Excel-object mag niet langer zijn dan 255 tekens.

Door de gebruiker gedefinieerde functies die automatisch worden herberekend

Met de methode **AltijdHerberekenen** zorgt u ervoor dat de door de gebruiker gedefinieerde functies automatisch worden herberekend. Een **AltijdHerberekenen**-functie wordt telkens opnieuw berekend als de berekening voorkomt in een werkblad. (Doorgaans wordt een cel die een door de gebruiker gedefinieerde, Niet-AltijdHerberekenen-functie bevat, alleen herberekend als een onderdeel van de volledige formule in die cel wordt herberekend). Roep de methode **AltijdHerberekenen** op als eerste instructie in een procedure. Zie *methode AltijdHerberekenen* in de online *Visual Basic Naslaggids* voor meer informatie over de methode **AltijdHerberekenen**.

Waarden opslaan in Visual Basic-matrices

Visual Basic-matrices zijn vergelijkbaar met de matrices en matrixformules van Microsoft Excel. Zie "Wat is een matrixconstante?" en "Werken met matrices" in hoofdstuk 10 van het *Microsoft Excel Handboek* voor meer informatie over Microsoft Excel-matrices.

In Visual Basic kunt u gebruik maken van matrices om naar een reeks variabelen te verwijzen, die allemaal dezelfde naam hebben en waarbij u gebruik maakt van een getal (een index) om onderscheid te maken tussen de verschillende variabelen. Zo kunt u in veel situaties de programmacode beperkt en eenvoudig houden door lussen te definiëren die op een efficiënte wijze gebruik maken van de variabelen. Matrices hebben zowel bovengrenzen als ondergrenzen. De elementen in een matrix zijn opeenvolgend gerangschikt binnen die grenzen. Omdat Visual Basic geheugenruimte toewijst aan elk indexnummer, is het van belang dat u de matrices niet groter maakt dan nodig is. U kunt de grootte of de dimensies van een matrix wijzigen met het sleutelwoord **HerDim**. Zie "De grootte van een matrix wijzigen terwijl de programmacode wordt uitgevoerd" verderop in dit hoofdstuk voor meer informatie over de instructie **HerDim**.

Alle elementen in een matrix zijn van hetzelfde gegevenstype. Wanneer dit het gegevenstype **Variant** is, kunnen de afzonderlijke elementen natuurlijk verschillende soorten gegevens bevatten (tekenreeksen, getallen of datum- en tijdwaarden). U kunt voor een matrix elk primair gegevenstype declareren, met inbegrip van door de gebruiker gedefinieerde gegevenstypen (die zijn beschreven in "Uw eigen gegevenstypen maken" eerder in dit hoofdstuk) en object-variabelen.

U kunt een matrix met een vaste grootte op een van de volgende drie manieren declareren, afhankelijk van het gewenste bereik.

- Als u een *lokale matrix* wilt maken, declareert u de matrix binnen een procedure met de instructie **Dim**.
- Als u een *matrix op module-niveau* wilt maken, plaatst u de instructie **Dim** of **Persnl** in de declaratie-sectie van een Visual Basic-module.
- Als u een *openbare matrix* wilt maken, plaatst u de instructie **Openbaar** in de declaratie-sectie van een Visual Basic-module.

De bereiken lokaal, op module-niveau en openbaar worden beschreven in "Variabelen beschikbaar maken in een procedure of module, of openbare variabelen maken" eerder in dit hoofdstuk. Het begrip bereik heeft voor matrices dezelfde betekenis als voor variabelen.

Wanneer u een *matrix met een variabele grootte* (een matrix waarvan de grootte tijdens de uitvoering van de programmacode kan worden aangepast) wilt maken, moet u rekening houden met een aantal extra regels. Deze regels worden beschreven in "Waarden in matrices met een variabele grootte bewaren" verderop in dit gedeelte.

In de declaratie van een matrix plaatst u de bovengrens van de matrix tussen ronde haken achter de naam van de matrix. In de declaratie-sectie van een module kunt u bijvoorbeeld de volgende programmacode gebruiken om matrices te declareren.

```
Dim Tellers(14) Als Integer
Dim Sommen(20) Als Dubbel
```

Wanneer u een openbare matrix wilt maken, gebruikt u simpelweg het sleutelwoord **Openbaar** in plaats van **Dim**.

```
Openbaar Tellers(14) Als Integer
Openbaar Sommen(20) Als Dubbel
```

Voor dezelfde declaraties binnen een procedure kunt u ook het sleutelwoord **Statisch** gebruiken.

```
Statisch Tellers(14) Als Integer
Statisch Sommen(20) Als Dubbel
```

Met de eerste declaratie maakt u een matrix van 15 elementen met de indexnummers 0 tot en met 14. Met de tweede declaratie maakt u een matrix van 21 elementen met de indexnummers 0 tot en met 20. De standaardondergrens is 0 (matrix met ondergrens nul). U kunt deze ondergrens wijzigen in 1 (matrix met ondergrens één) door de instructie **Optie Basis** in de declaratie-sectie van een module op te nemen.

```
Optie Basis 1
```

Tip Het is misschien verstandig om de instructie `Optie Basis 1` altijd in uw modulen te gebruiken. Microsoft Excel-verzamelingen en alle matrices die het resultaat zijn van een Microsoft Excel-methode of -eigenschap hanteren ook één als ondergrens. Als u een matrix met de ondergrens één toewijst aan een matrix met de ondergrens nul, wordt de informatie wel overgenomen, maar wijkt elk element net een indexnummer af. Hierdoor wordt de programmacode minder gemakkelijk te lezen en is het ook moeilijk om fouten op te sporen.

U kunt een ondergrens ook expliciet opgeven (als een lang geheel getal) met het sleutelwoord **Tot**.

```
Dim Tellers(1 Tot 15) Als Integer
Dim Sommen(100 Tot 120) Als Tekenreeks
```

In de voorgaande declaraties lopen de indexnummers in `Tellers` van 1 tot en met 15, en in `Sommen` van 100 tot en met 120.

Lussen bieden vaak een efficiënte methode om matrices te bewerken. In de volgende **Voor...Volgende**-lus krijgen alle elementen in de matrix de waarde 5.

```
Statisch Tellers(1 Tot 15) Als Integer
Voor X = 1 Tot 15
    Tellers(X) = 5
Volgende
```

Matrices opslaan in Variant-variabelen

U kunt een matrix met **Variant**-variabelen declareren, maar u kunt ook met de functie **Matrix** een **Variant**-variabele maken die een matrix bevat met variabelen van een willekeurig gegevenstype.

```
Dim MijnMatrix(10) ' Maakt een matrix met tien Variant-variabelen.

Dim WeekDag
WeekDag = Matrix("Ma"; "Di"; "Wo"; "Do"; "Vr"; "Za"; "Zo")
' Maakt een Variant-variabele die een matrix bevat
```

Werken met multidimensionale matrices

In Visual Basic kunt u een matrix declareren met maximaal 60 dimensies. Met de volgende instructie declareert u bijvoorbeeld een tweedimensionale matrix van 10 bij 10 elementen (de matrix heeft als ondergrens nul).

```
Statisch MatrixA(9; 9) Als Dubbel
```

U kunt voor een van beide of voor beide dimensies een expliciete ondergrens in de declaratie opnemen.

```
Statisch MatrixA(1 Tot 10; 1 Tot 10) Als Dubbel
```

Met geneste **Voor**-lussen kunt u een multidimensionale matrix efficiënt bewerken. In het volgende voorbeeld krijgt elk element in `MatrixA` een waarde die is gebaseerd op de positie van het element in de matrix.

```
Dim I Als Integer; J Als Integer
Statisch MatrixA(1 Tot 10; 1 Tot 10) Als Dubbel
Voor I = 1 Tot 10
    Voor J = 1 Tot 10
        MatrixA(I; J) = I * 10 + J
    Volgende J
Volgende I
```

U kunt ook een matrix uitbreiden tot meer dan twee dimensies.

```
Dim MultiD(3; 1 Tot 10; 1 Tot 15)
```

Met de voorgaande declaratie maakt u een driedimensionale matrix van 4 bij 10 bij 15 elementen. Het totaal aantal elementen in deze matrix is gelijk aan het produkt van deze drie dimensies, oftewel 600.

Opmerking Wanneer u dimensies aan een matrix toevoegt, neemt de totale geheugenruimte die door deze matrix in beslag wordt genomen, gigantisch toe. Maak daarom spaarzaam gebruik van matrices. Wees vooral voorzichtig met **Variant**-matrices, want deze nemen nog veel meer plaats in dan matrices van andere gegevenstypen.

De grootte van een matrix wijzigen terwijl de programmacode wordt uitgevoerd

Soms weet u vooraf niet hoe groot een matrix moet zijn. Dan is het handig als de grootte van de matrix kan worden aangepast terwijl de programmacode wordt uitgevoerd. De omvang van matrices met een variabele grootte kan op elk moment worden gewijzigd. Door matrices met een variabele grootte kunt u doeltreffender omgaan met het geheugen. Zo kunt u bijvoorbeeld gedurende korte tijd een grote matrix gebruiken en zodra u deze niet meer nodig hebt, de geheugenruimte weer vrijmaken.

▶ Een matrix met een variabele grootte maken

1. Declareer de matrix met de instructie **Statisch** of **Dim** in een procedure (als u een lokale matrix wilt maken), met de instructie **Dim** in de declaratie-sectie van de module (als u een matrix op module-niveau wilt maken), of met de instructie **Openbaar** (als u een openbare matrix wilt maken).
2. Declareer de matrix als een matrix met een variabele grootte door er een lege dimensielijst aan toe te voegen.

   ```
   Dim VarMatrix()
   ```

3. Wijs eventueel het werkelijke aantal elementen toe met de instructie **HerDim**. Dit is alleen mogelijk in een procedure en niet op module-niveau.

   In de volgende programmacode is `X` een variabele waarvan de waarde wordt vastgesteld tijdens de uitvoering van de programmacode. U kunt een dergelijke variabele gebruiken om het vooraf gedefinieerde maximumaantal elementen te wijzigen.

   ```
   HerDim VarMatrix(X + 1)
   ```

De instructie **HerDim** kan alleen in een procedure worden gebruikt. In tegenstelling tot de instructies **Dim** en **Statisch** is **HerDim** een uitvoerbare instructie, een instructie die ervoor zorgt dat de toepassing een bepaalde handeling uitvoert als u de programmacode hebt gestart.

Voor de instructie **HerDim** wordt dezelfde syntaxis gehanteerd als voor matrices met een vaste grootte. Elke keer als u de instructie **HerDim** invoert, kunt u zowel het aantal elementen als de bovengrens en de ondergrens van elke dimensie opnieuw opgeven.

De matrix met een variabele grootte `Matrix1` wordt bijvoorbeeld eerst gemaakt door deze te declareren op module-niveau.

```
Dim Matrix1() Als Integer
```

Een procedure wijst vervolgens een bepaalde geheugenruimte aan deze matrix toe met een lege dimensielijst.

```
Sub BerWaardenNu ()
    .
    .
    .
    HerDim Matrix1(19; 29)
Einde Sub
```

De instructie **HerDim** maakt een matrix van 20 bij 30 gehele getallen (in totaal 600 elementen). Het is echter ook mogelijk om de grenzen voor een matrix met een variabele grootte in te stellen met variabelen.

```
HerDim Matrix1(X; Y)
```

Als u een lokale matrix met een variabele grootte maakt (in een procedure), valt het aan te bevelen om deze matrix met de instructie **Dim** of **Statisch** te declareren. Dit is echter niet noodzakelijk.

Waarden in matrices met een variabele grootte bewaren

Telkens wanneer de instructie **HerDim** wordt uitgevoerd, gaan alle huidige waarden in de matrix verloren. Visual Basic wijzigt de waarden in de waarde **Leeg** (in matrices van het gegevenstype **Variant**), nul (in numerieke matrices), een tekenreeks met de lengte nul (in matrices met tekenreeksen) of **Niets** (in matrices met objecten). Zie hoofdstuk 5, "Werken met objecten in Visual Basic", voor meer informatie over het gebruik van matrices met objecten.

Dit is handig als u de matrix wilt voorbereiden op nieuwe gegevens, of wanneer u de matrix kleiner wilt maken, zodat deze minder ruimte in beslag neemt. Soms zult u echter de grootte van de matrix willen wijzigen zonder de gegevens in de matrix kwijt te raken. U kunt dit bereiken door het sleutelwoord **Behouden** in de instructie **HerDim** op te nemen. Zo kunt u bijvoorbeeld een matrix met 10 elementen uitbreiden zonder de waarden van de bestaande elementen kwijt te raken.

```
HerDim Behouden MijnMatrix(BovenGrens(MijnMatrix) + 10)
```

Alleen de bovengrens van de laatste dimensie in een multidimensionale matrix kan worden gewijzigd wanneer u het sleutelwoord **Behouden** gebruikt. Als u een van de andere dimensies of de ondergrens van de laatste dimensie wijzigt, ontstaat er een "run-time"-fout. U kunt bijvoorbeeld wel de volgende programmacode gebruiken.

```
HerDim Behouden MatrixA(10; BovenGrens(MatrixA; 2) + 1)
```

Maar u kunt niet de volgende programmacode gebruiken.

```
HerDim Behouden MatrixA(BovenGrens(MatrixA; 1) + 1; 10)
```

Zie *HerDim* in de online *Visual Basic Naslaggids* voor meer informatie over **HerDim**. Zie hoofdstuk 9, "Foutafhandeling en foutwaarden", voor meer informatie over "run-time"-fouten.

Eigenschapsprocedures gebruiken

In de meeste gevallen stelt u de waarde van een eigenschap in volgens de syntaxis die wordt beschreven in hoofdstuk 5, "Werken met objecten in Visual Basic". Er zijn echter gevallen waarin het gebruik van procedures als eigenschappen in uw programmacode voordelen heeft. Zo kunt u bijvoorbeeld een procedure gebruiken om gegevens te valideren of een werkblad initialiseren door een waarde toe te kennen aan een door u gedefinieerde eigenschap.

In Visual Basic kunt u speciale procedures maken die u kunt gebruiken om de waarden van eigenschappen in te stellen of op te halen. Deze procedures worden *eigenschapsprocedures* genoemd. Wanneer u een eigenschapsprocedure maakt, wordt de nieuwe eigenschap een eigenschap van de Visual Basic-module waarin de eigenschapsprocedure staat. U hoeft aan de eigenschap niet de naam van de module toe te voegen die deze eigenschap bevat.

Visual Basic beschikt over drie soorten eigenschapsprocedures zoals u in de volgende tabel kunt zien.

Eigenschapsprocedure	Beschrijving
Eigenschap Bepalen	Een procedure waarmee u de waarde van een eigenschap bepaalt
Eigenschap Halen	Een procedure waarmee u de waarde van een eigenschap ophaalt
Eigenschap Toewijzen	Een procedure waarmee u een verwijzing naar een object instelt

U kunt met de instructies **Eigenschap Bepalen**, **Eigenschap Halen** en **Eigenschap Toewijzen** eigenschapsprocedures maken die dezelfde naam gebruiken. Op deze wijze maakt u een groep procedures die kunnen samenwerken. Zo kunt u bijvoorbeeld met **Eigenschap Bepalen** een eigenschapsprocedure schrijven waarmee u een waarde aan een eigenschap toewijst, om vervolgens met **Eigenschap Halen** een tweede eigenschapsprocedure te schrijven die dezelfde waarde weer ophaalt.

Hoewel eigenschapsprocedures over een gemeenschappelijke naam kunnen beschikken, hoeven ze niet vergelijkbare of verwante taken uit te voeren.

Een procedure maken die een waarde toewijst aan een eigenschap

Met de instructie **Eigenschap Bepalen** kunt u een procedure maken waarmee u een waarde voor een eigenschap instelt. Als u bijvoorbeeld gegevens wilt valideren voordat u die gegevens toewijst aan de waarde van een eigenschap, kunt u een eigenschapsprocedure maken om dit te doen.

Stel dat u een macro wilt maken waarmee u de korting voor een bepaald artikel kunt berekenen onder de voorwaarde dat het kortingspercentage niet meer dan 35 procent bedraagt. In plaats van er voetstoots van uit te gaan dat het kortingspercentage correct is, kunt u met de instructie **Eigenschap Bepalen** een eigenschapsprocedure maken die het kortingspercentage controleert voordat de uiteindelijke prijs wordt berekend.

```
Eigenschap Bepalen AfPrijs(ViaWaarde Korting)
    Indien Korting <= 35 Dan
        ActieveCel.Formule = Prijs * (1 - (Korting / 100))
    Anders
        BerichtVenster Aanwijzing:= "Korting te hoog."; _
        Titel:= "Fout"
    Einde Indien
Einde Eigenschap
```

De naam van deze eigenschapsprocedure is AfPrijs. De programmacode in de procedure controleert het kortingspercentage en berekent vervolgens de uiteindelijke prijs of geeft een foutbericht weer.

Zoals u ziet maakt de procedure gebruik van een argument. Eigenschapsprocedures die zijn gemaakt met de instructie **Eigenschap Bepalen** of **Eigenschap Toewijzen** moeten ten minste over één argument beschikken dat de waarde van de procedure ontvangt. (De instructie **Eigenschap Toewijzen** wordt verderop in dit hoofdstuk behandeld). Procedures roepen eigenschapsprocedures (die met de instructie **Eigenschap Bepalen** zijn gemaakt) op via de naam van de eigenschapsprocedure, zoals u kunt zien in het volgende voorbeeld.

```
Prijzen.AfPrijs = 25
```

In dit voorbeeld is de eigenschapsprocedure AfPrijs een eigenschap van de module Prijzen. Het opnemen van de naam van de Visual Basic-module in de syntaxis is optioneel. De waarde van de eigenschap wordt gebruikt als de waarde van het argument Korting in de eigenschapsprocedure.

Procedures die met de instructie **Eigenschap Bepalen** zijn gemaakt, kunnen over meer dan één argument beschikken. Als een eigenschapsprocedure wordt opgeroepen die met de instructie **Eigenschap Bepalen** is gemaakt, wordt het laatste argument altijd rechts van het "is gelijk"-teken geplaatst. De andere argumenten worden tussen ronde haken achter de naam van de eigenschap geplaatst. Stel dat de procedure AfPrijs zo wordt gewijzigd dat deze over twee argumenten kan beschikken: de oorspronkelijke prijs van het artikel en het kortingsbedrag. De eerste regel van de eigenschapsprocedure kan er dan als volgt uitzien.

```
Eigenschap Bepalen AfPrijs(ViaWaarde Prijs; ViaWaarde Korting)
```

Wanneer u de procedure nu oproept, wordt het argument Prijs doorgegeven tussen ronde haken (in het onderstaande geval wordt een variabele gebruikt).

```
Prijzen.AfPrijs(Grthandel) = 25
```

Zie *Eigenschap Bepalen* in de online *Visual Basic Naslaggids* voor meer informatie over de instructie **Eigenschap Bepalen**.

Een procedure maken die resulteert in de waarde van een eigenschap

Met de instructie **Eigenschap Halen** kunt u een eigenschapsprocedure maken die resulteert in de waarde van een eigenschap. Procedures die gebruik maken van de instructie **Eigenschap Halen**, lijken heel veel op **Functie**-procedures. Beide typen functies leveren namelijk een waarde op. Het enige verschil is dat u dezelfde naam kunt gebruiken voor procedures die u maakt met de instructie **Eigenschap Bepalen** en de instructie **Eigenschap Toewijzen**. Verder zijn procedures die u maakt met de instructie **Eigenschap Halen** identiek aan **Functie**-procedures.

In het volgende voorbeeld levert de procedure **Eigenschap Halen** de waarde van een cel op.

```
Eigenschap Halen InhoudCel()
    InhoudCel = Notatie(ActiefBlad.Bereik("Totalen").Waarde; "Valuta")
Einde Eigenschap
```

U kunt deze procedure bijvoorbeeld met de volgende instructie oproepen.

```
' Haalt de waarde van een eigenschap op met de procedure InhoudCel
X = InhoudCel
```

Zoals u ziet moet u net zoals bij een **Functie**-procedure in de programmacode van de procedure **Eigenschap Halen** de waarde opgeven die u met de naam van de procedure wilt ophalen. Eigenschapsprocedures die zijn gemaakt met de instructie **Eigenschap Halen** kunnen gebruik maken van meerdere argumenten, maar hoeven dit niet te doen.

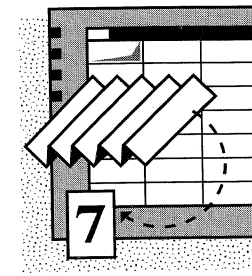
Zie *Eigenschap Halen* in de online *Visual Basic Naslaggids* voor meer informatie over de instructie **Eigenschap Halen**.

Een procedure maken die een verwijzing naar een object toewijst

Met de instructie **Eigenschap Toewijzen** kunt u een eigenschapsprocedure maken die een verwijzing naar een object instelt. U kunt met de instructie **Eigenschap Toewijzen** op dezelfde wijze eigenschapsprocedures maken en gebruiken als met de instructie **Eigenschap Bepalen**. Zie "Cellen en cellenbereiken (als objecten) toewijzen aan variabelen" eerder in dit hoofdstuk voor meer informatie over het gebruik van een verwijzing naar een object.

HOOFDSTUK 7

De uitvoering van de programmacode besturen



In dit hoofdstuk worden controlestructuren behandeld. *Controlestructuren* zijn procedures die controleren of aan bepaalde voorwaarden wordt voldaan en op grond van de resultaten beslissen welke instructies moeten worden uitgevoerd en welke niet. U zult ook leren hoe u een lus maakt waarmee u een serie instructies meerdere malen kunt uitvoeren en hoe u controlestructuren kunt nesten (een controlestructuur binnen een andere controlestructuur plaatsen).

Zonder controlestructuur worden de instructies in een procedure van links naar rechts en van boven naar beneden uitgevoerd. Hoewel het mogelijk is een aantal heel eenvoudige procedures zonder controlestructuur te maken, ligt de kracht en bruikbaarheid van elke programmeertaal in het vermogen om door middel van controlestructuren de volgorde te wijzigen waarin instructies worden uitgevoerd.

Inhoud

- Gedeelten van de programmacode uitvoeren op basis van voorwaarden
- Dezelfde programmacode meerdere keren uitvoeren (lussen)
- Een controlestructuur binnen een andere plaatsen (nesten)
- Controlestructuren en procedures verlaten

Gedeelten van de programmacode uitvoeren op basis van voorwaarden

Visual Basic-procedures kunnen controleren of aan voorwaarden wordt voldaan en op basis van het resultaat verschillende handelingen uitvoeren. De controlestructuren in Visual Basic zijn:

- **Indien...Dan**
- **Indien...Dan...Anders**
- **Kiezen Ingeval**

Indien...Dan

Gebruik de controlestructuur **Indien...Dan** om meerdere instructies uit te voeren, afhankelijk van de gestelde voorwaarden. U kunt een syntaxis van één regel of een "bloksyntaxis" van meerdere regels gebruiken.

```
Indien EenDatum < Nu Dan EenDatum = Nu
```

–Of–

```
Indien EenDatum < Nu Dan
    EenDatum = Nu
    BerichtVenster "Huidige datum is " & EenDatum
Einde Indien
```

Let erop dat u bij de formule van één regel **Indien...Dan** niet de **Einde Indien**-instructie kunt gebruiken. Als de voorwaarde **Waar** is en u meer dan één coderegel wilt uitvoeren, moet u de bloksyntaxis **Indien...Dan...Einde Indien** gebruiken.

Indien...Dan...Anders

Gebruik de controlestructuur **Indien...Dan...Anders** om een blok met instructies te definiëren waarvan er altijd één wordt uitgevoerd.

```
Indien Leeftijd < 18 Dan
    BerichtVenster "U bent niet oud genoeg voor een rijbewijs."
Anders
    BerichtVenster "U kunt uw rijbewijs halen."
Einde Indien
```

U kunt de instructie **AndersIndien** toevoegen om een aantal voorwaarden te testen. Zo vervangt u een aantal geneste instructies **Indien...Dan** en wordt uw programmacode korter en beter leesbaar. Stel dat de werknemers van uw bedrijf zijn ondergebracht in functiecategorieën waarop verschillende bonustarieven van toepassing zijn. In het volgende voorbeeld wordt in de **Functie**-procedure een serie instructies **AndersIndien** gebruikt om na te gaan wat de functiecategorie is.

```
Functie Bonus(FuncCat; Salaris; Waardering)
    Indien FuncCat = 1 Dan
        Bonus = Salaris * 0,1 * Waardering / 10
    AndersIndien FuncCat = 2 Dan
        Bonus = Salaris * 0,09 * Waardering / 10
    AndersIndien FuncCat = 3 Dan
        Bonus = Salaris * 0,07 * Waardering / 10
    Anders
        Bonus = 0
    Einde Indien
Einde Functie
```

Nu volgt een schematisch voorbeeld van de syntaxis voor **Indien...Dan...Anders**. Zie de online *Visual Basic Naslaggids* onder *Indien...Dan...Anders* voor meer informatie over **Indien...Dan...Anders**.

Indien *voorwaarde1* **Dan**
    instructies
AndersIndien.*voorwaarde2* **Dan**
    instructies
.
.
.
Anders
    instructies
Einde Indien

Visual Basic test eerst *voorwaarde1*. Als deze **Onwaar** is test Visual Basic *voorwaarde2*, en gaat zo verder tot een voorwaarde **Waar** is. Als een voorwaarde **Waar** is, voert het programma de instructies uit die aan die voorwaarde zijn verbonden en vervolgens de programmacode die op **Einde Indien** volgt. U kunt eventueel de instructie **Anders** toevoegen, die uitgevoerd zal worden als geen van de voorwaarden **Waar** zijn.

Opmerking Als de voorwaarde die u test de operator **Of** bevat, zoals `Indien (IsFout(A) Of (A>9))`, worden beide expressies getest ook al is de eerste **Waar**. Dit kan in sommige gevallen het resultaat van de instructie beïnvloeden. Het kan bijvoorbeeld leiden tot een "run-time"-fout, als een variabele in de tweede expressie een foutwaarde bevat.

Let erop dat u met een willekeurig aantal instructies **AndersIndien** kunt werken of met geen enkele. U kunt de instructie **Anders** gebruiken of u nu met de instructies **AndersIndien** werkt of niet. U kunt altijd andere instructies **AndersIndien** aan uw structuur **Indien...Dan** toevoegen. Dit kan echter nogal wat tijd vragen, als elke instructie **AndersIndien** dezelfde expressie met een andere waarde vergelijkt. In deze situatie kunt u de controlestructuur **Kiezen Ingeval** gebruiken.

Kiezen Ingeval

In Visual Basic fungeert de instructie **Kiezen Ingeval** als een alternatief voor **Indien...Dan...AndersIndien** voor het vergelijken van dezelfde expressie met meerdere waarden. De instructie **Kiezen Ingeval** heeft dezelfde mogelijkheden als de instructie **Indien...Dan...Anders**, maar maakt de programmacode efficiënter en gemakkelijker te lezen.

Stel dat u meer functiecategorieën aan het voorgaande voorbeeld wilt toevoegen. U kunt hiertoe instructies **AndersIndien** toevoegen, of u kunt de functie maken met **Kiezen Ingeval**. In het volgende voorbeeld wordt een aantal functiecategorieën toegevoegd, maar het gaat nog steeds over dezelfde **Ingeval**.

```
Functie Bonus(FuncCat; Salaris; Waardering)
    Kiezen Ingeval FuncCat
        Ingeval 1
            Bonus = Salaris* 0,1 * Waardering / 10
        Ingeval 2
            Bonus = Salaris * 0,09 * Waardering / 10
        Ingeval 3
            Bonus = Salaris * 0,07 * Waardering / 10
        Ingeval 4; 5'De expressielijst kan verschillende waarden
```

```
bevatten...
            Bonus = Salaris * 0,05 * Waardering / 5
        Ingeval 6 Tot 8 '...of een waardenbereik zijn
            Bonus = 150
        Ingeval Is > 8  '...of worden vergeleken met andere waarden
            Bonus = 100
        Ingeval Anders
            Bonus = 0
    Einde Kiezen
Einde Functie
```

Let erop dat de structuur **Kiezen Ingeval** een expressie éénmaal bovenaan de structuur test. De structuur **Indien...Dan...AndersIndien** daarentegen kan voor elke instructie **AndersIndien** een andere expressie testen. U kunt de structuur **Indien...Dan...AndersIndien** alleen vervangen door de structuur **Kiezen Ingeval** als elke instructie **AndersIndien** dezelfde expressie test.

Nu volgt een schematisch voorbeeld van de syntaxis voor **Kiezen Ingeval**. Zie de online *Visual Basic Naslaggids* onder *Kiezen* voor meer informatie, in het bijzonder voor informatie over de mogelijkheden en de flexibiliteit van de *expressielijst*.

Kiezen Ingeval *testexpressie*
  Ingeval *expressielijst1*
    instructies
  Ingeval *expressielijst2*
    instructies

.
.
.

  Ingeval Anders
    instructies
Einde Kiezen

De structuur **Kiezen Ingeval** werkt met een enkele *testexpressie* die éénmaal bovenaan de structuur wordt getest. Daarna vergelijkt Visual Basic het resultaat van deze expressie met de waarden van elke **Ingeval** in de structuur. Als een waarde voldoet aan dit resultaat, wordt het blok met instructies dat bij de **Ingeval** hoort, uitgevoerd.

Elke *expressielijst* is een lijst met een of meer waarden. Als er meerdere waarden in een lijst voorkomen, worden deze door puntkomma's gescheiden (of door de geldende lijstscheidingstekens). Als meer dan een **Ingeval** aan de *testexpressie* voldoet, wordt alleen het instructieblok dat bij de eerste overeenkomende **Ingeval** hoort, uitgevoerd. Als geen van de waarden in de *expressielijst* voldoet aan de *testexpressie* voert Visual Basic de instructies uit in de instructie **Ingeval Anders** (die optioneel is).

Dezelfde programmacode meerdere keren uitvoeren (lussen)

Met lusstructuren kunt u een of meerdere coderegels verschillende malen uitvoeren. De belangrijkste lusstructuren uit Visual Basic zijn:

- **Doorlopen...Lus**
- **Voor...Volgende**
- **Voor Elke...Volgende**

Met de instructie **Met** kunt u ook meerdere acties op hetzelfde object uitvoeren. Zie "Meerdere acties op een object uitvoeren" in hoofdstuk 5 voor meer informatie over de instructie **Met**.

Doorlopen...Lus

Met een **Doorlopen...Lus** kunt u een instructieblok uitvoeren zo vaak u wilt. Er zijn verschillende varianten van de instructie **Doorlopen...Lus**, maar elke variant test een numerieke voorwaarde en stelt zo vast of de handeling kan worden beëindigd of niet. Net als bij **Indien...Dan** moet *voorwaarde* een waarde of expressie zijn die **Onwaar** (nul) of **Waar** (niet nul) is. De verschillende varianten worden in dit gedeelte beschreven. Zie de online *Visual Basic Naslaggids* onder *Doorlopen...Lus* voor meer informatie over de instructie **Doorlopen...Lus**.

Een voorwaarde testen alvorens een lus uit te voeren

De volgende **Functie**-procedure telt hoe vaak een doeltekenreeks binnen een andere tekenreeks voorkomt door dezelfde programmacode uit te voeren zolang de doelreeks gevonden wordt. U kunt de lus uitvoeren zo vaak u wilt, maar de instructies binnen de lus worden niet uitgevoerd als de *voorwaarde* **Onwaar** is.

```
Functie ReeksenTellen (langereeks; doelreeks)
    positie = 1
    Doorlopen Terwijl InTReeks(positie; langereeks; doelreeks)
'Resulteert in Waar/Onwaar
        positie = InTReeks(positie; langereeks; doelreeks) + 1
        Aantal = Aantal + 1
    Lus
    ReeksenTellen = Aantal
Einde Functie
```

Als de doelreeks niet in de andere reeks voorkomt, resulteert **InTReeks** in nul en wordt de lus niet uitgevoerd.

Nu volgt een schematisch voorbeeld van de syntaxis voor een **Doorlopen...Lus**, waarbij de *instructies* worden uitgevoerd zolang *voorwaarde* **Waar** is.

Doorlopen Terwijl *voorwaarde*
    instructies
Lus

De *instructies* moeten uiteindelijk bewerkstelligen dat de *voorwaarde* **Onwaar** wordt, anders blijft de lus actief (een *oneindige lus*) en dit moet u proberen te vermijden. Als u een oneindige lus wilt afbreken, drukt u op ESC.

Een lus ten minste éénmaal uitvoeren alvorens een voorwaarde te testen

Een andere variant van de instructie **Doorlopen...Lus** voert eerst de instructies uit en test de *voorwaarde* telkens nadat de instructies zijn uitgevoerd. Deze variant zorgt ervoor dat uw procedure elke instructie ten minste één keer uitvoert.

```
Doorlopen
    GegevensVerwerken        'Procedure oproepen en gegevens ten minste
éénmaal verwerken.
    Antwoord = BerichtVenster("Wilt u nog meer gegevens verwerken?";
vbJaNee)
Lus Terwijl Antwoord = vbJa
```

Nu volgt een schematisch voorbeeld van de syntaxis voor deze variant van de instructie **Doorlopen...Lus**.

Doorlopen
instructies
Lus Terwijl *voorwaarde*

Een lus uitvoeren zolang een voorwaarde Onwaar is

Twee andere varianten zijn analoog met de vorige twee, maar blijven instructies uitvoeren zolang de *voorwaarde* **Onwaar** is en niet **Waar**.

```
Antwoord = BerichtVenster("Wilt u nog meer gegevens verwerken?";
vbJaNee)
Doorlopen Totdat Antwoord = vbNee
    GegevensVerwerken    'Procedure oproepen en gegevens verwerken
    Antwoord = BerichtVenster("Wilt u nog meer gegevens verwerken?";
vbJaNee)
Lus
```

In het volgende voorbeeld ziet u de versie die elke instructie ten minste één keer uitvoert.

```
Antwoord = BerichtVenster("Wilt u meer gegevens verwerken?" vbJaNee)
Doorlopen
    GegevensVerwerken        'Procedure oproepen en gegevens verwerken
    Antwoord = BerichtVenster("Wilt u meer gegevens verwerken?")
Lus Totdat Antwoord = vbNee
```

Nu volgen schematische voorbeelden van de syntaxis voor deze varianten van de instructie **Doorlopen...Lus**.

Doorlopen Totdat *voorwaarde*
instructies
Lus

– En –

Doorlopen
instructies
Lus Totdat *voorwaarde*

Let erop dat de *voorwaarde* voor **Doorlopen Totdat** exact gelijk is aan de *voorwaarde* voor **Terwijl Niet**.

Voor...Volgende

Als u niet weet hoe vaak u de instructies in een lus moet uitvoeren, kunt u **Doorlopen**-lussen gebruiken. Als u wel weet dat u de instructies een bepaald aantal keren uit moet voeren, gebruikt u de lus **Voor**. In tegenstelling tot de lus **Doorlopen**, gebruikt de lus **Voor** de variabele *teller* die in waarde toe- of afneemt als de lus wordt herhaald.

Met deze **Sub**-procedure kunt u bijvoorbeeld opgeven hoe vaak een geluidsignaal moet worden gegeven.

```
Sub PieptoonMeerdere()
    AantalTonen = InvoerVenster("Hoeveel pieptonen?")
    Voor Teller = 1 Tot AantalTonen
        Pieptoon
    Volgende Teller
Einde Sub
```

Het vorige voorbeeld is een eenvoudige lus. U kunt de lussen flexibeler maken door de teller een intervalwaarde te laten volgen die u zelf opgeeft. De volgende **Sub**-procedure vervangt elke tweede waarde in een matrix door nul.

```
Sub MatrixWissen (OpVerwijzing TeWissenMatrix ())
    Voor I = BenedenGrens(TeWissenMatrix) _
            Tot BovenGrens(TeWissenMatrix) Stappen 2
        TeWissenMatrix(i) = 0
    Volgende I
Einde Sub
```

Nu volgt een schematisch voorbeeld van de syntaxis voor **Voor...Volgende**.

Voor *teller* = *begin* **Tot** *Einde* [**Stappen** *intervalwaarde*]
instructies
Volgende [*teller*]

De argumenten *teller*, *begin*, *einde*, en *intervalwaarde* zijn numerieke argumenten. De vierkante haken geven aan dat het hier om optionele elementen gaat. U typt deze haken niet in de programmacode.

Opmerking Het argument *intervalwaarde* kan positief of negatief zijn. Als *intervalwaarde* positief is, moet *begin* kleiner dan of gelijk aan *einde* zijn, anders worden de instructies in de lus niet uitgevoerd. Als *intervalwaarde* negatief is, moet *begin* groter dan of gelijk aan *einde* zijn, anders worden de instructies in de lus evenmin uitgevoerd. Als **Stappen** niet wordt ingesteld, neemt *intervalwaarde* de standaardwaarde 1 aan.

Als Visual Basic de lus **Voor...Volgende** uitvoert, volgt het programma de volgende stappen:

1. Stelt *teller* zo in dat deze gelijk is aan *begin*.
2. Controleert of *teller* groter is dan *einde*. Zo ja, dan sluit Visual Basic de lus.

   (Als *intervalwaarde* negatief is, controleert Visual Basic of *teller* kleiner is dan *einde*).
3. Voert de *instructies* uit.
4. Stelt de intervalwaarde van *teller* op 1 in, of op *intervalwaarde* als dit argument is opgegeven.
5. Herhaalt stap 2 tot 4.

Voor Elke...Volgende

De lus **Voor Elke...Volgende** lijkt op de lus **Voor...Volgende**, maar herhaalt een groep instructies voor elk element in een verzameling objecten of in een matrix, in plaats van de instructies een bepaald aantal malen te herhalen. Dit is bijzonder handig als u niet weet hoeveel elementen in een verzameling voorkomen. Zie hoofdstuk 5, "Werken met objecten in Visual Basic", voor meer informatie over verzamelingen en meer voorbeelden van **Voor Elke...Volgende**.

De volgende **Sub**-procedure sluit bijvoorbeeld elke geopende werkmap.

```
Sub SluitElkeWerkmap()
'Map is een variabele die naar elementen verwijst in de
'Werkmapverzameling.
    Voor Elke Map in Werkmappen()
        Map.Sluiten
    Volgende Map
Einde Sub
```

De volgende procedure verhoogt de waarde van elke cel in het geselecteerde bereik, als deze waarde een cijfer is. Denk eraan dat een cellenbereik een object is en dat u de instructie **Toewijzen** moet gebruiken, als u een variabele maakt die naar een object verwijst.

```
Sub VerhogenHuidigeSelectie()
    Toewijzen TeVerhogenBereik = Selectie
    'C verwijst naar een bepaalde cel tijdens elke lus.
    Voor Elke C in TeVerhogenBereik
        Indien IsNumeriek(C.Waarde) Dan
            C.Waarde = C.Waarde + 1
        Einde Indien
    Volgende C
Einde Sub
```

Zie "Een object toewijzen aan een variabele" in hoofdstuk 6 voor meer informatie over de instructie **Toewijzen**.

Nu volgt een schematisch voorbeeld van de syntaxis voor **Voor Elke...Volgende**.

Voor Elke *element* **In** *groep*
    instructies
Volgende *element*

Als Visual Basic de lus **Voor Elke...Volgende** uitvoert, volgt het programma de volgende stappen:

1. Definieert *element* zodat deze variabele het eerste element in *groep* benoemt (op voorwaarde dat er ten minste één element is).
2. Voert de *instructies* uit.

3. Controleert of *element* het laatste element in *groep* is. Zo ja, dan voert Visual Basic de lus uit.
4. Definieert *element*, zodat het volgende element in *groep* wordt benoemd.
5. Herhaalt stap 2 tot 4.

Denk eraan dat de volgende beperkingen gelden als u **Voor Elke...Volgende** gebruikt:

- Bij verzamelingen kan *element* alleen een **Variant**-variabele, een algemene **Object**-variabele of een specifieke OLE-objectvariabele zijn. (Zie hoofdstuk 10, " Samenwerken met andere toepassingen", voor meer informatie over OLE-objecten). Bij matrices kan *element* alleen een **Variant**-variabele zijn.
- U kunt de instructie **Voor Elke...Volgende** niet gebruiken bij een door de gebruiker gedefinieerde matrix, omdat een **Variant** een dergelijk type matrix niet kan bevatten.

Zie de online *Visual Basic Naslaggids* onder *Voor* voor meer informatie over **Elke...Volgende**.

Een controlestructuur binnen een andere plaatsen (nesten)

Zoals het vorige voorbeeld laat zien, kunt u controlestructuren in andere controlestructuren onderbrengen (bijvoorbeeld een **Indien...Dan**-blok binnen een **Voor Elke...Volgende**-lus binnen een **Indien...Dan**-blok, enz.). Als een controlestructuur binnen een andere controlestructuur wordt geplaatst, wordt dit nesten genoemd.

De volgende **Functie**-procedure lijkt op de **Sub**-procedure VerhogenHuidigeSelectie in het voorgaande gedeelte, maar deze procedure doorzoekt een cellenbereik en telt het aantal cellen dat voldoet aan de waarde die u opgeeft.

```
Functie WaardenTellen(ZoekBereik; ZoekWaarde)
    Indien TypeNaam(ZoekBereik) <> "Bereik" Dan
        BerichtVenster "U kunt alleen een cellenbereik doorzoeken."
    Anders
        Voor Elke C in ZoekBereik
            Indien C.Waarde = ZoekWaarde Dan
                Teller = Teller + 1
            Einde Indien
        Volgende C
    Einde Indien
    WaardenTellen = Teller
Einde Functie
```

U ziet dat de eerste **Einde Indien**-instructie de binnenste **Indien**-lus sluit en dat de laatste **Einde Indien**-instructie de buitenste **Indien**-lus sluit. Zo geldt ook voor de geneste instructies **Voor** en **Voor Elke** dat de instructie **Volgende** automatisch van toepassing is op de dichtstbijzijnde voorafgaande instructie **Voor** of **Voor Elke**. De geneste structuren **Doorlopen...Lus** werken op een vergelijkbare wijze met de binnenste instructie **Lus** die voldoet aan de instructie **Doorlopen**.

Controlestructuren in Visual Basic kunnen onbeperkt genest worden. In de praktijk zult u geneste controlestructuren en lusstructuren overzichtelijker maken door de bij elkaar horende onderdelen van de controlestructuur of lus even ver te laten inspringen, zoals in de vorige programmacode het geval is.

Controlestructuren en procedures verlaten

Normaal gesproken geeft u een initiële voorwaarde op als u met een controlestructuur werkt. De programmacode doorloopt de controlestructuur, totdat aan deze voorwaarde voldaan is, verlaat vervolgens de controlestructuur en doorloopt de rest van de procedure. Soms is het echter mogelijk uw programmacode sneller te laten werken, door een controlestructuur vroegtijdig te verlaten (zodra aan een bepaalde voorwaarde voldaan is).

Als u bijvoorbeeld met de lus **Voor...Volgende** een waarde zoekt in een matrix en u deze waarde de eerste keer via de lus vindt, is het niet nodig de rest van de matrix te doorlopen. U kunt de lus onmiddellijk verlaten. Ook als een fout optreedt of zich iets anders voordoet waardoor de procedure niet hoeft worden uitgevoerd, kunt u de procedure verlaten. Met een van de instructies **Verlaten** kunt u een controlestructuur of een procedure verlaten.

Controlestructuren verlaten

Met de instructie **Verlaten Voor** kunt u de lussen **Voor** of **Voor...Volgende** direct verlaten. Met de instructie **Verlaten Doorlopen** verlaat u de lus **Doorlopen**.

In de volgende bewerking van het vorige voorbeeld telt het programma het aantal cellen dat doorlopen moet worden, voordat de cel wordt gevonden die u hebt opgegeven (dat wil zeggen dat de procedure de positie van de waarde vindt in de lijst met cellen die doorlopen moet worden). Zodra de overeenkomende waarde is gevonden, stopt het programma met tellen en zoeken.

```
Functie Positie(ZoekBereik; ZoekWaarde)
    Gevonden = Onwaar
    Voor Elke C in ZoekBereik
        Teller = Teller + 1
        Indien C.Waarde = ZoekWaarde Dan
            Gevonden = Waar
            Verlaten Voor 'Tellen en testen beëindigen
        Einde Indien
```

```
    Volgende C
    Indien Gevonden Dan
        Positie = Teller
    Anders
        Positie = 0
    Einde Indien
Einde Functie
```

Zoals het voorgaande voorbeeld laat zien, komt de instructie **Verlaten** vrijwel altijd voor in de instructie **Indien** of de instructie **Kiezen Ingeval** genest in een lus. De reden hiervoor is dat u alleen onder buitengewone omstandigheden de lus moet verlaten. In alle andere gevallen wacht u gewoon tot de lus is uitgevoerd.

De syntaxis van de instructie **Verlaten** is eenvoudig. **Verlaten Voor** mag net zo vaak in de lussen **Voor...Volgende** of **Voor Elke...Volgende** voorkomen als u nodig acht. Ook **Verlaten Doorlopen** mag u naar eigen inzicht binnen een **Doorlopen...Lus** gebruiken.

Voor *teller* = *begin* **Tot** *einde* **Stappen** *intervalwaarde*
    instructies
    Verlaten Voor
    instructies
Volgende *teller*

–Of–

Voor Elke *element* **In** *groep*
    instructies
    Verlaten Voor
    instructies
Volgende *element*

De instructie **Verlaten Doorlopen** werkt met alle versies van de syntaxis van **Doorlopen...Lus**.

Doorlopen Terwijl *voorwaarde*
    instructies
    Verlaten Doorlopen
    instructies
Lus

Procedures verlaten

U verlaat een procedure op dezelfde manier als een controlestructuur (met de instructies **Verlaten Sub** en **Verlaten Functie**). De syntaxis van deze instructies is gelijk aan die van **Verlaten Voor** en **Verlaten Doorlopen**, zoals beschreven in het vorige gedeelte. **Verlaten Sub** mag onbeperkt voorkomen binnen de structuur van de **Sub**-procedure. Dit geldt ook voor **Verlaten Functie**.

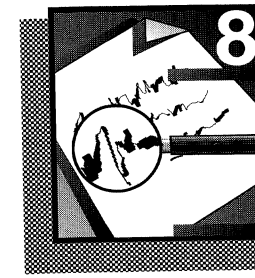
De mogelijkheid om de procedure te verlaten kan van pas komen als de procedure volledig is afgehandeld en kan terugkeren naar de procedure van waaruit deze was opgeroepen. Stel dat u het voorbeeld WaardenTellen zo wilt wijzigen dat de procedure onmiddellijk wordt verlaten nadat de reeks "Onjuiste gegevens" gevonden is. U kunt dan de instructie **Verlaten Functie** toevoegen.

```
Functie WaardenTellen(ZoekBereik; ZoekWaarde)
    Voor Elke C in ZoekBereik
        Indien C.Waarde = ZoekWaarde Dan
            Teller = Teller + 1
        AndersIndien C.Waarde = "Onjuiste gegevens" Dan
            WaardenTellen = Nulwaarde
            'Testen beëindigen en procedure direct verlaten.
            Verlaten Functie
        Einde Indien
    Volgende C
    WaardenTellen = Teller
Einde Functie
```

Tot zover de inleiding over controlestructuren in Visual Basic. Op basis van de kennis die u in dit en vorige hoofdstukken hebt opgedaan, kunt u effectieve Visual Basic-programmacode maken om uw werk te automatiseren. Maar naarmate uw programmacode ingewikkelder wordt, is de kans op fouten groter. In het volgende hoofdstuk zult u leren hoe u uw programmacode moet controleren en hoe u fouten moet opsporen, als uw programmacode niet zo wordt uitgevoerd als u had verwacht.

HOOFDSTUK 8

Fouten in de programmacode opsporen en programmacode testen



Visual Basic biedt een aantal hulpmiddelen om te analyseren hoe uw programmacode functioneert. Deze hulpmiddelen zijn vooral nuttig om de bron van haperingen in het programma op te sporen, de zogenaamde *bugs* (programmafouten). Maar u kunt met deze hulpmiddelen voor foutopsporing (ook wel *debugging tools* genoemd) ook experimenteren met het effect van wijzigingen in uw programmacode of nagaan hoe programmacode werkt die door anderen is geschreven.

In dit hoofdstuk wordt behandeld hoe u de hulpmiddelen voor foutopsporing van Visual Basic moet gebruiken.

Inhoud

Hoe de hulpmiddelen voor foutopsporing helpen

Na het uitvoeren van een lange reeks berekeningen kan het voorkomen dat het resultaat niet juist is. Met de hulpmiddelen voor foutopsporing kunt u nagaan wat er fout is gegaan. Misschien hebt u vergeten de juiste waarde aan een variabele toe te wijzen, of u hebt de verkeerde operator gekozen of niet de juiste formule gebruikt.

De hulpmiddelen voor foutopsporing vormen geen wondermiddel dat alle problemen meteen oplost, maar bieden een aantal hulpmiddelen voor het lokaliseren van fouten. Stel eerst vast welk type fout er optreedt wanneer u de programmacode uitvoert. Maak daarna gebruik van de hulpmiddelen voor foutopsporing die in dit hoofdstuk wordt beschreven om de programmacode te doorlopen en te lokaliseren waar de fout optreedt. Hoe beter u begrijpt hoe de programmacode werkt, des te sneller u een eventuele programmeerfout zult ontdekken.

De hulpmiddelen voor foutopsporing kunnen geen diagnose van de fouten maken of fouten herstellen. U kunt met deze hulpmiddelen echter wel analyseren hoe de uitvoering van elk afzonderlijk gedeelte van de programmacode verloopt en hoe variabelen en de instellingen van eigenschappen veranderen gedurende de uitvoering van de instructies. Hulpmiddelen voor foutopsporing zijn instrumenten om binnen de programmacode te kijken en vast te stellen wat er gebeurt en waarom.

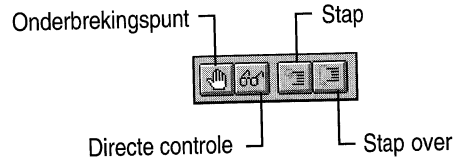
Soorten fouten

Als u de volgende lijst met mogelijke soorten fouten bekijkt, zult u het nut van foutopsporing zeker inzien.

- *Programmeerfouten* zijn een gevolg van onjuist geformuleerde programmacode en vergrijpen tegen de syntaxis van de programmeertaal. Het kan zijn dat u een woord verkeerd hebt ingetypt, noodzakelijke interpunctie hebt vergeten of een instructie **Volgende** hebt gebruikt zonder een corresponderende **Voor**-instructie. Deze fouten worden door Visual Basic ontdekt wanneer u de regel verlaat of onmiddellijk voordat de programmacode wordt uitgevoerd.
- *"Run-time"-fouten* treden op wanneer een bewerking moet worden uitgevoerd die niet mogelijk is. Dit is bijvoorbeeld het geval wanneer wordt verwezen naar een object dat in de huidige context niet kan bestaan. Stel dat u een stuk programmacode hebt geschreven waarin wordt verwezen naar een cellenbereik op het actieve blad. Als het actieve blad echter geen werkblad is, zal er tijdens de uitvoering van de programmacode een "run-time"-fout optreden, aangezien alleen een werkblad een cellenbereik kan bevatten.
- *Logische fouten* zijn fouten die optreden in programmacode die weliswaar syntactisch juist is en wordt uitgevoerd zonder dat er onjuiste bewerkingen plaatsvinden, maar die toch niet tot een correct resultaat leidt. Alleen door de programmacode te testen en de resultaten te analyseren, kunt u nagaan of de programmacode de gewenste uitkomst oplevert. Logische fouten kunnen ook leiden tot "run-time"-fouten.

Knoppen voor foutopsporing op de Visual Basic-werkbalk

Op de Visual Basic-werkbalk bevinden zich verschillende knoppen die van pas komen bij het opsporen van fouten. De Visual Basic-werkbalk verschijnt automatisch op het scherm als u naar een Visual Basic-module overschakelt. In de volgende figuur ziet u het gedeelte van de werkbalk waarop deze knoppen zijn afgebeeld.



Het volgende overzicht geeft een korte beschrijving van de functie van iedere knop. In de volgende gedeelten van dit hoofdstuk worden de situaties beschreven waarin u met deze knoppen uw programmacode efficiënter kunt testen op fouten of kunt analyseren.

Foutopsporingsknop	Functie
Onderbrekingspunt	Maakt onderbrekingspunten of verwijdert deze. Een onderbrekingspunt is een plaats in de programmacode waar de uitvoering door Visual Basic wordt gestopt.
Directe controle	Geeft de huidige waarde van een expressie weer terwijl de onderbrekingsmodus actief is.
Stap	Voert alleen de eerstvolgende uitvoerbare regel van de programmacode uit. Als de programmacode een andere procedure oproept, wordt de opgeroepen procedure stap voor stap uitgevoerd. Zodra de procedure eindigt, gaat de uitvoering verder met de volgende regel van de oorspronkelijke programmacode.
Stap over	Voert de eerstvolgende uitvoerbare regel van de programmacode uit. Als de programmacode een andere procedure oproept, wordt deze in zijn geheel uitgevoerd, zonder dat u de afzonderlijke stappen van de opgeroepen procedure te zien krijgt.

De onderbrekingsmodus gebruiken

De onderbrekingsmodus legt de uitvoering van de programmacode stil en geeft een beeld van de toestand ervan op dat moment. Als u de onderbrekingsmodus activeert, gebeurt het volgende:

- Het venster **Foutopsporing** verschijnt op het scherm.
- U kunt de Visual Basic-werkbalk gebruiken, maar u kunt niet naar andere bladen in de werkmap schakelen.

In onderbrekingsmodus worden de instellingen van variabelen en eigenschappen bewaard, zodat u de actuele staat ervan kunt analyseren. U kunt ook wijzigingen aanbrengen die van invloed zijn op de wijze waarop de programmacode wordt uitgevoerd. Als de onderbrekingsmodus actief is, kunt u het volgende doen:

- Vaststellen welke actieve procedures zijn opgeroepen
- De waarden van variabelen, eigenschappen en instructies bekijken
- De waarden van variabelen en eigenschappen wijzigen
- Visual Basic-instructies direct uitvoeren
- De uitvoering van de programmacode per regel controleren.

Onderbrekingsmodus activeren bij een probleempunt

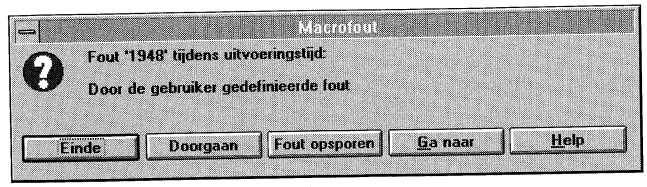
Tijdens het opsporen van fouten wilt u wellicht de programmacode laten stoppen op de plaats waar u denkt dat het probleem is ontstaan. In Visual Basic kunt u hiervoor gebruik maken van onderbrekingspunten en de instructie **Stoppen**. Een *onderbrekingspunt* definieert een instructie of een aantal voorwaarden waarbij Visual Basic automatisch de uitvoering van de programmacode stopt en de onderbrekingsmodus activeert. De instructie met het onderbrekingspunt wordt niet uitgevoerd. De instructie **Stoppen**, een sleutelwoord in Visual Basic, activeert eveneens de onderbrekingsmodus in Microsoft Excel.

De onderbrekingsmodus automatisch activeren

De onderbrekingsmodus wordt door Visual Basic geactiveerd wanneer er sprake is van een van de volgende situaties:

- Tijdens het uitvoeren van een procedure wordt een regel met een onderbrekingspunt bereikt.
- Tijdens het uitvoeren wordt een instructie **Stoppen** bereikt.
- Een onderbrekingsexpressie die u hebt gedefinieerd in het dialoogvenster **Controle toevoegen** ondergaat een wijziging of wordt **Waar**, afhankelijk van hoe u de expressie hebt gedefinieerd. Zie "Gegevens observeren met controle-expressies" verderop in dit hoofdstuk voor meer informatie over onderbrekingsexpressies.
- Een instructie in een coderegel blijkt een nog niet afgevangen "run-time"-fout te bevatten. Zie hoofdstuk 9, "Foutafhandeling en foutwaarden", voor meer informatie over het afvangen van fouten.

  In de volgende figuur ziet u een voorbeeld van een bericht dat op het scherm verschijnt wanneer er een fout optreedt. Afhankelijk van het soort fout dat optreedt, kunt u de uitvoering van de programmacode afbreken of de fout proberen op te sporen.



De onderbrekingsmodus handmatig activeren

Als u de onderbrekingsmodus wilt activeren terwijl de programmacode niet wordt uitgevoerd, kiest u de opdracht **Foutopsporing** in het menu **Beeld**. In Visual Basic kunt u de onderbrekingsmodus activeren tijdens de uitvoering van de programmacode. Dit doet u met behulp van het toetsenbord.

Systeem	Druk op
Windows	CTRL+BREAK of ESC
Macintosh	COMMAND+PUNT of ESC

De uitvoering selectief stoppen met een onderbrekingspunt

Een onderbrekingspunt zorgt ervoor dat Visual Basic stopt vlak voor een bepaalde regel programmacode. Als Visual Basic bij het uitvoeren van een procedure een regel programmacode met een onderbrekingspunt tegenkomt, wordt de onderbrekingsmodus geactiveerd.

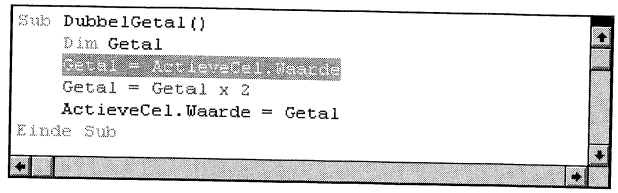
Opmerking U kunt op elk willekeurig moment een onderbrekingspunt instellen of verwijderen, ongeacht of de onderbrekingsmodus actief is of niet. U kunt echter geen onderbrekingspunt binnen een invoegmacro instellen, aangezien invoegmacro's niet kunnen worden bewerkt. Zie hoofdstuk 13, "Automatische procedures en invoegmacro's maken", voor meer informatie over invoegmacro's.

▶ Onderbrekingspunten instellen of verwijderen

1. Terwijl er een Visual Basic-module actief is, verplaatst u de invoegpositie naar de regel in de programmacode waar u een onderbrekingspunt wilt instellen of verwijderen.
2. Kies **Onderbrekingspunt** in het menu **Starten**.

   U kunt ook de knop Onderbrekingspunt op de Visual Basic-werkbalk kiezen.

Als u een onderbrekingspunt instelt, wordt de geselecteerde regel gemarkeerd in de door u aangegeven kleuren en wordt de tekst vet weergegeven. In de volgende figuur ziet u een voorbeeld van een procedure die door een onderbrekingspunt is gestopt.



De huidige instructie herkennen

Wanneer de onderbrekingsmodus actief is, wordt de *huidige instructie* (dat wil zeggen: de volgende instructie die zal worden uitgevoerd) gemarkeerd met een kader, mits de module waarin de instructie staat zichtbaar is.

Als de huidige instructie eveneens een onderbrekingspunt bevat, geeft alleen het kader de regel in de programmacode aan. Zodra een andere regel de huidige instructie wordt, wordt de regel met het onderbrekingspunt weer vet en in kleur weergegeven.

Als u de kleur van de tekst en achtergrond voor de huidige instructie of een regel met een onderbrekingspunt wilt bepalen, kiest u de opdracht **Opties** in het menu **Extra**. Kies daarna de tab Module-opmaak en selecteer vervolgens de gewenste kleuren in het vak "Kleuren".

De programmacode bij een onderbrekingspunt controleren

Als u een onderbrekingspunt bereikt en de programmacode stopt, kunt u de huidige status van de programmacode nader bekijken.

De uitvoering van de programmacode wordt door een onderbrekingspunt gestopt, net voordat de regel wordt uitgevoerd waarin het onderbrekingspunt staat. Als u wilt zien wat er gebeurt als de regel met het onderbrekingspunt wordt uitgevoerd, moet u minstens één instructie meer uitvoeren. Dit kunt u doen door de verschillende instructies stapsgewijs uit te voeren. Zie "Bepaalde gedeelten van de programmacode uitvoeren" verderop in dit hoofdstuk voor meer informatie over stapsgewijze uitvoering van instructies.

Wanneer u een probleem probeert te isoleren, houdt u er dan rekening mee dat een instructie ook indirect een fout kan hebben veroorzaakt, door een onjuiste waarde aan een variabele toe te kennen. U kunt de waarden van de variabelen en eigenschappen bekijken terwijl de onderbrekingsmodus actief is, door gebruik te maken van een controle-expressie of het deelvenster **Direct** in het venster **Foutopsporing**. Zie "Gegevens observeren met controle-expressies" verderop in dit hoofdstuk voor meer informatie over het definiëren en toepassen van controle-expressies. In het gedeelte "Gegevens en procedures testen met het deelvenster Direct" wordt uitgelegd hoe u expressies kunt testen en de waarden van variabelen kunt bekijken.

De onderbrekingsmodus activeren met de instructie Stop

Behalve een onderbrekingspunt kunt u ook een instructie **Stoppen** plaatsen om de uitvoering van de procedure te stoppen. Elke keer als Visual Basic een instructie **Stoppen** tegenkomt, wordt de uitvoering gestopt en de onderbrekingsmodus geactiveerd. Als u wilt terugkeren naar de uitvoering, kiest u de opdracht **Doorgaan** in het menu **Starten** of kiest u de knop Macro hervatten op de Visual Basic-werkbalk. Hoewel de instructie **Stoppen** zich op dezelfde wijze gedraagt als een onderbrekingspunt, wordt deze niet op dezelfde manier ingevoegd of gewist.

Opmerking Er zijn een aantal verschillen tussen de instructie **Stoppen** en onderbrekingspunten. Het belangrijkste verschil is, dat alle onderbrekingspunten worden gewist als u de huidige werkmap sluit en opnieuw opent. Dit in tegenstelling tot de instructie **Stoppen**, die in de programmacode blijft staan totdat u de instructie verwijdert.

De instructie **Stoppen** moet niet worden verward met de instructie **Einde**. De beide instructies zijn niet onderling verwisselbaar. Een instructie **Stoppen** onderbreekt de uitvoering alleen tijdelijk door de onderbrekingsmodus te activeren, terwijl de instructie **Einde** de uitvoering volledig stopt.

Het venster Foutopsporing gebruiken

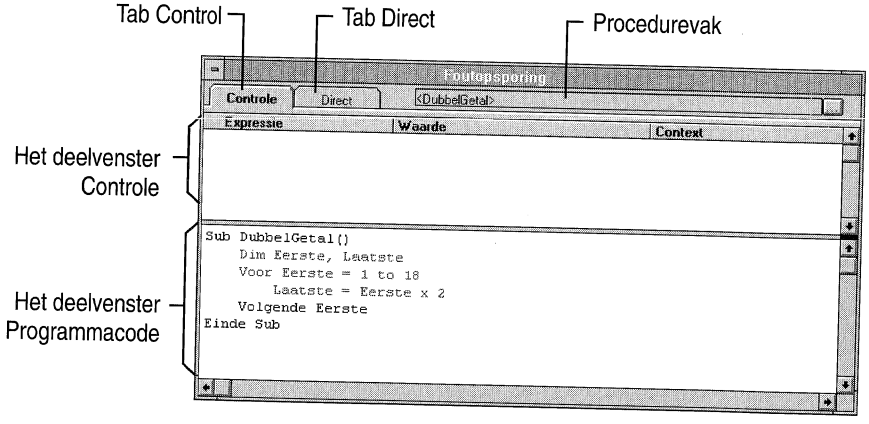
Soms kunt u de oorzaak van een probleem vinden door gedeeltes van de programmacode uit te voeren. Maar veel vaker moet u ook nagaan wat er met de gegevens gebeurt. Het kan bijvoorbeeld voorkomen dat u ontdekt dat aan een variabele of een eigenschap een onjuiste waarde is toegewezen. In dat geval moet u bepalen hoe en waarom aan die variabele of eigenschap een onjuiste waarde is toegewezen.

In het venster **Foutopsporing** kunt u de waarden van expressies en variabelen controleren terwijl u de instructies in de programmacode stapsgewijs uitvoert. U kunt in het venster **Foutopsporing** ook de waarden van variabelen en eigenschappen in onderbrekingsmodus wijzigen, zodat u kunt nagaan hoe verschillende waarden de programmacode beïnvloeden.

U kunt het venster **Foutopsporing** weergeven op de volgende manieren:

- Activeer de onderbrekingsmodus. Het venster **Foutopsporing** verschijnt automatisch wanneer Visual Basic de onderbrekingsmodus activeert.
- Kies de opdracht **Foutopsporing** in het menu **Beeld**.

Het venster **Foutopsporing** bestaat uit twee helften. Het deelvenster **Controle** of het deelvenster **Direct** verschijnt in de bovenste helft van het scherm, terwijl het deelvenster met de programmacode in de onderste helft verschijnt. Het *programmacodedeelvenster* geeft in de programmacode het punt weer waar de uitvoering van de programmacode op dat moment is aangeland, mits dat zichtbaar is. U kunt door het deelvenster met de programmacode schuiven, zodat u andere delen van de programmacode kunt bekijken.



Het venster Foutopsporing met het deelvenster Controle op de voorgrond

Het deelvenster **Controle** geeft de huidige controle-expressies weer, expressies waarvan u de waarden controleert terwijl de programmacode wordt uitgevoerd. Het deelvenster **Direct** geeft informatie weer over de resultaten van de foutopsporingsinstructies in de programmacode of informatie die u hebt opgevraagd door opdrachten direct in het deelvenster op te geven. De eerste keer dat het venster **Foutopsporing** wordt weergegeven, staat het deelvenster **Direct** op de voorgrond, daarna staat het laatst gekozen deelvenster op de voorgrond.

In het deelvenster **Controle** worden in de kolom Context de procedure, de module of modulen aangegeven waarin elke controle-expressie wordt geëvalueerd. Het deelvenster **Controle** kan alleen een waarde voor een controle-expressie weergeven als de huidige instructie in de aangegeven context staat. Als dat niet het geval is, bevat de kolom Waarde een bericht dat de instructie niet in de context staat.

Bepaalde gedeelten van de programmacode uitvoeren

Als u vermoedt welke instructie de fout heeft veroorzaakt, kan een enkel onderbrekingspunt u helpen om het probleem te vinden. Het zal echter vaker zo zijn dat u alleen het gebied kunt aanwijzen waarin de fout is veroorzaakt. In dat geval kunt u het probleemgebied met een onderbrekingspunt isoleren. Met de functies **Stap** en **Stap over** kunt u vervolgens het effect van elke instructie nagaan.

Werken met Stap

Stapsgewijze uitvoering houdt in dat u de instructies één voor één laat uitvoeren. Telkens wanneer u een instructie hebt laten uitvoeren, kunt u de resultaten ervan in het venster **Foutopsporing** bekijken. Aan het begin van een nieuwe procedure voert Visual Basic de huidige instructie uit, om dan door te gaan naar de volgende instructie van de betrokken procedure. Daarna schakelt Visual Basic weer terug naar de onderbrekingsmodus.

▶ **De instructies stapsgewijs laten uitvoeren**

- Kies de opdracht **Stap** in het menu **Starten**.

  U kunt ook de knop Stap kiezen op de Visual Basic-werkbalk.

Als de uitgevoerde instructie een oproep is voor een andere procedure in een niet-beschermde module, kunt u met Stap de instructies in de opgeroepen procedure één voor één laten uitvoeren. Als u het einde van de opgeroepen procedure bereikt, gaat u met Stap terug naar het punt in de programmacode waaruit de procedure werd opgeroepen. Als de opgeroepen procedure zich in een invoegmacro of een module met de eigenschap Zichtbaar ingesteld op xlSpeciaalverborgen bevindt, gedraagt de opdracht **Stap** zich als **Stap over**.

Opmerking U kunt de opdracht **Stap** tussen afzonderlijke instructies gebruiken, zelfs wanneer deze op dezelfde regel staan. Een regel programmacode kan twee of meer instructies bevatten, die van elkaar gescheiden worden door een dubbele punt (:). In Visual Basic wordt met een kader aangegeven welke van de instructies als volgende wordt uitgevoerd. onderbrekingspunten gelden alleen voor de eerste instructie op een regel.

Werken met Stap over

De opdracht **Stap over** werkt hetzelfde als **Stap**, behalve wanneer de huidige instructie een oproep voor een procedure bevat. Het verschil is dat **Stap over** de procedure als één geheel uitvoert om vervolgens verder te gaan met de volgende instructie van de huidige procedure, terwijl **Stap** ook de instructies in de opgeroepen procedure stapsgewijs uitvoert. Als voorbeeld nemen we een module die de volgende twee procedures bevat waarbij de eerste procedure de tweede oproept.

```
Sub BerekenPrijs()
    AantalAandelen = InvoerVenster("Aantal aandelen?")
    AandeelPrijs = InvoerVenster("Prijs per aandeel?")
    Kosten = Provisie(AantalAandelen; AandeelPrijs)
    BerichtVenster "De provisie bedraagt: " & Kosten
Einde Sub

Functie Provisie(AandelenVerkocht; PrijsPerAandeel)
    TotaleVerkPrijs = AandelenVerkocht * PrijsPerAandeel
    Indien TotaleVerkPrijs <= 15000 Dan
        Provisie = 25 + (0,03 * AandelenVerkocht)
    Anders
        Provisie = 25 + (0,03 * (0,9 * AandelenVerkocht))
    Einde Indien
Einde Functie
```

Als u de procedure BerekenPrijs stapsgewijs uitvoert, functioneren **Stap** en **Stap over** op dezelfde manier, totdat u de volgende regel wilt uitvoeren:

```
    Kosten = Provisie(AantalAandelen; AandeelPrijs)
```

Als u elke regel met de knop Stap uitvoert, verschijnt in het codedeelvenster van het venster **Foutopsporing** de programmacode van de procedure Provisie. De eerste regel van de opgeroepen procedure wordt de huidige instructie. De werkwijze met **Stap** is vooral nuttig als u de programmacode van de procedure Provisie wilt analyseren.

Als u op dezelfde regel echter de knop Stap over gebruikt, wordt de procedure Provisie door Visual Basic als één geheel uitgevoerd, zonder dat u de afzonderlijke regels te zien krijgt. Het programmacodedeelvenster geeft de volgende regel in de procedure BerekenPrijs weer, tenzij Provisie een onderbrekingspunt of een instructie **Stoppen** bevat. De werkwijze met **Stap over** is handig wanneer u op hetzelfde niveau in de programmacode wilt blijven en de procedure Provisie niet wilt bekijken.

Tip Als u onbedoeld in de oproep van een lange procedure terechtkomt, kan het even duren voordat u weer terug bent bij de procedure waar u vandaan kwam. U kunt sneller teruggaan door een onderbrekingspunt te plaatsen aan het eind van de procedure waarin u terecht bent gekomen en dan de knop Macro hervatten op de Visual Basic-werkbalk te kiezen. Als u het onderbrekingspunt hebt bereikt, kunt u de oorspronkelijke procedure verder weer stapsgewijs uitvoeren.

U kunt **Stap** en **Stap over** naast elkaar gebruiken, waarbij de juiste keus afhangt van de gedeelten van de programmacode die u op een bepaald moment wilt bekijken.

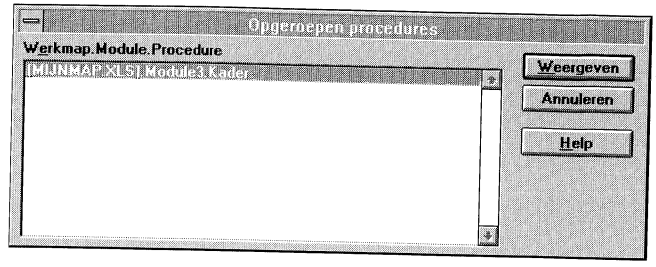
▶ **Werken met Stap over**

- Kies de opdracht **Stap over** in het menu **Starten**.

  U kunt ook de knop Stap over op de Visual Basic-werkbalk kiezen.

Het dialoogvenster Opgeroepen procedures gebruiken

Als u de werking van de programmacode wilt volgen terwijl deze een serie procedures uitvoert, dan helpt het dialoogvenster **Opgeroepen procedures** u daarbij. Deze functie is vooral handig wanneer u geneste procedures wilt controleren. Het kan bijvoorbeeld gebeuren dat de eerste procedure een tweede procedure oproept, die op zijn beurt weer een derde procedure oproept, en dat allemaal voordat de eerste procedure afgesloten is. Dergelijke geneste procedures zijn soms moeilijk te volgen.



U kunt het dialoogvenster **Opgeroepen procedures** alleen op het scherm weergeven als de onderbrekingsmodus is geactiveerd.

▶ **Het dialoogvenster Opgeroepen procedures weergeven**

- Kies in het venster **Foutopsporing** de knop rechts van het vak dat de naam van de huidige procedure bevat.

Geneste procedures controleren

Wanneer u te maken hebt met een reeks geneste oproepen, geeft het dialoogvenster **Opgeroepen procedures** de meest recente actieve procedure-oproep boven aan de lijst weer. De informatie die voor elke procedure wordt gegeven, begint met de bestandsnaam van de werkmap, gevolgd door de naam van de module en de naam van de opgeroepen procedure.

U kunt met het dialoogvenster **Opgeroepen procedures** de instructie in een procedure weergeven die de besturing van de programmacode doorgeeft aan de volgende procedure in de lijst.

▶ **Een instructie weergeven die een andere procedure oproept**

1. Selecteer in het dialoogvenster **Opgeroepen procedures** de procedure-oproep die u wilt weergeven.
2. Kies de knop Weergeven.

   De procedure verschijnt in het programmacodedeelvenster.

Gegevens observeren met controle-expressies

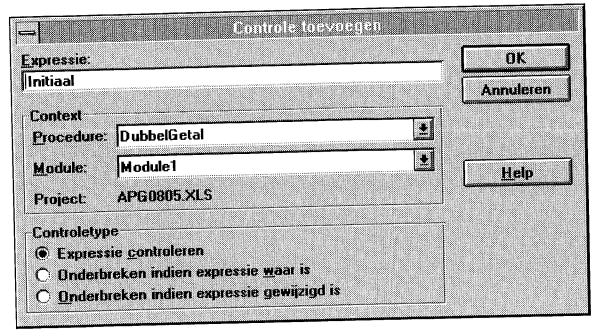
Het is mogelijk dat u tijdens het foutopsporingsproces merkt dat een specifiek probleem alleen voorkomt als een bepaalde variabele of eigenschap een specifieke waarde of reeks van waarden aanneemt. Of u komt erachter dat een berekening niet tot het beoogde resultaat leidt. Veel problemen die zich voordoen tijdens het opsporen van fouten kunnen niet onmiddellijk herleid worden tot een enkele instructie. In dit geval kan het dus noodzakelijk zijn dat u het gedrag van een variabele of expressie tijdens de gehele procedure bekijkt.

U kunt de waarde van een bepaalde variabele of expressie in de gaten houden met een *controle-expressie*. De controle-expressies worden door Visual Basic voor u gecontroleerd. In de onderbrekingsmodus verschijnen de controle-expressies in het deelvenster **Controle** van het venster **Foutopsporing**, waarin u de actuele waarden van de controle-expressies kunt controleren.

U kunt ook opgeven dat controle-expressies de onderbrekingsmodus moeten activeren zodra de waarde van de controle-expressie verandert of gelijk wordt aan een van tevoren bepaalde waarde. Zo kunt u een controle-expressie gebruiken om de onderbrekingsmodus te activeren zodra een lusteller een bepaalde waarde heeft bereikt, in plaats van de opdracht **Stap** te gebruiken en honderden lussen te moeten doorlopen. U kunt de onderbrekingsmodus ook activeren telkens wanneer een bepaalde variabele van waarde verandert.

Een controle-expressie toevoegen

U kunt een controle-expressie toevoegen voordat u een procedure start of nadat de onderbrekingsmodus is geactiveerd. Hiervoor gebruikt u het dialoogvenster **Controle toevoegen**.



In het volgende overzicht worden de onderdelen van het dialoogvenster **Controle toevoegen** beschreven.

Onderdeel	Beschrijving
Expressie	In dit vak voert u de expressie in die door de controle-expressie wordt gecontroleerd. De expressie kan een variabele zijn, een eigenschap, een functie-oproep of elke andere geldige expressie.
Context	Met deze opties bepaalt u het bereik van variabelen die in de expressie worden gecontroleerd. Gebruik de opties in deze groep als u variabelen hebt met dezelfde naam maar een ander bereik. U kunt het bereik beperken waarbinnen variabelen worden gecontroleerd die gedefinieerd zijn voor een specifieke procedure of module of die in de hele programmacode voorkomen. Hoe beperkter de context, hoe sneller Visual Basic een variabele kan controleren.
Controletype	Met deze opties bepaalt u hoe Visual Basic reageert op de controle-expressie. Visual Basic kan de expressie controleren en de waarde ervan weergeven in het deelvenster **Controle** zodra de onderbrekingsmodus wordt geactiveerd. U kunt ook automatisch de onderbrekingsmodus activeren als de expressie de waarde Waar (ongelijk aan nul) als resultaat geeft of wanneer de waarde van de expressie verandert.

▶ **Een controle-expressie toevoegen**

1. Kies de opdracht **Controle toevoegen** in het menu **Extra**.
2. Typ in het vak "Expressie" de expressie die u wilt controleren.
3. Selecteer de naam van de procedure of module in het bijbehorende vak, als u een bereik wilt bepalen voor de te controleren expressie.
4. Als u wilt bepalen hoe Visual Basic moet reageren op de controle-expressie, selecteert u het overeenkomstige keuzerondje onder optiegroep "Controletype".
5. Kies de knop OK.

Een controle-expressie bewerken of verwijderen

Iedere controle-expressie die u selecteert in het deelvenster **Controle** of het venster **Foutopsporing**, kan worden bewerkt of verwijderd. Als u de opdracht **Controle bewerken** in het menu **Extra** kiest, verschijnt het dialoogvenster **Controle bewerken** op uw scherm. Dit dialoogvenster lijkt op het dialoogvenster **Controle toevoegen**, maar bevat tevens de knop Verwijderen.

▶ **Een controle-expressie bewerken**

1. Kies in het deelvenster **Controle** van het venster **Foutopsporing** de controle-expressie die u wilt bewerken.
2. Kies de opdracht **Controle bewerken** in het menu **Extra**.
3. Wijzig in het dialoogvenster **Controle bewerken** de controle-expressie, het controlebereik voor de variabelen of het controletype.
4. Kies de knop OK.

Tip U kunt de controle-expressies die in het venster **Foutopsporing** worden weergegeven, ook beperken door te dubbelklikken op de gewenste expressie in het deelvenster **Controle**. Hierdoor wordt de geselecteerde controle-expressie door Visual Basic in het dialoogvenster **Controle bewerken** weergegeven.

▶ **Een controle-expressie verwijderen**

1. Selecteer in het deelvenster **Controle** van het venster **Foutopsporing** de controle-expressie die u wilt verwijderen.
2. Kies de opdracht **Controle bewerken** in het menu **Extra**.
3. Kies de knop Verwijderen in het dialoogvenster **Controle bewerken**.

Tip Als uw toetsenbord een toets DEL heeft, kunt u een controle-expressie verwijderen door deze in het deelvenster **Controle** te selecteren en dan op DEL te drukken.

Controletypen herkennen

Links van iedere controle-expressie in het deelvenster **Controle** van het venster **Foutopsporing** bevindt zich een pictogram dat het controletype van die expressie aangeeft. In de volgende figuur ziet u het pictogram voor elk van de drie controletypen.

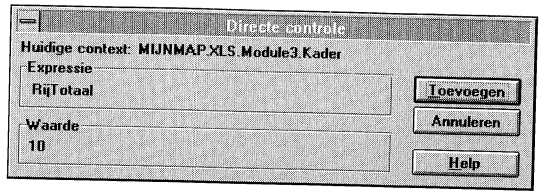
 – Controle-expressie

 – Onderbreken als de expressie Waar is

 – Onderbreken als de expressie is gewijzigd

Werken met Directe controle

U kunt de waarde van een expressie waarvoor u geen controle-expressie hebt gedefinieerd, controleren terwijl de onderbrekingsmodus actief is. Dat kunt u doen met het dialoogvenster **Directe controle**.



Het dialoogvenster **Directe controle** geeft de waarde weer van een expressie die u kiest uit:

- Een Visual Basic-module
- Het codedeelvenster van het venster **Foutopsporing**

Wilt u deze expressie blijven zien, dan kiest u de knop Toevoegen. Als de huidige expressie niet door Visual Basic kan worden gecontroleerd, is de knop Toevoegen niet beschikbaar.

▶ **Een controle-expressie toevoegen vanuit het dialoogvenster Directe controle**

1. Selecteer de expressie die u wilt controleren.
2. Kies de opdracht **Directe controle** in het menu **Extra**.

   U kunt ook de knop Directe controle op de Visual Basic-werkbalk kiezen.
3. Kies de knop Toevoegen.

Gegevens en procedures testen met het deelvenster Direct

Terwijl u fouten aan het opsporen bent of met programmacode aan het experimenteren bent, kunt u procedures testen, expressies controleren of nieuwe waarden toewijzen aan variabelen of eigenschappen. U doet dit door het deelvenster **Direct** van het venster **Foutopsporing** te kiezen. Expressies controleert u door de waarden van de expressies in het deelvenster **Direct** af te beelden.

Informatie afbeelden in het deelvenster Direct

Er zijn twee manieren om informatie in het deelvenster **Direct** af te beelden:

- Neem de methode **Afbeelden** op in de Visual Basic-programmacode.
- Geef de methode **Afbeelden** rechtstreeks in het deelvenster **Direct** op.

Het gebruik van de methode **Afbeelden** biedt een aantal voordelen ten opzichte van controle-expressies. Zo kunt u gegevens of andere berichten zien terwijl de onderbrekingsmodus actief is. Een ander voordeel is dat de respons wordt weergegeven in een apart gebied, namelijk het deelvenser **Direct**, zodat deze gescheiden blijft van de uitvoer die de gebruiker ziet. Tenslotte hoeft u de instructies niet iedere keer dat u aan de programmacode werkt opnieuw te definiëren, omdat u de programmacode als onderdeel van de werkmap kunt opslaan.

U kunt elke geldige expressie in het deelvenster **Direct** controleren, waaronder expressies die betrekking hebben op eigenschappen.

Informatie afbeelden vanuit de programmacode

De methode **Afbeelden** stuurt uitvoer naar het deelvenster **Direct** als de methode wordt voorafgegaan door het speciale object **Foutopsp**. De volgende instructie beeldt bijvoorbeeld elke keer dat deze wordt uitgevoerd de waarde van `Salaris` af in het deelvenster **Direct**:

```
Foutopsp.Afbeelden "Salaris = "; Salaris
```

Deze werkwijze werkt het beste als er een bepaalde plaats in de programmacode is waar de variabele, in dit geval `Salaris`, normaal gesproken verandert. Zo kunt u de vorige instructie in een lus plaatsen waarin de waarde van `Salaris` herhaaldelijk wordt gewijzigd.

Zie *Foutopsporing* in de online Visual Basic Naslaggids voor meer informatie over het object **Foutopsp**.

Informatie afbeelden vanuit het deelvenster Direct

Als de onderbrekingsmodus actief is, kunt u de methode **Afbeelden** zonder het object **Foutopsp** toepassen. Typ of plak een instructie in het deelvenster **Direct** en druk vervolgens op ENTER, waarna de instructie wordt uitgevoerd. De regels die de methode **Afbeelden** bevatten in het volgende voorbeeld, kunt u als invoer gebruiken in het deelvenster **Direct**. In de regels die daarop volgen, ziet u de reactie van Visual Basic op de instructies.

```
Afbeelden Beginwaarde
    4
Afbeelden Eindwaarde
    6
```

De methode **Afbeelden** kan ook worden aangeduid met een vraagteken (?). Het vraagteken betekent precies hetzelfde als **Afbeelden** en kan in dezelfde context worden gebruikt. Zo zouden de instructies uit het vorige voorbeeld ook als volgt kunnen worden ingevoerd.

```
? Beginwaarde
    4
? Eindwaarde
    6
```

Opdrachten uitvoeren in het deelvenster Direct

U kunt het deelvenster **Direct** niet alleen gebruiken voor het afbeelden van de waarden van variabelen en eigenschappen. In het deelvenster **Direct** kunt u ook de volgende Visual Basic-opdrachten uitvoeren:

- Opdrachten die de waarden van variabelen en eigenschappen wijzigen.
- Instructies die het gedrag van de programmacode wijzigen.

De opdrachten die u in het deelvenster **Direct** opgeeft, worden onmiddellijk door Visual Basic uitgevoerd.

Waarden aan variabelen en eigenschappen toewijzen

Als u de mogelijke oorzaak van een fout op het spoor bent gekomen, kan het nuttig zijn de effecten van bepaalde gegevenswaarden te testen. In de onderbrekingsmodus kunt u met instructies waarden instellen in het deelvenster **Direct**. Hierna volgen enkele voorbeelden:

```
ActieveWerkmap.Grafieken(1).AutoAanpassing = Waar
MaxBelast = ,36
```

De eerste instructie wijzigt een eigenschap van een grafiek, terwijl de tweede instructie een waarde toewijst aan een variabele.

Nadat u de waarden van een of meer eigenschappen en variabelen hebt ingesteld, kunt u met de uitvoering verdergaan om de resultaten te bekijken. U kunt ook het effect op procedures testen, zoals dat in het volgende gedeelte wordt beschreven.

Procedures testen

In het deelvenster **Direct** kunt u elke geldige uitvoerbare Visual Basic-instructie controleren, met uitzondering van gegevensdeclaraties. U kunt echter wel procedures oproepen, waardoor u de mogelijke effecten van elk type procedure kunt testen met een willekeurige reeks argumenten. U kunt gewoon een instructie in het deelvenster **Direct** invoeren, net zoals u dat in een Visual Basic-module doet. Bijvoorbeeld:

```
X = Kwadratisch(2; 8; 8)
GraphWeergeven 50; Matrix1
```

De instructie wordt door Visual Basic uitgevoerd en vervolgens wordt de onderbrekingsmodus weer geactiveerd. Dan kunt u de resultaten bekijken en de mogelijke effecten op waarden van variabelen en eigenschappen testen.

Opmerking Hoewel alle uitvoerbare instructies worden ondersteund in het deelvenster **Direct**, is een controlestructuur alleen geldig als deze in zijn geheel op een regel past. De volgende **Voor**-lus is geldig in het deelvenster **Direct**:

```
Voor I = 1 Tot 20 : Afbeelden 2 * I : Volgende I
```

Als de onderbrekingsmodus actief is, kunt u een andere procedure uitvoeren vanuit het deelvenster **Direct**. U kunt de onderbrekingsmodus nog een keer activeren door onderbrekingspunten te plaatsen of de instructie **Stoppen** in te voegen. Zodra de tweede procedure klaar is, omdat u deze in zijn geheel stapsgewijs hebt uitgevoerd of omdat u de uitvoering van de programmacode hebt hervat, keert u automatisch terug naar het venster **Foutopsporing** en de oorspronkelijke onderbrekingsmodus.

Effectiever werken met het deelvenster Direct

Hier volgen enkele handigheidjes die u kunt toepassen in het deelvenster **Direct**.

- Nadat u een instructie hebt ingevoerd, kunt u deze opnieuw uitvoeren door de invoegpositie terug te verplaatsen naar de betreffende instructie en op ENTER te drukken.
- Voordat u op ENTER drukt, kunt u de huidige instructie bewerken en de effecten ervan wijzigen.
- U kunt zich met de muis of de pijltoetsen verplaatsen in het deelvenster **Direct**. Druk niet op ENTER, tenzij u bij een instructie staat die u wilt uitvoeren.

Het foutopsporingsproces vereenvoudigen

Hoe nauwkeuriger u de programmacode schrijft in het begin, des te minder tijd u later aan het opsporen van fouten hoeft te besteden. Er zijn een aantal dingen die u kunt doen om te voorkomen dat er programmeerfouten in uw programmacode optreden:

- Voeg zoveel mogelijk commentaar toe. Als u teruggaat om uw programmacode te analyseren, zult u deze veel beter begrijpen als iedere procedure van een commentaar is voorzien.
- Maak een consistent overzicht van de benamingen van de variabelen in uw programmacode. De meeste fouten in programmacode treden op doordat namen van variabelen onjuist worden getypt of doordat variabelen met elkaar worden verward. U zou bijvoorbeeld het voorvoegsel `gv` kunnen gebruiken om globale of openbare variabelen aan te geven. Volgens het overzicht weet u dan dat een variabele die `gvBeginwaarde` heet, een globale variabele is.

  Een andere manier om verwarring over de namen van variabelen te vermijden, is gebruik te maken van de instructie **Optie Expliciet**. Zie "Variabelen en argumenten vooraf definiëren" in hoofdstuk 6 voor meer informatie over de instructie **Optie Expliciet**.

Tips bij het opsporen van fouten

Als u ontdekt dat er fouten in uw programmacode zitten, kunt u een aantal dingen doen om het opsporen van de fouten te vereenvoudigen.

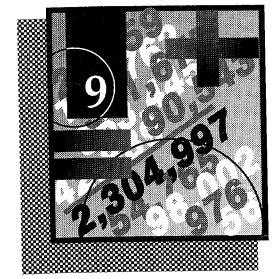
- Als de programmacode niet de juiste resultaten oplevert, bladert u door de programmacode en probeert u de instructie te vinden die het probleem kan hebben veroorzaakt. Zet onderbrekingspunten bij deze instructies en start de programmacode opnieuw. Zie "Onderbrekingsmodus activeren bij een probleempunt" eerder in dit hoofdstuk voor meer informatie over controle-expressies.

- Op het punt waar het programma stopt, test u de waarden van belangrijke variabelen en eigenschappen. Stel controle-expressies in om deze waarden te controleren. Gebruik het venster **Foutopsporing** om variabelen en expressies te onderzoeken. Zie "Gegevens observeren met controle-expressies" eerder in dit hoofdstuk voor meer informatie over controle-expressies.
- Voer de programmacode stapsgewijs uit, waarbij u met controle-expressies bekijkt hoe de waarden veranderen tijdens de uitvoering van de programmacode.
- Wanneer u ontdekt dat een variabele of eigenschap problemen veroorzaakt in de programmacode, definieert u een controle-expressie die de uitvoering stopt zodra de verkeerde waarde wordt toegewezen aan de desbetreffende variabele of eigenschap.

Hiermee eindigt de inleiding in de hulpmiddelen voor foutopsporing van Visual Basic. In dit hoofdstuk zijn de mogelijkheden behandeld die Visual Basic biedt om fouten in procedures op te sporen en de effecten te bekijken die de programmacode heeft op de waarden van de gebruikte variabelen en eigenschappen.

HOOFDSTUK 9

Foutafhandeling en foutwaarden



In dit hoofdstuk wordt uitgelegd hoe u "*run-time*"-*fouten* kunt afhandelen. Dit zijn fouten die optreden terwijl programmacode wordt uitgevoerd en die voortvloeien uit pogingen een ongeldige bewerking te voltooien. Tevens worden *foutwaarden* behandeld. Dit zijn waarden die door procedures worden gegeven wanneer u ze oproept. Zie hoofdstuk 8, "Fouten in de programmacode opsporen en programmacode testen", voor meer informatie over het opsporen van fouten in programmacode.

Het zou ideaal zijn als Visual Basic-procedures geen foutafhandelingscode nodig hadden. Helaas kunnen bestanden per ongeluk worden gewist, kunt u proberen toegang te krijgen tot een niet-bestaande eigenschap of de waarde in te stellen van een alleen-lezen-eigenschap, of kan het voorkomen dat een procedure onverwachte waarden moet verwerken. Om dergelijke fouten te kunnen afhandelen moet u foutafhandelingscode opnemen in procedures.

Inhoud

Wat gebeurt er als u fouten niet onderschept?

Wanneer er fouten optreden in programmacode en u deze niet onderschept, genereert Visual Basic vaak een "run-time"-fout die de uitvoering van de programmacode stopt. Gewoonlijk kan de gebruiker niet zelf de toepassing laten doorgaan. Andere fouten onderbreken de programmacode misschien niet, maar veroorzaken wel onvoorspelbare effecten.

De volgende **Functie**-procedure resulteert bijvoorbeeld in **Waar** als het opgegeven bestand bestaat en **Onwaar** als het niet bestaat, maar bevat geen foutafhandelingscode.

```
Functie BestandBestaat(bestandsnaam)
    BestandBestaat = (Dir(bestandsnaam) <> "")
Einde Functie
```

De functie **Dir** geeft het eerste bestand op uw schijf dat overeenkomt met de opgegeven bestandsnaam (met of zonder jokertekens, stationsaanduiding of pad). De functie geeft een reeks met een lengte van nul als er geen overeenkomende reeks wordt gevonden. Daarom geeft `Dir(bestandsnaam) <> ""` **Waar** als **Dir** een resultaat heeft en **Onwaar** als **Dir** in een reeks met een lengte van nul resulteert. De programmacode lijkt te voorzien in alle mogelijke uitkomsten van **Dir**.

Maar als de stationsaanduiding (een letter bij Microsoft Windows, een naam bij de Macintosh) die in het argument is opgegeven niet een geldig station is, verschijnt het foutbericht `Apparaat niet beschikbaar`. In dit bericht wordt niet vermeld of het opgegeven bestand bestaat of wat u nu moet doen.

Als u foutafhandelingscode opneemt in procedures wordt het vaak mogelijk een fout te herstellen en de gebruiker in de gelegenheid te stellen de taak te voltooien of de oorzaak van de fout weg te nemen.

Voorkomen dat programmacode stopt of onvoorspelbaar reageert

Met de foutafhandelingsfuncties in Visual Basic kunt u "run-time"-fouten zoals hiervoor beschreven *onderscheppen* en herstelacties ondernemen. Met de volgende instructies en functies kunt u "run-time"-fouten onderscheppen en afhandelen:

- De instructie **Bij Fout GaanNaar** maakt een foutafhandelingsroutine beschikbaar en geeft de plaats van de routine in een procedure op. Deze instructie kan ook worden gebruikt om een foutafhandelingsroutine uit te schakelen.

  ```
  Bij Fout GaanNaar FoutControleren
  ```

- De functie **FoutNr** geeft het nummer van de laatst opgetreden "run-time"-fout. Deze functie werkt als een **Openbaar**-variabele die een geheel getal bevat.

  ```
  BerichtVenster "De laatst opgetreden fout is " & FoutNr & ". Het
  bijbehorende bericht is: " & Fout(FoutNr)
  ```

  Zie "Meer leren over programmeren met Visual Basic" in hoofdstuk 6 voor meer informatie over **Openbaar**-variabelen.

- De functie **Fout** geeft het bericht dat bij een foutnummer hoort. Iedere "run-time"-fout heeft ter identificatie een foutnummer.

  ```
  TekstFoutbericht = Fout(68)
  ```

Andere foutafhandelingsinstructies en -functies (bijvoorbeeld de instructies **FoutNr** en **Fout**, die niet hetzelfde zijn als de hier beschreven functies **FoutNr** en **Fout**) worden beschreven onder "Geavanceerde foutafhandelingstechnieken" verderop in dit hoofdstuk. Zie ook *Foutafhandeling* in de online *Visual Basic Naslaggids.*

Het volgende voorbeeld demonstreert hoe schijfproblemen, bijvoorbeeld een ongeldig station, kunnen worden verwerkt met foutafhandelingscode.

```
Functie BestandBestaat (bestandsnaam)
'Foutonderschepping inschakelen, zodat de foutafhandeling
'reageert als er een fout optreedt.
Bij Fout GaanNaar FoutControleren
    BestandBestaat = (Dir(bestandsnaam) <> "")
    Verlaten Functie          ' Voorkomen dat foutafhandeling start
                              ' als er geen fout is.

FoutControleren:                  ' Hiernaartoe springen bij fout.
    ' Constanten definiëren die Visual Basic-foutcode vertegenwoordigen.
    Const FOUT_APPNIETBESCHIKBAAR = 68
    BestandBestaat = Onwaar
Indien FoutNr = FOUT_APPNIETBESCHIKBAAR Dan
'Geef berichtvenster weer met een uitroepteken als pictogram
        Bericht = "Dit station of pad bestaat niet: " & bestandsnaam
        BerichtVenster Bericht; vbWaarschuwing
        Hervatten Volgende
    Anders
        Bericht = "Onverwachte fout nr " & TReeks(FoutNr) & ": " _
            & Fout(FoutNr)
        ' Berichtvenster weergeven met pictogram Stop en knop OK.
        BerichtVenster Bericht; vbNoodsignaal
        Einde
    Einde Indien
Einde Functie
```

Bij de instructie **Bij Fout GaanNaar** moet u een regelnummer of -label opgeven om aan te geven waar de foutafhandelingscode zich bevindt, zoals u in het voorbeeld kunt zien. Een regellabel moet voldoen aan de standaard naamgevingsvoorschriften en met een dubbele punt eindigen. (`FoutControleren:` is de regellabel in het voorbeeld. De dubbele punt wordt echter niet gebruikt bij de instructie **Bij Fout GaanNaar**). Een regellabel moet binnen de module uniek zijn, maakt geen onderscheid tussen hoofdletters en kleine letters en kan overal op een regel beginnen zolang de label uitsluitend door spaties wordt voorafgegaan. (Voor een regelnummer geldt hetzelfde).

Opmerking In het vorige voorbeeld worden de foutnummers die de functie **FoutNr** geeft, vergeleken met constanten die u moet definiëren. Zie *Fouten die kunnen worden onderschept* in de online *Visual Basic Naslaggids* voor een lijst met deze foutnummers.

De constanten die met "vb" beginnen, hebben betrekking op standaarddialoogvensters van Visual Basic en worden weergegeven door de functie **BerichtVenster**. Zie *BerichtVenster* in de online *Visual Basic Naslaggids* voor meer informatie over de functie **BerichtVenster**.

Als de fout `Apparaat niet beschikbaar` optreedt, wordt er door de programmacode een bericht weergegeven dat het probleem beschrijft. Door de instructie **Hervatten Volgende** wordt vervolgens de uitvoering van het programma hervat bij de instructie die volgt op de instructie waar de fout is opgetreden.

Als er een onverwachte fout optreedt, wordt er een ander bericht afgebeeld en stopt de programmacode bij de instructie **Einde**.

Opmerking Het is raadzaam de instructie **Einde**, net als in het voorbeeld, te gebruiken in uw eigen toepassingen, omdat met deze instructie het programma op ordelijke wijze wordt gestopt: bestanden worden gesloten, variabelen worden gewist en Microsoft Excel gaat weer verder. Vermijd de instructie **Stoppen** omdat dan het programma abrupt wordt gestopt. Dit kan de gebruiker verwarren.

Fouten onderscheppen

In het vorige voorbeeld komt de foutafhandelingsroutine FoutControleren voor. Het voorbeeld illustreert drie stappen die u op de meeste foutafhandelingsroutines kunt toepassen:

- Foutonderschepping activeren
- Een foutafhandelingsroutine schrijven
- De foutafhandelingsroutine verlaten

Foutonderschepping activeren

De foutonderschepping wordt geactiveerd wanneer Visual Basic de instructie **Bij Fout** tegenkomt. Deze instructie geeft op naar welke foutafhandelingsroutine moet worden gesprongen binnen de procedure. De foutonderschepping blijft actief zolang de desbetreffende procedure actief is, dat wil zeggen totdat de instructie **Verlaten Sub**, **Verlaten Functie**, **Einde Sub** of **Einde Functie** in die procedure wordt uitgevoerd. (De instructies **Verlaten Eigenschap** en **Einde Eigenschap** zijn beschikbaar voor eigenschapsprocedures.)

Binnen een bepaalde procedure kan er maar één foutonderscheppingsinstructie tegelijk actief zijn. U kunt wel verschillende foutonderscheppingsinstructies opnemen die op verschillende momenten worden geactiveerd. U kunt foutonderschepping uitschakelen met behulp van de instructie **Bij Fout GaanNaar 0**. Zie "Foutafhandeling uitschakelen" verderop in dit hoofdstuk voor meer informatie over het uitschakelen van foutafhandeling.

Een foutafhandelingsroutine schrijven

De eerste stap in het schrijven van een foutafhandelingsroutine is het plaatsen van de instructie **Bij Fout GaanNaar** *regel*, waarin *regel* tekst is die de regellabel van de foutafhandelingscode aangeeft.

In de functie BestandBestaat, eerder in dit hoofdstuk, is de label `FoutControleren`. Het is gebruikelijk de foutafhandelingscode aan het einde van de procedure te plaatsen, voor de instructie **Einde Functie** of **Einde Sub**, met een instructie **Verlaten Sub** of **Verlaten Functie** onmiddellijk voor de regellabel. Dit voorkomt dat de foutafhandelingscode wordt gestart als er geen fout optreedt.

Gebruik de functie **FoutNr**, die het nummer van de laatst opgetreden fout geeft (inclusief van door u zelf gedefinieerde fouten), in combinatie met de instructie **Kiezen Ingeval** of **Indien...Dan...Anders** als u specifieke foutafhandelingscode wilt laten uitvoeren. Dit wordt in het voorbeeld in het volgende gedeelte gedemonstreerd.

Zie "Gedeelten van de programmacode uitvoeren op basis van voorwaarden" in hoofdstuk 7 voor meer informatie over de instructies **Kiezen Ingeval** en **Indien...Dan...Anders**.

Een foutafhandelingsroutine verlaten

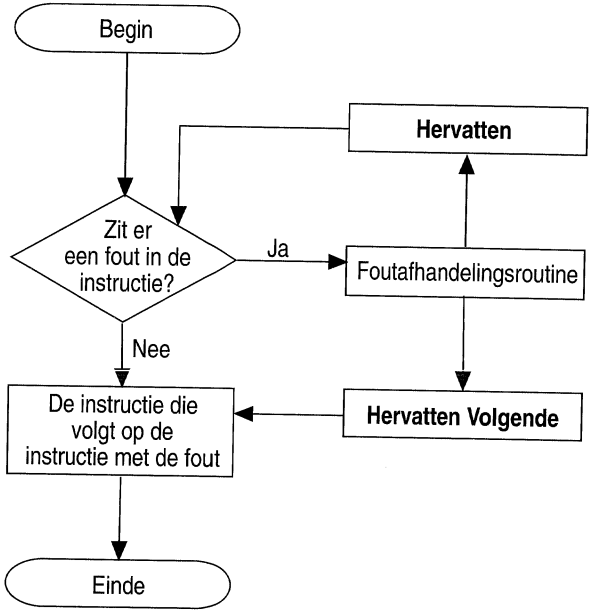
De functie BestandBestaat die eerder in dit hoofdstuk aan de orde kwam, gebruikt de instructie **Hervatten Volgende** om de instructie te laten uitvoeren die volgt op de instructie waarbij de fout is opgetreden.

U kunt deze en andere instructies gebruiken om een foutafhandelingsroutine te verlaten. De keuze hangt af van de omstandigheden, zoals wordt uitgelegd in de volgende tabel.

Instructie	Beschrijving
Hervatten Volgende	De procedure wordt voortgezet bij de instructie die onmiddellijk volgt op de instructie die de fout heeft veroorzaakt.
Hervatten	De procedure wordt voortgezet bij de instructie die de fout heeft veroorzaakt. Gebruik **Hervatten** om een bewerking te herhalen nadat de fout is hersteld.
Hervatten *regel*	De procedure wordt voortgezet bij de label opgegeven met *regel*, waarin *regel* een regelnummer (niet nul) of regellabel is in dezelfde procedure als de foutafhandelingsroutine.
Fout FoutNr	De laatst opgetreden "run-time"-fout wordt opnieuw veroorzaakt. Als deze instructie in de foutafhandelingsroutine wordt uitgevoerd, zoekt Visual Basic in de oproeplijst naar een andere foutafhandelingsroutine.

In de vorige tabel wordt aangenomen dat de fout in dezelfde procedure als de foutafhandelingsroutine is opgetreden. Is dit niet het geval, dan zoekt Visual Basic in de oproeplijst naar een andere foutafhandelingsroutine. (De oproeplijst is de keten van procedures waarnaar de huidige procedure bij voltooiing terugkeert. De oproeplijst wordt weergegeven in het dialoogvenster **Opgeroepen procedures**, dat verschijnt als u de knop kiest die zich rechts van het vak met de procedurenaam in het venster Foutopsporing bevindt. Zie hoofdstuk 8, "Fouten in de programmacode opsporen en programmacode testen", voor meer informatie).

Het verschil tussen **Hervatten** en **Hervatten Volgende** wordt in het volgende schema geïllustreerd.



Gebruik in het algemeen **Hervatten** wanneer de gebruiker een correctie moet aanbrengen en **Hervatten Volgende** wanneer de programmacode moet worden gecorrigeerd of wanneer u een fout in een lus voorziet en u de bewerking opnieuw moet starten. U kunt een foutafhandelingsroutine echter zodanig schrijven dat een bepaalde "run-time"-fout nooit zichtbaar is voor de gebruiker.

De volgende **Functie**-procedure gebruikt bijvoorbeeld foutafhandelingscode om een "veilige" deling op de argumenten te kunnen uitvoeren zonder eventuele fouten te laten zien. Als ze optreden, is het resultaat **Nulwaarde**.

```
Functie Delen(teller; noemer)
Const FOUT_DEL0 = 11; FOUT_OVERLOOP = 6; FOUT_ONGFUNC = 5
Bij Fout GaanNaar RekenAfhand
    Delen = teller / noemer
    Verlaten Functie
RekenAfhand:
    Indien FoutNr = FOUT_DEL0 Of FoutNr = FOUT_OVERLOOP Of FoutNr = _
        FOUT_ONGFUNC Dan
        ' Nulwaarde geven als de fout een deling door nul,
        ' een overloop of een ongeldige functie-oproep is.
        Delen = Nulwaarde
    Anders
        BerichtVenster "Onvoorziene fout " & FoutNr & ": " & Fout; _
            vbWaarschuwing
        Delen = Nulwaarde
    Einde Indien
    ' In alle gevallen gaat Hervatten Volgende door
    ' bij de volgende instructie Verlaten Functie.
    Hervatten Volgende
Einde Functie
```

Hoe zoekt Visual Basic naar een foutafhandelingsroutine?

Als er een fout optreedt in een procedure waaraan een geactiveerde foutafhandelingsroutine ontbreekt of in een foutafhandelingsroutine die momenteel wordt uitgevoerd, zoekt Visual Basic in de oproeplijst naar een andere geactiveerde foutafhandelingsroutine.

Veronderstel de volgende reeks oproepen:

1. Procedure A roept Procedure B op.
2. Procedure B roept Procedure C op.
3. Procedure C roept Procedure D op.

Als er een fout optreedt in Procedure D en deze procedure geen geactiveerde foutafhandelingsroutine heeft, wordt er achterwaarts gezocht in de nog actieve procedures in de oproeplijst, eerst Procedure C, dan Procedure B en dan Procedure A (niet verder) en wordt de eerst aangetroffen actieve foutafhandelingsroutine uitgevoerd. Als er geen geactiveerde foutafhandelingsroutine wordt aangetroffen in de oproeplijst, wordt het desbetreffende "run-time"-foutbericht weergegeven en de procedure gestopt.

Tip Het resultaat van de achterwaartse zoekbewerking door de oproeplijst is moeilijk te voorspellen, aangezien in geval van een instructie **Hervatten** of **Hervatten Volgende** in de foutafhandelingsroutine, de uitvoering verdergaat in de procedure waarin de foutafhandelingsroutine is gevonden, en niet noodzakelijk in de procedure waarin de fout is opgetreden. Schrijf om dit te vermijden een veiligheidsroutine die door alle andere foutafhandelingsroutines kan worden opgeroepen voor fouten die ze niet aankunnen. Deze veiligheidsroutine kan de gegevens van de gebruiker opslaan en de toepassing op ordelijke wijze beëindigen.

Foutafhandeling uitschakelen

Als in een procedure foutonderschepping is geactiveerd, wordt deze automatisch uitgeschakeld wanneer de procedure eindigt. Het is echter mogelijk dat u de foutonderschepping eerder wilt uitschakelen. Gebruik hiervoor de instructie **Bij Fout GaanNaar 0**, waarmee fouten wel worden opgespoord, maar niet binnen de procedure onderschept. U kunt de instructie **Bij Fout GaanNaar 0** overal in een procedure gebruiken om foutafhandeling uit te schakelen, zelfs binnen de foutafhandelingsroutine.

Ga bijvoorbeeld eens stapgsgewijs door de volgende procedure:

```
Sub FoutDemoSub()
Bij Fout GaanNaar SubAfhand
' Foutonderschepping activeren.
' Fouten worden hier onderschept en afgehandeld.
    BVerwijderen "OUDBEST.XYZ"
Bij Fout GaanNaar 0              ' Foutonderschepping wordt hier
uitgeschakeld.
    BVerwijderen "OUDBEST.XYZ"
Bij Fout GaanNaar SubAfhand      ' Foutonderschepping wordt weer
geactiveerd.
    BVerwijderen "OUDBEST.XYZ"
Verlaten Sub
```

```
SubAfhand:                    ' Foutafhandelingsroutine begint hier.
    BerichtVenster "Fout onderschept."
    Hervatten Volgende
Einde Sub
```

Met de volgende instructie kunt u voorkomen dat Microsoft Excel waarschuwingen weergeeft.

```
Toepassing.WaarschuwingenWeergeven = Onwaar
```

Zie *WaarschuwingenWeergeven* in de online *Visual Basic Naslaggids* voor meer informatie over waarschuwingsberichten.

Foutwaarden maken die de programmacode niet onderbreken

In het voorgaande gedeelte is behandeld hoe u "run-time"-fouten kunt afhandelen. Maar er bestaat nog een geheel ander type fout, namelijk *door de gebruiker gedefinieerde foutwaarden*. Dit zijn waarden die u voor procedures definieert en die altijd worden opgeslagen in variabelen van het gegevenstype **Variant**.

Deze foutwaarde wordt gewoonlijk gebruikt in procedures die verschillende argumenten accepteren en die als resultaat een waarde hebben. Als de resulterende waarde bijvoorbeeld alleen geldig is wanneer de argumenten binnen een bepaald waardenbereik vallen, kan de procedure de door de gebruiker gegeven argumenten testen. Bevinden deze zich niet binnen het voorgeschreven bereik, dan kunt u de procedure een passende foutwaarde laten geven.

Wat is een foutwaarde?

Variant-variabelen kunnen veel gegevenstypen bevatten: alle intrinsieke gegevenstypen plus **Fout**, **Leeg** en **Nulwaarde**. **Fout** is een subtype van het type **Variant**, en wanneer de term "**Fout**-waarde" wordt gebruikt, duidt deze gewoonlijk een variabele van het gegevenstype **Variant** aan die een waarde bevat die door Visual Basic wordt herkend als een door de gebruiker gedefinieerde fout. Zie "Andere soorten informatie die in Variant-variabelen kunnen worden opgeslagen" in hoofdstuk 6 voor meer informatie over de waarden **Leeg** en **Nulwaarde** en het subtype **Fout**.

Zulke **Fout**-waarden gebruikt u in een procedure om aan te geven dat er een fout is opgetreden, maar in tegenstelling tot normale "run-time"-fouten onderbreken deze fouten de programmacode niet omdat Visual Basic ze als gewone variabelen opvat. De procedure kan de **Fout**-waarden testen en gepaste actie ondernemen.

Hoe maakt u foutwaarden?

Fout-waarden worden gemaakt met de functie **CVarFout**, die als argument een geheel getal accepteert of een variabele die een geheel getal bevat.

```
GeenStraal = CVarFout(2010)

GeenGetal = 2020
OngeldArgument = CVarFout(GeenGetal)
```

In het volgende voorbeeld wordt de functie **CVarFout** gebruikt om een **Fout**-waarde te leveren. De procedure OppCirkel vraagt om een waarde van de straal. De **Functie**-procedure ContrData controleert de variabele Straal, en als deze geen acceptabel getal is, is het resultaat een van de twee mogelijke **Fout**-waarden.

```
Openbaar GeenStraal; GeenGetal

Sub OppCirkel()
    Const Pi = 3,14
    GeenStraal = CVarFout(2010)
    GeenGetal = CVarFout(2020)
    Straal = ContrData(Invoervak("Geef de straal op."))
    Indien IsFout(Straal) Dan
        Kiezen Ingeval Straal
            Ingeval GeenStraal
                BerichtVenster "Fout: Geen straal opgegeven"
            Ingeval GeenGetal
                BerichtVenster "Fout: Straal is geen getal."
            Ingeval Anders
                BerichtVenster "Onbekende fout."
        Einde Kiezen
    Anders      ' Er is geen fout
        BerichtVenster "Het oppervlak van de cirkel is " & _
            (Pi * Straal ^ 2)
    Einde Indien
Einde Sub

Functie ContrData(DeStraal)
    Indien Niet IsNumeriek(DeStraal) Dan
        ContrData = GeenGetal
    AndersIndien DeStraal = 0 Dan
        ContrData = GeenStraal
    Anders
        ContrData = DeStraal
```

```
    Einde Indien
Einde Functie
```

In de **Sub**-procedure OppCirkel resulteert de functie **IsFout** in **Waar** als het argument een **Fout**-waarde is. Denk erom dat er een "run-time"-fout optreedt als u een foutwaarde gebruikt in een expressie waarin Visual Basic een foutwaarde probeert te vergelijken met een getal of een andere waarde.

```
Indien GeenStraal = 2010 Dan...   'Veroorzaakt een fout._
Foutwaarde   kan niet worden vergeleken met een getal of_
een andere  waarde.
```

De volgende programmacode veroorzaakt geen fout omdat u foutwaarden kunt vergelijken met andere foutwaarden.

```
Indien GeenStraal = CVarFout(2010) Dan...
```

Zie *IsFout* of *CVarFout* in de online *Visual Basic Naslaggids* voor meer informatie over **IsFout** of **CVarFout**.

De foutwaarden van Microsoft Excel gebruiken

Namen en celformules in Microsoft Excel-werkbladen kunnen een van de volgende foutwaarden bevatten: #DEEL/0!, #N/B, #NAAM?, #LEEG!, #GETAL!, #VERW! en #WAARDE!. U kunt deze foutwaarden alleen vanuit werkbladen doorgeven naar procedures met argumenten van het type **Variant**. Als u met deze foutwaarden wilt werken, kunt u de argumenten testen met de functie **IsFout** en ermee werken alsof het door de gebruiker gedefinieerde foutwaarden betrof, die zijn beschreven in het voorgaande gedeelte.

U kunt foutwaarden vanuit een door de gebruiker gedefinieerde functie ook doorgeven aan cellen in werkbladen, zoals in het volgende voorbeeld wordt gedemonstreerd.

```
Functie Provisie(AandelenVerkocht; PrijsPerAandeel)
    Indien Niet(IsNumeriek(AandelenVerkocht) En _
        IsNumeriek(PrijsPerAandeel)) Dan
        Provisie = CVarFout(xlTypefoutGetal)     ' xlTypefoutGetal
correspondeert
                                                 ' met de foutwaarde #GETAL!
        Verlaten Functie
    Anders
        TotaleVerkPrijs = AandelenVerkocht * PrijsPerAandeel
        Indien TotaleVerkPrijs <= 15000 Dan
            Provisie = 25 + 0,03 * AandelenVerkocht
```

```
Anders
            Provisie = 25 + 0,03 * (0,9 * AandelenVerkocht)
        Einde Indien
    Einde Indien
Einde Functie
```

Met de foutwaarden van het Microsoft Excel-werkblad werkt u net zo als met de door de gebruiker gedefinieerde foutwaarden: als getallen die zijn omgezet in **Fout**-waarden met de functie **CVarFout**. Het enige verschil is dat voor werkbladfouten de foutwaarden als ingebouwde constanten beschikbaar zijn en dat er ook tekstuele foutwaarden zijn. De tekstuele foutwaarden moeten tussen vierkante haken staan, zoals u in de volgende tabel kunt zien.

Foutnummer (constante)	Tekstuele foutwaarde	Geconverteerde foutwaarde
xlTypefoutDiv0	[#DEEL/0!]	CVarFout(xlTypefoutDiv0)
xlTypefoutNB	[#N/B]	CVarFout(xlTypefoutNB)
xlTypefoutNaam	[#NAAM?]	CVarFout(xlTypefoutNaam)
xlTypefoutLeeg	[#LEEG!]	CVarFout(xlTypefoutLeeg)
xlTypefoutGetal	[#GETAL!]	CVarFout(xlTypefoutGetal)
xlTypefoutVerw	[#VERW!]	CVarFout(xlTypefoutVerw)
xlTypefoutWaarde	[#WAARDE!]	CVarFout(xlTypefoutWaarde)

Als u in het vorige voorbeeld de tekstuele foutwaarde voor #GETAL! wilt gebruiken, vervangt u op de volgende wijze CVarFout(xlTypefoutGetal) door [#GETAL!]:

```
Indien Niet(IsNumeriek(AandelenVerkocht) _
    En IsNumeriek(PrijsPerAandeel)) Dan
    Provisie = [#GETAL!]
    Verlaten Functie
.
.
.
```

Geavanceerde foutafhandelingstechnieken

Visual Basic heeft extra functies en instructies, waarmee u meer geavanceerde foutafhandelingscode kunt schrijven. In dit gedeelte worden de instructies **Fout** en **FoutNr** beschreven. (Verwar deze instructies niet met de functies **Fout** en **FoutNr** die eerder in dit hoofdstuk worden beschreven). In volgende gedeelten worden deze instructies gedemonstreerd in programmacode.

De instructie **Fout** simuleert een fout door de programmacode te onderbreken (tenzij u een foutafhandelingsroutine hebt geschreven) met de "run-time"-fout die u hebt opgegeven met het foutnummer. De instructie **Fout** stelt ook de waarde van de functie **FoutNr** in op het nummer dat u opgeeft. Een foutnummer is een geheel getal tussen 0 en 65.535. De lage nummers zijn gereserveerd voor Visual Basic.

```
Fout 68
```

U kunt zelf foutnummers definiëren. Zie de opmerking in het volgende gedeelte voor meer informatie over het nummeren van fouten.

De instructie **FoutNr** stelt de waarde in die de functie **FoutNr** als resultaat geeft.

```
FoutNr = 68
```

Als u de instructie **FoutNr** niet gebruikt, stelt Visual Basic de waarde van de functie **FoutNr** in op de laatst opgetreden "run-time"-fout. Ook de instructie **Fout** stelt de waarde in van de functie **FoutNr**, maar in tegenstelling tot de instructie **Fout** simuleert de instructie **FoutNr** geen fout door de programmacode te onderbreken.

"Run-time"-fouten simuleren

Het simuleren van "run-time"-fouten is nuttig wanneer u toepassingen test of wanneer u een bepaalde conditie wilt behandelen als "run-time"-fout. U bent bijvoorbeeld een module aan het schrijven die routines oproept in een dynamic-link library (DLL) en u wilt dat de door de DLL gegeven waarden door de rest van de toepassing worden behandeld als Visual Basic-fouten, dus dat ze een "run-time"-fout genereren in Microsoft Excel. (Zie hoofdstuk 10, " Samenwerken met andere toepassingen", voor meer informatie over DLL's).

U kunt iedere Visual Basic "run-time"-fout simuleren door de foutcode van de desbetreffende fout in de instructie **Fout** op te geven.

```
Fout 68      ' Fout "Apparaat niet beschikbaar" simuleren.
```

Met de instructie **Fout** kunt u door u zelf gedefinieerde fouten genereren door een foutcode op te geven die niet overeenkomt met een Visual Basic "run-time"-fout. Dit heeft natuurlijk alleen zin als u programmacode schrijft die de nieuwe fouten afhandelt die u definieert. Zie *Fouten die kunnen worden onderschept* in de online *Visual Basic Naslaggids* voor een lijst met standaardfouten.

Opmerking Microsoft Visual Basic gebruikt niet alle beschikbare nummers voor de eigen fouten. Wanneer in toekomstige versies van Visual Basic nieuwe fouten worden gedefinieerd, zal het aantal interne foutnummers stijgen. Als u zelf fouten wilt genereren en onderscheppen, kunt u het beste bij nummer 50.000 beginnen en naar boven nummeren.

De codelengte beperken door fouten centraal af te handelen

Wanneer u foutafhandelingscode aan uw eigen toepassingen toevoegt, zult u er snel achter komen dat u steeds weer dezelfde fouten afhandelt. U kunt zich veel moeite besparen als u een paar procedures schrijft die uw foutafhandelingscode kunnen oproepen in geval van veel voorkomende fouten.

De volgende **Functie**-procedure, BestandsFouten, geeft een bericht weer waarvan de inhoud afhangt van de opgetreden fout en stelt de gebruiker zo mogelijk in staat een knop te kiezen die aangeeft wat de functie moet doen. Vervolgens zendt de functie een codegetal aan de oproepende procedure (dit wordt niet gedemonstreerd). Dit codegetal geeft aan welke actie uit een groep van acties het programma moet uitvoeren.

```
Openbaar Const HERVATTEN_INSTRUCTIE = 0              'Hervatten
Openbaar Const HERVATTEN_VOLGENDE = 1                'Hervatten Volgende
Openbaar Const ONHERSTELBAAR = 2                     'Onherstelbare fout
Openbaar Const ONBEKEND = 3                          'Onbekende fout

Openbaar Const FOUT_APPNIETBESCHIKBAAR = 68
Openbaar Const FOUT_FOUTINNAAMOFNRBESTAND = 52: FOUT_PADBESTAATNIET = 76
Openbaar Const FOUT_FOUTINBESTANDSMODUS = 54
```

```
Functie BestandsFouten(foutWrde Als Integer) Als Integer
Dim BerichtType Als Integer; Bericht Als Reeks; Antwoord Als Integer

    BerichtType = vbWaarschuwing
    Kiezen Ingeval foutWrde
        Ingeval FOUT_APPNIETBESCHIKBAAR                    'Fout 68
            Bericht = "Apparaat is niet beschikbaar."
            BerichtType = BerichtType + vbAfbrekenNogmaalsNegeren
        Ingeval FOUT_FOUTINNAAMOFNRBESTAND                 'Fouten 64 & 52
            Bericht = "Die bestandsnaam is niet geldig."
            BerichtType = BerichtType + vbOKAnnuleren
        Ingeval FOUT_PADBESTAATNIET                        'Fout 76
            Bericht = "Dat pad bestaat niet."
            BerichtType = BerichtType + vbOKAnnuleren
        Ingeval FOUT_FOUTINBESTANDSMODUS                   'Fout 54
            Bericht = "Bestand kan niet worden geopend voor dat type _
                toegang."
            BerichtType = BerichtType + vbOKAnnuleren
        Ingeval Anders
            BestandsFouten = ONBEKEND
            Verlaten Functie
    Einde Kiezen

    Antwoord = BerichtVenster(Bericht; BerichtType; "Schijffout")
    Kiezen Ingeval Antwoord
        Ingeval vbOK; vbNogmaals
            BestandsFouten = HERVATTEN_INSTRUCTIE
        Ingeval vbNegeren
            BestandsFouten = HERVATTEN_VOLGENDE
        Ingeval vbAnnuleren; vbAfbreken
            BestandsFouten = ONHERSTELBAAR
        Ingeval Anders
            BestandsFouten = ONBEKEND
    Einde Kiezen
Einde Functie
```

De voorgaande **Functie**-procedure handelt een kleine groep fouten af die met bestanden en schijven (diskettes) te maken hebben. Als de fout geen deel uitmaakt van de groep, is het resultaat de waarde **3**. De procedure die deze functie oproept moet vervolgens de fout zelf afhandelen of een andere procedure oproepen die de fout afhandelt, of simpelweg de oorspronkelijke "run-time"-fout opnieuw laten optreden, zoals in het volgende voorbeeld wordt gedemonstreerd.

```
Functie BestandsVerwerkingenDemo(Bestandsnaam Als Tekenreeks)
'Demonstratie van hoe u de functie BestandsFouten zou oproepen
Bij Fout GaanNaar BevestigBestandsFout
.
.
.
'Programmacode die een bestandsbewerking probeert uit te voeren
.
.
.
BevestigBestandsFout:
Actie = BestandsFouten(FoutNr)                    'De functie BestandsFouten
                                                  'oproepen
    Kiezen Ingeval Actie
        Ingeval HERVATTEN_INSTRUCTIE              'Gebruiker heeft OK of
                                                  'Opnieuw gekozen toen
                                                  'BestandsFouten bericht
                                                  'weergaf, dus opnieuw
                                                  'proberen.
            Hervatten
        Ingeval HERVATTEN_VOLGENDE                'Gebruiker heeft Negeren
                                                  'gekozen toen
                                                  'BestandsFouten bericht
                                                  'weergaf, dus fout negeren
                                                  'en verdergaan.
            Hervatten Volgende
        Ingeval ONHERSTELBAAR                     'Gebruiker heeft Annuleren
                                                  'of Afbreken gekozen toen
                                                  'BestandsFouten bericht
                                                  'weergaf, dus procedure
                                                  'afbreken.
            Verlaten Functie
        Ingeval Anders                            'Oorspronkelijke "run-time"
                                                  '-fout kan niet worden
                                                  'afgehandeld, dus herhalen
                                                  'en de normale systeemfout-
                                                  'melding weergeven.
            Fout FoutNr
    Einde Kiezen
Einde Functie
```

Fouten afhandelen binnen de procedure

In voorgaande gedeelten van dit hoofdstuk is de instructie **Bij Fout GaanNaar** gebruikt om naar een foutafhandelingsroutine te springen, waarna de rest van de programmacode later werd uitgevoerd. U kunt ook fouten afhandelen zonder naar een subroutine te springen.

Als u voorziet dat een bepaalde instructie een fout kan veroorzaken, kunt u de instructie **Bij Fout Hervatten Volgende** invoegen voor de desbetreffende instructie om te voorkomen dat de toepassing wordt onderbroken. U kunt vervolgens de waarde van de functie **FoutNr** testen en passende maatregelen nemen.

De volgende procedure is een gewijzigde versie van de procedure BestandBestaat, die eerder in dit hoofdstuk aan de orde is geweest. Deze gewijzigde versie springt niet naar een subroutine maar handelt fouten binnen het programma af.

```
Functie BestandBestaat(bestandsnaam)
Dim Bericht

Const FOUT_APPNIETBESCHIKBAAR = 68

' FoutNr instellen op nul. Fouten binnen programma afhandelen
Bij Fout Hervatten Volgende

BestandControleren:
    BestandBestaat = (Dir(bestandsnaam) <> "")
    Indien FoutNr = FOUT_APPNIETBESCHIKBAAR Dan
        Bericht = "Dit station of pad bestaat niet: " & bestandsnaam
        BerichtVenster Bericht; vbWaarschuwing
        BestandBestaat = Onwaar
    'Afhandelen als mislukken niet volgt uit een van voorgaande fouten
    AndersIndien FoutNr <> 0 Dan
        Bericht = "Onverwachte fout " & TReeks(FoutNr) & ": " _
            & Fout(FoutNr)
        BerichtVenster Bericht; vbNoodsignaal
        BestandBestaat = Onwaar
    Einde Indien
Einde Functie
```

Gebruikers-interrupts afhandelen

Wanneer een Visual Basic-procedure wordt uitgevoerd, kan een gebruiker de procedure onderbreken door op CTRL+BREAK of ESC te drukken (of COMMAND +PUNT op de Macintosh). Hoewel interrupts van pas kunnen komen bij het opsporen van fouten, wilt u ze misschien uitschakelen voor procedures in voltooide toepassingen. Als u toch gebruikers-interrupts in een voltooide toepassing toelaat, kunt u ervoor zorgen dat de procedure van tevoren wordt gewaarschuwd, zodat bestanden kunnen worden gesloten, gedeelde resources kunnen worden losgekoppeld en gewijzigde variabelen in de oorspronkelijke waarde kunnen worden hersteld voordat de besturing van de toepassing aan de gebruiker wordt teruggegeven.

U kunt interrupts in procedures onderscheppen door de eigenschap **KnopAnnulerenActiveren** in te stellen op xlFoutafhandeling. Als u dat hebt gedaan, genereren alle interrupts "run-time"-foutnummer 18, een fout die u kunt onderscheppen met de instructie **Bij Fout**. U kunt deze fout afhandelen om de procedure te onderbreken en het programma te verlaten. Als u echter de instructie **Hervatten** gebruikt om door te gaan na een onderschepte "run-time"-fout, wordt de interrupt genegeerd.

Het volgende voorbeeld betreft een procedure die veel tijd nodig heeft. Als een gebruiker de procedure onderbreekt, wordt er een fout onderschept: eerst om te bevestigen dat de procedure moet worden gestopt en vervolgens om de procedure op ordelijke wijze af te sluiten. (U kunt slechts één interrupt-afhandelingsroutine per procedure gebruiken. Dezelfde procedure wordt gebruikt voor alle "run-time"-fouten die de procedure tegenkomt).

```
Sub GegevensVerwerken
    ' Onderscheppen van gebruikers-interrupt instellen
    ' als "run-time"-fout
    Bij Fout GaanNaar GebruikersInterrupt
    Toepassing.KnopAnnulerenActiveren = xlFoutafhandeling
    ' Lang durende taak starten
    DataBestandOpenen
    Voor x = 1 Tot 1000000
        VerwerkRecord x
        SchrijfRecord x
    Volgende x
    DataBestandSluiten
    Verlaten Sub
```

```
GebruikersInterrupt:
    Indien FoutNr = 18 Dan
        Indien BerichtVenster ("Records verwerken stoppen?"; vbJaNee) _
            = vbNee Dan
            ' Verdergaan op het punt van onderbreking
            Hervatten
        Anders
            ' Geopende bestanden sluiten voor terugkeer
            DataBestandSluiten
            Verlaten Sub
        Einde Indien
    Anders
        ' Andere optredende fouten afhandelen
        BerichtVenster Fout(FoutNr)
        Hervatten Volgende
    Einde Indien
Einde Sub
```

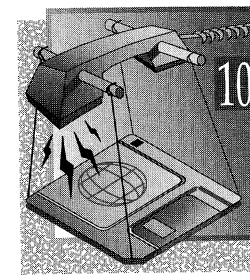
U kunt gebruikers-interrupts ook volledig negeren door de eigenschap **KnopAnnulerenActiveren** op xlUitgeschakeld in te stellen. Dit heeft als gevolg dat alle pogingen van de gebruiker om de procedure te onderbreken, worden genegeerd.

Opmerking Wees voorzichtig bij het uitschakelen of negeren van gebruikers-interrupts. Het is mogelijk programmacode te schrijven die nooit eindigt. Als u interrupts uitschakelt door de eigenschap **KnopAnnulerenActiveren** op xlUitgeschakeld in te stellen of als u altijd de instructie **Hervatten** gebruikt om terug te keren van een onderschepte fout, geeft de procedure de besturing niet terug aan de gebruiker.

U kunt de standaardmanier van Microsoft Excel om interrupts te verwerken herstellen door de eigenschap **KnopAnnulerenActiveren** op xlOnderbreken in te stellen. Om te voorkomen dat een procedure blijvend gebruikers-interrupts uitschakelt, herstelt Microsoft Excel altijd de eigenschap **KnopAnnulerenActiveren** in de waarde xlOnderbreken nadat de uitvoering van de programmacode is beëindigd. Steeds wanneer een procedure wordt uitgevoerd, moet deze uitdrukkelijk interrupts uitschakelen of onderscheppen om ervoor te zorgen dat interrupts op de juiste wijze worden afgehandeld.

HOOFDSTUK 10

Samenwerken met andere toepassingen



Terwijl u werkt met Microsoft Excel, kunt u gegevens uitwisselen met andere toepassingen. Zo kunt u bijvoorbeeld een Microsoft Word-document maken van de gegevens in een werkblad. Met Visual Basic kunt u onder andere dit eenvoudige scenario automatiseren. Verder kunt u andere toepassingen vanuit Microsoft Excel aansturen. Het is zelfs mogelijk om geavanceerde toepassingen te maken waarin Microsoft Excel, Microsoft Word, Microsoft Project en andere toepassingen zijn geïntegreerd.

Omdat er veel verschillende soorten toepassingen bestaan, zijn er ook veel manieren waarop u deze toepassingen vanuit Visual Basic kunt benaderen. In dit hoofdstuk wordt uitgelegd op welke manieren u Visual Basic kunt gebruiken om met andere toepassingen te communiceren en opdrachten uit te voeren in andere toepassingen. Eerst wordt een overzicht gegeven van de verschillende mogelijkheden en vervolgens wordt elke manier afzonderlijk behandeld.

Inhoud

Verschillende manieren om samen te werken

Visual Basic kent vier verschillende manieren om met andere toepassingen samen te werken.

- OLE (Object linking and embedding, d.w.z. koppelen en insluiten van objecten)
- Gekoppelde en ingesloten objecten
- DDE (Dynamic data exchange, d.w.z. dynamische gegevensuitwisseling)
- Verzenden van toetsaanslagen

OLE (Object linking and embedding) is een manier waarop u andere toepassingen vanuit Visual Basic kunt aansturen. Toepassingen die OLE ondersteunen, bevatten objecten die u op dezelfde manier kunt gebruiken als Microsoft Excel-objecten. U kunt gebruik maken van Visual Basic om in toepassingen die OLE ondersteunen, de volgende taken uit te voeren:

- Documenten openen, maken en opslaan.
- Gegevens invoegen, verwijderen of ophalen.
- Menu-opdrachten of macro's uitvoeren.

Koppelen en *insluiten* zijn manieren om gegevens uit andere toepassingen in Microsoft Excel weer te geven. Op deze wijze kunt u bijvoorbeeld grafische afbeeldingen, opgemaakte tekst, geluid en andere speciale soorten gegevens in een werkblad opnemen. Met Visual Basic kunt u gekoppelde of ingesloten objecten invoegen, bijwerken of wijzigen.

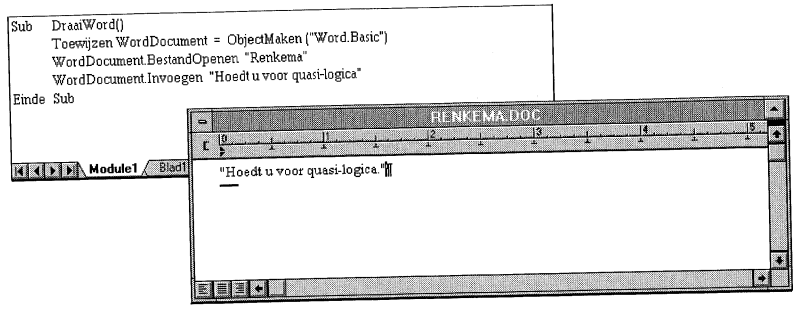
Met *DDE (Dynamic data exchange)* kunt u gegevens uitwisselen met of opdrachten verzenden naar toepassingen die OLE niet ondersteunen. Aangezien steeds meer toepassingen OLE-objecten of het gebruik van gekoppelde en ingesloten objecten ondersteunen, zal DDE in de toekomst minder worden gebruikt.

Ten slotte kan het verzenden van toetsaanslagen van pas komen als een toepassing geen andere communicatiemogelijkheid biedt.

Opmerking Toepassingen die voor een Macintosh-omgeving zijn geschreven, ondersteunen het verzenden van toetsaanslagen niet. Koppelen, insluiten of DDE wordt alleen ondersteund door Macintosh Systeem 7 of een latere versie. OLE wordt alleen ondersteund door Apple Macintosh Systeem 6.0.5 of een latere versie.

OLE gebruiken met toepassingen

U kunt OLE gebruiken om toepassingen te starten of objecten te maken. De objecten die op deze wijze zijn gemaakt, worden doorgaans niet weergegeven als deel van een werkblad. Meestal worden ze weergegeven in de toepassing waarin ze zijn gemaakt.



OLE is vooral handig als u de voorzieningen van een bepaalde toepassing, bijvoorbeeld Microsoft Word, wilt gebruiken in een andere toepassing, bijvoorbeeld Microsoft Excel. Met OLE kunt u voorzieningen van beide toepassingen in één procedure integreren.

▶ **OLE gebruiken**

1. Maak een nieuw object of haal een bestaand object op.
2. Schrijf een procedure die gebruik maakt van de methoden en eigenschappen van het object.
3. Sluit het object zodra u klaar bent.

In de volgende gedeelten worden deze stappen gedetailleerd beschreven.

Objecten maken of ophalen

Als u een nieuw object maakt of een bestaand object ophaalt, start Visual Basic de brontoepassing voor het object, voorzover dit nog niet is gebeurd. U kunt een toepassing die nog niet actief is, op deze wijze echter niet activeren.

De manier waarop u een object maakt of ophaalt, is afhankelijk van het feit of de toepassing al dan niet beschikt over een objectenbibliotheek. De meeste toepassingen, zoals bijvoorbeeld Microsoft Project, beschikken over een dergelijke bibliotheek.

Als u dit wilt	Doet u dit
Een nieuw object maken in of een bestaand object ophalen uit een toepassing die over een objectenbibliotheek beschikt.	Verwijs direct naar het object of maak met de instructie **Toewijzen** een variabele die naar het object verwijst.
Een nieuw object maken in een toepassing die niet over een objectenbibliotheek beschikt.	Maak met de instructie **ObjectMaken** een variabele die naar het object verwijst.
Een bestaand object ophalen uit een toepassing die niet over een objectenbibliotheek beschikt.	Maak met de instructie **ObjectHalen** een variabele die naar het object verwijst.

Zie "Verwijzingen naar toepassingen en werkmappen toevoegen" verderop in dit hoofdstuk voor meer informatie over de manier waarop u kunt vaststellen of een toepassing over een objectenbibliotheek beschikt.

Rechtstreeks naar objecten verwijzen

Rechtstreeks naar een object verwijzen kan in een groot aantal gevallen van pas komen, bijvoorbeeld wanneer u een bewerking op een object wilt uitvoeren. Rechtstreekse verwijzingen naar objecten in externe toepassingen hanteert u op dezelfde manier als verwijzingen naar objecten in Microsoft Excel.

In het volgende voorbeeld wordt de toepassing Microsoft Project gestart en wordt de eigenschap **WindowState** ingesteld op maximized.

```
MSProject.WindowState = pjmaximized
```

Variabelen maken voor regelmatig terugkerende objecten

Als u een bepaald object meerdere malen wilt gebruiken, kunt u de instructie **Toewijzen** gebruiken om een variabele te maken die naar dit object verwijst. Nadat u deze variabele hebt gemaakt, kunt u de kortere variabelenaam gebruiken om bewerkingen uit te voeren op het object.

Als u een variabele wilt maken die verwijst naar een nieuw object, gebruikt u de volgende syntaxis.

Toewijzen *objectvariabele* = *toepassing.objectnaam*

In het volgende voorbeeld wordt een nieuw project gestart in Microsoft Project en wordt de variabele `MijnProject` gedefinieerd, die naar dit nieuwe project verwijst.

```
Toewijzen MijnProject = MSProject.ActiveProject
MijnProject.Manager = "Ik"        ' De eigenschap Manager toewijzen.
```

Het woord `MSProject` is de naam van de toepassing zoals die is geregistreerd in Visual Basic. De eigenschap **ActiveProject** geeft het project aan waarnaar wordt verwezen.

Als u een variabele wilt maken die verwijst naar een bestaand object, gebruikt u de volgende syntaxis.

Toewijzen *objectvariabele* = *toepassing*.*objectnaam*(*index*)

In het volgende voorbeeld wordt Microsoft Project gestart en wordt de variabele `DitProject` gedefinieerd, die verwijst naar het eerste geopende project.

```
Toewijzen DitProject = MSProject.Projects(1)
```

Welke gegevens u voor het argument *index* moet invoeren, wordt bepaald door het object dat u wilt ophalen. In Microsoft Excel moet u bij het bereikobject bijvoorbeeld een cellenbereik opgeven.

```
Toewijzen MijnBereik = Bereik("A1";"C5")
```

In het vorige voorbeeld kunt u de naam van de toepassing (`Excel`) weglaten, omdat er geen andere objecten bestaan die van dezelfde naam gebruik maken. Zie "Hoe Visual Basic naar verwijzingen zoekt" verderop in dit hoofdstuk als u wilt weten wanneer u de naam van een toepassing moet gebruiken bij de instructie **Toewijzen**.

Een nieuw object in een toepassing maken

Als een toepassing niet is geregistreerd bij Visual Basic, moet u de functie **ObjectMaken** gebruiken om een nieuw object in die toepassing te maken. Hanteer daarbij de volgende syntaxis.

Toewijzen *objectvariabele* = **ObjectMaken(Klasse :=** *programma-aanduiding*)

In het volgende voorbeeld wordt Microsoft Word gestart en wordt de variabele `MSWord` geïntroduceerd die naar deze toepassing verwijst.

```
Toewijzen MSWord = ObjectMaken(Klasse := "Word.Basic")
```

Het argument `"Word.Basic"` is de *programma-aanduiding* voor Microsoft Word. Programma-aanduidingen worden in uw systeem geregistreerd wanneer u een toepassing installeert.

Een bestaand object ophalen uit een toepassing

Met de functie **ObjectHalen** kunt u een bestaand object ophalen uit een toepassing. Het object kan een bestand op schijf of een ander object van een gestarte toepassing zijn.

Gebruik de volgende syntaxis om een object uit een gestarte toepassing op te halen.

Toewijzen *objectvariabele* **= ObjectHalen(Klasse :=** *programma-aanduiding*)

Wanneer u werkt met Microsoft Word, versie 6.0 of lager, kunt u geen gebruik maken van benoemde argumenten bij **ObjectHalen**. In het volgende voorbeeld verwijst de variabele `ActiefMSWord` naar een gestarte sessie van Microsoft Word.

```
Toewijzen ActiefMSWord = ObjectHalen("". "Word.Basic")
```

Net zoals bij de functie **ObjectMaken**, is het argument `"Word.Basic"` de programma-aanduiding voor Microsoft Word. In tegenstelling tot **ObjectMaken** wordt met **ObjectHalen** Microsoft Word niet gestart als dit nog niet is gebeurd. Wanneer er meerdere sessies van Microsoft Word zijn gestart, valt niet te voorspellen naar welke sessie `ActiefMSWord` verwijst.

U kunt deze vorm van de functie **ObjectHalen** samen met de functie **ObjectMaken** gebruiken, zoals u in het volgende voorbeeld kunt zien.

```
Functie EenObjectHalen(Programma_aanduiding)
    Bij Fout Hervatten Volgende
    Toewijzen EenObjectHalen = ObjectHalen("". Programma_aanduiding)
    Indien FoutNr Dan Toewijzen EenObjectHalen =_
ObjectMaken(Programma_aanduiding)
    Indien FoutNr Dan
        BerichtVenster "Object kon niet worden gemaakt."
        EenObjectHalen = Onwaar
    Einde Indien
Einde Functie
```

In het vorige voorbeeld maakt de functie EenObjectHalen gebruik van **ObjectHalen** om een object op te halen uit een gestarte toepassing. Wanneer de toepassing niet is gestart, start de functie EenObjectHalen de toepassing met behulp van **ObjectMaken**. Als dit niet lukt, wordt een bericht weergegeven en resulteert de functie in **Onwaar**.

Als u een object wilt ophalen uit een bestand, maakt u gebruik van de volgende syntaxis.

Toewijzen *objectvariabele* = **ObjectHalen**(*bestandsnaam*; _
Programma_aanduiding)

In het volgende voorbeeld wordt Microsoft Draw gestart, als deze toepassing nog niet is gestart, en wordt het bestand GUERNICA geopend.

```
Toewijzen Picasso = ObjectHalen("guernica"; "MSDraw.Metabestand")
```

Nadat u de variabele `Picasso` hebt toegewezen, kunt u `Picasso` verder in de code gebruiken om naar de tekening te verwijzen.

Deze manier om bestanden te openen kan niet in alle toepassingen worden gehanteerd. Zo kunt u op deze wijze bijvoorbeeld geen bestanden openen in Microsoft Word versie 6.0, omdat de programmeertaal WordBasic uitgaat van selectie in plaats van objecten. Wanneer u een bestand in Microsoft Word wilt selecteren, gebruikt u de functie **ObjectMaken** om Word te starten en opent u vervolgens het bestand met de methode **BestandOpenen** uit WordBasic, zoals u in het volgende voorbeeld kunt zien.

```
Toewijzen MSWord = ObjectMaken("Word.Basic")
' Start een Word-sessie.
MSWord.BestandOpenen Naam := "Renkema"
' Opent het bestand Renkema.
```

Er bestaat een belangrijk verschil tussen de vorige twee voorbeelden. De variabele `MSWord` verwijst naar een gestarte sessie van Word. In tegenstelling hiermee verwijst de variabele `Picasso` naar een specifiek bestand. In het Word-voorbeeld heeft uw code betrekking op het actieve document. Zodra er een ander document actief wordt in Word, heeft alle code vanaf dat moment betrekking op het nieuwe actieve document. In het Draw-voorbeeld heeft de code betrekking op een bepaald bestand, dat wordt aangeduid door de variabele `Picasso`, ongeacht welk document actief is.

Zie de documentatie bij de desbetreffende toepassing als u wilt weten of u bestanden in een bepaalde toepassing kunt ophalen.

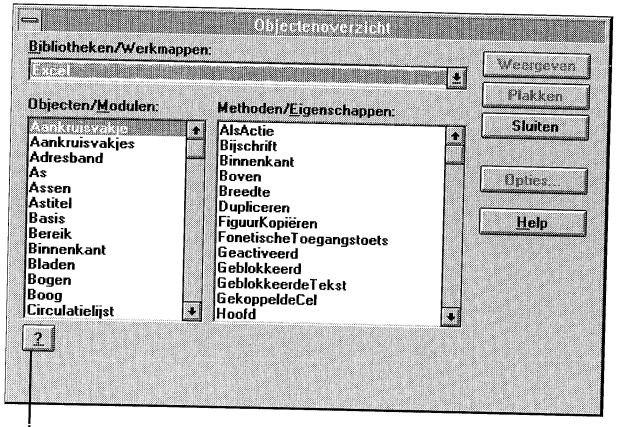
Toepassingen en objecten weergeven

In het dialoogvenster **Objectenoverzicht** kunt u de toepassingen en de objecten weergeven die op uw systeem zijn geïnstalleerd. In dit dialoogvenster worden ook alle procedures weergegeven die zijn opgenomen in geopende werkmappen of in invoegmacro's van Microsoft Excel. Zie hoofdstuk 13, "Automatische procedures en invoegmacro's maken", voor meer informatie over het gebruik van procedures uit andere werkmappen en over invoegmacro's.

▶ **Toepassingen die op uw systeem zijn geïnstalleerd en de bijbehorende objecten weergeven**

1. Schakel over naar een Visual Basic-module.
2. Kies de opdracht **Objectenoverzicht** in het menu **Beeld**.
3. Typ of selecteer in het vak "Bibliotheken/Werkmappen" de naam van de toepassing die u wilt weergeven.

   Visual Basic geeft de objecten van de toepassing weer in het vak "Objecten/Modulen".



Geeft volledige gegevens uit de online Visual Basic Naslaggids voor het geselecteerde onderwerp weer.

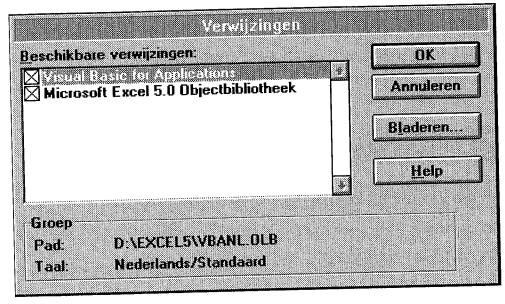
Verwijzingen naar toepassingen en werkmappen toevoegen

Wanneer u wilt dat Visual Basic de onderdelen weergeeft waarover een andere toepassing beschikt, moet u een verwijzing naar die toepassing in uw module opnemen. Nadat u deze verwijzing hebt opgenomen, zijn de objecten van die toepassing beschikbaar voor Visual Basic en worden deze weergegeven in het dialoogvenster **Objectenoverzicht**.

▶ Een verwijzing naar een toepassing of een werkmap opnemen

1. Schakel over naar een Visual Basic-module.
2. Kies de opdracht **Verwijzingen** in het menu **Extra**.
3. Selecteer in het vak "Beschikbare verwijzingen" de naam van de toepassing of de werkmap waarnaar u een verwijzing wilt opnemen.

   Als een toepassing niet beschikt over een objectenbibliotheek, verschijnt de naam van de toepassing niet in dit vak.



Tip Als u het aantal verwijzingen beperkt houdt, verloopt de compilatie sneller, omdat Visual Basic alle geselecteerde toepassingen doorzoekt op het voorkomen van symboolnamen.

Zie het vorige gedeelte, "Toepassingen en objecten weergeven", voor meer informatie over het weergeven van de objecten waarover een toepassing of werkmap beschikt.

Hoe Visual Basic naar verwijzingen zoekt

Als u in uw code naar een object verwijst, doorzoekt Visual Basic elke toepassing die in het dialoogvenster **Verwijzingen** is geselecteerd, in de volgorde waarin deze toepassingen worden weergegeven. Wanneer twee toepassingen dezelfde naam hanteren voor een object, gebruikt Visual Basic de naam die als eerste wordt aangetroffen. Dit principe is ook van toepassing op de methoden, eigenschappen en vooraf gedefinieerde constanten die in het dialoogvenster **Objectenoverzicht** worden weergegeven.

Over het algemeen valt het aan te raden om verwijzingen naar objecten in andere toepassingen of werkmappen zo gedetailleerd mogelijk op te geven, zoals u in het volgende voorbeeld kunt zien.

```
' Volledig opgegeven objectnaam wijst een bijschrift toe aan het
' venster van Microsoft Project.
MSProject.Bijschrift = "Project Bijschrift"

' Niet nader opgegeven naam zorgt ervoor dat Visual Basic het eerste
' gevonden object met de naam Bijschrift gebruikt, in dit geval
' Microsoft Excel.
Bijschrift = "Microsoft Excel Bijschrift"
```

De objecten van een toepassing gebruiken

Nadat u een variabele hebt gemaakt die verwijst naar een OLE-object, kunt u in Visual Basic bewerkingen uitvoeren op dit object op dezelfde wijze als op elk ander Microsoft Excel-object. Gebruik de syntaxis *object.eigenschap* om eigenschappen aan het object toe te wijzen of de waarden ervan op te vragen of om methoden op het object uit te voeren.

Sommige objecten bevatten andere objecten. Een taakobject bijvoorbeeld bevindt zich in een projectobject van Microsoft Project. Met de puntsyntaxis kunt u meerdere objecten, eigenschappen en methoden in dezelfde coderegel insluiten, op dezelfde wijze als u bij Microsoft Excel-objecten zou doen, zoals u in het volgende voorbeeld kunt zien.

```
MSProject.ActiveProject.Tasks(1).Start = Now
```

Sommige toepassingen, zoals Microsoft Word, beschikken over bestaande instructies in macrotaal als methoden. Deze methoden hanteren de argumenten en de syntaxis die worden beschreven in de documentatie bij de macrotaal van die toepassing, zoals u in het volgende voorbeeld kunt zien.

```
' Haalt de actieve sessie van Microsoft Word op.
Toewijzen MijnWord = ObjectHalen("". "Word.Basic")

' Slaat alle geopende documenten op
' (hetzelfde als instructie BestandAllesOpslaan van WordBasic).
MijnWord.BestandAllesOpslaan
```

Een toepassing afsluiten

Veel OLE-objecten ondersteunen een methode waarmee u het object en de bijbehorende toepassing kunt afsluiten. Omdat OLE-objecten een fors beslag kunnen leggen op de geheugenruimte die voor een procedure beschikbaar is, is het raadzaam om een object dat u niet meer nodig hebt, expliciet te sluiten. Gebruik de juiste methode om een object te sluiten, zoals u in het volgende voorbeeld kunt zien.

```
' Sluit het vensterobject.
Venster.Sluiten
' Sluit de toepassing af waarmee het object is gemaakt.
MijnToepassing.BestandAfsluiten
```

Wanneer een object is afgesloten, ofwel door de procedure of door de gebruiker, zijn alle variabelen die naar dit object verwijzen niet langer geldig, zoals u in het volgende voorbeeld kunt zien.

```
Toewijzen MijnToepassing = ObjectMaken("ToepassingXYZ")
MijnToepassing.BestandNieuw
MijnToepassing.Invoegen "Willekeurige tekst"
MijnToepassing.BestandAfsluiten
MijnToepassing.BeginVanDocument      ' Veroorzaakt een fout!
```

"Run-time"-fouten afhandelen

In het vorige voorbeeld wordt met de instructie `MijnToepassing.BestandAfsluiten` de toepassing afgesloten, zodat de daaropvolgende instructie een fout veroorzaakt. Omdat toepassingen om onvoorspelbare redenen kunnen worden gesloten door gebruikers, maar ook door systeemfouten, is het raadzaam om op deze fout te controleren wanneer u met objecten uit andere toepassingen werkt. De volgende code geeft hier een voorbeeld van.

```
Functie ZoekenInBestanden(GezochteTekst; BestandenLijst())
    Bij Fout GaanNaar FoutOpsporen
    Toewijzen MijnWord = ObjectMaken("Word.Basic")
    Voor Elke Naam In BestandenLijst
        MijnWord.BestandOpenen Naam
        MijnWord.BewerkenZoeken GezochteTekst
        Indien MijnWord.BewerkenZoekenGevonden() Dan
            LijstGevonden = LijstGevonden & ", " & Naam
        Einde Indien
        MijnWord.BestandSluiten
    Volgende
    ZoekenInBestanden = LijstGevonden
    Verlaten Functie

FoutOpsporen:
    Kiezen Ingeval FoutNr
        Ingeval 440                    ' OLE-fout.
            Pogingen = Pogingen + 1
            ' Maximaal 5 pogingen om Word opnieuw te starten.
            Indien Pogingen < 5 Dan
                Toewijzen MijnWord = ObjectMaken("Word.Basic")
                Hervatten
            Anders
                BerichtVenster "Word kan niet opnieuw worden gestart."
                Stoppen
            Einde Indien
        Ingeval Anders
            BerichtVenster "De volgende fout is opgetreden: " & FoutNr
            Stoppen
    Einde Kiezen
Einde Functie
```

Als er in het vorige voorbeeld een andere fout optreedt dan de fout met nummer 440, geeft de procedure het foutnummer weer en wordt de procedure afgesloten. Het is mogelijk dat de toepassing die het object bevat, een eigen foutnummer doorgeeft. Bijvoorbeeld WordBasic-fout 1078 waarmee wordt aangegeven dat een document niet bestaat. In sommige gevallen is dit foutnummer hetzelfde als een foutnummer dat in Visual Basic voor een andere fout wordt gebruikt. In dergelijke gevallen moet u de instructie **Bij Fout Hervatten Volgende** gebruiken en op fouten controleren direct na elke regel die een fout kan veroorzaken. Dit soort foutcontrole wordt *foutafhandeling binnen de procedure* genoemd.

Zie de documentatie bij een toepassing voor meer informatie over de foutnummers die de desbetreffende toepassing kan doorgeven. Zie hoofdstuk 9, "Foutafhandeling en foutwaarden", voor meer informatie over het afhandelen van fouten.

Opmerking Wanneer het bereik van een variabele niet meer gedefinieerd is, worden het object en de bijbehorende toepassing niet gesloten. U kunt de variabele echter niet langer gebruiken om naar het object te verwijzen. Als het object nog steeds actief is, kunt u de functie **ObjectHalen** gebruiken om een andere variabele aan het object toe te wijzen.

Afwijkende namen gebruiken

Sommige toepassingen gebruiken bij het benoemen van objecten, eigenschappen of methoden tekens die in Visual Basic een speciale betekenis hebben. Zo beschikt Microsoft Word bijvoorbeeld over methoden die resulteren in tekenreeksen. De naam van een dergelijke methode eindigt altijd op een dollarteken ($). Het dollarteken heeft echter een speciale betekenis in Visual Basic. Als u een van de tekenreeksfuncties van Microsoft Word in Visual Basic wilt gebruiken, moet u de naam tussen vierkante haken plaatsen, zoals u in het volgende voorbeeld kunt zien.

```
NaamVanOpmaakprofiel = MijnWord.[OpmaakprofNaam$]()
```

OLE gebruiken met DLL's

Dynamic Link Libraries (DLL's) en *programmacodebronnen* bevatten procedures die u op ongeveer dezelfde wijze kunt gebruiken als de functies en instructies van Visual Basic. Met DLL's en programmacodebronnen worden de mogelijkheden waarover u in een procedure beschikt, aanmerkelijk uitgebreid.

Opmerking Hoewel er in technisch opzicht verschillen bestaan tussen DLL's en de programmacodebronnen van Macintosh, zijn deze in het gebruik bijna identiek. Omwille van de beknoptheid wordt in dit hoofdstuk het begrip *DLL* gebruikt om te verwijzen naar Dynamic Link Libraries en programmacodebronnen.

Enkele DLL's ondersteunen OLE, waardoor deze eenvoudiger zijn te gebruiken in Visual Basic. DLL's die OLE ondersteunen, worden weergegeven in het dialoogvenster **Verwijzingen** en de bijbehorende procedures worden opgenomen in het dialoogvenster **Objectenoverzicht**.

▶ **Een procedure vanuit een DLL gebruiken**

1. Als de DLL OLE ondersteunt, selecteert u de gewenste DLL in het dialoogvenster **Verwijzingen**.

   Zie "Verwijzingen naar toepassingen en werkmappen toevoegen" eerder in dit hoofdstuk voor verdere instructies.

   –Of–

   Als de DLL OLE niet ondersteunt, moet u de DLL declareren.

   Zie het volgende gedeelte, "DLL's declareren", voor verdere instructies.

2. Roep de procedure aan.

   Zie "DLL-procedures aanroepen" verderop in dit hoofdstuk voor nadere instructies.

Belangrijk Visual Basic kan niet controleren of u de juiste waarden aan een DLL doorgeeft. Als u een onjuiste waarde doorgeeft, kunnen er storingen optreden in de procedure, hetgeen er toe kan leiden dat de Visual Basic-procedure of de toepassing vastloopt. Indien u de wijzigingen in de werkmap niet regelmatig opslaat, kunnen er gegevens verloren gaan. Misschien zult u het besturingssysteem opnieuw moeten starten.

DLL-procedures declareren

Als een DLL OLE niet ondersteunt, kunt u de instructie **Declareren** gebruiken om Visual Basic de informatie te geven die nodig is om de DLL-procedures te vinden en te starten. Nadat u een DLL-procedure hebt gedeclareerd, kunt u deze even gemakkelijk in uw code gebruiken als elke andere procedure.

U declareert een DLL-procedure door de instructie **Declareren** in het begin van de module te plaatsen. Gebruik hiervoor de volgende syntaxis.

Declareren Functie *algemenenaam* **Biblio** "*bibliotheeknaam*" [**Alias** "*alias*"] _
(*argumenten*) [**Als** *type*]

–Of–

Declareren Sub *algemenenaam* **Biblio** "*bibliotheeknaam*" [**Alias** "*alias*"] _
(*argumenten*)

Bijvoorbeeld:

```
' Windows-DLL declareren.
Declareren Functie SysteemGegevensOphalen Biblio "User" _
    Alias "GetSystemMetrics" (ViaWaarde N Als Integer) Als Integer

' Programmacodebron voor Macintosh declareren.
Declareren Sub Waarschuwing Biblio "MijnVS:MijnWrs" _
    Alias "XLVB$MijnWrs" (ViaWaarde N Als Integer)
```

De clausule **Biblio** "*bibliotheeknaam*" zorgt ervoor dat Visual Basic weet waar de DLL te vinden is. Als de DLL zich in het pad van uw systeem bevindt, hoeft u de naam van de directory en het station niet op te geven.

De clausule **Alias** "*alias*" geeft aan dat de procedure in de DLL een andere naam heeft dan in de Visual Basic-code. U moet deze clausule gebruiken als de procedure een naam heeft die niet geldig is in Visual Basic.

De aanduiding *argumenten* staat voor een lijst met argumenten die worden gebruikt wanneer de DLL wordt opgeroepen. De volgende overwegingen zijn van belang wanneer u de argumenten opgeeft voor de declaratie van een DLL-procedure:

- Standaard worden in Visual Basic alle argumenten doorgegeven via een verwijzing. Bij veel DLL-procedures is het echter vereist dat de argumenten worden doorgegeven als een waarde. Plaats het sleutelwoord **ViaWaarde** voor de declaratie van het argument in de instructie **Declareren**, als u wilt dat het argument als een waarde wordt doorgegeven.
- In een aantal DLL-procedures kunt u voor hetzelfde argument soms verschillende gegevenstypen gebruiken. Als u meer dan één gegevenstype wilt doorgeven, declareert u het argument met de sleutelwoorden **Als Alle** om eventuele restricties op te heffen.
- Als u tekenreeksen wilt doorgeven, moet u het sleutelwoord **ViaWaarde** gebruiken om een tekenreeks uit Visual Basic om te zetten in een tekenreeks die eindigt op een nulwaarde.

In de volgende tabel ziet u een overzicht van de meest voorkomende DLL-argumenten en de bijbehorende equivalenten in Visual Basic.

DLL-argument	Declareer als volgt in Visual Basic	Oproepen met
Boole-argument	*I* **Als Integer**	Willekeurige **Integer-** of **Variant-**variabele
Pointer naar een tekenreeks (`LPSTR`)	**ViaWaarde** *T* **Als Tekenreeks**	Willekeurige **Tekenreeks-** of **Variant-**variabele
Pointer naar een geheel getal (`LPINT`)	*I* **Als Integer**	Willekeurige **Integer-** of **Variant-**variabele
Pointer naar een lang geheel getal (`LPDWORD`)	*L* **Als Lang**	Willekeurige **Lang-** of **Variant-**variabele
Pointer naar een structuur (bijvoorbeeld: `LPRECT`)	*T* **Als** *Rechthoek*	Willekeurige variabele van het door de gebruiker gedefinieerde gegevenstype
Geheel getal (`INT`, `UINT`, `WORD`, `BOOLE`)	**ViaWaarde** *I* **Als Integer**	Willekeurige **Integer-** of **Variant-**variabele
Ingang (`hWnd`, `hDC`, `hMenu` enz.)	**ViaWaarde** *g* **Als Integer**	Willekeurige **Integer-** or **Variant-**variabele
Ingang (Windows NT™)	**ViaWaarde** *g* **Als Lang**	Willekeurige **Lang-** of **Variant-**variabele
Lang (`DWORD`, `LONG`)	**ViaWaarde** *L* **Als Lang**	Willekeurige **Lang-** of **Variant-**variabele
Pointer naar een matrix met gehele getallen	*I* **Als Integer**	Het eerste element van de matrix, zoals `I(0)`
Pointer naar een void (`void *`)	**Als Alle**	Willekeurige variabele (gebruik **ViaWaarde** om de tekenreeks door te geven)
void (resultaatwaarde van een functie)	**Sub** procedure	Niet van toepassing
`NULL (nulwaarde)`	**Als Alle**	**ViaWaarde 0&**

Als de DLL-procedure andere procedures aanroept op dezelfde manier als C (in tegenstelling tot de manier die in Pascal wordt gebruikt), moet u het sleutelwoord **CDecl** opnemen in de declaratie van de DLL-procedure. Zie *Declareren* en *Roepen* in de online Visual Basic Naslaggids voor meer informatie over het aanroepen en declareren van DLL's.

DLL-procedures aanroepen

Nadat u een DLL hebt geselecteerd in het dialoogvenster **Verwijzingen** of de procedure hebt gedeclareerd met de instructie **Declareren**, kunt u de DLL-procedure op dezelfde wijze aanroepen als een instructie of functie van Visual Basic, zoals u in het volgende voorbeeld kunt zien.

```
' Windows DLL aanroepen.
Const SG_MUISAANWEZIG = 19
Indien SysteemGegevensOphalen(SG_MUISAANWEZIG) _
    Dan BerichtVenster "Muis geïnstalleerd"

' Programmacodebron voor Macintosh aanroepen.
Waarschuwing 0
```

Als een DLL geen OLE ondersteunt, moet u met de volgende zaken rekening houden:

- Tekenreeksargumenten moeten met het sleutelwoord **ViaWaarde** worden gedeclareerd.
- DLL's kunnen geen tekenreeksen als resultaat geven.
- DLL's kunnen niet de lengte vergroten van tekenreeksen die eraan worden doorgegeven. U kunt dit probleem omzeilen door de tekenreeks lang genoeg te maken, zodat de DLL-procedure niet voorbij het einde van de tekenreeks schrijft. Zie het volgende voorbeeld.

  ```
  Dim TekenreeksArgument Als Tekenreeks * 255
  ```

- DLL-procedures kunnen geen toegang verkrijgen tot matrices.
- Door de gebruiker gedefinieerde gegevenstypen kunnen niet worden doorgegeven als een waarde.
- Door de gebruiker gedefinieerde gegevenstypen die worden doorgegeven aan een DLL-procedure, mogen geen tekenreeksen met variabele lengte bevatten.
- Eigenschappen moeten als een waarde worden doorgegeven.
- U kunt geen object, zoals een werkblad, doorgeven aan een DLL-procedure.

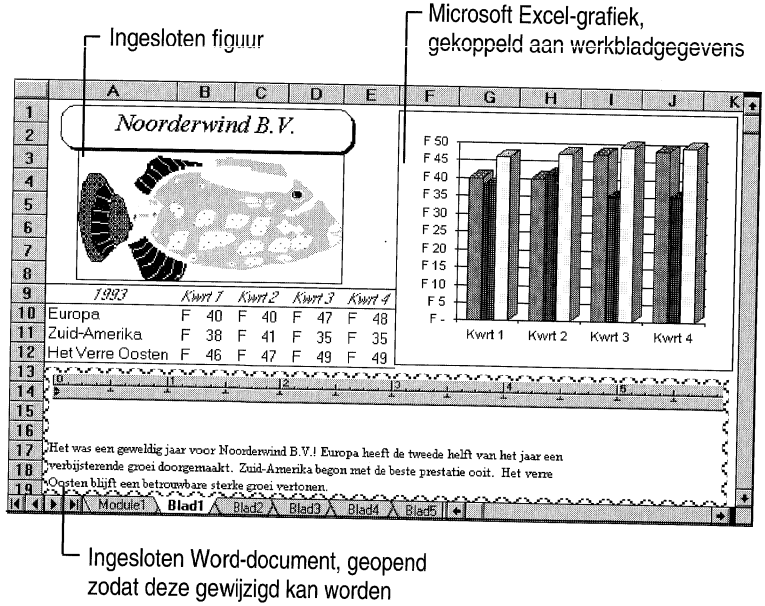
Onder bepaalde omstandigheden maken DLL's gebruik van specifieke soorten argumenten:

- Als de DLL-procedure een geheugenbuffer nodig heeft, moet u gebruik maken van tekenreeksen. Zorg ervoor dat de tekenreeks lang genoeg is om de gegevens te bevatten die het resultaat zijn van de procedure.

- Als u een null-pointer wilt doorgeven aan een procedure, declareert u het argument met de sleutelwoorden **Als Alle** en geeft u de expressie **ViaWaarde 0&** door. U kunt van deze techniek gebruik maken om een null-pointer door te geven aan elke willekeurige procedure die een argument via een verwijzing accepteert.
- Als een procedure een ingang als argument accepteert, moet u deze altijd declareren met **ViaWaarde Integer** of **ViaWaarde Lang**. DLL-functies die een ingang als resultaat geven, kunnen worden gedeclareerd met **Integer** of **Lang**. Ingangen zijn identificatienummers, geen pointers of numerieke waarden. U kunt geen wiskundige bewerkingen op ingangen uitvoeren.

Gekoppelde en ingesloten objecten gebruiken

Vanuit toepassingen die het koppelen en insluiten van objecten ondersteunen, kunt u objecten in een werkblad van Microsoft Excel opnemen. Zo kunt u bijvoorbeeld een grafische afbeelding uit Microsoft Draw in een werkblad van Microsoft Excel insluiten. Gekoppelde en ingesloten objecten worden als onderdeel van het werkblad weergegeven. Ze kunnen op dezelfde wijze als andere onderdelen van het werkblad worden verplaatst, verwijderd of bewerkt.



Het belangrijkste verschil tussen gekoppelde en ingesloten objecten, is de plaats waar de gegevens worden opgeslagen. De gegevens die betrekking hebben op een gekoppeld object, worden door de toepassing waarin deze zijn gemaakt in het oorspronkelijke document opgeslagen. Gegevens die betrekking hebben op een ingesloten object, worden opgeslagen in het bestand waarin dit object zich bevindt. Een object dat is ingesloten in een werkmap van Microsoft Excel, wordt samen met de werkmap opgeslagen.

In de volgende gedeelten wordt beschreven hoe u gekoppelde en ingesloten objecten kunt gebruiken in de Visual Basic-taal. Zie hoofdstuk 41, "Gegevens en grafische afbeeldingen uitwisselen tussen toepassingen", in het *Microsoft Excel Handboek* voor meer informatie over het koppelen en insluiten van objecten met behulp van de menu's en dialoogvensters van Microsoft Excel.

Visual Basic beschikt over twee objecten, OLEObjecten en OLEObject, waarmee u gekoppelde en ingesloten objecten kunt invoegen of ophalen en bewerkingen hierop kunt uitvoeren. De volgende methoden hebben betrekking op gekoppelde en ingesloten objecten.

Methode	Doel
Verwijderen	Verwijdert een gekoppeld of ingesloten object.
Bijwerken	Werkt een gekoppeld object bij aan de hand van de brontoepassing.
Werkwoord *OLE-werkwoord*	Voert een actie uit op het object. Deze acties worden gedefinieerd door de toepassing waarin het object is gemaakt. De meeste gekoppelde of ingesloten objecten ondersteunen de werkwoorden xlPrimaire en xlOpen.

De volgende eigenschappen hebben betrekking op gekoppelde en ingesloten objecten.

Eigenschap	Doel
Object	Geeft het OLE-object als resultaat, als dit object OLE ondersteunt.
OLEType	Geeft **xlOLEGenest** als resultaat als het object is ingesloten of **xlOLEKoppelingen** als het object is gekoppeld.
AutomatischBijwerken	Geeft aan of een gekoppeld object al dan niet automatisch wordt bijgewerkt wanneer de brongegevens worden gewijzigd.

Zie de namen van het desbetreffende object, de methode of de eigenschap in de online Visual Basic Naslaggids voor meer informatie over een bepaald onderwerp.

▶ Gekoppelde of ingesloten objecten vanuit Visual Basic gebruiken

1. Gebruik een bestaand gekoppeld of ingesloten object of voeg een nieuw object in.
2. Voer de gewenste bewerkingen uit op het object, met behulp van methoden en eigenschappen.

In de volgende procedure worden bijvoorbeeld alle gekoppelde objecten in een werkblad bijgewerkt.

```
Sub AlleKoppelingenBijwerken()
' Elk gekoppeld of ingesloten object in het actieve werkblad ophalen.
    Voor Elke GekoppeldObject In ActiefBlad.OLEObjecten
        ' Indien gekoppeld, dan object bijwerken vanuit bron.
    Indien GekoppeldObject.OLEType = xlOLEKoppelingen _
        Dan GekoppeldObject.Bijwerken
    Volgende
Einde Sub
```

U kunt van een soortgelijke code gebruik maken om verschillende bewerkingen uit te voeren op gekoppelde en ingesloten objecten afzonderlijk. In de volgende gedeelten wordt dieper ingegaan op het ophalen en invoegen van gekoppelde en ingesloten objecten en het uitvoeren van specifieke bewerkingen hierop.

Bestaande gekoppelde en ingesloten objecten ophalen

Als u een gekoppeld of ingesloten object in een werkblad wilt invoegen met Visual Basic, maakt u gebruik van de volgende syntaxis.

blad.**OLEObjecten**(*onderdeel*)

Het object OLEObjecten kan in de instructie **Toewijzen** worden gebruikt om een variabele te maken die verwijst naar een specifiek gekoppeld of ingesloten object.

In het volgende voorbeeld verwijst `MijnObject` naar het eerste gekoppelde of ingesloten object in het werkblad Blad1.

```
Toewijzen MijnObject = Werkbladen("Blad1").OLEObjecten(1)
```

Als de brontoepassing voor een gekoppeld of ingesloten object OLE ondersteunt, kunt u veel meer te weten komen over de inhoud van het object. Zie "Gekoppelde en ingesloten objecten gebruiken met OLE" verderop in dit hoofdstuk voor meer informatie over dit onderwerp.

Een ingesloten object invoegen

Als u een nieuw ingesloten object wilt invoegen, gebruikt u de volgende syntaxis.

blad.**OLEObjecten.Toevoegen** *Klassetype*

Bijvoorbeeld:

```
' Sluit een nieuwe tekening in.
Toewijzen IngeslotenTekening1 = ActiefBlad.OLEObjecten.Toevoegen _
    (Klassetype := "MSDraw")
```

Als u een ingesloten object uit een ander bestand wilt invoegen, gebruikt u de volgende syntaxis.

blad.**OLEObjecten.Toevoegen Bestandsnaam :=** *bestandsnaam*; _
Koppeling := Onwaar

Bijvoorbeeld:

```
' Sluit een tekening uit een bestaand bestand in.
Toewijzen IngeslotenTekening2 = ActiefBlad.OLEObjecten.Toevoegen _
    (Bestandsnaam := "guernica";     Koppeling := Onwaar )
```

Omdat ingesloten objecten worden opgeslagen in de werkmap, worden in het vorige voorbeeld wijzigingen die worden aangebracht in het bestand GUERNICA niet weergegeven in het ingesloten object. Als u geen bestandsnaam opgeeft, neemt Visual Basic aan dat het object is ingesloten en wordt het argument **koppeling** genegeerd.

Een gekoppeld object invoegen

Als u een gekoppeld object wilt invoegen met Visual Basic, gebruikt u de volgende syntaxis.

blad.**OLEObjecten.Toevoegen bestandsnaam :=** *bestandsnaam*; _
koppeling := Waar

Bijvoorbeeld:

```
' Voegt een gekoppelde tekening in.
Toewijzen GekoppeldeTekening = ActiefBlad.OLEObjecten.Toevoegen _
    (Bestandsnaam := "guernica"; Koppeling := Waar)
```

Omdat gekoppelde objecten zijn opgeslagen in het oorspronkelijke bestand, moet u de bestandsnaam opgeven wanneer u het object invoegt. Wijzigingen die in het bestand GUERNICA uit het vorige voorbeeld worden aangebracht, worden weergegeven in het gekoppelde object zodra u dit bijwerkt.

Bijvoorbeeld:

```
' Werkt het gekoppelde object direct bij.
GekoppeldeTekening.Bijwerken
```

Acties uitvoeren op gekoppelde en ingesloten objecten

Met Visual Basic kunt u acties uitvoeren op gekoppelde en ingesloten objecten. Zo kunt u bijvoorbeeld een object activeren om dit te bewerken. Als u een actie wilt uitvoeren op een object, maakt u gebruik van de volgende syntaxis.

OLE-object.**Werkwoord** *OLE-werkwoord*

Bijvoorbeeld:

```
' Voer de standaardactie uit op elk gekoppeld of ingesloten object.
Voor Elke MijnObject In ActiefBlad.OLEObjecten
    MijnObject.Werkwoord xlPrimaire
Volgende MijnObject
```

Veel gekoppelde en ingesloten objecten beschikken over acties die uniek zijn voor die objecten. Zo kan een ingesloten geluidsopname beschikken over de actie Afspelen.

Zie de documentatie bij de brontoepassing van het object wanneer u wilt weten welke acties op een gekoppeld of ingesloten object kunnen worden uitgevoerd. Acties worden vaak in de documentatie opgenomen als OLE-werkwoorden.

Tip De meeste toepassingen ondersteunen de werkwoorden xlOpen en xlPrimaire. Met het werkwoord xlPrimaire start u de standaardactie van het object. Meestal is deze standaardactie het openen van het object zodat het kan worden bewerkt. Omdat deze actie wordt bepaald door de toepassing waarin het object is gemaakt, kan het echter ook een andere actie betreffen, zoals het afspelen van een geluidsopname.

Gekoppelde en ingesloten objecten bewerken

Sommige objecten kunt u direct bewerken, afhankelijk van de bijbehorende brontoepassing. *Direct bewerken* betekent dat u het te bewerken object activeert en dit vervolgens in het werkblad zelf bewerkt zonder een nieuw venster te openen.

Als het object is gekoppeld of als de brontoepassing directe bewerking niet ondersteunt, moet u het object in een nieuw venster openen wanneer u het wilt bewerken.

Als u een object wilt activeren voor directe bewerking, gebruikt u de volgende syntaxis.

OLE-object.**Activeren**

Bijvoorbeeld:

```
' Een tekening insluiten.
Toewijzen IngeslotenTekening = ActiefBlad.OLEObjecten.Toevoegen _
    (Bestandsnaam := "guernica";     Koppeling := Onwaar)
' Het object activeren.
IngeslotenTekening.Activeren
```

Als u een gekoppeld of ingesloten object in een nieuw venster wilt openen om dit te bewerken, gebruikt u de volgende syntaxis.

OLE-object.**Werkwoord xlOpen**

Bijvoorbeeld:

```
' De brontoepassing openen zodat het object kan worden bewerkt.
IngeslotenTekening.Werkwoord xlOpen
```

Gekoppelde objecten bijwerken

Als u een gekoppeld object met Visual Basic wilt bijwerken, gebruikt u de volgende syntaxis.

OLE-object.**Bijwerken**

Bijvoorbeeld:

```
' De koppeling aan de hand van de brontoepassing bijwerken.
GekoppeldeTekening.Bijwerken
```

Alleen gekoppelde objecten moeten op deze wijze worden bijgewerkt. Ingesloten objecten worden opgeslagen samen met het document waarin ze zijn ingesloten en zijn daarom altijd "up-to-date".

Gekoppelde en ingesloten objecten verwijderen

Als u een gekoppeld of ingesloten object wilt verwijderen met Visual Basic, gebruikt u de volgende syntaxis.

OLE-object.**Verwijderen**

Bijvoorbeeld:

```
' De koppeling verwijderen.
GekoppeldeTekening.Verwijderen
```

Gekoppelde en ingesloten objecten gebruiken met OLE

Als een toepassing OLE ondersteunt, kunt u voor de gekoppelde of ingesloten objecten van die toepassing de bij de toepassing behorende eigenschappen en methoden gebruiken. Zo kunt u bijvoorbeeld voor een Microsoft Word-document dat in een werkblad is ingesloten, methoden van Microsoft Word gebruiken.

▶ Een gekoppeld of ingesloten object gebruiken met OLE

1. Gebruik de eigenschap **Object** van het gekoppelde of ingesloten object om rechtstreeks naar het object te verwijzen of om een variabele te maken die naar het object verwijst.

   De eigenschap **Object** van het OLE-object heeft de volgende syntaxis.

   OLE-object.**Object**

2. Schrijf code die gebruik maakt van de eigenschappen en de methoden van de bij het object behorende toepassing.

Bijvoorbeeld:

```
' Een nieuw ingesloten object maken.
Toewijzen MijnIngeslotenWordDocument = ActiefBlad.OLEObjecten.Toevoegen
_
    (Klassetype := "word.document.6")
' Een variabele maken die naar Word verwijst via het ingesloten object.
Toewijzen MijnWordBasic =
MijnIngeslotenWordDocument.Object.Toepassing.WordBasic
MijnIngeslotenWordDocument.Activeren
' Tekst invoegen met de methode Invoegen van Word.
MijnWordBasic.Invoegen "Deze tekst in het ingesloten document invoegen."
```

U kunt gebruik maken van de eigenschappen en methoden van gekoppelde of ingesloten objecten om informatie op te halen over de inhoud van de objecten in een werkblad.

In het volgende voorbeeld worden de gekoppelde en ingesloten objecten in het actieve werkblad gecontroleerd en worden de Word-documenten die de naam "Dorien Elskens", bevatten in een lijst opgenomen.

```
Voor Elke MijnObject In ActiefBlad.OLEObjecten
    ' Neem aan dat alle OLE-objecten van Word afkomstig zijn.
    ' Activeer het object, want Word vereist een selectie.
    MijnObject.Activeren
    Met MijnObject.Object.Toepassing.Wordbasic
         'Naar de naam zoeken.
            BeginVanDocument
        .BewerkenZoeken Zoeken:="Dorien Elskens",Richting:=0
        ' Indien gevonden, bestandsnaam toevoegen aan lijst.
        Indien .BewerkenZoekenGevonden()=-1 Dan
            BestandenLijst = BestandenLijst & ", " & _
                .Bestandsnaam()
        Einde Indien
    Einde Met
    ' Deactiveer het OLE-object, als voorbereiding op het volgende _
object
    Bereik("A1").Selecteer
Volgende
```

DDE gebruiken

DDE (Dynamic Data Exchange) is een mechanisme dat twee toepassingen in staat stelt met elkaar te "praten" door continu en automatisch gegevens uit te wisselen. Dankzij DDE hoeft u niet meer handmatig gegevens te knippen en te plakken om deze te kunnen uitwisselen tussen twee toepassingen.

In de volgende tabel vindt u een overzicht van de taken die Microsoft Excel kan uitvoeren met DDE en de methoden die bij elke taak behoren.

Taak	Methode
DDE starten	**DDEKanaalOpenen**
Tekst uit een andere toepassing ophalen	**DDEVerzoeken**
Tekst naar een andere toepassing verzenden	**DDEInzetten**
Een opdracht uitvoeren in een andere toepassing	**DDEBestandUitvoeren**
DDE beëindigen	**DDEKanaalSluiten**

Opmerking Niet alle toepassingen ondersteunen DDE. Zie de documentatie bij de desbetreffende toepassing wanneer u wilt weten of een toepassing DDE ondersteunt.

DDE starten

Voordat u een andere taak kunt uitvoeren met DDE, moet u een verbinding tot stand brengen met de andere toepassing. Deze verbinding wordt een *DDE-kanaal* genoemd. Dit DDE-kanaal wordt als argument gebruikt in alle daaropvolgende DDE-methoden.

▶ DDE starten en een DDE-kanaal openen

1. Start de andere toepassing, als dit nog niet is gebeurd.
2. Gebruik de methode **DDEKanaalOpenen** om een verbinding tot stand te brengen en de waarde van het DDE-kanaal op te vragen.

   De methode **DDEKanaalOpenen** heeft de volgende syntaxis.

   kanaalNummer = **DDEKanaalOpenen**(*toepassingsnaam*; *onderwerpnaam*)

In het volgende voorbeeld wordt Microsoft Word gestart en wordt een DDE-kanaal geopend naar het bestand RENKEMA. De variabele `WordKanaal` kan nu worden gebruikt door de andere DDE-methoden.

Gebruik in Microsoft Excel voor Windows de volgende code.

```
Starten "WINWORD.EXE C:\DOCS\RENKEMA.DOC"
WordKanaal = DDEKanaalOpenen("WinWord"; "C:\Docs\Renkema.DOC")
```

Gebruik in Microsoft Excel voor de Macintosh de volgende code.

```
Starten "Microsoft Word"
WordKanaal = DDEKanaalOpenen("Microsoft Word"; "VS1:Documenten:Renkema")
```

In de volgende tabel ziet u een overzicht van de DDE-toepassingsnamen voor een aantal Microsoft-toepassingen.

Toepassing	DDE-toepassingsnaam
Microsoft Word voor Windows	WinWord
Microsoft Word voor de Macintosh	Microsoft Word
Microsoft Project voor Windows	Project
Microsoft Project voor de Macintosh	Microsoft Project
Microsoft Access ®	MSAccess
Microsoft FoxPro® voor Windows	FoxPro
Microsoft Windows Programmabeheer	ProgMan

Tekst en getallen ophalen

Nadat u DDE hebt gestart met de methode **DDEKanaalOpenen**, kunt u de methode **DDEVerzoeken** gebruiken om tekst en getallen via het DDE-kanaal op te halen. Gebruik hiervoor de volgende syntaxis.

variabele = **DDEVerzoeken**(*kanaalNummer*; *onderdeel*)

variabele moet behoren tot het gegevenstype **Variant**, aangezien DDERequest altijd een matrix als resultaat geeft.

In het volgende voorbeeld wordt de tekst toegewezen die zich bevindt bij de bladwijzer `DDE_Koppeling1` in een Word-document dat voor DDE is geopend.

```
WordNotitie1 = DDEVerzoeken(WordKanaal; "DDE_Koppeling1")
```

De methode **DDEVerzoeken** kan alleen tekst als resultaat geven. U kunt deze methode niet gebruiken om afbeeldingen of opmaak uit een ander document over te nemen.

Tekst en getallen verzenden

Nadat u DDE hebt gestart met de methode **DDEKanaalOpenen**, kunt u de methode **DDEInzetten** gebruiken om tekst en getallen te verzenden naar het DDE-kanaal. U gebruikt hiervoor de volgende syntaxis.

DDEInzetten(*kanaalnummer*; *onderdeel*; *gegevens*)

In het volgende voorbeeld wordt de tekst bij de bladwijzer `DDE_Koppeling1` vervangen in een Word-document met de waarde van cel A1 op het actieve blad.

```
DDEInzetten_WordKanaal; "DDE_Koppeling1"; Cellen(1;1)
```

De methode **DDEInzetten** kan alleen worden gebruikt om tekst of getallen te verzenden. U kunt deze methode niet gebruiken om een afbeelding of een bepaalde opmaak in een document in te voegen.

Opdrachten verzenden

Nadat u DDE hebt gestart met de methode **DDEKanaalOpenen**, kunt u de methode **DDEBestandUitvoeren** gebruiken om met behulp van DDE een opdracht in een andere toepassing uit te voeren. Gebruik hiervoor de volgende syntaxis.

DDEBestandUitvoeren(*kanaalNummer*; *DDE-opdracht*)

Met de code in het volgende voorbeeld wordt een nieuw document geopend in Word.

```
DDEBestandUitvoeren_WordKanaal; "[BestandOpenen .Naam = ""NIEUW.DOC""]"
```

Welke opdrachten u kunt verzenden via DDE, wordt bepaald door de toepassing waarmee u een verbinding legt. In Microsoft Word zijn bijvoorbeeld alle opdrachten van WordBasic beschikbaar via DDE, hoewel het gebruik van OLE veel sneller is. Zie "OLE gebruiken met toepassingen" eerder in dit hoofdstuk voor meer informatie over het verzenden van WordBasic-opdrachten naar Microsoft Word met OLE.

DDE beëindigen

Gebruik de methode **DDEKanaalSluiten** om de DDE-verbinding met een andere toepassing te beëindigen. Nadat u de methode **DDEKanaalSluiten** hebt gebruikt, kunt u het DDE-kanaal niet meer gebruiken, tenzij u DDE opnieuw start met de methode **DDEKanaalOpenen**.

Als u DDE wilt beëindigen, gebruikt u de volgende syntaxis.

DDEKanaalSluiten(*kanaalnummer*)

Met de code in het volgende voorbeeld sluit u de DDE-sessie af.

```
DDEKanaalSluiten WordKanaal
```

Het valt aan te bevelen om een DDE-sessie te beëindigen nadat u de onderhanden zijnde taak hebt afgesloten. Er kan slechts een beperkt aantal kanalen tegelijkertijd geopend zijn en DDE-kanalen blijven geopend totdat ze worden gesloten met de methode **DDEKanaalSluiten**.

Werken met DDE-koppelingen

Met de gegevens die resulteren uit de methode **Koppelingbronnen**, kunt u de Microsoft Excel- en DDE-koppelingen in een werkmap bijwerken of wijzigen. Hiervoor maakt u gebruik van de volgende syntaxis.

koppelingsmatrix = *werkmap*.**Koppelingbronnen**(*type*)

Het argument *type* geeft aan welk type koppeling de methode als resultaat geeft. **Koppelingbronnen** kan resulteren in vier verschillende typen koppelingen.

Type koppeling	Beschrijving
xlExcelKoppelingen (standaard)	Geeft een lijst met Microsoft Excel-koppelingen in de werkmap als resultaat
xlOLEKoppelingen	Geeft een lijst met de OLE- en de DDE-koppelingen in de werkmap als resultaat
xlPublicaties, xlAbonnees	Geeft een lijst met de edities en de verwijzingen van de editie als resultaat (alleen Macintosh)

In de volgende tabel ziet u een overzicht van de overige werkmapmethoden die betrekking hebben op koppelingen.

Methode	Doel
Koppelinginfo	Geeft informatie over een koppeling als resultaat
KoppelingenOpenen	Opent het brondocument om dit te bewerken
KoppelingBepalenBij Gegevens	Definieert een procedure die wordt gestart telkens als de gegevens in de koppeling worden gewijzigd
KoppelingBijwerken	Werkt een koppeling bij

In het volgende voorbeeld worden alle koppelingen naar Microsoft Word geactiveerd zodat deze kunnen worden bewerkt, en worden vervolgens alle koppelingen van Microsoft Excel in de werkmap bijgewerkt.

```
KoppelingsMatrix = ActieveWerkmap.Koppelingbronnen(xlOLEKoppelingen)
Voor ElementKoppeling = 1 Tot BovenGrens(KoppelingsMatrix)
    Indien KoppelingsMatrix(ElementKoppeling) Zoals "*Word*" Dan
        ActieveWerkmap.KoppelingenOpenen     (KoppelingsMatrix _
            (ElementKoppeling);      Type := xlOLEKoppelingen)
    Einde Indien
Volgende ElementKoppeling
' Wijzigingen doorvoeren in de OLE- en DDE-koppelingen.
ActieveWerkmap.KoppelingBijwerken Type := xlExcelKoppelingen
```

Onder de naam van de desbetreffende methode in de online Visual Basic Naslaggids vindt u meer informatie over de werkmapmethoden die betrekking hebben op koppelingen.

Fouten afhandelen

Wanneer u gebruik maakt van DDE, moet u bij het schrijven van de code rekening houden met onverwachte omstandigheden die kunnen optreden in de toepassing waarmee u gegevens uitwisselt. Zo kan bijvoorbeeld een gebruiker onverwacht een toepassing afsluiten terwijl u juist gegevens met deze toepassing uitwisselt. Ook kan het soms te lang duren voordat een toepassing reageert op een opdracht.

In een aantal gevallen kunt u nagaan wat er is gebeurd door de resulterende foutcode of de regel waar de toepassing is gestopt, te controleren. In andere situaties moet u misschien een opdracht een aantal malen opnieuw geven.

In de volgende code worden bijvoorbeeld een aantal opdrachtreeksen uitgevoerd die elders in de code zijn gedefinieerd. Als er een fout optreedt, zoals een "time-out"-fout, wordt de opdracht maximaal tien keer opnieuw geprobeerd. Wanneer het dan nog niet lukt de opdracht te starten, wordt een foutbericht weergegeven en wordt het programma afgebroken.

```
Sub DDEOpdrachtenAanroepen(TekenreeksenUitvoeren())
    Bij Fout GaanNaar OpdrachtHerhalen
    Voor Elke Opdr In TekenreeksenUitvoeren
        DDEBestandUitvoeren(KanaalNummer; Opdr)
    Volgende Teller
    Verlaten Sub
```

```
OpdrachtHerhalen:
    Indien Pogingen < 10 Dan
        Pogingen = Pogingen + 1
        SysActies                          ' Ga naar andere
                                           ' toepassing.
        Hervatten                          ' Probeer opnieuw.
    Anders
        BerichtVenster Fout(FoutNr)        ' Geef foutbericht weer.
    Einde Indien
Einde Sub
```

Wanneer u op een fout reageert door een opdracht opnieuw uit te voeren, zoals in dit voorbeeld, valt het aan te raden om de instructie **SysActies** in de code op te nemen, zodat het besturingssysteem de overige toepassingen kan uitvoeren. Als u dit niet doet, lijkt het alsof het systeem is geblokkeerd terwijl opnieuw wordt geprobeerd de opdracht te starten.

Zie hoofdstuk 9, "Foutafhandeling en foutwaarden", voor meer informatie over het afhandelen van fouten.

Toetsaanslagen verzenden

De eenvoudigste manier om binnen Microsoft Windows te communiceren met toepassingen die het gebruik van koppelen en insluiten, OLE of DDE niet ondersteunen, is het verzenden van toetsaanslagen.

Opmerking U kunt toetsaanslagen verzenden naar elke actieve Windows-toepassing. U kunt echter geen toetsaanslagen verzenden naar een toepassing die in een MS-DOS-sessie binnen Windows draait. Het is ook niet mogelijk toetsaanslagen te verzenden naar toepassingen binnen het Macintosh-besturingssysteem.

Toetsaanslagen verzenden naar een andere toepassing heeft hetzelfde resultaat als wanneer de gebruiker opdrachten invoert via het toetsenbord. Met deze methode heeft u dan ook de beschikking over alle bewerkingen die een gebruiker kan uitvoeren via het toetsenbord.

Met de volgende instructie wordt bijvoorbeeld de tekenreeks "ABC" verzonden alsof deze direct op het toetsenbord is getypt.

```
ToetsenZenden Tekenreeks := "ABC"
```

Wanneer de instructie **ToetsenZenden** op deze manier wordt gebruikt, worden de toetsaanslagen pas verzonden als de procedure *niet* meer actief is. Dat wil zeggen: als geen enkele Visual Basic-code meer wordt uitgevoerd of als de instructie **SysActies** wordt aangeroepen. U kunt er echter voor zorgen dat de toetsaanslagen onmiddellijk worden verzonden, zoals u in het volgende voorbeeld kunt zien.

```
ToetsenZenden Tekenreeks := "ABC"; Wachten := Waar
```

Als u de waarde **Waar** doorgeeft voor het optionele tweede argument van de aanroep voor **ToetsenZenden**, wordt Visual Basic tijdelijk onderbroken en worden de toetsaanslagen verwerkt voordat de rest van de code wordt uitgevoerd.

De toetsaanslagen worden doorgegeven aan de actieve toepassing. Als de actieve toepassing niet de toepassing is waarnaar u de toetsaanslagen wilt verzenden, moet u eerst naar de gewenste toepassing overschakelen.

Overschakelen naar een andere toepassing

U kunt alleen toetsaanslagen naar de actieve toepassing verzenden. Als u geen andere toepassing activeert, verzendt de procedure de toetsaanslagen naar zichzelf. U kunt een andere toepassing activeren met de instructie **ToepassingActiveren**. Als bijvoorbeeld de Windows-toepassing Terminal reeds is gestart, activeert u deze met de volgende instructie.

```
ToepassingActiveren Titel := "Terminal"
```

Wanneer de toepassing nog niet is gestart, moet u deze eerst starten met de functie **Starten** voordat u de toepassing kunt activeren.

Speciale toetsen opgeven

Als u toetsaanslagen wilt verzenden die u niet gewoon als een tekenreeks kunt typen, zoals ENTER, F1 en TAB, plaatst u de naam van de toets tussen accolades ({}). Als u bijvoorbeeld de toetsaanslagen TAB, F1 en ENTER wilt verzenden, gebruikt u de volgende instructie.

```
ToetsenZenden Tekenreeks := "{TAB}{F1}{enter}"
```

U hoeft bij het invoeren van deze toetsnamen geen onderscheid te maken tussen hoofdletters en kleine letters. Zie de instructie *ToetsenZenden* in de online Visual Basic Naslaggids voor een lijst met de namen van deze speciale toetsen.

Opmerking U kunt geen toetsaanslagen verzenden die interrupts genereren in plaats van tekencodes, zoals ALT+CTRL+DEL en PRINT SCREEN.

Werken vanuit andere toepassingen

In dit hoofdstuk is tot nu toe alleen aandacht besteed aan de communicatie met andere toepassingen vanuit Microsoft Excel. In dit gedeelte wordt uitgelegd hoe u met Microsoft Excel kunt werken vanuit andere toepassingen. Een groot aantal van deze toepassingen ondersteunt dezelfde mechanismen als Microsoft Excel.

In de volgende tabel ziet u een overzicht van de manieren waarop u met Microsoft Excel kunt werken vanuit een aantal andere Microsoft-toepassingen.

Toepassing	OLE	Koppelen en insluiten	DDE	Toetsaanslagen verzenden
Microsoft Access versie 1.0	Nee	Ja	Ja	Ja
Microsoft FoxPro versie 2.5	Nee	Nee	Ja	Nee
Microsoft Mail versie 3.2	Nee	Ja	Nee	Nee
Microsoft PowerPoint® versie 3.0	Nee	Ja	Nee	Nee
Microsoft Project versie 4.0	Ja	Ja	Ja	Ja
Microsoft Publisher versie 2.0	Nee	Ja	Nee	Nee
Microsoft Visual Basic versie 3.0	Ja	Ja	Ja	Ja
Microsoft Word versie 6.0	Nee	Ja	Ja	Ja
Microsoft Works versie 2.0	Nee	Nee	Nee	Nee

OLE

Wanneer u met andere toepassingen werkt die ook over Visual Basic for Applications beschikken, kunt u objecten maken en gebruiken zoals in dit boek is beschreven. Visual Basic-procedures die bijvoorbeeld in Microsoft Project zijn geschreven, kunnen gebruik maken van objecten uit Microsoft Excel volgens de in Excel voorgeschreven syntaxis.

Wanneer u met andere macrotalen werkt die OLE ondersteunen, kunt u zonder meer Microsoft Excel-objecten maken en gebruiken.

Microsoft Excel voegt de volgende Klassenamen toe aan het registratiebestand van uw systeem. U kunt gebruik maken van deze namen om objecten te benaderen via hulpprogramma's voor het programmeren, zoals het Programmeersysteem Microsoft Visual Basic voor Windows, versie 3.0.

Klassenaam	Geeft toegang tot
Excel.Toepassing	Een toepassingsobject
Excel.Blad	Een werkbladobject
Excel.Grafiek	Een grafiekobject

Zie de documentatie bij het Programmeersysteem Microsoft Visual Basic voor Windows, versie 3.0 voor meer informatie over het gebruik van Klassenamen met de functies **ObjectMaken** en **ObjectHalen**.

Koppelen en insluiten

Microsoft Excel beschikt over de volgende OLE-werkwoorden voor objecten die zijn gekoppeld met of ingesloten in andere toepassingen.

OLE-werkwoord	Doel
"Activeren"	Activeert een ingesloten object, zodat directe bewerking mogelijk is.
"Openen"	Opent de brontoepassing voor een gekoppeld object zodat bewerking ervan mogelijk is.

DDE

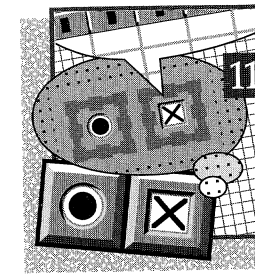
Microsoft Excel kent DDE-opdrachten voor elk van de functies van de Microsoft Excel 4.0-macrotaal. U kunt bijvoorbeeld de volgende code in WordBasic gebruiken om een werkmap van Microsoft Excel te openen. Zie de online Help voor macrofuncties voor meer informatie over de beschikbare opdrachten.

```
ExcelKanaal = DDEinitiate("Excel", "Systeem")
Opdracht$ = "[OPEN(" + Teken(34) + "C:\EXCEL\MIJNWRK.XLS" + Teken(34) +
")]"
DDEExecute(ExcelKanaal; Opdracht$)
```

In het volgende hoofdstuk wordt uitgelegd hoe u besturingselementen en dialoogvensters kunt gebruiken als een aanzet tot het maken van een aangepaste toepassing.

HOOFDSTUK 11

Besturingselementen en dialoogvensters



Bij het ontwerpen van aangepaste toepassingen in Microsoft Excel vormt de besturing van de interactie tussen de gebruiker en de toepassing een belangrijk aspect. Eerder in dit boek hebt u al gezien hoe u knoppen en andere grafische objecten in werkbladen kunt opnemen en macro's eraan kunt toewijzen. In dit hoofdstuk maakt u kennis met een aantal meer gevorderde manieren om informatie uit te wisselen tussen een toepassing en de gebruikers ervan. Dit kan met behulp van dialoogvensters die u ontwerpt en ondersteunt met Visual Basic-procedures.

Eerst zult u zien hoe u een toepassing met eenvoudige, van tevoren gedefinieerde dialoogvensters kunt uitbreiden. Verderop in dit hoofdstuk komt aan de orde hoe u een nieuw soort blad, het zogenaamde *dialoogblad*, in een werkblad kunt invoegen en hierin objecten zoals knoppen, Invoervakken en lijstvakken kunt opnemen. Dit stelt u in staat om aangepaste dialoogvensters te maken die net zo functioneren als de ingebouwde dialoogvensters van Microsoft Excel. Met behulp van de objecten in deze dialoogbladen, de zogenaamde *besturingselementen*, kunt u tevens uw eigen formulieren voor gegevensinvoer en wizards ontwerpen.

Met Visual Basic-programmacode kunt u de eigenschappen van besturingselementen instellen en wijzigen en groepen besturingselementen bijeenbrengen in een dialoogblad om zo aangepaste dialoogvensters te maken. Verder kunt u procedures maken die een aangepast dialoogvenster afbeelden en reageren op de informatie die door de gebruiker hierin wordt ingevoerd. Ten slotte kunt u procedures koppelen aan acties die in dialoogvensters worden uitgevoerd. U kunt bijvoorbeeld gebruik maken van een procedure om gegevens die in een InvoerVenster zijn opgegeven, te valideren zodra de gebruiker op ENTER drukt.

Inhoud

Een toepassing uitbreiden met vooraf gedefinieerde dialoogvensters

De makkelijkste manier om een dialoogvenster aan een toepassing toe te voegen is gebruik te maken van een vooraf gedefinieerd dialoogvenster. Een *vooraf gedefinieerd dialoogvenster* is een soort aangepast dialoogvenster dat u snel en gemakkelijk kunt ontwerpen en weergeven. U hebt echter slechts beperkt in de hand hoe dit venster eruitziet en welk type informatie u ermee van de gebruiker kunt opvragen of weergeven.

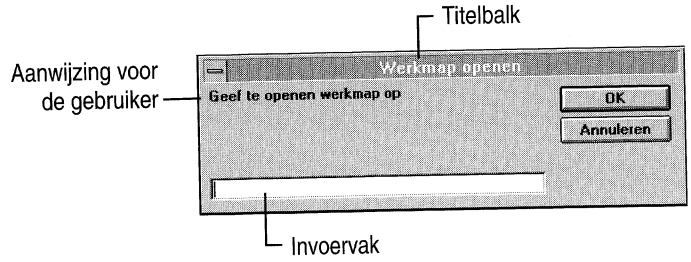
De volgende tabel geeft een overzicht van de beschikbare manieren om een vooraf gedefinieerd dialoogvenster aan een Visual Basic-toepassing toe te voegen.

Gebruik	Als u dit wilt doen
Functie **InvoerVenster**	Een aanwijzing in een dialoogvenster weergeven en de tekst die door de gebruiker wordt getypt, als resultaat geven.
Methode **InvoerVenster**	Een aanwijzing in een dialoogvenster weergeven en de informatie die door de gebruiker wordt ingevoerd, als resultaat geven. Deze methode lijkt op de functie **InvoerVenster**, maar biedt extra functionaliteit, bijvoorbeeld de mogelijkheid om te eisen dat de invoer van een bepaald gegevenstype is.
Functie **BerichtVenster**	Een bericht in een dialoogvenster weergeven en een waarde als resultaat geven die aangeeft welke opdrachtknop de gebruiker heeft gekozen. De resultaatwaarde kan worden genegeerd.

Informatie van de gebruiker opvragen

Met de functie **InvoerVenster** of de methode **InvoerVenster** kunt u informatie van de gebruiker vragen. In beide gevallen staan u alleen de bestaande componenten van het dialoogvenster ter beschikking. Het enige dat u kunt wijzigen, zijn de tekst in de titelbalk, de aanwijzing die de gebruiker te zien krijgt en de plaats van het dialoogvenster op het scherm. Als u een dialoogvenster met meer mogelijkheden wilt maken, moet u een dialoogblad gebruiken. Zie "Besturingselementen op een werkblad of dialoogblad plaatsen" verderop in dit hoofdstuk voor meer informatie over het maken van aangepaste dialoogvensters.

De functie **InvoerVenster** geeft een dialoogvenster weer waarin de gebruiker wordt gevraagd een tekenreeks of een **Variant**-variabele in te voeren. Het volgende dialoogvenster bevat bijvoorbeeld een InvoerVenster waarin de gebruiker de naam moet typen van de te openen werkmap.



Het volgende codevoorbeeld laat zien hoe u dit dialoogvenster kunt weergeven.

```
Sub MijnWerkmapOpenen()
    MijnMap = InvoerVenster("Geef te openen werkmap op"; "Werkmap
openen")
    Werkmappen.Openen Bestandsnaam:=MijnMap
Einde Sub
```

Raadpleeg *InvoerVenster* in de online Visual Basic Naslaggids voor meer informatie over de functie **InvoerVenster** en een overzicht van de volledige syntaxis.

Microsoft Excel kent ook de methode **InvoerVenster** van het toepassingsobject, die meer mogelijkheden heeft dan de functie **InvoerVenster**. Deze methode is vooral nuttig als u wilt bepalen welk type informatie de gebruiker kan invoeren. Raadpleeg *InvoerVenster (methode)* in de online Visual Basic Naslaggids voor meer informatie over de methode **InvoerVenster** en een overzicht van de volledige syntaxis.

Het volgende codevoorbeeld maakt gebruik van de methode **InvoerVenster** en vraagt de gebruiker een bereik op te geven waarin moet worden gezocht, alsmede de waarde waarnaar moet worden gezocht. De waarde moet een getal zijn.

```
Sub WaardenTellen()
    Aantal = 0
    Toewijzen ZoekBereik = Toepassing.InvoerVenster( _
        Aanwijzing:="Geef bereik voor zoeken op"; _
        Type:=8)
    ZoekWaarde = Toepassing.InvoerVenster(_
        Aanwijzing:=("Geef een zoekwaarde op")_
        Type:=1)
    Voor Elke AfzonderlijkeCel In ZoekBereik
        Indien AfzonderlijkeCel.Waarde = ZoekWaarde Dan
            Aantal = Aantal + 1
        Einde Indien
    Volgende AfzonderlijkeCel
    Berichtvak CStr(Aantal)
Einde Sub
```

U kunt ook aangepaste schermhulp afbeelden voor het dialoogvenster dat door de functie **InvoerVenster** of de methode **InvoerVenster** wordt weergegeven. Hieronder wordt een voorbeeld gegeven voor de functie **InvoerVenster**.

```
' Gebruikt HelpBestand en Context. De Help-knop wordt
' automatisch toegevoegd.
Waarde = InvoerVenster(Aanwijzing:="Welk item wilt u wissen?"; _
        HelpBestand:="DEMO.HLP"; Context:=10)
```

Raadpleeg *InvoerVenster* in de online Visual Basic Naslaggids voor meer informatie over het weergeven van aangepaste schermhulp voor een dialoogvenster. Meer informatie vindt u ook in Bijlage C, "Aangepaste Help-onderwerpen weergeven".

Informatie weergeven met de functie InvoerVenster

Het kan voorkomen dat u een kort bericht wilt weergeven, zoals een foutbericht of een waarschuwing. Met de functie **BerichtVenster** kunt u dergelijke berichten in een dialoogvenster laten afbeelden. Nadat de gebruiker het bericht heeft gelezen, kan deze een knop kiezen om het dialoogvenster te sluiten.

Met de volgende programmacode kunt u het dialoogvenster van de vorige figuur weergeven.

```
BerichtVenster "Fout gevonden bij het openen van het bestand, " & _
    "probeer het opnieuw."; vbWaarschuwing; "Tekst-editor"
```

Opmerking De functie **BerichtVenster** geeft als resultaat een waarde die aangeeft welke knop de gebruiker heeft gekozen. Voor sommige dialoogvensters is het mogelijk dat er geen keuze is of dat het niet van belang is welke knop wordt gekozen. In deze gevallen kunt u de resultaatwaarde negeren. De functie **BerichtVenster** werkt dan als een instructie en niet als een functie, en daarom staan de argumenten ook niet tussen haakjes. Raadpleeg *BerichtVenster (functie)* in de online Visual Basic Naslaggids voor meer informatie over de functie **BerichtVenster**.

U kunt met de functie **BerichtVenster** uitgebreide dialoogvensters maken met uiteenlopende pictogrammen en knoppen. Hiervoor voegt u ingebouwde constanten toe en gebruikt u de som als het argument *knoppen* van de functie **BerichtVenster**. U kunt ook vaststellen welke knop een gebruiker in een dialoogvenster heeft gekozen, aan de hand van de resultaatwaarde van de functie **BerichtVenster**.

In het volgende voorbeeld wordt gebruik gemaakt van de functie **BerichtVenster** om een foutbericht weer te geven in een dialoogvenster met een knop Ja en een knop Nee. (De ernst van het foutbericht wordt aangegeven door het pictogram Stop dat u in het dialoogvenster kunt opnemen). De knop Nee is het standaardantwoord. De variabele `DialoogStijl` bevat de som van de ingebouwde constanten die de voorzieningen van het dialoogvenster definiëren (het argument *knoppen*). De functie **BerichtVenster** geeft als resultaat een waarde die afhangt van de knop die de gebruiker heeft gekozen.

```
Sub VoorbeeldVakMaken()
    'Voorbeeld van een berichtvenster met alle bijbehorende componenten
    Bericht = "Dit is een voorbeeld van een foutbericht."
    Bericht = Bericht & " Wilt u doorgaan?"
    'Beschrijving van het dialoogvenster.
    DialoogStijl = vbJaNee + vbNoodsignaal + vbStandaardKnop2
    Titel = "Voorbeeld van een berichtvenster"
    'Antwoord van de gebruiker opvragen
    Antwoord = BerichtVenster(Bericht; DialoogStijl; Titel)
    Indien Antwoord = vbJa Dan                    'Antwoord controleren
        Bericht = "U hebt Ja gekozen."            'en bijbehorende actie
    Anders                                        'ondernemen.
        Bericht = "U hebt Nee gekozen of op Enter gedrukt."
    Einde Indien
    BerichtVenster Bericht                        'Actie weergeven.
Einde Sub
```

Opmerking De constanten vbJaNee, vbNoodsignaal en dergelijke zijn in Visual Basic gedefinieerd. Raadpleeg *BerichtVenster (functie)* in de online Visual Basic Naslaggids voor een lijst van deze constanten.

Zoals u ziet, worden de knoppen en het pictogram voor het foutbericht gemaakt door de constanten bij elkaar op te tellen en de som op te slaan in de variabele `DialoogStijl`. Aangezien `DialoogStijl` wordt gebruikt als een argument voor de functie **BerichtVenster** in de regel `Antwoord = BerichtVenster(Bericht; DialoogStijl; Titel)`, wordt het dialoogvenster weergegeven in de bijbehorende stijl.

Voor elk dialoogvenster dat door de functie **BerichtVenster** wordt weergegeven, kunt u tevens een knop Help en aangepaste schermhulp opnemen. Zie Bijlage C, "Aangepaste Help-onderwerpen weergeven", voor meer informatie over aangepaste schermhulp.

Besturingselementen op een werkblad of dialoogblad plaatsen

Eerder in dit boek is aan de orde geweest hoe u een knop of een ander object (zoals een rechthoek of een figuur) in een werkblad opneemt en een macro toewijst die wordt gestart telkens wanneer de gebruiker op het object klikt. Zie hoofdstuk 13, "Grafische objecten in werkbladen en grafieken maken", in het *Microsoft Excel handboek* voor meer informatie over het opnemen van knoppen en andere grafische objecten in werkbladen.

Als u een besturingselement wilt toevoegen aan een aangepaste toepassing, begint u er eveneens mee dit element in een werkblad of dialoogblad op te nemen. U kunt besturingselementen opnemen in werkbladen, grafiekbladen en in aangepaste dialoogvensters, maar niet in een grafiek of een Visual Basic-module.

Een van de voordelen van het opnemen van besturingselementen in een werkblad is dat u de elementen op deze manier dicht bij de gegevens kunt plaatsen waarop ze betrekking hebben. Zo kunt u een reeks aankruisvakjes opnemen in een werkblad dat een aangifteformulier voor de inkomstenbelasting nabootst en vervolgens een aantal procedures aan het blad koppelen die toegang verschaffen tot belastingtabellen, inkomensoverzichten en dergelijke. Een ander voordeel van het opnemen van besturingselementen in een werkblad is dat u een document kunt maken dat eruitziet als een dialoogvenster, maar cellen bevat die automatisch opnieuw worden berekend.

Door groepen besturingselementen bijeen te brengen in dialoogvensters, kunt u anderzijds voorkomen dat de gebruiker verward raakt in een complex systeem van werkbladen en procedures die u hebt ontworpen. Zo kunt u de gebruikers-interface voor uw toepassing overzichtelijk houden, zodat de gebruiker niet wordt overstelpt met opties die hij vrijwel nooit gebruikt. U kunt het proces van gegevensinvoer gescheiden houden van het overbrengen van gegevens naar een werkblad, om het aantal fouten in het werkblad tot een minimum te beperken. Bovendien kunt u ook dialoogvensters maken die betrekking hebben op een groot aantal werkbladen of grafieken, of op de Microsoft Excel-omgeving in zijn geheel.

Aangepaste besturingselementen gebruiken in een toepassing

Nadat u een besturingselement hebt opgenomen, bepaalt u de standaardinstellingen van de eigenschappen ervan, bijvoorbeeld of een aankruisvakje standaard is ingeschakeld en of het formaat van een knop moet worden gewijzigd wanneer het formaat van de onderliggende cel in een werkblad verandert.

Vervolgens kunt u een procedure toewijzen aan de gebeurtenis die is gekoppeld aan een besturingselement. Gebeurtenissen treden op bij een gebruikersactie, bijvoorbeeld zodra de gebruiker een knop, een aankruisvakje of een keuzerondje kiest, of wanneer de gebruiker de tekst in een InvoerVenster bewerkt.

U kunt een besturingselement ook rechtstreeks koppelen aan een cel in een werkblad, zonder procedures te gebruiken. Op deze manier kunt u de gebruikersinterface voor een werkblad vereenvoudigen, zodat de gebruiker geen gegevens meer in een cel hoeft te typen om bepaalde opties in te stellen, maar dit met de muis kan doen. U kunt bijvoorbeeld boven aan een werkblad een aantal aankruisvakjes opnemen die opties in een haalbaarheidsstudie vertegenwoordigen en elk aankruisvakje vervolgens koppelen aan een cel die elders in het werkblad wordt gebruikt. Als een gebruiker een van de aankruisvakjes selecteert, krijgt de overeenkomstige cel in het werkblad de waarde **Waar** en wordt het werkblad opnieuw berekend. Zie "Besturingselementen koppelen aan werkbladen" verderop in dit hoofdstuk voor meer informatie over het koppelen van besturingselementen aan cellen.

Ten slotte kunt u de onderlinge samenwerking van besturingselementen coördineren door de bijbehorende dialoogvensters weer te geven of te verwijderen, te controleren of de gegevens die in het dialoogvenster worden ingevoerd correct zijn en deze gegevens beschikbaar te maken voor werkbladen en andere procedures in de Microsoft Excel-omgeving. Zie "Een aangepast dialoogvenster weergeven" verderop in dit hoofdstuk voor meer informatie over het weergeven van dialoogvensters.

Een dialoogblad maken

Als u een aangepast dialoogvenster wilt maken, moet u eerst een nieuw dialoogblad invoegen.

▶ Een nieuw dialoogblad invoegen

1. Kies de opdracht **Macro** in het menu **Invoegen** en vervolgens de opdracht **Dialoogblad**.

   Microsoft Excel voegt een nieuw dialoogblad in vóór het huidige blad in de actieve werkmap. Het nieuwe blad bevat een kader van een dialoogvenster met daarin de knoppen OK en Annuleren.

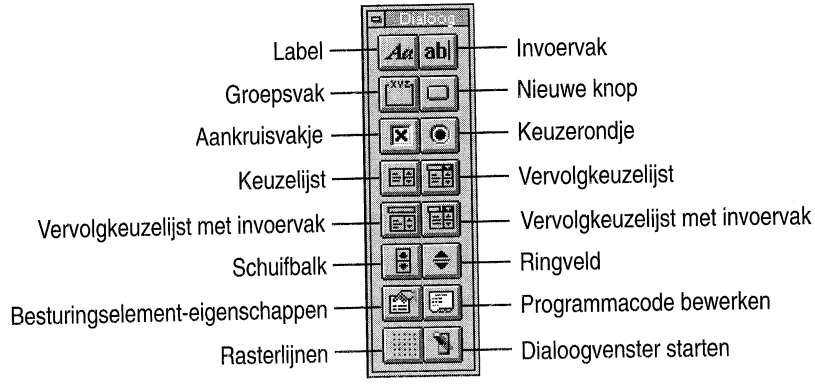
2. Als u de naam boven het dialoogvenster wilt wijzigen, kiest u de tekst boven aan het kader van het dialoogvenster en typt u een nieuwe naam.
3. Als u de afmetingen van het dialoogvenster wilt aanpassen, sleept u de grepen totdat het kader het gewenste formaat heeft.

Opmerking Hoewel u de positie van het kader van een dialoogvenster in het bijbehorende dialoogblad kunt wijzigen, wordt het resulterende dialoogvenster steeds weergegeven op dezelfde positie als het laatst weergegeven dialoogvenster, ongeacht of dat een ingebouwd of een aangepast dialoogvenster was. U kunt er echter voor zorgen dat een aangepast dialoogvenster op een specifieke positie verschijnt door een procedure te schrijven die de positie van het dialoogvenster bepaalt alvorens dit weer te geven. Dit komt aan de orde in "Een aangepast dialoogvenster weergeven" verderop in dit hoofdstuk.

U kunt bekijken hoe een dialoogvenster tijdens de uitvoering op het scherm zal verschijnen door de knop Dialoogvenster starten op de werkbalk Dialoog te kiezen. Dit heeft tot gevolg dat het dialoogvenster actief wordt. Nu kunt u bijvoorbeeld gegevens typen in Invoervakken en als u een knop in het dialoogvenster kiest, zal de bijbehorende procedure worden gestart.

Besturingselementen plaatsen

Als u eenmaal een nieuw dialoogblad hebt gemaakt, kunt u met de werkbalk Dialoog nieuwe besturingselementen in het kader van het dialoogvenster plaatsen. U kunt de besturingselementen ook rechtstreeks in een werkblad plaatsen.



▶ Een besturingselement plaatsen in een werkblad of een dialoogblad

1. Kies op de werkbalk Dialoog de knop die correspondeert met het besturingselement dat u wilt toevoegen.
2. Sleep het besturingselement binnen het werkblad of het dialoogblad totdat het element het formaat en de vorm heeft die u wenst.

De volgende tabel geeft een overzicht van de besturingselementen die door de overeenkomstige knoppen op de werkbalk Dialoog worden gemaakt.

Naam knop	Beschrijving besturingselement
Label	Tekst die voor de gebruiker bestemd is, waaronder namen, aanwijzingen en waarschuwingen
Invoervak	Een vak waarin de gebruiker tekst, getallen of celverwijzingen kan invoeren
Groepsvak	Een kader dat een groep knoppen en andere besturingselementen bevat.
Nieuwe knop	Een opdrachtknop, zoals OK of de knop Annuleren
Aankruisvakje	Een vakje dat aangeeft of een optie is ingesteld, ongeacht de status van andere opties in het dialoogvenster
Keuzerondje	Een rondje waarmee u één optie kunt selecteren van een groep opties die elkaar onderling uitsluiten. Plaats een serie bij elkaar horende keuzerondjes samen in een groepsvak.
Keuzelijst	Een lijst met tekenreeksen, waarvan er één of meerdere kunnen worden geselecteerd
Vervolgkeuzelijst	Een tekstvak dat niet kan worden bewerkt, gekoppeld aan een vervolgkeuzelijst die verschijnt als de gebruiker een pijl naast het tekstvak selecteert
Keuzelijst met Invoervak	Een tekstvak dat kan worden bewerkt, gecombineerd met een vervolgkeuzelijst.
Vervolgkeuzelijst met Invoervak	Een leeg Invoervak, gekoppeld aan een vervolgkeuzelijst die verschijnt als de gebruiker een pijl naast het Invoervak selecteert
Schuifbalk	Een horizontale of verticale schuifbalk voor het wijzigen van numerieke waarden. Sleep horizontaal om een horizontale schuifbalk te maken
Ringveld	Een knoppenpaar voor het verhogen of verlagen van een weergegeven waarde

De werkbalk Dialoog bevat daarnaast nog een viertal knoppen om te werken met besturingselementen en dialoogvensters.

Naam knop	Beschrijving
Besturingselement-eigenschappen	Hiermee kunt u de eigenschappen van het geselecteerde object bekijken of wijzigen. Dit komt overeen met het kiezen van de opdracht **Objecteigenschappen** in het menu **Opmaak**.
Programmacode bewerken	Hiermee kunt u programmacode die is toegewezen aan het geselecteerde object, maken of bewerken.
Rasterlijnen	Hiermee kunt u besturingselementen uitlijnen op rasterlijnen.
Dialoogvenster starten	Hiermee kunt u bekijken hoe het dialoogvenster wordt weergegeven wanneer het wordt gestart.

Opmerking Met de werkbalk Tekenen kunt u ook standaardfiguren opnemen in een dialoogvenster. U kunt bijvoorbeeld een bitmap-figuur of een ander grafisch object, zoals een bedrijfslogo, in een dialoogvenster opnemen en vervolgens een procedure toewijzen die wordt gestart wanneer de gebruiker op het object klikt. Kies de opdracht **Werkbalken** in het menu **Beeld** om een verborgen werkbalk weer te geven.

Besturingselementen selecteren

Wanneer u een besturingselement in een werkblad of een dialoogvenster kiest, heeft dit gewoonlijk hetzelfde resultaat als u verwacht van besturingselementen in een ingebouwd dialoogvenster. Als u bijvoorbeeld een knop in een werkblad kiest, wordt deze ingedrukt weergegeven, en als u een aankruisvakje selecteert, verschijnt er een kruis in het vakje. Als er een procedure is verbonden aan een besturingselement, start Microsoft Excel deze procedure. Als een besturingselement is gekoppeld aan een cel in een werkblad, heeft een wijziging van de inhoud of de status van het besturingselement tot gevolg dat de overeenkomstige waarde wordt overgebracht naar het werkblad.

Als een besturingselement is geselecteerd, kunt u de eigenschappen ervan wijzigen, het besturingselement verplaatsen in het werkblad of een procedure toewijzen aan een van de gebeurtenissen die het besturingselement ondersteunt.

▶ **Een besturingselement selecteren in een werkblad**

- Houd CTRL ingedrukt in Microsoft Excel voor Windows, of houd COMMAND ingedrukt in Microsoft Excel voor de Macintosh, en kies het besturingselement.

  U kunt ook de knop Selectie op de werkbalk Tekenen kiezen en vervolgens een rechthoek trekken om het gewenste besturingselement. Als de werkbalk verborgen is, kiest u de opdracht **Werkbalken** in het menu **Beeld** om deze weer te geven.

Opmerking Als u een groep besturingselementen in een dialoogblad wilt selecteren, trekt u een rechthoek die de gewenste besturingselementen omsluit. U kunt ook besturingselementen toevoegen aan een reeds geselecteerde groep elementen, door SHIFT ingedrukt te houden en aanvullende besturingselementen te selecteren.

Besturingselementen verplaatsen, wissen en het formaat ervan wijzigen

Als u de afmetingen van een besturingselement wilt wijzigen, selecteert u dit en sleept u een van de grepen ervan. Als u de positie van het element wilt wijzigen, sleept u een van de randen ervan. Rasterlijnen in het dialoogblad zorgen ervoor dat bij verplaatsingen van het besturingselement steeds een bepaald interval wordt aangehouden (de grootte van dit interval hangt af van het type van uw beeldscherm). Dit verschijnsel staat bekend als *vastzetten op rasterlijn.*

▶ **Vastzetten op rasterlijn inschakelen en uitschakelen**

- Kies de knop Rasterlijnen op de werkbalk Dialoog.

U kunt de voorziening voor het vastzetten op rasterlijnen tijdelijk buiten werking stellen door ALT (in Microsoft Excel voor Windows) of COMMAND (in Microsoft Excel voor de Macintosh) ingedrukt te houden terwijl u een greep, kader of besturingselement sleept. Als u ALT ingedrukt houdt terwijl u in een werkblad sleept, bedragen verplaatsingen steeds een veelvoud van een celbreedte of celhoogte.

▶ **Een besturingselement verwijderen**

1. Selecteer het besturingselement.
2. Druk op DEL.

   U kunt ook de opdracht **Wissen** in het menu **Bewerken** kiezen en vervolgens **Alles**.

De eigenschappen van een besturingselement instellen

Als u een besturingselement maakt, wijst Microsoft Excel hieraan een reeks standaardeigenschappen toe, die lijken op de eigenschappen die u in de ingebouwde dialoogvensters vindt. Sommige van deze eigenschappen kunt u wijzigen met de opdracht Objecteigenschappen in het menu **Opmaak**. Andere eigenschappen kunnen alleen worden gewijzigd door een Visual Basic-procedure uit te voeren. Zie "Programmacode toewijzen aan besturingselementen en dialoogvensters" verderop in dit hoofdstuk voor meer informatie over het wijzigen van de eigenschappen van een besturingselement.

▶ **Een eigenschap van een besturingselement weergeven of wijzigen**

1. Selecteer het besturingselement.
2. Kies .de opdracht Objecteigenschappen in het menu **Opmaak**.
3. Selecteer een tabblad en bekijk de instelling van de gewenste eigenschap of wijzig deze.
4. Kies de knop OK.

Een tabvolgorde toewijzen aan besturingselementen

Er zijn verschillende manieren waarop een gebruiker een besturingselement in een dialoogvenster kan selecteren. Eén manier bestaat erin dat gewoon wordt geklikt op het besturingselement, zoals een knop, een Invoervak of een keuzelijst. Een andere manier bestaat erin dat ALT (in Microsoft Excel voor Windows) of COMMAND (in Microsoft Excel voor de Macintosh) wordt ingedrukt samen met de toegangstoets voor het besturingselement (hierop wordt nader ingegaan in het volgende gedeelte van dit hoofdstuk).

U kunt ook een tabvolgorde instellen voor de besturingselementen in een dialoogvenster. Wanneer de gebruiker op TAB drukt, worden de besturingselementen actief in de door u ingestelde *tabvolgorde*. Een logische tabvolgorde is vooral van belang in aangepaste formulieren voor gegevensinvoer, zodat het aantal toetsaanslagen dat nodig is om de verschillende Invoervakken te doorlopen, tot een minimum beperkt kan blijven. Een tabvolgorde kan alleen worden ingesteld voor besturingselementen in een dialoogvenster, niet in een werkblad.

▶ **De tabvolgorde van besturingselementen wijzigen**

1. Schakel naar het dialoogblad.
2. Kies de opdracht **Tabvolgorde** in het menu **Extra**.
3. Selecteer het element waarvan u de plaats in de tabvolgorde wilt wijzigen.

   Als u meer dan één besturingselement wilt selecteren, houdt u tijdens het selecteren CTRL ingedrukt.
4. Kies de knop Verplaatsen omhoog of Verplaatsen omlaag.
5. Kies de knop OK.

Een toegangstoets toewijzen aan een besturingselement

Een toegangstoets is de toets die u samen met ALT (in Microsoft Excel voor Windows) of COMMAND (in Microsoft Excel voor de Macintosh) indrukt om een besturingselement te selecteren, gelijkwaardig met het kiezen van een knop. Net als bij het instellen van de tabvolgorde, kunnen toegangstoetsen ook alleen worden gedefinieerd voor besturingselementen in een dialoogvenster. U kunt een toegangstoets toewijzen aan een label, een opdrachtknop, een aankruisvakje, een keuzerondje of een groepsvak. De toewijzing houdt tevens in dat één van de tekens van de tekst die aan het besturingselement is gekoppeld, onderstreept wordt weergegeven in het dialoogvenster.

▶ **Een toegangstoets voor een besturingselement instellen**

1. Selecteer het besturingselement.
2. Kies .de opdracht Objecteigenschappen in het menu **Opmaak**.
3. Selecteer de tab Besturingselement.
4. Typ in het vak "Toegangstoets" de toets die u als toegangstoets wilt gebruiken.
5. Kies de knop OK.

Opmerking Als u het effect van een toegangstoets wilt instellen voor een besturingselement in een werkblad, kunt u een procedure ontwikkelen die is gekoppeld aan de gebeurtenis **BijToets**, en die wordt gestart zodra een gebruiker een bepaalde toetscombinatie indrukt. In hoofdstuk 13, "Automatische procedures en invoegmacro's maken", wordt nader ingegaan op het maken van **BijToets**-procedures.

Besturingselementen koppelen aan werkbladen

U kunt de waarde van bepaalde besturingselementen koppelen aan een cel in een werkblad, zonder dat u een aparte procedure hoeft te ontwikkelen die vervolgens aan een besturingselement wordt toegewezen. Zodra de gebruiker de waarde in één van beide lokaties wijzigt, wordt de waarde op beide plaatsen gewijzigd. U kunt bijvoorbeeld in een werkblad een reeks aankruisvakjes opnemen die globale opties voor een hele werkmap instellen, waarbij elk aankruisvakje is gekoppeld aan een cel in een verschillend werkblad. Als een gebruiker een aankruisvakje selecteert, verschijnt in de gekoppelde cel de waarde "WAAR".

Cellen in een werkblad kunnen worden gekoppeld aan aankruisvakjes, keuzerondjes, keuzelijsten, schuifbalken en ringvelden. De ontstane koppeling is een eigenschap van het besturingselement, maar niet van de gekoppelde cel.

U kunt een bepaalde cel aan meerdere besturingselementen koppelen. Elk besturingselement kan echter slechts aan één cel gekoppeld zijn, en de koppeling werkt alleen als deze cel zich bevindt in een geopend blad. Als u de waarde van een besturingselement wilt overbrengen naar meer dan één cel, koppelt u het besturingselement aan een van de cellen en verwijst u vervolgens naar deze cel in elke andere cel waarin u de betreffende waarde wilt gebruiken.

U kunt koppelingen instellen voor besturingselementen die u in een dialoogvenster hebt opgenomen. Houd er echter rekening mee dat koppelingen altijd actief zijn. Dit betekent dat bij elke wijziging van de status van het besturingselement tevens de waarde in de gekoppelde cel wordt gewijzigd, zelfs als de gebruiker de knop Annuleren kiest om het dialoogvenster te sluiten. Daarom is het meestal beter gebruik te maken van een procedure om informatie van een dialoogvenster over te brengen naar de cellen in een werkblad.

Wanneer u een besturingselement aan een cel in een werkblad koppelt, wordt het werkblad opnieuw berekend zodra de gebruiker een waarde in de cel invoert of de status van het besturingselement wijzigt. Als een aankruisvakje bijvoorbeeld is gekoppeld aan een cel en het aankruisvakje is ingeschakeld, is de waarde in de cel **Waar**. Bij elke wijziging van deze waarde worden de formules die verwijzen naar de cel opnieuw berekend.

▶ **Een besturingselement koppelen aan een cel in een werkblad**

1. Selecteer het besturingselement.
2. Kies de opdracht **Objecteigenschappen** in het menu **Opmaak**.
3. Selecteer de tab Besturingselement.
4. Typ in het vak "Koppeling met selectie" de naam of de verwijzing van de cel waarmee u een koppeling wilt instellen.

   U kunt ook op de cel klikken om de bijbehorende verwijzing in het vak in te voeren.
5. Geef aanvullende informatie op met betrekking tot het besturingselement dat u wilt koppelen, zoals in de volgende gedeelten van dit hoofdstuk wordt beschreven.
6. Kies de knop OK.

Opmerking U kunt een koppeling ook instellen door het besturingselement in het werkblad of dialoogblad te selecteren en vervolgens in de formulebalk de verwijzing te typen van de cel waarmee u een koppeling wilt instellen.

U kunt een besturingselement aan een cel koppelen en een procedure toewijzen aan hetzelfde besturingselement. In zo'n geval wordt de koppeling bijgewerkt voordat de procedure start. In "Programmacode toewijzen aan besturingselementen en dialoogvensters" verderop in dit hoofdstuk wordt beschreven hoe u een procedure aan een besturingselement toewijst.

Aankruisvakjes

De koppeling tussen een cel en een aankruisvakje werkt in beide richtingen (*bidirectionele koppeling*). Zodra de waarde in de cel of het aankruisvakje verandert, wordt ook de waarde op de corresponderende plaats gewijzigd. Het aankruisvakje kan ingeschakeld, uitgeschakeld of grijs gekleurd verschijnen. Het aankruisvakje kan eveneens worden ingeschakeld door de waarde **Waar** of een getal (behalve nul) in de gekoppelde cel te typen; het kan worden uitgeschakeld door de waarde **Onwaar** of **0** (nul) in de cel te typen. Het aankruisvakje wordt grijs gekleurd als u de waarde **#N/B** invoert.

Keuzerondjes

Net als aankruisvakjes, kunnen keuzerondjes wel of niet geselecteerd verschijnen. Wanneer u een van de keuzerondjes selecteert, wordt de selectie van de andere keuzerondjes in de groep geannuleerd. Als u een van de keuzerondjes in de groep koppelt, worden alle overige keuzerondjes in de groep eveneens gekoppeld. Ook hier kunt u keuzerondjes in- en uitschakelen door waarden in de gekoppelde cellen te typen. Wanneer u een getal in een gekoppelde cel invoert, wordt het corresponderende keuzerondje geselecteerd. Als het getal niet overeenkomt met een keuzerondje, wordt de selectie van alle keuzerondjes in de groep geannuleerd.

Als u meerdere keuzerondjes in één groepsvak bijeen brengt, is de resultaatwaarde die naar de gekoppelde cel wordt gezonden, gelijk aan het volgnummer van het geselecteerde keuzerondje zoals dat in het groepsvak verschijnt, van achteren naar voren geteld. (U kunt deze volgorde wijzigen met de opdracht **Tabvolgorde** in het menu **Extra**).

Als u de keuzerondjes niet omgeeft door een groepsvak, worden alle keuzerondjes in het werkblad of het dialoogvenster behandeld als één groep.

Keuzelijsten

Voor keuzelijsten geeft u een cellenbereik in het werkblad op dat de lijst bevat van de elementen die in de keuzelijst moeten verschijnen. Daarnaast kunt u de verwijzing opgeven van één enkele cel, die het volgnummer bevat van het element dat op een bepaald moment in de keuzelijst is geselecteerd. Ook hier kunt u een getal invoeren in de gekoppelde cel om het corresponderende element in de keuzelijst te selecteren.

Formules die afhankelijk zijn van de gekoppelde cel, worden opnieuw berekend zodra een ander element wordt geselecteerd. Dit is nuttig als u "wat-als"-modellen wilt maken waarin de gebruiker kan zien hoe de resultaten veranderen naar gelang verschillende elementen in de lijst worden geselecteerd.

Schuifbalken

Schuifbalken ondersteunen een gekoppelde cel waarin de positie van het schuifblokje op de schuifbalk als een getal is vastgelegd. Met de opdracht Objecteigenschappen in het menu **Opmaak** kunt u het dialoogvenster **Objecteigenschappen** oproepen, waarin u de begin- en eindwaarde kunt opgeven, alsmede een getal dat aangeeft hoeveel het schuifblokje wordt verplaatst elke keer wanneer de gebruiker op de pijlen of de schuifbalk klikt. Telkens wanneer het getal in de gekoppelde cel wordt gewijzigd, verandert de positie van het schuifblokje.

Ringvelden

Ringvelden lijken op schuifbalken, met dit verschil dat de gebruiker alleen op de pijlen kan klikken. Er is geen balk beschikbaar. De waarde in de gekoppelde cel wordt hierdoor telkens gewijzigd met het interval dat is opgegeven in het dialoogvenster **Objecteigenschappen**. Als u de muisknop langere tijd ingedrukt houdt, wordt de waarde in de gekoppelde cel voortdurend gewijzigd. Het werkblad wordt bij elke wijziging opnieuw berekend.

Programmacode toewijzen aan besturingselementen en dialoogvensters

Nadat u besturingselementen in een werkblad of een dialoogvenster hebt opgenomen en de beginwaarden van de eigenschappen ervan hebt vastgelegd, moet u een reeks procedures ontwerpen die het dialoogvenster ondersteunen.

Om te beginnen wordt het aangepaste dialoogvenster op het scherm weergegeven, zodat de gebruiker wijzigingen kan aanbrengen in de besturingselementen ervan. Door u geschreven procedures hebben toegang tot de status van de besturingselementen via de eigenschappen ervan en kunnen de reactie van de toepassing op wijzigingen in het dialoogvenster definiëren. U kunt ingevoerde gegevens laten valideren, het gedrag van besturingselementen of het dialoogvenster wijzigen terwijl het dialoogvenster zichtbaar is of het dialoogvenster sluiten en de ingevoerde gegevens doorgeven aan een werkblad.

Net zoals u een procedure kunt toewijzen aan een knop of een grafisch object in een werkblad, kunt u ook een procedure toewijzen aan een besturingselement, met de eigenschap **BijActie**. In dat geval speekt men van een *gebeurtenisafhandelingsprocedure*. Als het besturingselement wordt bewerkt, voert Microsoft Excel de procedure uit die wordt toegewezen door de eigenschap **BijActie**. In de volgende tabel wordt een lijst van acties gegeven waarop elk besturingselement reageert.

Besturingselementen	Acties
Opdrachtknoppen, groepsvakken, aankruisvakjes, keuzerondjes	Het besturingselement wordt geactiveerd. Deze gebeurtenis wordt gebruikt voor besturingselementen die geen waarde hebben of slechts een eenvoudige waarde, zoals **Waar** of **Onwaar**.
Invoervakken, keuzelijsten, vervolgkeuzelijsten, keuzelijsten met Invoervak, vervolgkeuzelijsten met Invoervak, schuifbalken, ringvelden	Het besturingselement wordt door de gebruiker gewijzigd. Deze gebeurtenis wordt gebruikt voor besturingselementen die een complexe waarde ondersteunen.
Dialoogvensters	Het dialoogvenster verschijnt op het scherm.

U kunt bijvoorbeeld de eigenschap **BijActie** gebruiken om een procedure aan een bewerkingsvak toe te wijzen. De procedure kan de ingevoerde gegevens controleren, telekens wanneer de gebruiker een onderdeelnummer invoert of wijzigt, zodat er alleen gevalideerde gegevens als resultaat op het werkblad worden gezet en het dialoogvenster pas sluit als overal geldige informatie is ingevoerd.

▶ Een bestaande procedure koppelen aan een gebeurtenis die bij een besturingselement hoort

1. Selecteer het besturingselement.
2. Kies de opdracht **Macro toewijzen** in het menu **Extra**.

   U kunt ook met de rechtermuisknop op het besturingselement klikken om het snelmenu weer te geven en vervolgens de opdracht **Macro toewijzen** kiezen.
3. Typ of selecteer in het vak "Macronaam of verwijzing" de naam van de procedure die u aan de gebeurtenis wilt toewijzen.
4. Kies de knop OK.

U kunt ook een nieuwe procedure maken en deze koppelen aan de gebeurtenis die bij een besturingselement hoort.

▶ Een nieuwe procedure maken en deze koppelen aan de gebeurtenis die bij een besturingselement hoort

1. Selecteer het besturingselement.
2. Kies de knop Programmacode bewerken op de werkbalk Dialoog.

   Microsoft Excel opent een Visual Basic-module en geeft een leeg procedure-skelet weer.
3. Schrijf tussen de instructies **Sub** en **End Sub** de procedure die u aan de gebeurtenis wilt koppelen.

Als u bijvoorbeeld een gebeurtenisafhandelingsprocedure wilt schrijven voor een knop Annuleren, die het vierde object is in het kader van het dialoogvenster, maakt Microsoft Excel het skelet voor de procedure, waarna u tussen de instructies **Sub** en **End Sub** de volgende instructie kunt invoegen.

```
Sub Knop4_Klikken()
    BerichtVenster "Weet u zeker dat u wilt annuleren?"; vbJaNee
Einde Sub
```

Als de werkmap met het dialoogblad geen Visual Basic-modulen bevat, voegt Microsoft Excel een nieuwe module in en schakelt vervolgens naar het programmaskelet.

▶ **Een procedure bewerken die al eerder is gekoppeld aan de gebeurtenis die bij een besturingselement hoort**

1. Selecteer het besturingselement.
2. Kies de knop Programmacode bewerken op de werkbalk Dialoog.

   Microsoft Excel opent een Visual Basic-module en geeft de gekoppelde procedure weer. Als de gekoppelde programmacode is geschreven in de macrotaal van Microsoft Excel 4.0, schakelt Microsoft Excel naar het macroblad dat de gekoppelde macro bevat.
3. Bewerk de programmacode.

Een aangepast dialoogvenster weergeven

U kunt een dialoogvenster oproepen met de methode **Weergeven** in een procedure die is toegewezen aan een knop, een menu-opdracht of een gebeurtenis in de Microsoft Excel-omgeving. U kunt bijvoorbeeld de volgende procedure toewijzen aan een knop die zich in een werkblad bevindt.

```
Sub Knop4_Klikken()
    Dialoogbladen("MijnBestandOpenDlg").Weergeven
Einde Sub
```

Zodra de gebruiker de knop kiest, wordt de methode **Weergeven** voor het opgegeven dialoogvenster opgeroepen. Het dialoogvenster verschijnt op het scherm en de gebruiker kan werken met de bijbehorende besturingselementen: tekst in een Invoervak invoeren, een keuzerondje selecteren, elementen selecteren in een keuzelijst, enzovoort. Als een van deze besturingselementen is gekoppeld aan een gebeurtenisafhandelingsprocedure **BijActie**, wordt deze procedure uitgevoerd.

Als de gebruiker de knop OK of de knop Annuleren kiest en de eigenschap **DialoogSluiten** van de knop de waarde **Waar** heeft en tevens de eigenschap **KnopAnnuleren** de waarde **Onwaar** heeft, worden de volgende acties uitgevoerd.

- De inhoud van elk Invoervak wordt gevalideerd op basis van de eigenschap Bewerkingscontrole, zoals die is ingesteld in het tabblad Besturingselement van het dialoogvenster **Objecteigenschappen**.

  Als de inhoud van een van de Invoervakken niet geldig is, wordt een waarschuwingsbericht weergegeven. Tevens wordt de selectie verplaatst naar het betreffende Invoervak, zodat de gebruiker een correcte waarde kan opgeven. Dit wordt herhaald totdat elk Invoervak een geldige waarde bevat.
- Als er een afhandelingsprocedure **BijActie** aan de knop is toegewezen, wordt deze uitgevoerd.
- Het dialoogvenster verdwijnt van het scherm en de uitvoering van de toepassing gaat verder met de eerste instructie na de instructie die de methode **Weergeven** gebruikte.

Gewoonlijk wilt u de wijzigingen die in het dialoogvenster zijn aangebracht, alleen verwerken als de gebruiker het dialoogvenster sluit door OK te kiezen. In dat geval zal de methode **Weergeven** als resultaat **Waar** geven.

Stel dat u een knop met de naam Knop 4 in een werkblad hebt opgenomen. Als de gebruiker een werkmap wil openen, kiest deze Knop4 om een dialoogvenster weer te geven in een dialoogblad met de naam MijnBestandOpenDlg. De procedure die aan de knop is toegewezen, zou eruit kunnen zien als in het volgende voorbeeld.

```
Sub Knop4_Klikken()
    Indien Dialoogbladen("MijnBestandOpenDlg").Weergeven Dan
        DezeWerkmapOpmaken
    Einde Indien
Einde Sub
```

Wanneer de gebruiker Knop 4 kiest, zorgt de procedure ervoor dat het dialoogvenster MijnBestandOpenDlg wordt weergegeven en wacht vervolgens op een actie van de gebruiker. Als de gebruiker de knop OK kiest, wordt **Waar** als resultaat doorgegeven aan de instructie **Indien...Dan** en wordt de procedure DezeWerkmapOpmaken uitgevoerd. Als de gebruiker de knop Annuleren kiest of op ESC drukt, wordt **Onwaar** als resultaat doorgegeven en voert de procedure geen actie uit.

Opmerking U kunt een dialoogvenster ontwerpen waarin aan geen van de besturingselementen een gebeurtenisafhandelingsprocedure is gekoppeld. In dat geval moet de procedure die het dialoogvenster weergeeft echter de resultaten ervan verwerken.

Een afhandelingsprocedure voor de gebeurtenis BijActie maken

Steeds wanneer u een aangepast dialoogvenster weergeeft, wordt er een gebeurtenis **BijActie** gegenereerd voor het dialoogvenster. Als u een procedure hebt gemaakt die is gekoppeld aan de gebeurtenis **BijActie**, wordt deze procedure uitgevoerd voordat het dialoogvenster verschijnt. U kunt hier gebruik van maken om ervoor te zorgen dat de besturingselementen in het dialoogvenster bij het verschijnen ervan zijn ingesteld op bepaalde standaardwaarden, die bijvoorbeeld bepalen bij welk besturingselement de selectie staat of welk keuzerondje in een groepsvak is geselecteerd. Anders hebben de besturingselementen in het dialoogvenster de status zoals die was toen het dialoogvenster de laatste keer werd gesloten.

▶ **Een procedure koppelen aan de gebeurtenis BijActie van een dialoogvenster**

1. Selecteer het kader van het dialoogvenster door op de rand ervan te klikken.
2. Kies de opdracht **Macro toewijzen** in het menu **Extra**.

   U kunt ook met de rechtermuisknop op het dialoogvenster klikken om het snelmenu weer te geven en vervolgens de opdracht **Macro toewijzen** kiezen.
3. Typ of selecteer in het vak "Macronaam of verwijzing" de naam van de procedure die u aan de gebeurtenis wilt toewijzen.
4. Kies de knop OK.

Als een dialoogvenster met de naam InvoerDialoog bijvoorbeeld een serie aankruisvakjes heeft en een Invoervak met de naam InvoerBereik, kunt u een afhandelingsprocedure voor de gebeurtenis **BijActie** maken die de selectie in de aankruisvakjes annuleert en de inhoud van het Invoervak gelijkstelt aan de inhoud van de eerste cel van Blad1.

```
Sub WeergevenDialoogkader1()
    Voor Elke Kruis In Dialoogbladen("Dialoog1").Aankruisvakjes
        Kruis.Waarde = xlUit
    Volgende
    Dialoogbladen("Dialoog1").Invoervakken("Invoervak4").Tekst _
        = Werkbladen("Blad1").Cellen(1;1).Waarde
Einde Sub
```

Een ingebouwd dialoogvenster weergeven

Aan elk ingebouwd dialoogvenster is een constante gekoppeld die de naam van het dialoogvenster bevat. Zie *Dialogen (methode)* in de online Visual Basic Naslaggids voor een volledige lijst van gereserveerde namen voor ingebouwde dialoogvensters. Elk ingebouwd dialoogvenster ondersteunt een lijst van argumenten waarmee u de beginstatus van de besturingselementen van het dialoogvenster kunt instellen.

In Microsoft Excel voor Windows kunt u bijvoorbeeld met de volgende programmacode het ingebouwde dialoogvenster **Bestand openen** weergeven. Als standaarddirectory wordt in het dialoogvenster XLDOCS weergeven.

```
Toepassing.Dialogen(xlDlgOpenen).Weergeven("C:\XLDOCS")
```

In Microsoft Excel voor de Macintosh kunt u hiervoor de volgende programmacode gebruiken.

```
Toepassing.Dialogen(xlDlgOpenen).Weergeven("MijnSchijf:XLDocs")
```

Zodra de gebruiker de knop OK kiest, worden de acties uitgevoerd die zijn opgegeven door de wijzigingen in het dialoogvenster, alsof de overeenkomstige ingebouwde menu-opdracht was gekozen. Als de gebruiker de knop Annuleren kiest of op ESC drukt, wordt de uitvoering van het programma hervat bij de volgende instructie.

Aan de hand van de resultaatwaarde van de methode **Weergeven** kunt u bepalen of de gebruiker de knop OK of de knop Annuleren heeft gekozen. De volgende programmaroutine voert bijvoorbeeld de procedure WerkmapVerwerken uit als de gebruiker in het dialoogvenster **Bestand openen** een werkmap heeft geselecteerd.

```
Sub TestVenster()
    Toepassing.Dialogen(xlDlgOpenen).Weergeven ("c:\")
    Indien ActiefVenster.Type = "xlWerkmap" Dan mijnProcesWerkmapMacro
Einde sub
```

Opmerking U kunt een door uzelf gemaakt dialoogvenster dupliceren door het dialoogblad te kopiëren en vervolgens de kopie aan te passen. Als u echter de structuur van een ingebouwd dialoogvenster opnieuw wilt gebruiken in een aangepaste versie, die een lichtelijk afwijkende reeks voorzieningen ondersteunt, moet u het dialoogvenster geheel opnieuw maken, element voor element, en zelf een reeks procedures schrijven die de voorzieningen van het ingebouwde dialoogvenster dupliceren. U kunt besturingselementen toevoegen of verwijderen of het gedrag van de besturingselementen wijzigen met behulp van de procedures die het dialoogvenster ondersteunen.

Informatie ophalen uit een dialoogvenster

Zoals eerder in dit hoofdstuk besproken, kunt u koppelingen gebruiken om informatie van een besturingselement over te brengen naar een cel in een werkblad. Als een besturingselement is gekoppeld aan een cel in een werkblad, wordt het geselecteerde element in een keuzelijst of de waarde die de status van een aankruisvakje aangeeft, onmiddellijk bijgewerkt in het werkblad met de gekoppelde cel.

Met dialoogvensters kunt u uw procedures echter zo schrijven dat de informatie die in een besturingselement is ingevoerd, of een wijziging in de status van een besturingselement, worden meegenomen, zodat de nieuwe informatie beschikbaar wordt voor andere procedures in de toepassing.

In een dialoogvenster met een enkel Invoervak kunt u de gegevens in het Invoervak bijvoorbeeld met de volgende programmacode overbrengen naar de actieve cel.

```
Sub Invoervak4_wijziging()
        DezeTekst = Dialoogbladen("MijnDialoog").Invoervakken(1).Tekst
        Toepassing.Bladen(2).Activeren
        ActieveCel.Waarde = DezeTekst
Einde Sub
```

U kunt werken met alle besturingselementen van een bepaald type door de verzameling van de besturingselementen te indexeren. De verzameling van de keuzerondjes is bijvoorbeeld Keuzerondjes. Als u de status van alle keuzerondjes in een dialoogvenster wilt inlezen in een matrix, kunt u de volgende programmacode gebruiken.

```
Sub KnopControleren()
N = 0
Voor Elke Keuzerond In Dialoogbladen(1).Keuzerondjes
    N = N + 1
        Indien Keuzerond.Waarde=xlAan Dan
            Bladen("Blad1").Cellen(N;1)="Gecontroleerd"
    Anders
            Bladen("Blad1").Cellen(N;1)="Niet gecontroleerd"
    Einde Indien
Volgende
Einde Sub
```

Opmerking Een procedure kan de eigenschappen van een besturingselement op elk willekeurig moment lezen en instellen: voordat u het dialoogvenster weergeeft, terwijl dit op het scherm staat of nadat het is gesloten.

Besturingselementen wijzigen terwijl er een dialoogvenster zichtbaar is

De beginwaarden van de eigenschappen die aan een besturingselement zijn gekoppeld, wijzigt u tijdens het ontwerp van een dialoogvenster met .de opdracht Objecteigenschappen in het menu **Opmaak**. U kunt een eigenschap echter ook wijzigen tijdens de uitvoering van een procedure. Het is bijvoorbeeld mogelijk dat u een andere tekst naast een aankruisvakje wilt laten verschijnen als de gebruiker een bepaald keuzerondje selecteert of dat u een serie opties beschikbaar wilt maken als de gebruiker een bepaald aankruisvakje selecteert.

Besturingselementen hebben veel methoden en eigenschappen gemeen met figuren, zoals afmetingen en plaatsing. Sommige eigenschappen, zoals de eigenschap **AutomatischeGrootte** (die bepaalt of het formaat van een knop kan worden gewijzigd als het formaat van de onderliggende cel wordt gewijzigd), zijn alleen van toepassing wanneer het besturingselement zich op een werkblad bevindt. Andere eigenschappen, zoals **Toegangstoets**, zijn alleen van toepassing als het besturingselement zich in een dialoogvenster bevindt.

Welke eigenschappen beschikbaar zijn voor een besturingselement, is verschillend voor ieder element. Onder de naam van een besturingselement in de online Visual Basic Naslaggids vindt u meer informatie over de eigenschappen van het desbetreffende element.

Een besturingselement inschakelen wanneer aan een voorwaarde is voldaan

Met de eigenschap **Geactiveerd** van een besturingselement kunt u voorkomen dat de gebruiker een optie wijzigt, tenzij is voldaan aan een bepaalde voorwaarde. Hiermee kunt u bijvoorbeeld een reeks keuzerondjes beschikbaar maken wanneer de gebruiker een aankruisvakje inschakelt. Een voorbeeld hiervan ziet u in de volgende programmacode, die is toegewezen aan een gebeurtenisafhandelingsprocedure **BijActie** voor het aankruisvakje.

```
Sub AankruisVak4_Klik()
    Indien Dialoogbladen(1).Aankruisvakjes(4).Waarde = xlAan Dan
        Dialoogbladen(1).Keuzerondjes.Geactiveerd = Waar
    Anders Dialoogbladen(1),Keuzerondjes.Geactiveerd = Onwaar
    Einde Indien
Einde Sub
```

Verder kunt u de eigenschap **Geactiveerd** gebruiken om een afhandelingsprocedure **BijActie** te maken voor een Invoervak, zodat de knop OK beschikbaar wordt wanneer een onderdeelnummer dat de gebruiker heeft ingevoerd, in overeenstemming is met een standaardpatroon.

De selectie bij een besturingselement plaatsen

U kunt de selectie bij een bepaald besturingselement in een dialoogvenster plaatsen door de eigenschap **Nadruk** van het dialoogvenster in te stellen.

```
Sub DialoogKader1_Weergeven()
        Dialoogbladen("MijnBestandOpenDialoog").Nadruk = "Knop 4"
Einde Sub
```

Deze programmacode zorgt ervoor dat de nadruk bij Knop 4 komt te staan.

Koppelingen tussen een besturingselement en een werkblad wijzigen

U kunt koppelingen tussen een besturingselement en een cel in een werkblad instellen wanneer het besturingselement wordt gemaakt. U kunt dergelijke koppelingen tussen een besturingselement en een cel echter ook maken, wijzigen en verwijderen tijdens de uitvoering van een procedure. Hiervoor kunt u de eigenschap **GekoppeldeCel** van het besturingselement gebruiken. De eigenschap **GekoppeldeCel** hoort bij het gekoppelde besturingselement en niet bij de gekoppelde cel. Daarnaast kunt u gebruik maken van de eigenschap **BereikDoorvoerenInKeuzelijst**, die het invoerbereik voor keuzelijsten opgeeft.

De eigenschappen van een dialoogvenster wijzigen

Sommige eigenschap van het dialoogvenster zelf kunt u wijzigen terwijl het venster zichtbaar is, zoals het formaat en de plaats op het scherm. Op deze manier kunt u bijvoorbeeld een knop Opties ondersteunen. U kunt besturingselementen plaatsen onder de normale benedenrand van het dialoogvenster. Als de gebruiker in dat geval de knop Opties kiest, wijzigt de procedure het verticale formaat van het dialoogvenster, zodat aanvullende opties verschijnen die de gebruiker kan instellen.

```
'Stelt het uitgebreide formaat in
Sub Knop8_Klik()
        Dialoogbladen("MijnBestandOpenDlg").Dialoogkader.Hoogte = 500
Einde Sub
```

Een aangepast dialoogvenster verbergen

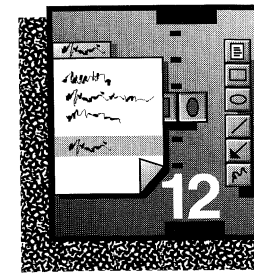
Een dialoogvenster wordt van het scherm verwijderd, zodra de gebruiker een knop kiest waarvan de eigenschap **DialoogSluiten** de waarde **Waar** heeft, zoals de knop OK. U kunt echter ook een gebeurtenis-procedure maken voor een besturingselement, zoals het volgende voorbeeld laat zien.

```
Sub Knop1_Klik()
    N = BerichtVenster("Weet u zeker dat u wilt afsluiten?"; vbJaNee)
    Indien N = vbJa Dan
        Dialoogbladen(1).Verbergen
    Einde Indien
Einde Sub
```

In het volgende hoofdstuk wordt een volgende stap in het maken van een aangepaste toepassing behandeld, namelijk hoe u menu's en werkbalken aanpast.

HOOFDSTUK 12

Menu's en werkbalken



In hoofdstuk 11, "Besturingselementen en dialoogvensters", wordt beschreven hoe u besturingselementen kunt toevoegen aan werkbladen en dialoogvensters en hoe u deze met Visual Basic-procedures kunt ondersteunen. In dit hoofdstuk komen twee andere mogelijkheden voor het uitbreiden van de gebruikers-interface van een Microsoft Excel-toepassing aan de orde: aangepaste menu's en aangepaste werkbalken. Een groep complexe opties kan het beste worden gepresenteerd in een dialoogvenster, maar een optie die slechts twee mogelijkheden kent, bijvoorbeeld een aankruisvakje dat in- of uitgeschakeld kan zijn, is erg geschikt voor een aangepaste menu-opdracht of een werkbalkknop.

Microsoft Excel heeft een aantal standaardmenu's, die deel uitmaken van een geheel van menu's, menu-opdrachten en snelmenu's. Met behulp van de Menu-editor kunt u dit menusysteem aanpassen aan uw eigen behoeften. Aan elke menubalk kunt u een menu toevoegen, aan dat menu kunt u een menu-opdracht toevoegen en aan deze opdracht kunt u een procedure koppelen die wordt uitgevoerd wanneer de gebruiker de opdracht kiest. Met de Visual Basic-procedures kunt u de eigenschappen van een menu instellen en wijzigen. U kunt bijvoorbeeld bepalen of een menu-opdracht een vinkje krijgt en of de opdracht beschikbaar is voor de gebruiker.

Werkbalken en de knoppen daarop zijn de grafische tegenhangers van menu's en de opdrachten daarin en zijn het meest zinvol voor bewerkingen die vaak worden gekozen. Met de Knop-editor kunt u werkbalkknoppen ontwerpen die de betrokken bewerking uitbeelden. U kunt ook aangepaste werkbalken maken die net zo werken als de standaardwerkbalken, en aan werkbalkknoppen procedures toewijzen die starten wanneer de gebruiker de knop kiest.

Inhoud

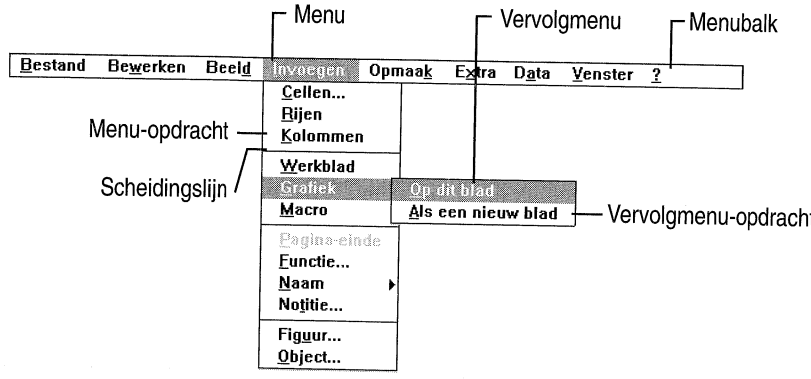
Het menusysteem wijzigen met de Menu-editor

Het *menusysteem* in Microsoft Excel bestaat uit de verzameling beschikbare menubalken, de menu's op elke menubalk, de menu-opdrachten in elk menu en de vervolgmenu's die bij sommige menu-opdrachten horen. In dit hoofdstuk wordt de term *menu-element* gebruikt voor alle elementen van een menusysteem: menubalk, menu, menu-opdracht, vervolgmenu-opdracht, snelmenu of snelmenu-opdracht.

U hebt al gewerkt met het ingebouwde menusysteem. Als u bijvoorbeeld een grafiek wilt invoegen in een werkblad, opent u het menu **Invoegen** op de menubalk Werkblad, kiest u de menu-opdracht **Grafiek** en kiest u vervolgens de vervolgmenu-opdracht **Op dit blad**.

Opmerking In dit hoofdstuk wordt de algemene term *opdracht* vermeden als het niet duidelijk is of er een menu-opdracht, een vervolgmenu-opdracht of een snelmenu-opdracht wordt bedoeld.

De verschillende elementen van het menusysteem verschijnen in verschillende omgevingen. Als u bijvoorbeeld een grafiek bewerkt, staan er op de menubalk een aantal menu's die van toepassing zijn op grafieken. U kunt procedures toewijzen aan menu-, vervolgmenu- en snelmenu-opdrachten die u zelf maakt en u kunt procedures ontwikkelen die de huidige menubalk vervangen door een menubalk die past bij het blad dat op het scherm staat.



Er zijn ook snelmenu's die verschijnen als de gebruiker op de rechtermuisknop drukt terwijl de muisaanwijzer zich op een object bevindt (Windows) of als de gebruiker op CTRL drukt en tegelijk de muisknop indrukt (Macintosh). Als u bijvoorbeeld met de rechtermuisknop op een grafisch object klikt, verschijnt er een snelmenu waarin u opdrachten kunt kiezen die een bewerking op de grafiek uitvoeren of de eigenschappen ervan wijzigen.

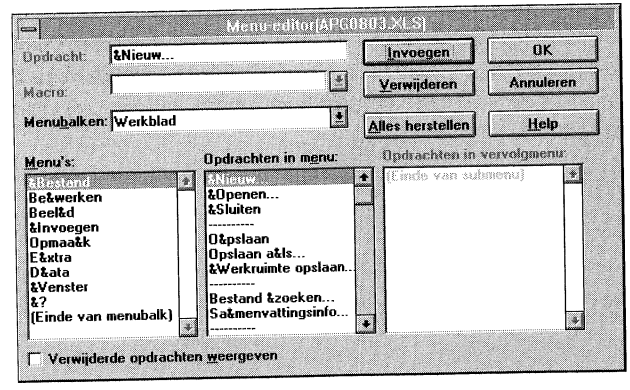
Menu-elementen maken

Als u het ingebouwde menusysteem wilt wijzigen, kunt u de Menu-editor of een of meer Visual Basic-procedures gebruiken, zoals verderop in dit hoofdstuk wordt uitgelegd. U kunt alle functies van het dialoogvenster **Menu-editor** ook uitvoeren met Visual Basic-programmacode, maar tenzij u menu-elementen wilt toevoegen of verwijderen terwijl er programmacode wordt uitgevoerd, is het gewoonlijk gemakkelijker om met de Menu-editor een menusysteem te maken dat gekoppeld is aan de actieve werkmap.

▶ Het dialoogvenster Menu-editor weergeven

- Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.

  U kunt ook de knop Menu-editor op de werkbalk Visual Basic kiezen als er een werkblad actief is.



Een nieuwe menubalk maken

Meestal hoeft u maar één menu-opdracht toe te voegen aan een standaardmenu, bijvoorbeeld een nieuwe menu-opdracht aan het menu **Extra**. U kunt ook een geheel nieuw menu toevoegen aan een standaardmenubalk en vervolgens een aantal opdrachten toevoegen aan dat menu. Deze taken worden in de volgende gedeelten van dit hoofdstuk beschreven.

Als u een aangepaste toepassing wilt maken, moet u misschien een nieuwe menubalk maken, compleet met menu's, menu-opdrachten en vervolgmenu-opdrachten. U kunt aan elk menu, standaard of aangepast, opdrachten toevoegen en u kunt deze ook verwijderen. U kunt eveneens een aangepast menu toevoegen aan iedere menubalk, standaard of aangepast.

▶ Een menubalk toevoegen aan een werkmap

1. Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.
2. Selecteer in het vak "Menubalken" de naam van een menubalk.

   Door deze stap wordt de selectie van elementen in de vakken "Menu's", "Opdrachten in een menu" en "Opdrachten in vervolgmenu" geannuleerd. Als u deze stap overslaat, wordt de tekst die u typt in het vak "Opdracht" de eerste opdracht in het eerste menu dat is geselecteerd in het vak "Menu's".
3. Kies de knop Invoegen.
4. Typ in het vak "Opdracht" de naam van de nieuwe menubalk of accepteer de standaardnaam, als deze van toepassing is.
5. Voeg desgewenst andere menubalken, menu-opdrachten of vervolgmenu-opdrachten toe.
6. Kies de knop OK.

De nieuwe menubalk wordt toegevoegd aan de lijst "Menu-balken", onder het laatste element (Snelmenu's 3) in de lijst met standaardmenubalken, die u in de volgende tabel kunt bekijken. De aangepaste menubalken die u maakt, worden geplaatst in de volgorde waarin u ze maakt. U kunt maximaal vijftien menubalken gebruiken.

Menubalk	Beschrijving
Werkblad	De menubalk die verschijnt als er een werkblad actief is
Grafiek	De menubalk die verschijnt als er een grafiek actief is
Geen documenten geopend	De menubalk die verschijnt als er geen werkmap geopend is
Visual Basic-module	De menubalk die verschijnt als er een Visual Basic-module actief is
Snelmenu's 1	Een groep snelmenu's die verschijnt wanneer de gebruiker op een werkbalk, werkbalkknop, geselecteerd cellenbereik, werkmaptab, titelbalk van venster of desktop klikt met de rechtermuisknop
Snelmenu's 2	Een groep snelmenu's die verschijnt wanneer de gebruiker op een figuur, knop of tekstvak klikt met de rechtermuisknop
Snelmenu's 3	Een groep snelmenu's die verschijnt wanneer de gebruiker met de rechtermuisknop op een grafiek klikt of op een van de volgende onderdelen ervan: gegevensreeks, tekst, plotgebied, as, rasterlijn, basis, pijl, legenda

Opmerking Een snelmenu wordt opgevat als een type menu en elke groep snelmenu's in de lijst "Menubalken" wordt opgevat als een type menubalk. Wanneer u een snelmenu wilt weergeven op de Apple Macintosh, houdt u CTRL ingedrukt en klikt u met de muisknop.

Omdat standaardmenubalken bij een bepaald type blad horen (de menubalk voor grafieken verschijnt bijvoorbeeld als u een grafiek activeert), is het misschien nodig dat u een menu, menu-opdracht of vervolgmenu-opdracht met dezelfde naam toevoegt aan meer dan één menubalk. U kunt bijvoorbeeld aan het menu **Bestand** van elke menubalk de menu-opdracht **Openen speciaal** toevoegen, die een speciaal dialoogvenster weergeeft waarmee bestanden kunnen worden geïmporteerd die niet door Microsoft Excel worden ondersteund.

Een nieuw menu maken

U kunt aan iedere standaardmenubalk of aangepaste menubalk een menu toevoegen. U kunt bijvoorbeeld een menu **Boekhouden** aan elke menubalk toevoegen, zodat de werknemers de bijbehorende macro's vanuit elk blad kunnen starten. Voor dit menu kunt u ook een toegangstoets opgeven. (Toegangstoetsen worden onderstreept weergegeven in het menu).

▶ **Een menu toevoegen aan een menubalk**

1. Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.
2. Selecteer in het vak "Menubalken" de naam van de menubalk waaraan u het menu wilt toevoegen.
3. Selecteer in het vak "Menu's" de naam van het menu waarvoor u links het nieuwe menu wilt invoegen.

   –Of–

   Selecteer "Einde van menubalk" als u een menu rechts van het laatste menu op de menubalk wilt toevoegen.
4. Kies de knop Invoegen om de nieuwe menu-opdracht te maken.
5. Typ in het vak"Opdracht" de naam van het nieuwe menu.

   Typ het teken & voor het teken dat u als toegangstoets voor het menu wilt gebruiken.
6. Voeg desgewenst andere menubalken, menu-opdrachten of vervolgmenu-opdrachten toe.
7. Kies de knop OK.

Opmerking Een snelmenu kunt u niet toevoegen of wissen. Maar als u een snelmenugroep selecteert in de lijst "Menubalken", verschijnt in de lijst "Menu's" de naam van ieder type object waarvoor een snelmenu is gedefinieerd. Als u een object in deze lijst selecteert, kunt u opdrachten toevoegen aan het snelmenu dat bij het object hoort.

Een nieuwe menu-opdracht of snelmenu-opdracht maken

U kunt een opdracht toevoegen aan een standaardmenu, een snelmenu of een aangepast menu.

▶ **Een opdracht toevoegen aan een menu of snelmenu**

1. Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.
2. Selecteer in het vak "Menubalken" de naam van een menubalk of groep snelmenu's.
3. Selecteer in het vak "Menu's" de naam van het menu waaraan u de opdracht wilt toevoegen of het object waarvoor een snelmenu is gedefinieerd.
4. Selecteer in het vak "Opdrachten in menu" de naam van de (snel)menu-opdracht waarboven u de nieuwe opdracht wilt invoegen.

   –Of–

   Selecteer "Einde van menu" als u een opdracht onder in het menu wilt toevoegen.

7. Kies de knop Invoegen om de nieuwe menu-opdracht te maken.
6. Typ in het vak"Opdracht" de naam van de nieuwe opdracht die u wilt toevoegen.

   Typ het teken **&** voor het teken dat u als toegangstoets voor de menu-opdracht wilt gebruiken.

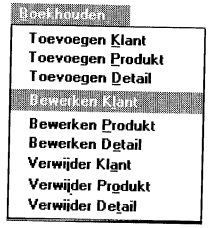
   Typ één afbreekstreepje (-) als u een scheidingslijn wilt maken.

7. Typ of selecteer de naam van een procedure in het vak "Macro" om op te geven dat de procedure wordt gestart wanneer de gebruiker de (snel)menu-opdracht kiest.
8. Voeg desgewenst andere menubalken, menu-opdrachten of vervolgmenu-opdrachten toe.
9. Kies de knop OK.

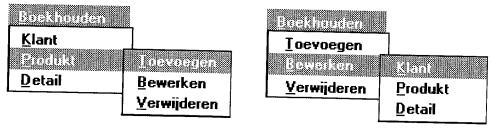
Een nieuw vervolgmenu maken

Vervolgmenu's verschijnen wanneer de gebruiker een menu-opdracht kiest waaraan een of meer vervolgmenu-opdrachten zijn gekoppeld. Dank zij vervolgmenu's kan de gebruiker opdrachten bekijken die anders alleen via een reeks dialoogvensters beschikbaar zouden zijn.

Met vervolgmenu's kunt u menu's effectiever ordenen en een toepassing helderder maken doordat minder informatie tegelijk wordt gepresenteerd. Stel bijvoorbeeld dat u een menu maakt dat een aantal opties presenteert, zoals in het volgende voorbeeld.



Met vervolgmenu's kunt u dezelfde elementen op een van de volgende wijzen presenteren.



U kunt vervolgmenu-opdrachten toevoegen aan aangepaste menu-opdrachten, maar niet aan standaardmenu-opdrachten. Als u een vervolgmenu wilt toevoegen aan een standaardmenu-opdracht, moet u eerst deze opdracht verwijderen, een aangepaste menu-opdracht toevoegen die dezelfde taak verricht en vervolgens de vervolgmenu-opdracht koppelen aan de aangepaste menu-opdracht.

▶ Een vervolgmenu-opdracht toevoegen aan een menu-opdracht

1. Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.
2. Selecteer in het vak "Menubalken" de naam van een menubalk.
3. Selecteer in het vak "Menu's" de naam van het menu waaraan u de opdracht wilt toevoegen.
4. Selecteer in het vak "Opdrachten in menu" de naam van de menu-opdracht waaraan u de vervolgmenu-opdracht wilt koppelen.
5. Selecteer in het vak "Opdrachten in vervolgmenu" de naam van de vervolgmenu-opdracht waarboven u de vervolgmenu-opdracht wilt invoegen.

   –Of–

   Selecteer "Einde van vervolgmenu" als u een opdracht onder in het vervolgmenu wilt toevoegen.

6. Kies de knop Invoegen om de nieuwe vervolgmenu-opdracht te maken.
7. Typ in het vak "Opdracht" de naam van de nieuwe vervolgmenu-opdracht.

   Typ het teken **&** voor het teken dat u als toegangstoets voor de opdracht wilt gebruiken.

   Typ één afbreekstreepje (-) als u een scheidingslijn wilt maken.
8. Typ of selecteer de naam van een procedure in het vak "Macro" om op te geven dat de procedure wordt gestart wanneer de gebruiker de vervolgmenu-opdracht kiest.
9. Voeg desgewenst andere menubalken, menu-opdrachten of vervolgmenu-opdrachten toe.
10. Kies de knop OK.

Menu-elementen verwijderen, herstellen en een nieuwe naam geven

U kunt elementen verwijderen uit iedere standaardmenubalk en u kunt complete aangepaste menubalken verwijderen, maar een standaardmenubalk kunt u niet verwijderen. U kunt wel alle menu's op een standaardmenubalk verwijderen en deze desgewenst later herstellen.

Stel dat u een standaardmenu-opdracht voor grafieken wilt vervangen door een opdracht die een aangepaste functie Wizard Grafieken presenteert met een reeks speciale grafiekopmaaksjablonen voor uw gegevens. Of misschien wilt u bepaalde menu-opdrachten verwijderen om te voorkomen dat onervaren gebruikers een juiste werking van de toepassing verstoren of om de gebruikers-interface van een toepassing begrijpelijker te maken.

▶ **Een menu-element verwijderen**

1. Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.
2. Selecteer in het desbetreffende vak de naam van de menubalk, het menu of de menu-, vervolgmenu- of snelmenu-opdracht die u wilt verwijderen.
3. Kies de knop Verwijderen.
4. Kies de knop OK.

U kunt standaardmenubalken, -menu's of -menu-opdrachten die u hebt verwijderd, herstellen. Maar aangepaste menubalken, menu's en menu-opdrachten die zijn verwijderd, kunnen niet meer worden hersteld.

▶ **Een verwijderd menu-element herstellen**

1. Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.
2. Schakel het aankruisvakje "Verwijderde opdrachten weergeven" in.

   Verwijderde menu-elementen worden lichter gekleurd weergegeven.
3. Selecteer in het desbetreffende vak het verwijderde element dat u wilt herstellen.
4. Kies de knop Wissen ongedaan maken.
5. Kies de knop OK.

Een andere mogelijkheid is alle standaardmenubalken, standaardmenu's, standaardmenu-opdrachten en vervolgmenu-opdrachten te herstellen door de knop Alles herstellen te kiezen. Als u dit doet, worden ook alle aangepaste menubalken, menu's, menu-opdrachten en vervolgmenu-opdrachten uit de actieve werkmap verwijderd.

Van aangepaste menu-elementen kunt u ook de naam wijzigen.

▶ **De naam van een menu-element wijzigen**

1. Terwijl er een Visual Basic-module actief is, kiest u **Menu-editor** in het menu **Extra**.
2. Selecteer in het desbetreffende vak het menu-element waarvan u de naam wilt wijzigen.
4. Typ in het vak "Opdracht" de nieuwe naam voor het element.
5. Kies de knop OK.

Van standaardmenu-elementen kunt u de naam niet wijzigen.

Waar het menusysteem wordt opgeslagen

Standaardmenu-elementen horen bij Microsoft Excel, maar alle wijzigingen die u aanbrengt in het menusysteem, inclusief toegevoegde en gewiste menu-elementen, worden opgeslagen als een *menuwijzigingslijst* in de actieve werkmap. Alle geopende werkmappen kunnen een eigen menuwijzigingslijst hebben. Deze lijsten hebben een gecombineerd effect op het menusysteem.

Stel bijvoorbeeld dat u één geopende werkmap hebt: u wijzigt het menusysteem in die werkmap en u ontwikkelt procedures die de menu's, werkbladen en andere objecten in de toepassing ondersteunen. Als u dan de werkmap opslaat en sluit, keert het menusysteem terug in de oorspronkelijke staat. Wanneer u de werkmap weer opent, wordt het menusysteem aangepast aan de menuwijzigingslijst die met de werkmap is opgeslagen.

Neem nu bijvoorbeeld eens het standaardmenu **Bestand**, waarin de eerste drie menu-opdrachten **Nieuw**, **Openen** en **Sluiten** zijn. Stel dat u een werkmap opent die MIJNMAP1.XLS heet en dat u de menu-opdracht **Openen speciaal** toevoegt aan het menu **Bestand**, vlak boven de opdracht **Sluiten**. Vervolgens sluit u MIJNMAP1.XLS en opent u de werkmap MIJNMAP2.XLS, waarin u de menu-opdracht **Database openen** toevoegt aan het menu **Bestand**, vlak boven de opdracht **Sluiten**. Als u dan MIJNMAP2.XLS geopend laat en MIJNMAP1.XLS opnieuw opent, bevat het menu **Bestand** alle menu-opdrachten: **Nieuw**, **Openen**, **Database openen**, **Openen speciaal** (op deze plaats omdat de andere werkmap al geopend was) en **Sluiten**.

Als u echter naar de Menu-editor gaat terwijl MIJNMAP1.XLS actief is en de werkbalk Werkblad en het menu **Bestand** selecteert, ziet u slecht de menu-opdrachten **Nieuw**, **Openen**, **Openen speciaal** en **Sluiten**. Als u daarentegen naar de Menu-editor gaat terwijl MIJNMAP2.XLS actief is, ziet u de menu-opdrachten **Nieuw**, **Openen**, **Database openen** en **Sluiten**. Dit komt doordat de Menu-editor in zo'n geval alleen de effecten van de menuwijzingslijst van de actieve werkmap weergeeft.

Menu's beheren met Visual Basic

Zoals al eerder in dit hoofdstuk is gesteld, kunt u met Visual Basic-procedures alle functies van de Menu-editor verrichten. In de meeste gevallen is het gemakkelijker om met de Menu-editor te werken, maar er zijn ook taken die de Menu-editor niet aankan.

Stel dat u een menu-opdracht wilt opnemen in een menu, maar dan zodanig dat de opdracht niet beschikbaar is voor de gebruiker, dus uitgeschakeld is (dit heet ook wel *lichter gekleurd* weergeven). Of bij een optie die twee mogelijkheden heeft, wilt u aangeven dat een van deze twee actief is door een vinkje te plaatsen naast de desbetreffende menu-opdracht. Zo verschijnt er een vinkje naast de menu-opdracht **Formulebalk** in het menu **Beeld** wanneer de formulebalk zichtbaar is.

Ten slotte kan het voorkomen dat u de naam van een menu-opdracht wilt laten afhangen van de huidige toestand. Als u bijvoorbeeld de menu-opdracht **Titels blokkeren** kiest in het menu **Venster**, verandert de naam van de opdracht in **Titelblokkering opheffen**. Deze taken kunnen alleen worden verricht met Visual Basic-procedures.

Werken met menubalken

Visual Basic ondersteunt een scala van methoden en eigenschappen voor het beheer van menubalken. De verzameling Menubalken bevat alle menubalken die in de Microsoft Excel-omgeving beschikbaar zijn. Met de eigenschap **Aantal** kunt u het aantal beschikbare menubalken te weten komen (maar niet het aantal snelmenugroepen die als menubalken in de lijst staan in het dialoogvenster **Menu-editor**).

Met de methode **Onderdeel** kunt u een verwijzing naar één menubalk genereren, gegeven de index of de naam van de menubalk. Met de volgende constanten kunt u de bijbehorende standaardmenubalken opgeven.

Constante	Menubalk die hoort bij
xlWerkblad	Werkbladen en macrobladen
xlGrafiek	Grafiekbladen
xlGeenDocumenten	Geen documenten geopend
xlInfo	Informatievenster
xlModule	Visual Basic-module

Hoewel er een lijst van snelmenugroepen staat in het vak "Menubalken" in het dialoogvenster **Menu-editor**, is op deze wijze toegang tot de groepen niet mogelijk. Als u een verwijzing wilt genereren naar een snelmenu, gebruikt u de methode **Snelmenus**, zoals verderop in dit hoofdstuk wordt beschreven.

Een nieuwe menubalk toevoegen

Met de methode **Toevoegen** kunt u een nieuwe menubalk maken en desgewenst de naam van de menubalk opgeven.

```
'Een menubalk toevoegen met naam
Menubalken.Toevoegen "MijnWerkbladMenubalk"
```

U kunt ook de eigenschap **Index** gebruiken om het indexnummer te zoeken van een menubalk die u aan de verzameling Menubalken hebt toegevoegd, en de eigenschap **Bijschrift** om de naam van een menubalk op te zoeken of in te stellen, gegeven het indexnummer. Omdat u de naam van een menubalk in de methode **Toevoegen** kunt opgeven en omdat het indexnummer van een menubalk onverwacht gewijzigd kan worden door het toevoegen of wissen van menubalken, kunt u beter geen gebruik maken van het indexnummer van een menubalk.

Met de methode **Menus** kunt u een verzameling van alle menu's in de opgegeven menubalk krijgen. Bijvoorbeeld:

```
Menubalken(xlWerkblad).Menus
```

Deze verzameling bevat het menu **Bestand**, het menu **Bewerken**, enzovoort.

U kunt erachter komen of een menubalk tot Microsoft Excel of tot een werkmap behoort door de eigenschap **Ingebouwd** te gebruiken. Deze eigenschap resulteert in **Waar** als de menubalk een standaardmenubalk is. In het volgende voorbeeld wordt bepaald of de actieve menubalk standaard is.

```
IsStandaard = ActieveMenubalk.Ingebouwd
```

Een menubalk weergeven

Nadat u een menubalk hebt gemaakt, kunt u deze met de methode **Activeren** weergeven.

```
Menubalken("MijnWerkbladMenubalk").Activeren
```

Welke standaardmenubalken kunnen worden weergegeven, hangt af van het type blad dat actief is. U kunt bijvoorbeeld niet de grafiekmenubalk weergeven wanneer er een werkblad actief is.

Een menubalk verwijderen

Met de methode **Verwijderen** kunt u een menubalk verwijderen, gegeven de naam of het indexnummer van de menubalk.

```
Menubalken("MijnWerkbladMenubalk").Verwijderen
```

Zoals al eerder in dit hoofdstuk is gesteld, kunt u alleen aangepaste menubalken verwijderen.

Een menubalk herstellen

Met de methode **OpnieuwInstellen** kunt u een standaardmenubalk en alle bijbehorende menu-elementen in de oorspronkelijke staat herstellen.

```
Menubalken(xlWerkblad).OpnieuwInstellen
```

Gebruik de methode **OpnieuwInstellen** alleen om een menubalk te herstellen als u er zeker van bent dat u een menu, menu-opdracht of vervolgmenu-opdracht op die menubalk niet meer nodig hebt. Anders moet u de menubalk opdracht voor opdracht reconstrueren om mogelijke interacties met de menuwijzigingslijsten van andere werkmappen te voorkomen.

Werken met menu's

De verzameling Menus bevat de menu's van een opgegeven menubalk. Iedere menubalk heeft een methode **Menus** die als resultaat een verzameling geeft van alle bijbehorende menu's. Het volgende voorbeeld betreft een verzameling die de menu's **Bestand**, **Bewerken**, **Beeld** en andere bevat:

```
Toewijzen GrafiekMenus = Menubalken(xlGrafiek).Menus
```

Een object Menu definieert een menu op een menubalk, een snelmenu-opdracht of een vervolgmenu-opdracht. U kunt verwijzen naar een menu met de naam of met de positie ervan op de menubalk. Omdat de positie van een menu op een menubalk kan worden gewijzigd (door andere procedures, door de Menu-editor of door het effect van een menuwijzigingslijst van een andere werkmap dan de werkmap die de procedure bevat), kunt u beter met de naam dan met de positie verwijzen. De volgende twee expressies zijn bijvoorbeeld alleen gelijkwaardig indien het standaardmenu **Bewerken** op de grafiekmenubalk niet is gewijzigd.

```
Toewijzen TweedeOpdracht = Menubalken(xlGrafiek).Menus("Bewerken")
Toewijzen TweedeOpdracht = Menubalken(xlGrafiek).Menus(2)
```

Een menu toevoegen aan een menubalk

Met de methode **Toevoegen** kunt u een menu toevoegen aan een opgegeven menubalk. In het volgende voorbeeld wordt het menu MijnWerkMenu links geplaatst van het menu Help op de menubalk die bij grafiekbladen hoort.

```
Toewijzen NieuwMenu = Menubalken(xlGrafiek).Menus.Toevoegen( _
    Bijschrift := "&MijnWerkMenu" ; _
    Voor := "Help")
```

Gebruik het teken & om de toegangstoets voor het menu op te geven. Wanneer het menu is toegevoegd, hoeft u het teken & niet te gebruiken om naar het menu te verwijzen. Als u het menu op de laatste positie wilt toevoegen, laat u het argument *voor* weg.

De methode **Toevoegen** resulteert in een verwijzing naar het nieuwe menu.

Als u Microsoft Excel voor de Macintosh gebruikt, veroorzaakt de methode **Toevoegen** een onderschepbare fout als er niet voldoende ruimte is voor de naam van het menu.

Een menu verwijderen

Met de methode **Verwijderen** kunt u een aangepast menu of standaardmenu verwijderen. In het volgende voorbeeld wordt het menu **Bewerken** uit alle menubalken verwijderd.

```
Sub VerwijderBewerken()
    Voor Elke mb In Menubalken
        Indien mb.Index <> xlGeenDocumenten En mb.Index <> xlInfo Dan
            mb.Menus("Bewerken").Verwijderen
        Einde Indien
    Volgende mb
Einde Sub
```

Een standaardmenu herstellen

Als u een menu-opdracht wilt herstellen, gebruikt u het argument *herstellen* van de methode **Toevoegen**. In het volgende voorbeeld worden de menu's Bewerken hersteld, die in het vorige voorbeeld zijn verwijderd.

```
Sub HerstelBewerken()
    Voor Elke mb In Menubalken
        Indien mb.Index <> xlGeenDocumenten En mb.Index <> xlInfo Dan
            mb.Menus.Toevoegen Bijschrift:="&Bewerken".Voor:=2_
            Herstellen:=Waar
        Einde Indien
    Volgende mb
Einde Sub
```

Als u een aangepast menu wilt herstellen, moet u het helemaal opnieuw opbouwen met de methode **Toevoegen**, zoals eerder in dit hoofdstuk is beschreven.

Werken met menu-opdrachten en vervolgmenu-opdrachten

De verzameling MenuOpdrachten bevat de menu-opdrachten in een opgegeven menu, inclusief scheidingslijnen en vervolgmenu-opdrachten. De lijst van de laatst gebruikte bestanden in het menu Bestand telt als één menu-opdracht en ook de lijst van geopende vensters in het menu Venster telt als één menu-opdracht.

Een object MenuOpdracht definieert één menu-opdracht. In het volgende voorbeeld wordt door de instructie de variabele `BestandNieuwMenuOpdracht` ingesteld op de menu-opdracht **Nieuw** in het menu **Bestand** op de grafiekmenubalk.

```
Toewijzen BestandNieuwMenuOpdracht = _
Menubalken(xlGrafiek).Menus("Bestand").MenuOpdrachten("Nieuw")
```

Een menu-opdracht toevoegen aan een menu

Met de methode **Toevoegen** kunt u een nieuwe menu-opdracht toevoegen aan het opgegeven menu. De argumenten van deze methode gebruikt u om het volgende op te geven: de naam van het menu, de naam van de procedure die wordt gestart wanneer de gebruiker de menu-opdracht kiest de positie in het menu. In het volgende voorbeeld wordt de menu-opdracht **Database openen** toegevoegd aan het menu **Bestand**.

```
Toewijzen MenuOpdrachtToegevoegd = _
Menubalken("MijnMenubalk").Menus("Bestand").MenuOpdrachten _
    .Toevoegen (Bijschrift := "&Database openen" ; _
        BijActie:= "DatabaseOpenen" ; _
        Voor := "Sluiten"    )          'Weergeven voor opdracht
                                        'Sluiten
```

Als u de menu-opdracht aan het einde van het opgegeven menu wilt plaatsen, laat u het argument *voor* weg.

Met de volgende instructie kunt u een scheidingslijn maken.

```
'Afbreekstreepje maakt de scheidingslijn.
Menubalken("MijnMenubalk").Menus("Bestand").MenuOpdrachten.Toevoegen _
    Bijschrift := "-" ; _
    Voor := "Sluiten"                   'Weergeven voor opdracht
                                        'Sluiten.
```

Een menu-opdracht verwijderen

Met de methode **Verwijderen** kunt u een menu-opdracht uit het opgegeven menu verwijderen. In het volgende voorbeeld wordt de menu-opdracht **Openen** uit ieder menu **Bestand** verwijderd.

```
Bij Fout Hervatten Volgende
Voor Elke DitMenu In Menubalken
    DitMenu.Menus("Bestand").MenuOpdrachten("Openen...").Verwijderen
Volgende
```

Een menu-opdracht activeren

Met de eigenschap **Geactiveerd** maakt u een aangepaste menu-opdracht wel of niet beschikbaar voor de gebruiker. Als een menu-opdracht niet geactiveerd is (dus lichter gekleurd), resulteert de eigenschap **Geactiveerd** in **Onwaar**. In het volgende voorbeeld wordt de aangepaste menu-opdracht **MijnOpenen** in het menu **Bestand** op de werkbladmenubalk uitgeschakeld maar niet verwijderd.

```
Menubalken(xlWerkblad).Menus("Bestand").MenuOpdrachten( _
"MijnOpenen...")     .Geactiveerd = Onwaar
```

Een vinkje plaatsen naast een menu-opdracht

Met de eigenschap **Ingeschakeld** kunt u een vinkje naast een menu-opdracht plaatsen of een vinkje verwijderen. Ook kunt u met deze eigenschap vaststellen of een menu-opdracht afgevinkt is. De eigenschap **Ingeschakeld** resulteert in **Waar** als er een vinkje naast de menu-opdracht staat. In het volgende voorbeeld is de procedure toegewezen aan de menu-opdracht **Database**, die is toegevoegd onder de menu-opdracht **Werkbalken** in het menu **Beeld** op de standaardmenubalk voor werkbladen. Deze procedure plaatst een vinkje naast de menu-opdracht **Database** of verwijdert het vinkje, voordat er naar een andere weergave wordt geschakeld.

```
Sub DatabaseWeergave()
    Met Menubalken(xlWerkblad).Menus("Beeld").MenuOpdrachten("Database")
        .Ingeschakeld = Niet.Ingeschakeld
        Indien .Ingeschakeld Dan
            'Schakelen naar databaseweergave
            .
            .
            .
        Anders
            'Schakelen naar werkbladweergave
            .
            .
            .
        Einde Indien
    Einde Met
Einde Sub
```

De naam van een menu-opdracht wijzigen

Met de eigenschap **Bijschrift** van een menu-opdracht kunt u de naam van de opdracht wijzigen als reactie op veranderende voorwaarden in de Visual Basic-programmacode.

```
Menubalken("MijnMenubalk").Menus("Bestand").MenuOpdrachten( _
"Database openen") _
    .Bijschrift = " &Database sluiten"
```

Als u de naam van een menu-opdracht op deze manier wijzigt, moet u ervoor zorgen dat de andere procedures in de toepassing met de nieuwe naam verwijzen naar de menu-opdracht, dus in dit voorbeeld met "Database sluiten". U kunt ook met variabelen naar een menu-opdracht verwijzen. Dit heeft het voordeel dat de variabelen blijven functioneren, ook als het bijschrift van de opdracht wordt gewijzigd. In het volgende voorbeeld wordt een variabele toegewezen aan de menu-opdracht **Database openen**.

```
Toewijzen MijnMenu = Menubalken("Mijn menubalk").Menus( _
    "Bestand").MenuOpdrachten("Database Openen")
```

U kunt het bijschrift later herstellen met de volgende programmacode:

```
MijnMenu.Bijschrift = "&Database sluiten"
```

Een menu-opdracht herstellen

U kunt een standaardmenu-opdracht die is verwijderd, herstellen met het argument *herstellen* van de methode **Toevoegen**. In het volgende voorbeeld wordt de verwijderde menu-opdracht **Openen** hersteld in het menu **Bestand** op de werkbladmenubalk.

```
Menubalken(xlWerkblad).Menus("Bestand").MenuOpdrachten.Toevoegen _
    Bijschrift := "Openen";_
    Herstellen := Waar
```

Als u in dit voorbeeld het argument *herstellen* weglaat, wordt er een nieuwe menu-opdracht **Openen** toegevoegd onder de menu-opdracht **Afsluiten** in het menu **Bestand**.

Een vervolgmenu-opdracht toevoegen

Met de methode **MenuToevoegen** kunt u een nieuw vervolgmenu toevoegen aan een menu-opdracht of kunt u een verwijderd standaardvervolgmenu herstellen. Als u **MenuToevoegen** toepast op menu-opdrachten, moet u, net als bij de methode **Toevoegen**, de naam van de vervolgmenu-opdracht opgeven met het argument *bijschrift*. U kunt de positie van de opdracht opgeven met het argument *voor*, en met het argument *herstellen* geeft u op of u een standaardvervolgmenu wilt herstellen.

Werken met snelmenu's

In tegenstelling tot de gewone menu's voor werkbladen en grafieken staan snelmenu's niet op een menubalk. Gebruik de methode **Snelmenus** om toegang te krijgen tot een snelmenu. U geeft een standaardsnelmenu op door de bijbehorende constante te gebruiken. Zie *Snelmenu's* in de online Visual Basic Naslaggids als u de constante van een standaardsnelmenu wilt weten.

In het volgende voorbeeld wordt verwezen naar het snelmenu dat verschijnt wanneer de gebruiker met de rechtermuisknop op een gegevensreeks in een grafiek klikt (Windows), of op een gegevensreeks klikt en CTRL ingedrukt houdt (Apple Macintosh).

```
Snelmenus(xlGrafiekreeks)
```

Het is niet mogelijk een nieuw snelmenu te maken. U kunt een standaardsnelmenu alleen wijzigen met de verzameling MenuOpdrachten. In het volgende voorbeeld wordt er een aangepaste snelmenu-opdracht, **Opmaken speciaal**, toegevoegd onder in het snelmenu voor werbladcellen.

```
Snelmenus(xlWerkbladCel).MenuOpdrachten.Toevoegen _
    bijschrift := "Opmaken &Speciaal"
```

Werkbalken en knoppen beheren met Visual Basic

Een *werkbalk* is de verzameling besturingselementen die verschijnt als u de opdracht **Werkbalken** kiest in het menu **Beeld**, het aankruisvakje naast de naam van een werkbalk inschakelt en de knop OK kiest. Op werkbalken staan werkbalkknoppen. Een werkbalkknop heeft twee componenten: de knop zelf, dat wil zeggen het besturingselement waarop de gebruiker klikt om een bewerking te starten, en het knopvlak, dat wil zeggen het plaatje op de knop.

Veel taken die kunnen worden verricht met werkbalken worden beschreven in hoofdstuk 34, "De werkruimte aanpassen", in het *Microsoft Excel handboek.* Hierin vindt u informatie over het gebruik van de optie "Aanpassen" in het dialoogvenster **Werkbalken** om een nieuwe werkbalk te maken of om nieuwe of standaardwerkbalkknoppen toe te voegen aan een werkbalk; over het gebruik van de Knop-editor om de afbeelding op een bepaalde werkbalkknop aan te passen; over het gebruik van het dialoogvenster **Macro toewijzen** om een procedure toe te wijzen die wordt gestart wanneer de gebruiker de werkbalkknop kiest.

In dit hoofdstuk komen veel van dezelfde onderwerpen aan de orde, maar dan aan de hand van de Visual Basic-programmacode in plaats van het dialoogvenster **Werkbalken** of de Knop-editor. Net als met de Menu-editor verdient het gebruik van het dialoogvenster **Werkbalken** en de Knop-editor de voorkeur bij het werken met werkbalken. Daar staat tegenover dat het samenstellen van werkbalken met Visual Basic-procedures meer mogelijkheden biedt.

U kunt programmacode schrijven die werkbalken een positie op het scherm geeft, u kunt bepalen of een werkbalkknop beschikbaar (geactiveerd) is of ingedrukt wordt weergegeven en u kunt aangepaste werkbalken koppelen aan bepaalde documenten. Verder kunt u met de opdracht **Werkbalken toevoegen** in het menu **Extra** een aangepaste werkbalk in een werkmap opslaan wanneer er een Visual Basic-module actief is.

Werken met werkbalken

De verzameling Werkbalken bevat alle standaardwerkbalken en aangepaste werkbalken die in de Microsoft Excel-omgeving zijn gedefinieerd. Met de eigenschap **Aantal** van Werkbalken kunt u nagaan hoeveel werkbalken er beschikbaar zijn. Een object Werkbalk definieert één standaardwerkbalk of aangepaste werkbalk.

Met de methode **Onderdeel** kunt u een verwijzing naar een bepaalde werkbalk genereren, gegeven het indexnummer of de naam van de werkbalk. U kunt beter geen numerieke index bij de methode **Onderdeel** gebruiken, omdat het indexnummer van een werkbalk kan veranderen en omdat iedere werkbalk een naam heeft.

Met de eigenschap **Naam** kunt u de naam van een werkbalk instellen of genereren. Als het een standaardwerkbalk betreft, is de eigenschap **Naam** alleen-lezen. Met de eigenschap **Ingebouwd** kunt u bepalen of een werkbalk een standaardwerkbalk of een aangepaste werkbalk is.

Een nieuwe werkbalk maken

Met de methode **Toevoegen** maakt u een nieuwe werkbalk. U moet een naam opgeven. Doet u dit niet, dan krijgt de werkbalk de naam "Werkbalk *n*", waarin *n* een nummer is.

In het volgende voorbeeld wordt de werkbalk MijnToepWerkbalk gemaakt maar niet weergegeven.

```
Werkbalken.Toevoegen "MijnToepWerkbalk"
```

Het uiterlijk van een werkbalk wijzigen

Werkbalken hebben een aantal eigenschappen die u kunt gebruiken om het formaat te wijzigen, ze te koppelen aan de boven-, onder-, linker- of rechterkant van de werkruimte of om ze elders op het scherm te plaatsen (mits het *zwevende* werkbalken zijn).

Met de eigenschap **Breedte** kunt u de breedte van de werkbalk instellen of genereren, uitgedrukt in punten. Bij gebruik in een procedure resulteert deze eigenschap in de actuele breedte van de werkbalk. Met de eigenschap **Breedte** kunt u ook het formaat van een werkbalk wijzigen: zowel de hoogte als de breedte worden gewijzigd wanneer de werkbalk de opgegeven breedte aanneemt. Als de eigenschap **Breedte** door een procedure wordt ingesteld, neemt de werkbalk zoveel mogelijk de opgegeven breedte aan terwijl alle knoppen worden weergegeven.

Met de eigenschap **Hoogte** kunt u de hoogte van een werkbalk genereren, uitgedrukt in punten.

Met de eigenschap **Positie** wordt de positie van de werkbalk ingesteld of gegenereerd. Een werkbalk kan gekoppeld zijn aan een van de vier randen van de werkruimte (xlBoven, xlLinks, xlRechts, xlBeneden) of zweven (xlZwevend). Een werkbalk die een vervolglijst bevat kan niet gekoppeld zijn aan de linkerrand of rechterrand van de werkruimte.

Als de werkbalk gekoppeld is, kunt u met de eigenschap **Boven** de afstand instellen of genereren, uitgedrukt in punten, van de bovenrand van de werkbalk tot de bovenrand van het gebied waaraan de werkbalk is gekoppeld. Als de werkbalk zwevend is (dat wil zeggen dat de eigenschap **Positie** xlZwevend is), kunt u met de eigenschap **Boven** het aantal punten instellen of genereren vanaf de bovenrand van de Microsoft Excel-werkruimte tot aan de bovenrand van de werkbalk.

Als de werkbalk gekoppeld is, kunt u met de eigenschap **Links** de afstand instellen of genereren, uitgedrukt in punten, van de linkerrand van de werkbalk tot de linkerrand van het gebied waaraan de werkbalk is gekoppeld. Als de werkbalk zwevend is (dat wil zeggen dat de eigenschap **Positie** xlZwevend is), kunt u met de eigenschap **Links** het aantal punten instellen of genereren vanaf de linkerrand van de Microsoft Excel-werkruimte tot aan de linkerrand van de werkbalk.

Een werkbalk weergeven

Nadat u de aanvangspositie van een werkbalk hebt ingesteld, kunt u de werkbalk weergeven door de eigenschap **Zichtbaar** in te stellen op **Waar**. Dit komt overeen met het inschakelen van het aankruisvakje naast de naam van de werkbalk in het dialoogvenster **Werkbalken** en het kiezen van de knop OK. U kunt ook bepalen of een werkbalk wordt weergegeven door de waarde van de eigenschap **Zichtbaar** te lezen.

In het volgende voorbeeld wordt de werkbalk geplaatst en weergegeven door de procedure die is toegewezen aan de menu-opdracht MijnToepWerkbalk in het menu Beeld.

```
Sub MijnToepWerkbalkZien()
    Met Menubalken(xlWerkblad).Menus("Beeld").MenuOpdrachten( _
"MijnWerkbalk bekijken")
        .Ingeschakeld = Niet.Ingeschakeld
        Indien .Ingeschakeld Dan
            'De werkbalk weergeven
            Werkbalken("MijnToepWerkbalk").Zichtbaar = Waar
        Anders
            'De werkbalk verbergen
            Werkbalken("MijnToepWerkbalk").Zichtbaar = Onwaar
        Einde Indien
    Einde Met
Einde Sub
```

Als een werkbalk zichtbaar is, kan de gebruiker alle knoppen kiezen om de bijbehorende toegewezen procedures te starten.

Een werkbalk verwijderen

Met de methode **Verwijderen** kunt u een aangepaste werkbalk verwijderen uit de Microsoft Excel-werkruimte. Deze methode is niet van toepassing op standaardwerkbalken. In het volgende voorbeeld wordt de werkbalk MijnToepWerk verwijderd.

```
Werkbalken("MijnToepWerk").Verwijderen
```

Het verwijderen van werkbalken heeft alleen betrekking op de Microsoft Excel-werkruimte. Een werkbalk die bij een werkmap hoort, moet handmatig worden gewist met behulp van het dialoogvenster **Werkbalken toevoegen**, zoals verderop in dit hoofdstuk wordt besproken.

Een standaardwerkbalk herstellen in de oorspronkelijke staat

Als een standaardwerkbalk door een gebruiker of procedure is gewijzigd, kunt u de werkbalk in de oorspronkelijke staat herstellen met de methode **OpnieuwInstellen**. Deze methode komt overeen met het kiezen van de opdracht **Werkbalken** in het menu **Beeld**, het inschakelen van het aankruisvakje naast de naam van de aangepaste standaard werkbalk in het dialoogvenster **Werkbalken** en het kiezen van de knop Opnieuw instellen.

In het volgende voorbeeld worden alle werkbalken in de oorspronkelijke staat hersteld en tegelijk alle aangepaste werkbalken verwijderd.

```
Voor Elke DezeWerkbalk In Werkbalken
    Indien DezeWerkbalk.Ingebouwd Dan
        DezeWerkbalk.OpnieuwInstellen
    Anders
        DezeWerkbalk.Verwijderen
    Einde Indien
Volgende
```

Werken met werkbalkknoppen

De verzameling WerkbalkKnoppen bevat alle werkbalkknoppen die zich op een bepaalde werkbalk bevinden. De eigenschap **Aantal** geeft als resultaat het aantal knoppen op een werkbalk, inclusief de tussenruimtes. De methode **Onderdeel** geeft als resultaat één knop op een werkbalk. Een object WerkbalkKnop definieert één knop op een werkbalk.

Met de eigenschap **Ingebouwd** kunt u bepalen of een bepaalde werkbalkknop een standaardknop is. De eigenschap resulteert in **Waar** als de knop een standaardknop is. Nadat in een procedure is bepaald dat een werkbalkknop standaard is, kunt u met de eigenschap **ID** het identificatienummer van de knop achterhalen. De eigenschap **Naam** resulteert in de naam (tekst) van een standaardwerkbalkknop.

Zie bijlage D, "Werkbalkknoppen in Microsoft Excel", voor een lijst met standaardwerkbalkknoppen en de bijbehorende identificatienummers.

Een werkbalkknop toevoegen aan een werkbalk

Met de methode **Toevoegen** kunt u een nieuwe knop toevoegen aan een bestaande standaardwerkbalk of aangepaste werkbalk. In de argumenten bij deze methode wordt het volgende opgegeven voor standaardknoppen:

- Naam of identificatienummer
- Positie op de werkbalk
- Naam van de te starten procedure wanneer de gebruiker de knop kiest
- Of de knop aanvankelijk als lichter gekleurd of ingedrukt wordt weergegeven

In het volgende voorbeeld worden twee knoppen, gescheiden door een tussenruimte, toegevoegd aan de werkbalk MijnToepWerk.

```
Toewijzen knppn = Werkbalken("MijnToepWerk").WerkbalkKnoppen
' Knop met afbeelding van klok toevoegen
knppn.Toevoegen _
    Knop:=213; _
    Voor:=1; _
    BijActie:="Module1.MijnAgenda"; _
    Geactiveerd:=Waar; _
    Ingedrukt:=Onwaar
' Knop met afbeelding van vuilnisbak toevoegen
knppn.Toevoegen _
    Knop:=225; _
    Voor:=2; _
    BijActie:="Module1.MijnLegeBak"; _
    Geactiveerd:=Waar; _
    Ingedrukt:=Onwaar
' Ruimte tussen de knoppen toevoegen
knppn.Toevoegen knop:=0; voor:=2
```

Als u geen positie opgeeft, wordt de werkbalkknop toegevoegd aan het einde van de werkbalk. Als u geen procedure opgeeft die wordt gestart wanneer de gebruiker de knop kiest, wordt in geval van een standaardwerkbalkknop de standaardbewerking uitgevoerd.

Werkbalkknoppen kopiëren en verplaatsen

Met de methode **Kopiëren** kunt u een knop naar een plaats op dezelfde werkbalk of op een andere werkbalk kopiëren. De methode **Kopiëren** heeft als argument de bestemming van de werkbalkknop, uitgedrukt als een object Werkbalk en een knoppositienummer. De knoppen op de bestemmingswerkbalk worden naar rechts (of beneden) opgeschoven om plaats te maken voor de gekopieerde knop. In het volgende voorbeeld wordt de werkbalkknop op de tiende positie van de werkbalk Standaard gekopieerd naar de vierde positie op de werkbalk MijnToepWerk.

```
Werkbalken(xlStandaard).WerkbalkKnoppen(10)_
    .Kopiëren Werkbalken("MijnToepWerk"); 4
```

Als u een knop naar de laatste positie op een werkbalk wilt kopiëren, telt u 1 op bij de waarde die door de eigenschap **Aantal** is gegeven.

Als u een werkbalkknop wilt verplaatsen in plaats van kopiëren, kunt u de methode **Verplaatsen** gebruiken, die net als de methode **Kopiëren** als argumenten werkbalk- en positiebestemmingen heeft.

In het volgende voorbeeld wordt dezelfde werkbalkknop verplaatst als in het vorige voorbeeld, maar wordt tevens het originele exemplaar verwijderd.

```
Werkbalken(xlStandaard).WerkbalkKnoppen(10)_
    .Verplaatsen Werkbalken("MijnToepWerk"); 4
```

Een werkbalkknop verwijderen uit een werkbalk

Met de methode **Verwijderen** kunt u een knop verwijderen uit een werkbalk. In het volgende voorbeeld wordt de derde knop verwijderd uit de werkbalk Standaard.

```
Werkbalken("Standaard").WerkbalkKnoppen(3).Verwijderen
```

Het uiterlijk van een werkbalkknop wijzigen

Nadat u een knop hebt gemaakt en op een werkbalk hebt geplaatst, kunt u het uiterlijk ervan op verscheidene manieren wijzigen. U kunt de knop beschikbaar maken voor de gebruiker, u kunt de knop ingedrukt weergeven of u kunt de afbeelding op de knop wijzigen.

Met de eigenschap **Geactiveerd** kunt u instellen of genereren of een knop geactiveerd (kan worden gekozen) of uitgeschakeld is (is lichter gekleurd en geeft een pieptoon wanneer de gebruiker erop klikt). In het volgende voorbeeld wordt de derde knop op de werkbalk Standaard uitgeschakeld.

```
Werkbalken("Standaard").WerkbalkKnoppen(3).Geactiveerd = Onwaar
```

Met de eigenschap **Ingedrukt** kunt u instellen of genereren of een knop ingedrukt wordt weergegeven. De eigenschap **Ingedrukt** resulteert in **Waar** als de knop ingedrukt wordt weergegeven. Deze eigenschap kan gebruikt worden (analoog aan het vinkje naast een menu-opdracht) om aan te geven dat een optie zich momenteel in één van twee mogelijke toestanden bevindt. In het volgende voorbeeld verandert de procedure, die is toegewezen aan een nieuwe werkbalkknop DatabaseWeergave, het uiterlijk van de knop alvorens te schakelen tussen speciale weergaven van het werkblad.

```
Sub DatabaseWeergave()
    Met Werkbalken("MijnToepWerkbalk").WerkbalkKnoppen(3)
        .Ingedrukt = Niet.Ingedrukt
        Indien .Ingedrukt Dan
            'Naar database-weergave schakelen
            .
            .
            .
        Anders
            'Naar werkbladweergave schakelen
            .
            .
            .
        Einde Indien
    Einde Met
Einde Sub
```

U kunt de gebruiker in staat stellen de afbeelding op een knop te wijzigen met de methode **Bewerken** voor de knop, die de Knop-editor activeert. In het volgende voorbeeld wordt de gebruiker in staat gesteld de afbeelding op de derde werkbalkknop te bewerken.

```
Werkbalken("Standaard").WerkbalkKnoppen(3).Bewerken
```

Met de methode **KnopvlakKopiëren** kunt u de bitmap-afbeelding op een werkbalkknop overbrengen naar het Klembord. Nadat u een knopvlak in het Klembord hebt geplaatst, door het te kopiëren van een andere toepassing of door het knopvlak van een bestaande werkbalkknop te kopiëren, kunt u met de methode **KnopvlakPlakken** een knopvlak van het Klembord op een bepaalde knop plakken.

Met de eigenschap **IngebouwdKnopvlak** kunt u nagaan of de knop het standaardknopvlak gebruikt of een aangepast knopvlak dat over het standaardknopvlak is heengeplakt. Als een procedure de waarde van deze eigenschap instelt op **Waar**, wordt het geplakte knopvlak verwijderd en krijgt de knop weer de standaardafbeelding.

Met de eigenschap **GroteKnoppen** van het object Toepassing kunt u opgeven dat er grotere knoppen moeten worden gebruikt. Grote knoppen zijn handig op grote beeldschermen of op beeldschermen die een groot aantal pixels per vierkante centimeter weergeven.

Werkbalken koppelen aan werkmappen

U kunt een werkbalk die u hebt gemaakt, van de Microsoft Excel-werkruimte verplaatsen naar een werkmap. Werkbalken op werkmapniveau vergemakkelijken het maken van een verzorgde gebruikers-interface voor een aangepaste toepassing, bijvoorbeeld een invoegfunctie, of het verplaatsen van aangepaste werkbalken en bijbehorende procedures van de ene omgeving naar de andere.

▶ **Een werkbalk verplaatsen naar een werkmap**

1. Terwijl er een Visual Basic-module actief is, kiest u **Werkbalken toevoegen** in het menu **Extra**.
2. Selecteer in het vak "Aangepaste werkbalken" de naam van de werkbalk die u wilt kopiëren naar de actieve werkmap.
3. Kies de knop Kopiëren.

   De naam van de werkbalk verschijnt in het vak "Werkbalken in werkmap".
4. Kies de knop OK.

U kunt ook de oorspronkelijke werkbalk op werkruimteniveau verwijderen door **Werkbalken** te kiezen in het menu **Beeld**, het aankruisvakje te selecteren naast de naam van de werkbalk die u wilt verwijderen en vervolgens de knop Verwijderen te kiezen. Als u het werkruimte-exemplaar van de werkbalk niet verwijdert, kunt u het wijzigen zonder gevolgen voor het exemplaar in de werkmap. U kunt natuurlijk de gewijzigde werkruimte-werkbalk weer naar de werkmap kopiëren en zodoende de kopie vervangen.

Nadat u een werkbalk naar een werkmap hebt gekopieerd, wordt de werkbalk pas beschikbaar als de gebruiker die werkmap heeft geopend. Een dergelijke werkbalk behoudt niet alleen de naam en de inhoud van het origineel, maar ook de toewijzing van programmacode aan knoppen, de lokatie, grootte en vorm ervan, de positie op het scherm (wel of niet gekoppeld) en het kenmerk zichtbaar of verborgen.

U kunt ook een werkbalk op werkmapniveau verwijderen.

▶ **Een werkbalk op werkmapniveau verwijderen**

1. Terwijl er een Visual Basic-module actief is, kiest u **Werkbalken toevoegen** in het menu **Extra**.
2. Selecteer in het vak "Werkbalken in werkmap" de naam van de werkbalk die u wilt verwijderen.
3. Kies de knop Verwijderen.
4. Kies de knop OK.

Waar werkbalken worden opgeslagen

Wanneer u Microsoft Excel afsluit, worden de werkbalken in de werkruimte opgeslagen in EXCEL5.XLB (in de directory \WINDOWS in Microsoft Excel voor Windows) of in EXCEL WERKBALKEN (5) (in de map VOORKEUREN in de systeemmap in Microsoft Excel voor de Macintosh). De werkbalken bij een werkmap worden opgeslagen in het werkmapbestand.

Als u een werkmap opent die een of meer werkbalken bevat, stelt Microsoft Excel eerst vast of er al een werkruimtewerkbalk met die naam voorkomt. Is dit niet het geval, dan wordt er een nieuwe werkruimtewerkbalk gemaakt en wordt de werkmapwerkbalk ernaartoe gekopieerd. Op deze wijze krijgt de gebruiker een nieuwe kopie van de werkbalk en kan hij de balk wijzigen door deze te verbergen of door werkbalkknoppen te kopiëren van en naar het werkruimte-exemplaar van de werkbalk. Als de gebruiker Microsoft Excel afsluit, worden de wijzigingen aan dit exemplaar opgeslagen in EXCEL5.XLB of EXCEL WERKBALKEN (5).

Omdat de naam van de werkbalk niet kan worden gewijzigd, wordt de werkbalk niet opnieuw gekopieerd vanuit de werkmap wanneer de werkmap of invoegmacro weer wordt geopend. De werkruimte bevat namelijk al een werkbalk met die naam. Maar de procedures die horen bij de werkbalkknoppen in de geopende werkmap worden wel gestart als de gebruiker die werkbalkknoppen kiest.

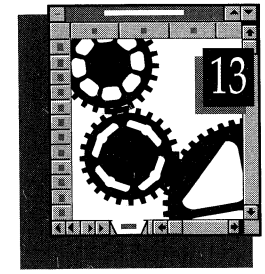
Doordat werkbalken op deze wijze worden opgeslagen en gekopieerd, kunnen ontwikkelaars van invoegmacro's werkmappen distribueren die ongewijzigde exemplaren van werkbalken bevatten, zodat gebruikers werkbalkknoppen kunnen verplaatsen naar hun eigen aangepaste werkbalken. Een ontwikkelaar van een invoegwerkmap kan een werkbalk ontwerpen met het dialoogvenster **Werkbalk** en de Knop-editor en kan vervolgens de werkbalk koppelen aan de invoegwerkmap met het dialoogvenster **Werkbalken toevoegen**.

Als de gebruiker de invoegwerkmap opent, verschijnt de aangepaste invoegwerkbalk. De gebruiker kan deze vervolgens bewerken en werkbalkknoppen verplaatsen naar persoonlijke werkbalken, zonder dat dit effect heeft op het exemplaar dat is opgeslagen in de invoegwerkmap. De gewijzigde werkbalken van de gebruiker worden opgeslagen in EXCEL5.XLB of EXCEL WERKBALKEN (5) wanneer de gebruiker Microsoft Excel afsluit. Als de gebruiker Microsoft Excel opnieuw start, verschijnt de gewijzigde werkbalk. Wanneer de gebruiker een van de werkbalkknoppen van de ontwikkelaar kiest, wordt de invoegwerkmap geladen die de procedure bevat die gekoppeld is aan de werkbalkknop. Als de gebruiker een nieuwe kopie van de werkmapwerkbalk wenst, kan hij het gewijzigde exemplaar verwijderen.

In het volgende hoofdstuk wordt met een uitleg van de begrippen *gebeurtenis*, *automatische procedure* en *invoegoplossing* de discussie over het maken van aangepaste toepassingen afgerond.

HOOFDSTUK 13

Automatische procedures en invoegmacro's maken



In dit hoofdstuk wordt de laatste belangrijke vorm van interactie tussen de gebruiker en Microsoft Excel, en tussen Microsoft Excel en andere toepassingen behandeld. Centraal bij deze vorm van interactie staan gebeurtenissen. Een *gebeurtenis* is een actie, zoals het openen van een werkmap, het overschakelen op een blad, het drukken op een bepaalde toetsencombinatie of het herberekenen van een werkblad. Sommige gebeurtenissen worden in gang gezet door de gebruiker, andere door Microsoft Excel of door andere toepassingen. U kunt de manier waarop de interactie tussen gebruikers en uw toepassing plaatsvindt, wijzigen en de mogelijkheden die hun hierbij ter beschikking staan uitbreiden door aan gebeurtenissen procedures toe te wijzen.

Gebeurtenissen kunnen worden onderverdeeld in drie hoofdcategorieën, afhankelijk van de manier waarop u de gebeurtenis aan een procedure koppelt.

- Met de opdracht **Toewijzen aan object** in het menu **Extra** kunt een procedure koppelen aan het kiezen van een knop of een ander object, zoals is beschreven in de voorgaande hoofdstukken.
- U kunt ervoor zorgen dat een procedure automatisch wordt uitgevoerd wanneer een bepaalde gebeurtenis plaatsvindt door de naam van de procedure te laten beginnen met Auto_.
- U kunt een AlsGebeurtenis-procedure maken (bijvoorbeeld met de eigenschap **BijVenster** of de eigenschap **BijBerekenen**) die wordt uitgevoerd als de daaraan gekoppelde gebeurtenis plaatsvindt.

In dit hoofdstuk leest u ook hoe u werkbladen, besturingselementen, menu's, werkbalken en ondersteunende Visual Basic-modulen kunt samenvoegen tot een invoegmacro. Met invoegmacro's kunt u aangepaste voorzieningen samenstellen en verspreiden, die voor andere gebruikers lijken te zijn ingebouwd in Microsoft Excel.

Inhoud

- Automatische procedures maken
- AlsGebeurtenis-procedures maken
- Een invoegmacro maken
- Er een geheel van maken

Automatische procedures maken

Een *automatische procedure* wordt opgeslagen in een werkmap en wordt automatisch uitgevoerd, bijvoorbeeld als de werkmap wordt geopend of gesloten. U kunt automatische procedures in een willekeurige Visual Basic-module in de werkmap plaatsen. Automatische procedures op werkmapniveau worden herkend aan hun namen zoals deze in de module verschijnen, te weten Auto_openen of Auto_sluiten.

▶ **Een automatische procedure definiëren**

1. Schakel over op de Visual Basic-module waarin u de automatische procedure wilt opslaan.
2. Maak een nieuwe procedure met de naam Auto_openen of Auto_sluiten.
3. Schrijf de programmacode van de procedure.

U kunt een Auto_openen- of Auto_sluiten-procedure testen zonder de werkmap expliciet te openen of te sluiten. Hiertoe kiest u de opdracht **Macro** in het menu **Extra**, selecteert u de naam van de procedure en kiest u de knop Starten.

U kunt per werkmap maximaal één Auto_openen- en één Auto_sluiten-procedure gebruiken. Als er meerdere Auto_sluiten- en Auto_openen-procedures bestaan, wordt geen van deze procedures uitgevoerd.

Auto_openen-procedures

Een Auto_openen-procedure wordt uitgevoerd als een werkmap wordt geopend. U kunt een Auto_openen-procedure gebruiken om menubalken in te stellen (als u menubalken hebt gemaakt met een procedure in plaats van met de Menu-editor), de werkmap aan te passen aan het besturingssysteem waarmee u werkt, een aangepast beginscherm weer te geven of om koppelingen met andere bestanden of andere toepassingen aan te brengen.

De volgende procedure voegt bijvoorbeeld automatisch een aangepaste menuopdracht toe aan het menu **Extra** van het werkblad.

```
Sub Auto_openen()
    Menubalken(xlWerkblad).Menus("Extra").MenuOpdrachten.Toevoegen _
        Bijschrift := "Mijn Analyse"; _
        BijActie := DezeWerkmap.Naam & "!Module2.ProcMijnAnalyse"; _
        Voor := 1
Einde Sub
```

Auto_sluiten-procedures

Een Auto_sluiten-procedure wordt uitgevoerd vlak voordat de werkmap met de procedure wordt gesloten. U kunt met deze voorziening menubalken, werkbalken en andere elementen van de gebruikers-interface weer in de oorspronkelijke staat terugbrengen en bestanden die een toepassing ondersteunen, opslaan en sluiten.

Met een Auto_sluiten-procedure kunt u er ook voor zorgen dat de communicatie met een andere toepassing op een ordelijke manier wordt beëindigd. Als uw toepassing bijvoorbeeld gebruik maakt van Windows Terminal om aandelenkoersen van een externe informatiedienst te laden, wilt u waarschijnlijk dat Terminal afmeldt bij de informatiedienst voordat de werkmap wordt gesloten, zoals in de volgende procedure.

```
Sub Auto_sluiten()
    Menubalken(xlWerkblad).Menus("Extra"). _
        MenuOpdrachten("Mijn Analyse").Verwijderen
    LogbestandSluiten           'Procedure: mutatiebestand sluiten.
    'Procedure: verbinding met informatiedienst beëindigen.
    AfmeldenBijInfodienst
Einde Sub
```

Automatische procedures met een gedefinieerde naam gebruiken

Microsoft Excel ondersteunt automatische procedures die zijn gekoppeld aan specifieke werkbladen in plaats van aan de hele werkmap. In de meeste gevallen kunt u beter standaard automatische procedures gebruiken in plaats van procedures met een gedefinieerde naam. Gebruikt u namelijk een procedure met een gedefinieerde naam, dan moet u voor ieder werkblad dat u wilt ondersteunen expliciet procedures definiëren en zijn deze procedures niet beschikbaar voor grafiekbladen en Visual Basic-modulen.

▶ Een automatische procedure met een gedefinieerde naam maken

1. Schakel over op het werkblad waarvoor u een automatische procedure wilt definiëren.

2. Kies de opdracht **Naam** in het menu **Invoegen** en kies vervolgens de opdracht **Bepalen**.
3. Typ in het vak "Naam" een naam die begint met Auto_openen, Auto_sluiten, Auto_activeren of Auto_desactiveren. Laat de naam voorafgaan door de bladnaam gevolgd door een uitroepteken, bijvoorbeeld **Blad1!Auto_activeren_mijnactiveren**.
4. Typ in het vak "Verwijst naar" een "is gelijk"-teken en de naam van de procedure die u aan de gedefinieerde naam wilt koppelen, zoals **=Map1!Module1.ControleerGegevens**.

   Deze procedure wordt uitgevoerd als het werkblad wordt geactiveerd of gedesactiveerd of wanneer de werkmap wordt geopend of gesloten. U kunt dezelfde naam gebruiken die u in het vak "Naam" hebt getypt of een andere naam.
5. Kies de knop Toevoegen.

Het verschil tussen automatische procedures met een gedefinieerde naam en standaard automatische procedures is dat in procedures met een gedefinieerde naam de naam die wordt bepaald in het werkblad alleen hoeft te beginnen met de naam van het type procedure dat u bij een gebeurtenis wilt uitvoeren. U kunt de namen Auto_openen_Openbaar, Auto_openen_BeginScherm, en Auto_openen_Bestanden in hetzelfde werkblad definiëren. Wanneer de werkmap met dat werkblad wordt geopend, worden de procedures uitgevoerd die aan deze drie namen zijn gekoppeld. Als u wilt voorkomen dat de procedure per ongeluk of opzettelijk wordt gewijzigd, kunt u een gedefinieerde naam verbergen.

De uitvoering van standaard automatische procedures heeft voorrang op de uitvoering van procedures met een gedefinieerde naam. Stel bijvoorbeeld dat u een Auto_openen-procedure in een module plaatst en in een bepaald werkblad de naam Auto_openen_DitWerkblad definieert als de tekst Openingswerkblad. Als de gebruiker de werkmap opent, wordt eerst op werkbladniveau de Auto_openen-procedure en daarna de procedure Openingswerkblad uitgevoerd. U kunt de Auto_openen-procedures die in afzonderlijke werkbladen zijn gedefinieerd ook niet in iedere gewenste volgorde laten uitvoeren.

Opmerking U kunt voorkomen dat een automatische procedure wordt uitgevoerd door op SHIFT te drukken als u een werkblad of een werkmap opent, sluit of erop overschakelt. Zo kunt u op SHIFT drukken terwijl u een toepassing opent die in ontwikkeling is, om te voorkomen dat de Auto_openen-procedures van de toepassing worden uitgevoerd.

Automatisch een werkmap openen bij het starten van Microsoft Excel

Als een werkmap in de Microsoft Excel-opstartdirectory of -opstartmap is geplaatst, wordt de werkmap automatisch geopend bij het starten van Microsoft Excel. De opstartdirectory in Microsoft Excel voor Windows is XLSTART. Voor de Macintosh heet de opstartmap EXCEL OPSTARTMAP. Zie hoofdstuk 35, "Bepalen wat er gebeurt als u Microsoft Excel start", in het *Microsoft Excel handboek* voor meer informatie over het gebruik van een opstartdirectory of opstartmap.

U kunt met de eigenschap **Opstartpad** van het toepassingsobject het pad van de opstartdirectory opvragen, met de eigenschap **AltOpstartpad** het pad van de alternatieve opstartdirectory of -map instellen of opvragen, en met de eigenschap **Pad** het pad lezen waarin Microsoft Excel zich bevindt.

In Microsoft Excel voor Windows hebben deze eigenschappen de vorm `C:\EXCEL`. In Microsoft Excel voor de Macintosh luidt de vorm van deze eigenschappen `HD80:MICROSOFT EXCEL`.

Met de eigenschap **PadScheidingsteken** van het toepassingsobject vraagt u het juiste scheidingsteken op voor het opgeven van de lokatie van de directory's in Windows (\) en mappen op de Macintosh (:).

BijGebeurtenis-procedures maken

Een *BijGebeurtenis-procedure* (ook wel een Gebeurtenisafhandeling genoemd) is een procedure die wordt uitgevoerd wanneer zich een opgegeven gebeurtenis voordoet. Dit type gebeurtenis is een actie die in gang wordt gezet door de Microsoft Excel-omgeving, maar niet noodzakelijkerwijs door de gebruiker. Voorbeelden van een dergelijke actie zijn het verstrijken van een opgegeven tijdsinterval, het activeren van een venster, het herberekenen van een werkblad of het aanslaan van een bepaalde reeks toetsen.

In hoofdstuk 11, "Besturingselementen en dialoogvensters", hebt u al kennis gemaakt met één type AlsGebeurtenis-procedure. In dat hoofdstuk werd behandeld hoe u een procedure toewijst aan de **BijActie**-eigenschap van een besturingselement.

In hoofdstuk 9, "Foutafhandeling en foutwaarden", en hoofdstuk 10, "Samenwerken met andere toepassingen", wordt eveneens aandacht besteed aan AlsGebeurtenis-procedures. In hoofdstuk 9 wordt uiteengezet hoe met behulp van de instructie **Bij Fout** verschillende typen fouten kunnen worden onderschept tijdens de uitvoering van een Visual Basic-procedure. In hoofdstuk 10 wordt uitgelegd hoe met de methode **KoppelingBepalenBijGegevens** gegevens uit toepassingen buiten Microsoft Excel kunnen worden opgevraagd.

Een AlsGebeurtenis-procedure wordt op soortgelijke wijze gemaakt en gebruikt, ongeacht het type gebeurtenis.

▶ Een BijGebeurtenis-procedure definiëren

1. Schrijf een procedure (de Gebeurtenisafhandeling zelf) die moet worden uitgevoerd als de gebeurtenis plaatsvindt.
2. Gebruik in een andere procedure een Visual Basic-methode of -eigenschap (bijvoorbeeld de eigenschap **BijTijd**) in een instructie die de gebeurtenis koppelt aan de bijbehorende Gebeurtenisafhandeling.

Nadat de instructie is uitgevoerd die de gebeurtenis aan de bijbehorende afhandelingsroutine koppelt, wordt de afhandelingsroutine telkens uitgevoerd wanneer de gebeurtenis plaatsvindt. Dit wordt het *onderscheppen* van een gebeurtenis genoemd. Als u de onderschepping van gebeurtenissen wilt uitschakelen, gebruikt u dezelfde Visual Basic-methode of -eigenschap om de gebeurtenis en de bijbehorende afhandelingsroutine te ontkoppelen.

In de volgende gedeelten ziet u hoe verschillende typen gebeurtenissen worden onderschept.

De methode BijTijd

Een BijTijd-Gebeurtenisafhandeling wordt op een opgegeven datum en tijd uitgevoerd, mits Microsoft Excel actief is en de werkmap met de BijTijd-procedure in het geheugen is geladen. Met de methode **BijTijd** van het toepassingsobject kunt u een BijTijd-Gebeurtenisafhandeling uitvoeren op een bepaald tijdstip of als een bepaalde periode is verstreken. U kunt het tijdstip waarop u de afhandelingsroutine wilt uitvoeren en de naam van de afhandelingsroutine opgeven. Wilt u bijvoorbeeld dagelijks om twaalf uur 's middags een serie rapporten samenstellen en afdrukken, dan kunt u de volgende programmacode gebruiken:

```
'Onderschepping initialiseren met methode BijTijd
Sub OnderschepTijd()
'BijTijd-argumenten instellen
    Toepassing.BijTijd _
        VroegsteTijd := Tijdwaarde("12:00:00"); _
        Procedure := "RapportenVerzorgen"
Einde Sub
```

```
Sub RapportenVerzorgen()                    'BijTijd afhandelingsroutine
    RapportenSamenstellen
    RapportenAfdrukken
Einde Sub
```

De procedure OnderschepTijd initialiseert de onderschepping van de **BijTijd**-gebeurtenis. De eerste keer dat het twaalf uur 's middags is, wordt na de uitvoering van OnderschepTijd de RapportenVerzorgen-afhandelingsroutine gestart. U kunt ervoor zorgen dat de procedure slechts één keer wordt uitgevoerd door het argument *VroegsteTijd* in te stellen op een datum en een tijd in plaats van alleen op een tijd.

Als op het moment dat de gebeurtenis zich voordoet een andere procedure in uitvoering is, wacht Microsoft Excel gedurende een extra tijdsinterval dat is opgegeven met het argument *LaatsteTijd* van de methode **BijTijd**. De volgende versie van de OnderschepTijd-procedure wacht bijvoorbeeld een half uur en beëindigt de gebeurtenis als Microsoft Excel in dit half uur niet beschikbaar komt.

```
Sub OnderschepTijd()                    'Onderschepping BijTijd initialiseren.
    Toepassing.BijTijd _
        VroegsteTijd := Tijdwaarde("12:00:00"); _
        Procedure := "MijnBijTijdAfhandelingsroutine"; _
        LaatsteTijd := Tijdwaarde("12:30:00")
Einde Sub
```

Als u het argument *LaatsteTijd* niet opgeeft, wordt de procedure uitgevoerd bij de eerstvolgende gelegenheid dat Microsoft Excel beschikbaar is. U kunt meer dan één BijTijd-afhandelingsroutine programmeren op verschillende tijdstippen. Als u meer dan één BijTijd-afhandelingsroutine op hetzelfde tijdstip programmeert, wordt de procedure die u het laatst hebt geprogrammeerd het eerst uitgevoerd.

Tip Een **BijTijd**-gebeurtenis geldt slechts voor de duur van een sessie. Als u wilt dat de gebeurtenis wordt opgeslagen, kunt u in de opstartdirectory of -map een werkmap opnemen met daarin zowel de BijTijd-afhandelingsroutine zelf als een Auto_openen-procedure. De Auto_openen-procedure moet de onderschepping van de BijTijd-gebeurtenis voor een bepaalde tijdsduur initialiseren. De onderschepping van die gebeurtenis wordt dan telkens opnieuw geïnitialiseerd, wanneer Microsoft Excel wordt gestart.

Als u een **BijTijd**-gebeurtenis wilt verwijderen die nog gaat plaatsvinden, geeft u de exacte tijd van de gebeurtenis en de naam van de bijbehorende afhandelingsroutine op en stelt u het argument *schema* in op **Onwaar**. Staan er bijvoorbeeld om twaalf uur 's middags twee **BijTijd**-gebeurtenissen op het programma, waarvan de ene wordt afgehandeld door RapportenVerzorgen en de andere door een procedure met de naam Opruimen, dan kunt u de afhandelingsverzoek Opruimen stoppen met de volgende programmacode.

```
Toepassing.BijTijd _
    VroegsteTijd := Tijdwaarde("12:00:00"); _
    Procedure := "Opruimen"; _
    Schema := Onwaar
```

Opmerking Met de methode **BijTijd** kan de gebruiker werken totdat een **BijTijd**-gebeurtenis plaatsvindt. De methode **BijTijd** verschilt hierin van de methode **Wachten**, waarbij geen interactie met Microsoft Excel mogelijk is. Als u alle activiteiten behalve afdrukken en herberekenen wilt opschorten, gebruikt u de methode **Wachten** zoals in de volgende instructie.

```
'15 sec. wachten
Toepassing.Wachten Nu + Tijdwaarde("00:00:15")
```

Zie *Wachten (methode)* in de online Visual Basic Naslaggids voor meer informatie over de methode **Wachten**.

De eigenschappen BijBladActiveren en BijBladDesactiveren

Een BijBladActiveren-afhandelingsroutine wordt telkens uitgevoerd wanneer de gebruiker overschakelt op een blad in een geopende werkmap. Deze Gebeurtenisafhandeling wordt niet uitgevoerd als het blad wordt geactiveerd door een procedure. U kunt de eigenschap **BijBladActiveren** instellen voor het toepassings- of werkmapobject of voor een willekeurig bladobject om een procedure uit te voeren als de gebruiker overschakelt op een willekeurig blad in de toepassing, een willekeurig blad in een bepaalde werkmap of een specifiek blad in een werkmap.

Met **BijBladActiveren** kunt u ervoor zorgen dat een werkbalk die bij een werkmap hoort telkens wordt weergegeven als de gebruiker op die werkmap overschakelt. De volgende procedure in een werkmap start bijvoorbeeld de onderschepping van de BijBladActiveren-gebeurtenis voor de werkmap en voert een procedure uit die een werkbalk Technische analyse weergeeft als de gebruiker overschakelt op de werkmap.

```
Sub GeefAnalyseWerkbalkWeer()
    DezeWerkmap.BijBladActiveren = "WMActiveringAfhandelingsroutine"
Einde Sub

Sub WMActiveringAfhandelingsroutine()
    Indien ActiefBlad.Naam = "Mijn_Geg" Dan
            Werkbalken("WerkbalkTechAnalyse").Zichtbaar = Waar
    Anders
            Werkbalken("WerkbalkTechAnalyse").Zichtbaar = Onwaar
    Einde Indien
Einde Sub
```

U kunt een BijBladActiveren-procedure ook gebruiken om een aangepaste menubalk weer te geven als de gebruiker overschakelt op een bepaald soort blad. U kunt er bijvoorbeeld voor zorgen dat automatisch de ingebouwde grafiekmenubalk wordt weergegeven als de gebruiker op een grafiekblad overschakelt. Een aangepaste grafiekmenubalk wordt alleen weergegeven als u een procedure maakt die wordt uitgevoerd wanneer de gebruiker op een grafiekblad overschakelt.

Op vergelijkbare wijze wordt een BijBladDesactiveren-procedure uitgevoerd als de gebruiker een werkblad of -map verlaat. BijBladDesactiveren-procedures zijn handig wanneer u de oorspronkelijke instellingen van een aangepaste gebruikersinterface wilt herstellen, bijvoorbeeld om menubalken, werkbalken en weergave-instellingen weer terug te brengen in hun oorspronkelijke staat.

Opmerking De volgende werkmap wordt actief voordat de BijBladDesactiveren-procedure in de vorige werkmap wordt uitgevoerd. Het gebruik van de eigenschap **ActiefBlad** of de eigenschap **ActieveWerkmap** in de BijBladDesactiveren-afhandelingsroutine kan daarom ongewenste resultaten opleveren.

De eigenschap BijVenster

Een BijVenster-Gebeurtenisafhandeling wordt uitgevoerd als de gebruiker overschakelt op het opgegeven venster of wanneer de gebruiker het venster van de Microsoft Excel-toepassing activeert of opent. Vergeet niet dat u een hele werkmap bekijkt via een venster. (Indien u een procedure wilt toewijzen aan een enkel blad in een werkmap, kunt gebruik maken van een Auto_activeren-procedure).

De volgende programmacode bevat bijvoorbeeld een procedure die telkens wanneer de gebruiker overschakelt op een tweede venster met de naam FietsDB, dit venster op een bepaalde lokatie op het scherm laat verschijnen.

```
Sub OnderschepFiets2()
    Vensters("FietsDB.XLS:2").BijVenster = "VensterPlaatsen"
Einde Sub

Sub VensterPlaatsen()
Vensters("FietsDB.XLS:2").VensterStatus = xlNormaal
    Met Vensters("FietsDB.XLS:2")
        .Links = 0
        .Boven = 100
        .Breedte = 300
        .Hoogte = 50
    Einde Met
Einde Sub
```

Als u een venster en de bijbehorende BijVenster-afhandelingsroutine wilt ontkoppelen, stelt u de eigenschap **BijVenster** in op `""`, zoals in het volgende voorbeeld.

```
Vensters("FietsDB.XLS:2").BijVenster = ""
```

Als u een procedure wilt koppelen aan het overschakelen op een willekeurig venster in de Microsoft Excel-toepassing, gebruikt u de eigenschap **BijVenster** van het toepassingsobject, zoals in het volgende voorbeeld:

```
Toepassing.BijVenster = "AlleVenstersAfhandelingsroutine"
```

Als een werkmap, werkblad of ander blad is gekoppeld aan een BijBladActiveren-procedure, wordt deze procedure uitgevoerd na de procedure die is opgegeven bij de eigenschap **BijVenster**.

De methode BijToets

Een BijToets-Gebeurtenisafhandeling wordt uitgevoerd als de gebruiker op een bepaalde toetsencombinatie drukt. U koppelt een procedure aan een opgegeven toetsencombinatie door gebruik te maken van de methode **BijToets** van het toepassingsobject. De volgende programmacode voert bijvoorbeeld de procedure RapportenVerzorgen uit als de gebruiker op F12 drukt.

```
Sub OnderschepToetsen()                    'Toetsencombinaties onderscheppen.
    Toepassing.BijToets _
        toets := "{F12}"; _
        Procedure := "RapportenVerzorgen"
Einde Sub

Sub RapportenVerzorgen()                   'F12 BijToets-afhandelingsroutine.
    RapportenSamenstellen
    AfdrukkenRapporten
Einde Sub
```

Als u een complete lijst van speciale codes voor toetsencombinaties wilt raadplegen, zoekt u in de online Visual Basic Naslaggids naar *BijToets-methode.*

Opmerking Een BijToets-afhandelingsroutine wordt niet uitgevoerd als de bijbehorende toetsencombinatie wordt aangeslagen terwijl een andere procedure in uitvoering is. U kunt een BijToets-afhandelingsroutine dus niet gebruiken als u een speciale procedure wilt uitvoeren wanneer de gebruiker op ESC drukt om een andere procedure te stoppen die op dat moment wordt uitgevoerd. (In Microsoft Excel voor de Macintosh kan de gebruiker een procedure ook beëindigen door op COMMAND+PUNT te drukken).

De eigenschap BijBerekenen

Een BijBerekenen-Gebeurtenisafhandeling wordt direct uitgevoerd nadat een werkblad is herberekend. Met de eigenschap **BijBerekenen** van het werkblad- of toepassingsobject kunt u een procedure koppelen aan de herberekening van één opgegeven werkblad of aan de herberekening van een willekeurig geopend werkblad.

U kunt een BijBerekenen-afhandelingsroutine bijvoorbeeld gebruiken om kolombreedten bij te werken wanneer nieuwe gegevens worden herberekend, zoals te zien is in het volgende voorbeeld.

```
Sub OnderschepBerek()
    Toepassing.BijBerekenen = "PasKolommenAan"
Einde Sub

Sub PasKolommenAan()
    Kolommen("A:H").GeheleKolom.AanSelectieAanpassen
Einde Sub
```

Als u een procedure hebt toegewezen aan de eigenschap **BijBerekenen** van het toepassingsobject, wordt de procedure uitgevoerd voor elk herberekend werkblad dat niet voorzien is van een eigen BijBerekenen-afhandelingsroutine. Als u de herberekening en de afhandelingsroutine wilt ontkoppelen, stelt u de bijbehorende eigenschap **BijBerekenen** in op `""`.

De eigenschap BijInvoer

Een BijInvoer-afhandelingsroutine wordt uitgevoerd als de gebruiker gegevens invoert in een werkblad. Met de eigenschap **BijInvoer** van het werkblad- of toepassingsobject kunt u een procedure koppelen aan de invoer van gegevens in een werkblad. De gebeurtenis vindt plaats nadat de gebruiker gegevens heeft ingevoerd in een cel en op ENTER heeft gedrukt of een andere cel heeft geselecteerd. De gebeurtenis vindt niet plaats als de gebruiker de opdracht **Knippen** of **Plakken** in het menu **Bewerken** kiest en evenmin als de inhoud van een cel wordt gewijzigd door een andere procedure.

De volgende programmacode koppelt de gebeurtenis BijInvoer bijvoorbeeld aan een procedure die gegevens valideert die in een cel zijn ingevoerd, mits de cel zich in kolom B bevindt.

```
Sub OnderschepInvoer()
    ActieveWerkmap).Werkbladen("AlgDataDB").BijInvoer = _
        "KolBValideren"
Einde Sub

Sub KolBValideren()
    Met ActieveCel
        Indien .Kolom = 2 Dan        'Cel moet zich in kolom 2 bevinden.
            Indien IsNumeriek(.Waarde) Dan
                Indien .Waarde < 0 Of .Waarde > 255 Dan
                    BerichtVenster "Waarde moet tussen 0 en 255 liggen."
                    .Waarde = ""
                Einde Indien
            Anders
                'Niet-numerieke invoer afhandelen
                BerichtVenster "Waarde moet een getal tussen 0 en 255 _
zijn."
                .Waarde = ""
            Einde Indien
        Einde Indien
    Einde Met
Einde Sub
```

De eigenschap BijGegevens

Een BijGegevens-afhandelingsroutine wordt uitgevoerd als gegevens binnenkomen van een andere toepassing dan Microsoft Excel. U kunt de eigenschap **BijGegevens** van het werkblad-, toepassings-, koppelings- of abonnee-object gebruiken om een procedure te koppelen aan de binnenkomst van gegevens die aan een van deze objecten is gekoppeld door middel van DDE (Dynamic Data Exchange) of OLE (Object Linking and Embedding). In Microsoft Excel voor de Macintosh (Systeem 7) kunnen ook gegevens binnenkomen via de voorziening voor publicaties en abonnementen.

De volgende programmacode koppelt bijvoorbeeld de binnenkomst van aandelenkoersen in een werkblad aan een procedure die de gegevens afzonderlijk valideert.

```
Sub OnderschepGegevens()          'Onderschepping BijGegevens instellen
    Werkbladen("Koersanalyse").BijGegevens = "GegevensValideren"
Einde Sub

Sub GegevensValideren()                    'BijGegevens-
afhandelingsroutine
'Invoerbereik is een Openbaar-variabele
    Voor Elke Cel In Invoerbereik
        GegevensControleren                    'Mijn validatieprocedure
                                               'uitvoeren
    Volgende
Einde Sub
```

Zie hoofdstuk 10, "Samenwerken met andere toepassingen", voor meer informatie over het gebruik van DDE en OLE.

Opmerking Een BijGegevens-afhandelingsroutine wordt niet uitgevoerd als deze is gekoppeld aan een andere actieve kopie van Microsoft Excel.

De eigenschap BijToetsAnnuleren

Een BijToetsAnnuleren-Gebeurtenisafhandeling wordt uitgevoerd als de gebruiker op ESC drukt om de procedure in uitvoering te annuleren. (In Microsoft Excel voor de Macintosh kunt u een procedure in uitvoering ook annuleren door op COMMAND+PUNT te drukken). Met de eigenschap **BijToetsAnnuleren** van het toepassingsobject kunt u de ene procedure koppelen aan de annulering van de andere procedure.

Vaak is het belangrijk te voorkomen dat de gebruiker de uitvoering van een procedure onderbreekt. De procedure kan bijvoorbeeld bezig zijn een database bij te werken of bestanden hebben geopend die in een bepaalde volgorde moeten worden gesloten. U kunt de toets ESC buiten bedrijf stellen met de volgende instructie.

```
Toepassing.BijToetsAnnuleren = xlUitgeschakeld
```

In plaats van de eigenschap **BijToetsAnnuleren** te gebruiken, wilt u misschien wel reageren op het verzoek van de gebruiker om de procedure te beëindigen, maar de procedure op een meer ordelijke manier beëindigen. De volgende programmacode onderschept bijvoorbeeld de poging om een willekeurige procedure in uitvoering te beëindigen en roept de procedure ToepAfsluiten op.

```
Sub OnderschepAnnuleren()
    Toepassing.BijToetsAnnuleren = "ToepAfsluiten"
Einde Sub

Sub ToepAfsluiten()                          'BijToetsAnnuleren-
afhandelingsroutine.
    BestandenSluiten
    MenubalkenHerstellen
Einde Sub
```

De methoden BijHerhalen en BijOngedaanMaken

Een BijHerhalen-Gebeurtenisafhandeling wordt uitgevoerd als de gebruiker een van de vormen van de opdracht **Herhalen** in het menu **Bewerken** kiest. De gebruiker kan bijvoorbeeld tekst in een cel plakken en deze bewerking vervolgens herhalen met de opdracht **Herhalen Plakken**. Verder komt een BijHerhalen-Gebeurtenisafhandeling goed van pas nadat de gebruiker een macro start die een reeks gecompliceerde bewerkingen in een werkblad uitvoert.

Met de methode **BijHerhalen** van het toepassingsobject kunt u de vorm van de opdracht **Herhalen** aanpassen en een procedure koppelen aan de menu-opdracht **Herhalen**.

Stel bijvoorbeeld dat u nadat een datum in een cel is geplakt de datum telkens met een dag wilt laten toenemen als deze wordt gekopieerd en geplakt in de onderliggende cel. U kunt dit bijvoorbeeld bereiken met de programmacode in de volgende procedure.

```
Sub OnderschepHerhaling()
    Toepassing.BijHerhalen _
        Tekst:= "Volgende dag plakken"; _
        Procedure := "ToenamePlakken"
Einde Sub
```

```
Sub ToenamePlakken()                                    'BijHerhalen-
afhandelingsroutine.
    Met ActieveCel
        'Een dag laten toenemen.
        .Waarde = .Verschuiving(-1;0) + 1
    Einde Met
Einde Sub
```

Op soortgelijke wijze wordt een BijOngedaanMaken-Gebeurtenisafhandeling uitgevoerd als de gebruiker de opdracht **Ongedaan maken** in het menu **Bewerken** kiest. Met de methode **BijOngedaanMaken** van het toepassingsobject kunt u de vorm van de opdracht **Ongedaan maken** aanpassen en een procedure koppelen aan het kiezen van de opdracht **Ongedaan maken**.

Stel dat in een procedure die een rij met gegevens voor een nieuwe record invoegt, als laatste handeling gegevens in een cel worden ingevoerd. De opdracht **Ongedaan maken** luidt op dat moment **Ongedaan maken Invoer**. Met deze opdracht wordt echter alleen de informatie verwijderd die in de cel is ingevoerd, terwijl u waarschijnlijk liever de gehele rij zou willen wissen met een opdracht **Ongedaan maken Nieuwe record**.

Een invoegmacro maken

U kunt een invoegmacro gebruiken om exemplaren van uw macro te verspreiden in een onleesbare, onbewerkbare vorm die anderen kunnen gebruiken. In de meest eenvoudige vorm kan een invoegmacro een reeks door de gebruiker gedefinieerde functies bevatten. Een complexere invoegmacro kan bestaan uit een aangepast dialoogvenster, aangepaste menu's, een speciale werkbalk en verder een reeks **Functie**- en **Sub**-procedures die de toepassing tot een samenhangend geheel maken. Als een gebruiker een invoegmacro in de opstartdirectory of -map plaatst, werken de voorzieningen van de invoegmacro als ingebouwde voorzieningen. Uw zelfgemaakte functies verschijnen bijvoorbeeld in het dialoogvenster **Wizard Functies** (dat verschijnt als u de opdracht **Functie** in het menu **Invoegen** kiest).

Een werkmap converteren naar een invoegmacro

Als u een werkmap opslaat met de bestandsindeling voor invoegmacro's, converteert Microsoft Excel de programmacode van de werkmap naar een gecomprimeerde vorm. Het bestand kan in deze vorm door niemand worden gelezen en kan niet worden teruggeconverteerd naar een werkmap. Dit proces heet *compileren.* Compileren versnelt de uitvoering van de programmacode en voorkomt dat uw werk door anderen wordt gelezen en gewijzigd.

Invoegmacro's ondersteunen veel van de kenmerken van werkmappen, maar niet alle. De procedures in een invoegmacro kunnen werken met de gegevens in de bijbehorende werkbladen en kunnen interne grafieken kopiëren en plakken in een geopende werkmap, maar de gebruiker kan deze bladen niet bewerken of bekijken.

Belangrijk Omdat invoegmacro's niet kunnen worden bewerkt, is het zeer belangrijk een kopie van uw oorspronkelijke werkmap te bewaren, zodat u de procedures daarin kunt bijwerken en er nieuwe procedures aan kunt toevoegen. Sla een werkmap waarvan u een invoegmacro maakt daarom altijd onder een andere bestandsnaam op. (Als u in Microsoft Excel voor Windows de voorgestelde .XLA-extensie voor de invoegmacro gebruikt, is de bestandsnaam van de invoegmacro anders dan die van de werkmap. In Microsoft Excel voor de Macintosh wordt voor de invoegmacro dezelfde naam voorgesteld als die van het huidige bestand, gevolgd door "Add-In").

Voordat u een exemplaar van een werkmap opslaat als invoegmacro (ervan uitgaand dat u uw programmacode al grondig hebt getest en alle fouten hebt gecorrigeerd), moet u alle foutopsporingscode verwijderen, zoals instructies waarin gebruik is gemaakt van de methoden **Stoppen** en **Foutenopsp.Afbeelden**.

▶ **Een invoegmacro maken**

1. Maak een Visual Basic-module, Microsoft Excel 4.0-macroblad of dialoogblad actief in de werkmap die u wilt converteren en kies de opdracht **Invoegmacro maken** in het menu **Extra**.
2. Ga naar het vak "Directory's" in Microsoft Excel voor Windows of naar het vak "Mappen" in Microsoft Excel voor de Macintosh en selecteer de directory of de map waarin u het bestand wilt opslaan.
3. Typ in het vak Bestandsnaam de naam die u aan het gecompileerde invoegmacrobestand wilt geven.
4. Kies OK.

Een invoegmacro transporteren

Als u een invoegmacro naar een andere computer wilt transporteren, moet u ook alle type- of objectbibliotheken en ondersteunende bestanden en werkmappen overbrengen waarnaar in de invoegmacro wordt verwezen. Invoegmacro's zijn platform-onafhankelijk omdat zowel de op Windows gebaseerde versie als de Macintosh-versie van Microsoft Excel de gecompileerde programmacode kan interpreteren.

Zie Bijlage A, "Programmacode schrijven voor internationaal gebruik", voor meer informatie over ojectbibliotheken en het ontwikkelen van programmacode voor internationale toepassingen.

Een invoegmacro gebruiken

Invoegmacro's worden *op afroep geladen*. Dat wil zeggen dat bepaalde onderdelen van de invoegmacro onmiddellijk worden geladen, voordat de gehele invoegmacro wordt geladen. Bevat een invoegmacro bijvoorbeeld een menubewerkingslijst (een aangepaste lijst die menu-opdrachten toevoegt of verwijdert), dan worden de effecten van deze lijst zichtbaar in het menusysteem als de gebruiker Microsoft Excel start. Als de gebruiker een menu-opdracht kiest die toebehoort aan de invoegmacro, wordt de gehele invoegmacro van de schijf in het geheugen geladen.

Bevat een invoegmacro echter een Auto_openen-procedure, dan wordt bij het starten van Microsoft Excel direct de gehele invoegmacro geladen.

Als u een invoegmacro installeert, worden de bijbehorende menubewerkingslijst en werkbalken gelezen en opgeslagen in het bestand EXCEL5.INI in Microsoft Excel voor Windows of in het bestand EXCEL VOORKEUREN (5) in Microsoft Excel voor de Macintosh. De volgende keer dat u Microsoft Excel start, verschijnen alle menu-opdrachten van de invoegmacro, ook al is de invoegmacro niet geladen. Kiest u een menu-opdracht die aan de invoegmacro toebehoort, dan wordt de invoegmacro geladen. Als u de invoegmacro verwijdert, wordt de bijbehorende menubewerkingslijst gewist uit EXCEL5.INI of EXCEL VOORKEUREN.

Als u een werkmap opent, wordt aan de hand van de bijbehorende menubewerkingslijst het huidige menusysteem gewijzigd voordat eventuele Auto_openen-procedures in de werkmap worden uitgevoerd. Met deze voorziening kunt u Auto_openen-procedures schrijven die gebruik maken van de toegevoegde menu-opdrachten.

Als Microsoft Excel wordt gestart nadat een invoegmacro is geïnstalleerd, zijn toetsencombinaties die aan procedures zijn toegewezen beschikbaar voor de gebruiker. De namen van door de gebruiker gedefinieerde functies verschijnen dan in het dialoogvenster **Wizard Functies** (menu **Invoegen**) en de namen van macro's verschijnen in het dialoogvenster **Macro** (menu **Extra**), ook wanneer de invoegmacro niet is geladen.

Invoegmacro's beheren met Visual Basic

De verzameling Invoegmacro's bevat een lijst met alle beschikbare invoegmacro's. Ook niet-geïnstalleerde macro's staan in deze lijst. Deze lijst correspondeert met de lijst met invoegmacro's die wordt weergegeven in het dialoogvenster **Invoegbeheer** (menu **Extra**, opdracht **Invoegmacro's**). Een invoegobject bestaat uit een enkele invoegmacro, ongeacht of deze is geïnstalleerd.

U kunt toegang krijgen tot andere invoegmacro's met de methode **Toevoegen** van het invoegobject, die nieuwe invoegmacro's toevoegt aan de lijst met beschikbare invoegmacro's. Hetzelfde resultaat bereikt u als u de knop Bladeren kiest in het dialoogvenster **Invoegbeheer** (menu **Extra**, opdracht **Invoegmacro's**) en vervolgens de naam van een bestand selecteert. De volgende programmacode maakt bijvoorbeeld een invoegmacro met de naam MIJNMACR.XLA beschikbaar voor Microsoft Excel voor Windows, maar installeert deze niet.

```
Invoegmacros.Toevoegen bestandsNaam := "C:\MIJNTOEP\MIJNMACR.XLA"
```

De eigenschap **Geïnstalleerd** van het invoegobject stelt **Waar** in of resulteert in **Waar** als de invoegmacro is geïnstalleerd. Deze eigenschap correspondeert met het aankruisvakje naast de naam van de invoegmacro in het dialoogvenster **Invoegbeheer**. Als u de eigenschap **Geïnstalleerd** instelt op **Waar**, installeert u de invoegmacro. Als de invoegmacro wordt geïnstalleerd, worden de bijbehorende Auto_toevoegen-functies opgeroepen. Stelt u deze eigenschap in op **Onwaar**, dan verwijdert u de invoegmacro. Als de invoegmacro wordt verwijderd, worden de bijbehorende Auto_verwijderen-functies opgeroepen.

```
'MIJNMACRO installeren
Invoegmacros("MIJNMACR").Geïnstalleerd = Waar
Invoegmacros.Toevoegen("C:\MIJNTOEP\MIJNMACR.XLA").Geïnstalleerd = Waar
```

Invoegmacro's hebben veel van dezelfde eigenschappen als werkmappen in het algemeen, zoals de eigenschappen **Auteur**, **Titel**, **Onderwerp**, **Sleutelwoorden** en **Opmerkingen** van het werkbladobject.

De eigenschap **Naam** van het invoegobject resulteert in de bestandsnaam van de invoegmacro. De eigenschap **VolledigeNaam** resulteert in de naam van de invoegmacro, met inbegrip van het volledige pad. Deze eigenschap is equivalent aan de eigenschap **Pad** gevolgd door het huidige padscheidingsteken en de eigenschap **Naam** van de invoegmacro. Met de eigenschap **Pad** wordt het pad naar de invoegmacro opgehaald, zonder het laatste scheidingsteken.

Er een geheel van maken

Een Microsoft Excel-toepassing is een pakket procedures, bladen en gebruikersinterface-onderdelen die samen één einddoel dienen. Sommige toepassingen kunnen alleen bestaan uit een reeks **Functie**-procedures die u wilt verspreiden in de vorm van invoegmacro's. Als de gebruiker deze toevoegt aan de Microsoft Excel-omgeving, zijn de functies niet te onderscheiden van ingebouwde functies. Andere toepassingen kunnen bestaan uit een enkele werkbalkknop, die wordt ondersteund door een of meer procedures die één taak uitvoeren.

Een uitgebreide bedrijfstoepassing, zoals een voorraadcontrolesysteem voor een klein bedrijf, kan het volgende bevatten:

- Speciaal opgemaakte werkbladen met ondersteunende gegevens en gedefinieerde namen en grafieksjablonen voor het weergeven van resultaten
- Menu-onderdelen, zoals aangepaste menubalken, menu's en opdrachten
- Werkbalken en werkbalkknoppen
- Dialoogvensters met besturingselementen die worden ondersteund door een reeks procedures
- Visual Basic-modulen met de procedures die uw toepassing ondersteunen

U kunt de werkmap met deze onderdelen samenvoegen tot een pakket, verbergen, beveiligen en vervolgens naar een invoegmacro converteren.

Voorbeeldtoepassing

Op de installatiediskettes van Microsoft Excel staat een voorbeeldwerkmap die veel van de in dit hoofdstuk behandelde principes aanschouwelijk maakt. Deze werkmap beheert een eenvoudige database, waarin inventarisgegevens van een boekhandel worden bijgehouden, en bestaat uit de volgende onderdelen:

- Een werkblad met een database waarin verscheidene gefingeerde records zijn opgeslagen
- Een dialoogvenster waarin de gebruiker records uit de database kan opvragen en bewerken
- Een werkbalk en menu-opdrachten
- Een Visual Basic-module met de procedures die de toepassing beheren

De bestandsnaam van deze voorbeeldwerkmap is BOEKWNKL.XLS in Microsoft Excel voor Windows en INVENTARIS BOEKWINKEL in Microsoft Excel voor de Macintosh.

BIJLAGE A

Programmacode schrijven voor internationaal gebruik

Met Visual Basic kunt u gemakkelijk Microsoft Excel-toepassingen (meestal invoegmacro's) uitwisselen met gebruikers in andere landen van wie de moedertaal niet de taal is waarin u de code hebt geschreven. In deze bijlage wordt onder een *toepassing* een werkmap of verzameling werkmappen met bijbehorende documenten verstaan die een bepaalde taak verricht en die onder Microsoft Excel draait.

In deze bijlage wordt uitgelegd hoe u een taal kunt kiezen voor het schrijven van programmacode en hoe u toepassingen voor internationaal gebruik kunt schrijven en distribueren. Ook komen er technieken aan de orde die garanderen dat programmacode overal op de wereld naar verwachting functioneert, met name naar de verwachting van de gebruikers.

Programmacode universeel toepasbaar maken met objectbibliotheken

U kunt Visual Basic-programmacode in vele talen schrijven. Het functioneren van die programmacode is onafhankelijk van de taalversie van Microsoft Excel (uitgezonderd de beperking die onder "Belangrijk" even verderop aan de orde komt). Dit is mogelijk doordat Visual Basic objectbibliotheken gebruikt bij het compileren van programmacode. Een *objectbibliotheek* is een bestand dat bepaalt welke woorden en symbolen Visual Basic kan begrijpen: de namen van objecten, eigenschappen, methoden, Visual Basic-instructies, enzovoort die u in een Visual Basic-module typt. Deze woorden en symbolen verschillen van taal tot taal.

Belangrijk U kunt geen programmacode transporteren die is geschreven op een systeem dat is ingesteld op DBCS-tekens (Verre Oosten), naar een systeem waarbij dat niet het geval is, tenzij de programmacode in het Engels is geschreven.

Afgezien van de terminologische informatie in objectbibliotheken gebruikt Microsoft Excel ook bepaalde instellingen van het besturingssysteem om te achterhalen hoe in een bepaald land datums, tijdseenheden, valutatekens, scheidingstekens voor decimalen en duizendtallen en het lijstscheidingsteken worden gebruikt. De informatie over taal en land geeft taal/landcombinaties, bijvoorbeeld Engels/Australië, Engels/Canada, Frans/Frankrijk en Frans/Canada. Als u internationale programmacode schrijft, moet u denken in taal/landcombinaties, niet gewoon talen of gewoon landen.

Met objectbibliotheken kunt u programmacode op een voor u gerieflijke wijze schrijven en kunnen gebruikers in andere landen die programmacode op een voor hun gerieflijke wijze benutten. Als bijvoorbeeld in uw taal/land de komma het decimaalscheidingsteken is, zult u de komma willen gebruiken als u getallen typt in de programmacode. Als een ander teken, bijvoorbeeld de punt, het decimaalscheidingsteken is in de taal/landcombinatie van de gebruiker, wilt u natuurlijk dat er een punt wordt weergegeven waar dat nodig is. U wilt ook dat uw programmacode de punt juist interpreteert als een gebruiker een getal invoert in een cel of dialoogvenster. Objectbibliotheken maken dit mogelijk.

Een taal/land kiezen voor het schrijven en bewerken van Visual Basic-programmacode

Als de benodigde objectbibliotheken op uw schijf zijn geïnstalleerd en bij Visual Basic zijn geregistreerd, kunt u de taal kiezen waarin u de programmacode wilt schrijven. U kunt ook een land opgeven voor de notatie van datums, tijden, lijstscheidingsteken, decimaalscheidingsteken, enzovoort. Zo geeft u de taal/landcombinatie voor de Visual Basic-omgeving op. (Deze taal/landcombinatie hoeft niet dezelfde te zijn als de taal/landinformatie in het besturingssysteem die van invloed is op de uitvoer van de programmacode).

▶ **Een taal/land kiezen voor het schrijven of bewerken van Visual Basic-programmacode**

1. Kies **Opties** in het menu **Extra**.
2. Selecteer het tabblad Module.
3. Selecteer in het vak "Land/Taal" de gewenste taal/landcombinatie.
4. Kies de knop OK.

Als de taal/landcombinatie die u wenst niet in het vak "Land/Taal" voorkomt, moet u de desbetreffende objectbibliotheken installeren, zoals wordt uitgelegd in "Objectbibliotheken installeren en registreren" verderop in deze bijlage.

Voor gebruikers die programmacode willen schrijven zonder een van de vooraf gedefinieerde taal/landcombinaties te gebruiken, voorziet Visual Basic in aangepaste mogelijkheden in stap 3. In dat geval interpreteert Visual Basic de datumnotaties, scheidingstekens, enzovoort die in uw programmacode voorkomen, op basis van de taal/landinformatie in uw besturingssysteem. Deze mogelijkheid zult u meestal niet kiezen, omdat toepassingen die met aangepaste instellingen zijn gemaakt waarschijnlijk niet werken op andere systemen.

Onder de groep "Internationaal" in het tabblad "Module" bevinden zich de keuzerondjes "Huidige instellingen" en "Standaardinstellingen" en ook een weergavegebied met het lijstscheidingsteken, de getalnotatie, enzovoort. Als u het keuzerondje "Huidige instellingen" selecteert, worden de taal/landinstellingen voor de actieve werkmap weergegeven. Als u het keuzerondje "Standaardinstellingen" selecteert, worden de standaardinstellingen weergegeven voor nieuwe, door u gemaakte werkmappen. Dus als u gewoonlijk met de combinatie Engels/Verenigde Staten werkt en u een Frans/Frankrijk-toepassing ontvangt, wordt informatie hierover in de laatstgenoemde combinatie weergegeven, maar de standaardinstelling voor de rest van uw werk blijft Engels/Verenigde Staten.

Werken in meerdere taal/landcombinaties

Bij elke werkmap wordt een taal/landcombinatie opgeslagen, dus de taal/landcombinatie kan per werkmap verschillen, zelfs als er een aantal tegelijk geopend zijn. U kunt procedures schrijven en oproepen die in verschillende talen zijn geschreven.

Visual Basic vertaalt bestaande programmacode niet automatisch van de ene taal in de andere. Als u de programmacode bewerkt, moet u dezelfde taal/landcombinatie gebruiken als waarin de programmacode geschreven is. Het gebruik van meerdere taal/landcombinaties vereist meerdere werkmappen. (U kunt wel de programmacode bewerken met iedere taalversie van Microsoft Excel. U kunt bijvoorbeeld Nederlandse programmacode bewerken met de Noorse Microsoft Excel, hoewel de programmacode in het Nederlands blijft weergegeven).

Objectbibliotheken installeren en registreren

Als u de taal/landcombinatie die u wilt gebruiken, niet kunt vinden in het vak "Land/Taal" in het tabblad Module, moet u de benodigde objectbibliotheken installeren en registreren.

Visual Basic in Microsoft Excel heeft ten minste twee bestanden nodig voor elke taal waarin u programmacode schrijft of uitvoert: de Microsoft Excel-objectbibliotheek en de objectbibliotheek van Visual Basic for Applications. Microsoft Excel wordt altijd geleverd met de Engelse versies van deze bibliotheken plus die voor uw taalversie van Microsoft Excel (de Italiaanse Microsoft Excel heeft bijvoorbeeld de Engelse en Italiaanse objectbibliotheken, in totaal vier bestanden).

De namen van de objectbibliotheken en de wijze waarop u deze kunt verkrijgen worden verderop in deze bijlage besproken.

▶ Objectbibliotheken installeren

- Kopieer voor elke taal waarmee u wilt werken, de twee objectbibliotheken naar uw vaste schijf.

De volgende lokaties worden aanbevolen voor objectbibliotheken:

- Als u Microsoft Excel voor Windows gebruikt, kopieert u de objectbibliotheken (*.OLB) naar de subdirectory SYSTEEM van de directory waarin Microsoft Windows is geïnstalleerd. (Een veel voorkomend pad is C:\WINDOWS\SYSTEEM).
- Als u Microsoft Excel voor de Macintosh gebruikt, kopieert u de objectbibliotheken naar de map MICROSOFT GEDEELDE BESTANDEN in de systeemmap of de map EXTENSIES.

Als u van iemand een in een andere taal geschreven toepassing ontvangt waarvan de objectbibliotheken deel uitmaken, worden die bestanden automatisch geregistreerd wanneer u een procedure in die toepassing probeert te starten. Maar als u programmacode schrijft en de objectbibliotheken voor de eerste keer gebruikt, moet u ze zelf bij Microsoft Excel registreren. *Registreren* betekent dat u Microsoft Excel op de hoogte brengt van het bestaan van de bibliotheken. U hoeft een objectbibliotheek maar een keer te registreren, tenzij u de bibliotheek verplaatst of de naam ervan wijzigt.

▶ Een nieuwe taalversie van de objectbibliotheken registreren

1. Ga naar een Visual Basic-module.
2. Kies **Verwijzingen** in het menu **Extra**.
3. Kies de knop Bladeren.
4. Klik in het vak "Directory's" (of het vak "Mappen" op de Macintosh) op de namen van directory's of mappen totdat u de nieuwe Microsoft Excel-bibliotheek hebt gevonden.

   De bestandsnamen staan onder "Namen van objectbibliotheken" verderop in deze bijlage.

5. Typ of selecteer in het vak "Bestandsnaam" de bestandsnaam van de objectbibliotheek.
6. Kies de knop OK.
7. Herhaal de stappen 3 tot en met 6 voor de objectbibliotheek van Visual Basic for Applications.
8. Kies de knop OK om het dialoogvenster **Verwijzingen** te sluiten.

U kunt nu de opdracht **Opties** kiezen in het menu **Extra**, het tabblad Module selecteren en de gewenste taal/landcombinatie selecteren.

Als de taal die u wenst nog steeds niet beschikbaar is, is er al een taal toegewezen aan de werkmap. Open een nieuwe werkmap en herhaal de voorgaande stappen. Denk erom dat u slechts één keer een taal kunt toewijzen aan een werkmap, zoals is uitgelegd onder "Werken in meerdere taal/landcombinaties" eerder in deze bijlage.

Objectbibliotheken distribueren

Tenzij de Visual Basic-programmacode in het Engels is geschreven, moet u altijd de objectbibliotheken van de desbetreffende taal meesturen wanneer u een toepassing distribueert, voor het geval dat de ontvangers de bibliotheken nog niet hebben. Als de programmacode procedures oproept die in verschillende talen zijn geschreven, moet u de objectbibliotheken voor al die talen meesturen. (Microsoft staat toe dat geregistreerde gebruikers van Microsoft Excel, als dat nodig is, objectbibliotheken met de bijbehorende toepassingen aan andere geregistreerde gebruikers zenden. De namen van de objectbibliotheken staan in het volgende gedeelte, "Namen van objectbibliotheken"). U moet ook alle werkmappen en andere objectbibliotheken opnemen waarnaar wordt verwezen door de toepassing die u distribueert.

Objectbibliotheken zijn platform-specifiek, dus als u van plan bent werkmappen te distribueren aan gebruikers van zowel Microsoft Excel voor Windows als Microsoft Excel voor de Macintosh, moet u de Windows-versie en de Macintosh-versie van de objectbibliotheken meesturen.

Tip Doe het volgende om de lokatie van een objectbibliotheek op uw schijf te bepalen: ga naar een Visual Basic-module, kies de opdracht **Verwijzingen** in het menu **Extra**, selecteer de bibliotheeknaam onder "Beschikbare verwijzingen" en neem notitie van het pad in de lijst onder "Groep". Deze procedure is ook nuttig als u wilt weten welke objectbibliotheken u met de toepassing moet distribueren; de aankruisvakjes naast de benodigde bibliotheken zijn ingeschakeld.

Namen van objectbibliotheken

Voor Microsoft Excel voor de Macintosh zijn de bestandsnamen van de objectbibliotheken MS EXCEL 5.0 EN OLB en VBA EN OLB. De bestandsnamen van de objectbibliotheken voor andere talen vervangen de de letters "EN" ("English")door de afkorting van de desbetreffende taal (bijvoorbeeld MS EXCEL 5.0 FR OLB en VBA FR OLB).

Voor Microsoft Excel voor Windows staan een aantal van de namen van de objectbibliotheken in de volgende tabel.

Taal	Objectbibliotheeknaam voor Microsoft Excel voor Windows	Objectbibliotheeknaam voor Visual Basic for Applications
Engels	XLEN50.OLB	VBAEN.OLB
Frans	XLFR50.OLB	VBAFR.OLB
Duits	XLDE50.OLB	VBADE.OLB
Spaans	XLES50.OLB	VBAES.OLB
Italiaans	XLIT50.OLB	VBAIT.OLB
Portugees	XLPTG50.OLB (Portugal)	VBAPTG.OLB
Portugees	XLPTB50.OLB (Brazilië)	VBAPTB.OLB
Zweeds	XLSV50.OLB	VBASV.OLB
Noors	XLNO50.OLB	VBANO.OLB
Fins	XLFI50.OLB	VBAFI.OLB
Nederlands	XLNL50.OLB	VBANL.OLB
Deens	XLDA50.OLB	VBADA.OLB

Er kunnen later nog veel meer objectbibliotheken beschikbaar komen.

Hoe komt u aan nieuwe objectbibliotheken?

Als u nieuwe Microsoft Excel-objectbibliotheken wenst, kunt u contact opnemen met:

- CompuServe. Als u verbinding hebt met CompuServe, gaat u naar de Microsoft Software Library (typ **Go MSL**), zoekt u naar het trefwoord *object* of *library* en laadt u de gewenste bestanden.

- Microsoft Excel-gebruikers in andere landen met wie u toepassingen uitwisselt.

Zie de documentatie bij andere toepassingen (bijvoorbeeld Microsoft Project of Microsoft Word) voor informatie over het verkrijgen van objectbibliotheken voor deze toepassingen.

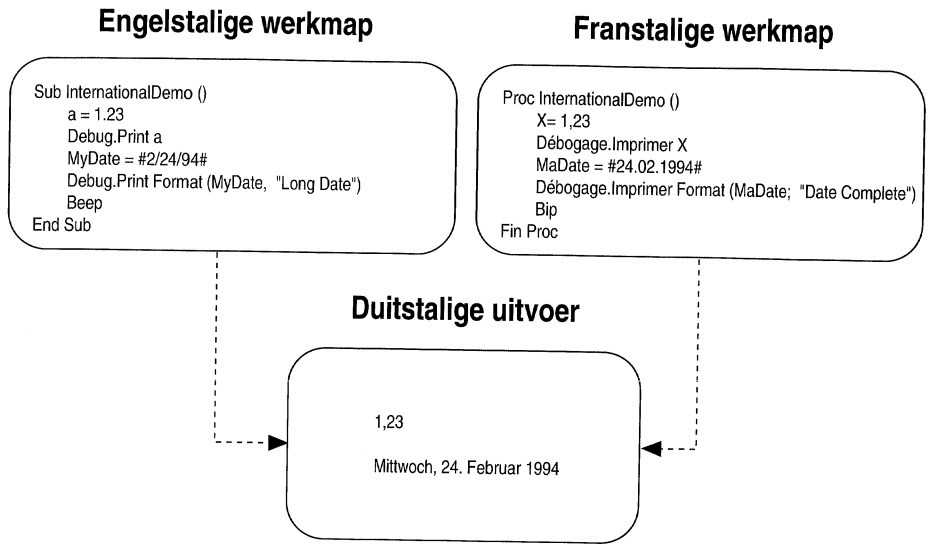
Opmerking Microsoft Word, versie 6.0 bevat geen afzonderlijke objectbibliotheekbestanden. Als gevolg hiervan moet bij het invoeren van Visual Basic programmacode in Microsoft Excel die dient voor het besturen van Microsoft Word, de taal gebruiken die overeenkomt met die versie van Microsoft Word. Als u bijvoorbeeld een Franse versie van Microsoft Word wilt besturen, gebruikt u de Franse Wordbasic instructie en functienamen, ongeacht de plaatselijke instellingen van Visual basic.

Programmacode in een andere taalomgeving uitvoeren

In het voorgaande is uiteengezet hoe de taal/landinstelling bepaalt hoe u als programmeur Visual Basic-programmacode moet invoeren. Maar de vorm waarin u datums, notaties en dergelijke invoert, is niet altijd gelijk aan de vorm waarin de gebruiker deze te zien moet krijgen; deze hangt af van zijn taal/landcombinatie.

Tijdens de uitvoering passen Visual Basic-functies zich meestal aan de gebruiker aan. Gewoonlijk worden de taal/landinstellingen van het besturingssysteem (dus niet die van de Visual Basic-bewerkingsomgeving) gebruikt om te bepalen hoe getallen en datums (bijvoorbeeld in dialoogvensters, cellen en dergelijke) moeten worden geïnterpreteerd die gebruikers van de programmacode opgeven. Visual Basic gebruikt deze instellingen ook om te bepalen hoe getallen en datums moeten worden weergegeven, bijvoorbeeld de resultaten van de functie **Notatie**.

In het volgende voorbeeld is de ene procedure in het Engels en de andere in het Frans geschreven, maar geven beide het resultaat in dezelfde taal weer wanneer ze op hetzelfde systeem worden uitgevoerd. De taal waarin wordt weergegeven hangt af van de taal/landinstellingen van het besturingssysteem. In het voorbeeld wordt uitgegaan van een Duits systeem waarop Duitse, Engelse en Franse objectbibliotheken zijn geïnstalleerd.



Zie "Richtlijnen voor het schrijven van universele Visual Basic-programmacode" verderop in deze bijlage voor meer voorbeelden van de verschillen tussen wat u als programmeur schrijft en wat de gebruiker in een andere taal/landcombinatie te zien krijgt.

Het kan voorkomen dat u als gebruiker zo nu en dan de taal/landinstellingen van uw besturingssysteem moet wijzigen. Of als programmeur wilt u misschien de instellingen wijzigen om toepassingen te testen die u aan anderen wilt distribueren. Door een wijziging van de taal/landinstellingen van het besturingssysteem wordt de uitvoer van taal/landgevoelige functies en instructies gewijzigd.

▶ De taal/landinstellingen in Microsoft Windows wijzigen

1. Open het Configuratiescherm van Microsoft Windows.
2. Dubbelklik op het pictogram Internationaal.
3. Typ of selecteer in het vak "Land" de naam van een land.

   Het lijstscheidingsteken en de notatie van datum, tijd, getallen en valuta's worden automatisch aangepast, hoewel u deze ook afzonderlijk kunt wijzigen.
4. Kies de knop OK.
5. Sluit het Configuratiescherm.

In het dialoogvenster **Internationaal** kunt u ook een taal opgeven als u de Microsoft Windows-installatiediskettes hebt met de dynamic-link library (DLL) van de benodigde taal. Dit is meestal niet nodig omdat stap 3 in de meeste landinstellingen voorziet.

▶ **Taal/landinstellingen op de Macintosh met Systeem 7.1 wijzigen**

1. Kies **Configuratiescherm** in het **Apple**-menu.
2. Dubbelklik op het configuratiescherm Datum en tijd.
3. Geef de gewenste datum- en tijdinstellingen op.
4. Sluit het configuratiescherm Datum en tijd.
5. Dubbelklik op het configuratiescherm Getallen.
6. Geef de gewenste getalnotaties op.
7. Sluit het configuratiescherm Getallen.

Misschien wilt u ook andere configuratie-instellingen wijzigen. Zie de Macintosh-documentatie voor informatie over het wijzigen van de notatie van datums, tijd en getallen op andere versies van de Macintosh.

Internationale programmacode schrijven die meerdere toepassingen stuurt

Als u van plan bent Microsoft Excel te gebruiken met een andere toepassing die Visual Basic ondersteunt, hebt u de juiste taalversie van de objectbibliotheken voor de betrokken toepassing nodig. Als u bijvoorbeeld Italiaanse programmacode wilt schrijven die functies in Microsoft Excel en Microsoft Project gebruikt, hebt u de volgende bestanden nodig:

- De Italiaanse objectbibliotheek van Microsoft Excel
- De Italiaanse objectbibliotheek van Visual Basic for Applications
- De Italiaanse objectbibliotheek van Microsoft Project

Zie hoofdstuk 10, " Samenwerken met andere toepassingen", voor meer informatie over het schrijven van programmacode die met andere toepassingen werkt.

De taalversie van Microsoft Excel en Microsoft Project zelf is onbelangrijk. De Italiaanse programmacode functioneert op elke versie van Microsoft Excel en Microsoft Project, mits u de objectbibliotheken uit de voorgaande lijst opneemt.

Als u in het voorgaande voorbeeld de Zweedse versie van Microsoft Project zou gebruiken en u niet beschikt over de Italiaanse objectbibliotheek voor Microsoft Project, kunt u desondanks de programmacode uitvoeren als u deze in twee talen schrijft: Italiaans voor alle functies waarbij Microsoft Excel betrokken is en Zweeds voor alle functies waarbij Microsoft Project betrokken is. In dit geval heeft u afzonderlijke werkmappen voor de procedures in beide talen nodig. U kunt dan met de opdracht **Verwijzingen** in het menu **Extra** een verwijzing definiëren tussen de Zweedse en de Italiaanse werkmap.

Richtlijnen voor het schrijven van universele Visual Basic-programmacode

Aan de hand van de volgende richtlijnen kunt u Visual Basic-programmacode schrijven die goed functioneert in andere taal/landomgevingen. Een goed begrip van deze richtlijnen is noodzakelijk om te kunnen voorzien hoe de programmacode zal werken. De richtlijnen betreffen de volgende gebieden:

- Door de gebruiker gemaakte werkbladformules verwerken
- Conversiefuncties gebruiken
- Informatie weergeven met taal/landgevoelige functies en instructies
- Werken met verschillende talen in de programmacode
- Andere overwegingen bij het schrijven van internationale Visual Basic-programmacode

Door de gebruiker gemaakte werkbladformules verwerken

Met de eigenschap **Formule** kunt u formules invoeren in cellen. Als deze formules werkbladfuncties bevatten, kan Microsoft Excel automatisch de functies in de gewenste taal omzetten. Als u daarentegen de programmacode zodanig schrijft dat de gebruiker dergelijke formules in invoervensters invoert, kan Visual basic nog steeds werkbladfuncties in de ingevoerde functies omzetten, maar dan moet u wel de eigenschap **LokaleFormule** gebruiken in plaats van de eigenschap **Formule**.

```
Cellen(1; 1).Formule = "=SOM(A2:A3)"          'Formule wordt automatisch
                                              'omgezet
FormuleTekst = InvoerVenster FormuleVragen
Cellen(1; 2).LokaleFormule = "=" &FormuleTekst       'Formule wordt ook
nu
                                              'automatisch omgezet
```

In dit voorbeeld had FormuleTekst ook zoiets kunnen bevatten als wat overeenkomt met de Nederlandse formule `"SOM(A2:A3)"`. Het invoeren van formules in cellen met de eigenschap **Formule** of **LokaleFormule** is het enige gebied waarin Visual Basic tekst van de ene taal in een andere kan omzetten. Maar een andere eigenschap, **LokaleGetalnotatie** (in tegenstelling tot **GetalOpmaak**), houdt ook rekening met de taal/landcombinatie van de gebruiker. Zie *Formule*, *LokaleFormule*, *GetalOpmaak* en *LokaleGetalnotatie* in de online Visual Basic Naslaggids voor meer informatie over deze eigenschappen.

Conversiefuncties gebruiken

Visual Basic heeft verschillende functies die gegevens van een bepaald type converteren naar een ander type met behulp van de taal/landinstellingen in het besturingssysteem. Sommige functies zijn echter niet taal/landgevoelig. Gebruik waar mogelijk de taal/landgevoelige functies om uw programmacode universeel toepasbaar te maken. Dat is vooral belangrijk als u reeksen converteert naar getallen of datums.

De functies **TReeks** en **Waarde** gaan bijvoorbeeld altijd uit van een punt als decimaalscheidingsteken, terwijl de functies **CTReeks**, **CDubbel**, **CEnkel**, **CInteger** en **CLang** de instellingen in het besturingssysteem gebruiken om het decimaalscheidingsteken te bepalen.

Zie *Conversie* in de online Visual Basic Naslaggids voor meer informatie over het converteren van het ene gegevenstype naar het andere.

Speciale aandacht voor datums en valuta's

Typ nooit datums in programmacode als reeksen. Zelfs taal/landgevoelige conversiefuncties kunnen op onvoorziene wijze reeksen verwerken. Als bijvoorbeeld de volgende instructies worden uitgevoerd, is het resultaat afhankelijk van de taal/landcombinatie.

```
BeginDatum = "2/3/94"
NieuwDatum = CDatum(BeginDatum)
```

Als deze programmacode in de taal/landcombinatie Engels/Verenigde Staten wordt uitgevoerd, bevat NieuwDatum 3 februari 1994. In de taal/landcombinatie Engels/Australië bevat NieuwDatum 2 maart 1994. Dit vormt een potentiële foutenbron bij het programmeren, maar is ook erg nuttig voor het verwerken van invoer van de gebruiker via een dialoogvenster. **CDatum** converteert de tekst van de gebruiker naar de datum die de gebruiker bedoelt.

Maar u als programmeur moet altijd in programmacode datums typen als datumliteralen, bijvoorbeeld **#2/3/94#**, zodat Visual Basic precies de bedoelde datum herkent. In het volgende stuk programmacode bevat NieuwDatum 3 februari 1994 of 2 maart 1994, afhankelijk van de taal/landcombinatie van de programmeur, niet die van de gebruiker. De datum is voor alle gebruikers dezelfde, waar de programmacode ook wordt uitgevoerd.

```
BeginDatum = #2/3/94#
NieuwDatum = CDatum(BeginDatum)
```

Typ ook valuta's nooit in programmacode als reeksen. Het volgende stuk programmacode werkt bijvoorbeeld alleen in een taal/landcombinatie waar het dollarteken ($) het valutasymbool is.

```
Geld = "$1.22"
NieuwGeld = CValuta(Geld)
```

Gebruik in plaats hiervan decimale getallen, zoals in het volgende voorbeeld. (Hierin wordt er van uitgegaan dat de punt het decimaalscheidingsteken is in de taal/landcombinatie van de programmeur, maar de programmacode werkt correct, onafhankelijk van het decimaalscheidingsteken in de taal/landcombinatie van de gebruiker).

```
Geld = 1.22
NieuwGeld = CValuta(Geld)
```

Informatie weergeven met taal/landgevoelige functies en instructies

Zoals al eerder is uitgelegd, hebben verschillende taal/landcombinaties verschillende regels voor het weergeven van datums, tijden, getallen, valuta's en andere informatie. Als programmeur kunt u niet de regels van alle taal/landcombinaties van uw gebruikers weten. Dat is ook niet nodig omdat veel Visual Basic-instructies gebruik maken van de taal/landinstellingen van het besturingssysteem. De schrijfwijzen worden automatisch bepaald wanneer een procedure wordt uitgevoerd, maar sommige instructies zijn taal/landgevoeliger dan andere. In het algemeen kunt u in de online Visual Basic Naslaggids te weten komen of een instructie of functie taal/landgevoelig is en zo ja, op welke wijze.

De instructie **Afbeelden** heeft bijvoorbeeld weinig flexibiliteit ten aanzien van de notatie van de uitvoer, maar maakt wel gebruik van de taal/landinstellingen van het besturingssysteem. Datums worden afgebeeld met een correcte korte datumnotatie, getallen worden afgebeeld met het correcte decimaalscheidingsteken en valuta's worden afgebeeld met het correcte symbool.

De functie **Notatie** accepteert notatiecodes, maar deze codes geven altijd hetzelfde type uitvoer, zonder rekening te houden met de taal/landcombinatie van de gebruiker. De notatiecode "`mm-dd-jj`" is bijvoorbeeld niet geschikt voor taal/landcombinaties waarin de dag voor de maand wordt genoteerd.

Ter verhoging van de flexibiliteit geeft de functie **Notatie** ook benoemde notaties die taal/landgevoelig zijn, inclusief Korte datumnotatie, Lange datumnotatie, Lange tijdnotatie en Standaard getalnotatie. Als u benoemde notaties gebruikt, wordt de uitvoer gebaseerd op de taal/landinstellingen van het besturingssysteem van de gebruiker.

```
MijnDatum = #27 januari 1994#
MijnReeks = Notatie(MijnDatum; "dd-mm-jj")       'Resulteert in 1-27-94
MijnReeks = Notatie(MijnDatum; "Korte datumnotatie")     'Resulteert in
een                                                    'korte datum
                                                'op basis van de taal/land-
                                                'combinatie
```

De benoemde notaties kunnen zelfs uitvoer genereren in de taal van de gebruiker, inclusief de namen van maanden en dagen. Dit kunt u zien in de figuur in het gedeelte "Programmacode uitvoeren in een andere taalomgeving" eerder in deze bijlage. Zie *Afbeelden* of *Notatie* in de online Visual Basic Naslaggids voor informatie over het afbeelden en de notatie bepalen in ander taal/landcombinaties.

Werken met verschillende talen in de programmacode

Visual Basic heeft veel voorzieningen die een juiste werking van uw programmacode in andere taal/landcombinaties bevorderen, maar is niet in staat tekst automatisch te vertalen. U wilt bijvoorbeeld in een toepassing tekst in menu's, dialoogvensters, berichten enzovoort opnemen. U bent misschien in eerste instantie geneigd deze tekst direct in de programmacode te typen, maar dit is een beletsel voor een werkelijk universele toepasbaarheid van het programma. In plaats van tekst tussen aanhalingstekens kunt u variabelen en verwijzingen naar werkbladnamen gebruiken. De cellen waarnaar de werkbladnamen verwijzen kunnen afhankelijk zijn van de taal/landcombinatie waarin de programmacode wordt uitgevoerd. Op deze wijze kunt u woorden en zinnen in andere talen weergeven, op basis van de taal/landcombinatie van de gebruiker.

Het volgende voorbeeld is een werkblad met een kolom met namen gevolgd door kolommen met tekst in drie talen.

	A	B	C	D
1	**Vertaalde tekst**	**English**	**French**	**German**
2	Welkom	Welcome to the West Coast Sales!	Bienvenue à la société les Ventes de la Côte Ouest!	Willkommen bei der Nordwind GmbH.
3	Klantnaam	What is your name?	Quel est votre nom?	Wie heißen Sie?
4	Klantnummer	Enter your account number.	Entrez vous numéro de compte.	Bitte geben Sie Ihre Kundennummer ein.
5	Boodschap ongeldig	That transaction is not valid.	Cette transaction n'est pas valide.	Dieser Vorgang kann nicht bearbeitet werden.

Om toegang te krijgen tot de tekst in een bepaalde taal, definieert u de tekst in de kolom Vertaalde tekst als werkbladnamen. Deze namen moeten verwijzen naar de Engelse kolom. Met de methode **Verwijzen** kunt u de tekst in de kolommen met de andere talen krijgen. Als u in het Engels wilt werken, is de verschuiving nul, in het Frans is de verschuiving een en in het Duits twee. De waarde van de verschuiving kunt u op een van de volgende manieren verkrijgen:

- Gebruik de eigenschap **Internationaal** van het object Toepassing om de taal/landcombinatie te bepalen van Microsoft Excel (xlLandcode) of de landinstelling van het besturingssysteem (xlLandinstelling). Zet de instelling met een wiskundige aanpassing om in een verschuivingswaarde en definieer vervolgens in overeenstemming daarmee een **Openbaar**-variabele. (Zie "Meer leren over programmeren met Visual Basic" in hoofdstuk 6 voor meer informatie over het bereik van **Openbaar**-variabelen).
- Geef een dialoogvenster weer waarin de gebruiker gevraagd wordt een taal of land op te geven en definieer vervolgens in overeenstemming daarmee een **Openbaar**-verschuivingsvariabele.
- Definieer een **Openbaar**-constante die de gebruiker instelt.

Met de volgende programmacode wordt er een verschuivingscode bepaald, gebaseerd op de versie van Microsoft Excel die u gebruikt.

```
Kiezen Ingeval Toepassing.Internationaal(xlLandcode)
    Ingeval 1:                          'Amerikaans Engels
        UnivVerschuivCode = 0
    Ingeval 33:                         'Frans
        UnivVerschuivCode = 1
    Ingeval 49:                         'Duits
        UnivVerschuivCode = 2
    Ingeval Anders:                     'Amerikaans Engels als
                                        'standaardtaal
        UnivVerschuivCode = 0
Einde Kiezen
```

In de volgende programmacode wordt deze verschuivingscode gebruikt om een berichtvenster weer te geven waarin de gebruiker wordt verwelkomd.

```
WelkomstBericht = DezeWerkmap.Werkbladen("VertaalTabel").Bereik _
    ("Welcome").Verschuiving(0; UnivVerschuivCode).Waarde
BerichtVenster WelkomstBericht
```

U kunt op overeenkomstige wijze de tekst, posities en afmetingen van onderdelen van dialoogvensters wijzigen, zodat de programmacode waarlijk universeel toepasbaar wordt.

Andere overwegingen bij het schrijven van internationale Visual Basic-programmacode

Twee overwegingen bij het schrijven van programmacode die per taal/landcombinatie kan variëren betreffen het vergelijken van reeksen en hoe u werkt met bestandsinvoer en -uitvoer.

De regels voor het vergelijken van reeksen variëren per taal/landcombinatie als u de operatoren "kleiner dan" (<), "groter dan" (>), "is gelijk" (=) en **Zoals** gebruikt, of de functie **TReeksVerg**. Zie *Optie Vergelijken*, *Zoals* en *TReeksVerg* in de online Visual Basic Naslaggids voor meer informatie over deze verschillen.

De taal/landcombinatie vormt ook een overweging bij bestandsinvoer en -uitvoer. De instructie **Afbeelden #** voert gegevens in een bestand in terwijl de gegevens op het scherm worden weergegeven in een taal/landgevoelige notatie. De instructie **Invoer #** kan deze informatie niet vanuit het bestand lezen. De instructie **Schrijven #** daarentegen, voert gegevens in een vaste notatie in een bestand in, zodat de instructie **Invoer #** deze later in elke taal/landcombinatie kan lezen. Zie *Afbeelden* of *Schrijven* in de online Visual Basic Naslaggids voor meer informatie over bestandsinvoer en -uitvoer. Zie *Invoer* en selecteer het onderwerp *Overzicht van sleutelwoorden voor invoer en uitvoer* voor meer informatie over bestandsbewerkingen in het algemeen.

BIJLAGE B

Omschakelen van de macrotaal van Microsoft Excel 4.0

In deze bijlage worden gebruikers van de macrotaal van Microsoft Excel 4.0 vertrouwd gemaakt met het programmeren in Visual Basic, de nieuwe programmeertaal die in Microsoft Excel versie 5.0 is opgenomen. In deze bijlage wordt behandeld welke verschillen er bestaan tussen Visual Basic en de macrotaal van Microsoft Excel 4.0, hoe u gebruik kunt blijven maken van macro's die afkomstig zijn uit Microsoft Excel 4.0 en waar u meer informatie kunt vinden over Visual Basic.

Visual Basic is een echte programmeertaal, met variabelen waaraan verschillende bereiken kunnen worden toegekend, een geïntegreerde editor en uitgebreide hulpprogramma's voor het maken van dialoogvensters en het opsporen van fouten. Als u eenmaal in Microsoft Excel 5.0 hebt kennisgemaakt met Visual Basic, zult u ook makkelijker leren programmeren met Microsoft Word, Microsoft Project, Microsoft Access en andere Microsoft-toepassingen die werken met Visual Basic. U kunt deze toepassingen ook eenvoudig besturen vanuit uw Visual Basic-programmacode. In Microsoft Excel kunt u leren programmeren met Visual Basic in uw eigen tempo. Bovendien blijven de macro's die u in Microsoft Excel 4.0 hebt gemaakt bruikbaar. De tijd die u in het maken van deze macro's hebt geïnvesteerd, is dus niet verloren.

In Microsoft Excel 5.0 kunt u nog steeds Microsoft Excel 4.0-macro's schrijven en uitvoeren. Alle nieuwe voorzieningen van Microsoft Excel 5.0 kunnen worden gebruikt door de macrofuncties. Microsoft Excel 5.0 kan bestaande Microsoft Excel 4.0-macro's niet automatisch vertalen in Visual Basic, maar u kunt deze macro's wel gebruiken vanuit uw Visual Basic-programmacode.

Zie hoofdstuk 2, "Opgenomen macro's bewerken", en hoofdstuk 5, "Werken met objecten in Visual Basic", voor een complete uitleg van Visual Basic in Microsoft Excel.

Informatie voor gebruikers van Microsoft Excel 4.0-macro's

In dit gedeelte worden ervaren gebruikers van Microsoft Excel 4.0-macro's verwezen naar informatie over het leren en gebruiken van Visual Basic. Voor uitgebreide informatie kunt u de materialen raadplegen waarnaar in dit gedeelte wordt verwezen.

Macro's opnemen in Visual Basic

In Microsoft Excel 5.0 kunt kiezen in welke taal u een macro wilt opnemen.

▶ Kiezen in welke programmeertaal u een macro wilt opnemen

1. Kies **Macro opnemen** in het menu **Extra** en kies vervolgens **Nieuwe macro opnemen**.
2. Kies de knop Opties.
3. Selecteer onder "Taal" het keuzerondje naast "MS Excel Visual Basic" of het keuzerondje naast "MS Excel 4.0 Macro".

Net als in Microsoft Excel versie 4.0 kunt u in versie 5.0 macro's opnemen in een werkmap die telkens wordt geopend als u Microsoft Excel start. In Microsoft Excel versie 4.0 voor Windows heet deze werkmap ALGEMEEN.XLM. In Microsoft Excel versie 4.0 voor de Macintosh heet deze werkmap ALGEMEEN MACROBLAD. In Microsoft Excel versie 5.0 heet deze werkmap respectievelijk PERSNLK.XLM en PERSOONLIJKE MACROWERKMAP. Macro's die u hebt gemaakt in versie 4.0 blijven bruikbaar in versie 5.0. Nieuwe macro's worden gewoon in de nieuwe werkmap opgenomen. In Microsoft Excel 5.0 wordt automatisch een nieuwe Persoonlijke macrowerkmap gemaakt als u uw eerste macro opneemt.

Zie hoofdstuk 1, "Terugkerende taken automatiseren", of zoek in de schermhulp naar *Macro opnemen* voor meer informatie over het opnemen van macro's.

Direct met objecten werken in Visual Basic

In Microsoft Excel versie 4.0 werken macro's in de volgorde "selecteren, dan uitvoeren", die op Microsoft Excel in het algemeen van toepassing is. Met Visual Basic hoeft u een object dat u door een procedure wilt laten wijzigen, niet te selecteren. U kunt het object direct laten wijzigen.

Wilt u bijvoorbeeld een tekstbereik vet maken in Microsoft Excel 4.0, dan moet u eerst het bereik selecteren met de functie SELECTEREN alvorens de opmaak van de tekst te wijzigen met de functie OPMAAK.LETTERTYPE. Als u in Visual Basic een tekstbereik vet wilt maken, hoeft u alleen de eigenschap **Vet** van het bereik in te stellen op **Waar**.

```
Bereik("C1:G5").Lettertype.Vet = Waar
```

U ziet dat u met Visual Basic een object, in dit geval het bereik C1:G5, direct kunt wijzigen, zonder het te selecteren of de huidige selectie te annuleren. Zie hoofdstuk 5, "Werken met objecten in Visual Basic", voor meer informatie over hoe u in Visual Basic cellen, bladen, grafieken, vormen en andere objecten kunt wijzigen.

Variabelen: flexibeler dan namen

Als u in Microsoft Excel versie 4.0 een waarde als variabele wilt opslaan, slaat u de waarde gewoonlijk op onder een naam. In Visual Basic wijst u in plaats daarvan waarden aan variabelen toe.

Variabelen zijn veel flexibeler dan namen. U kunt variabelen maken die beschikbaar zijn voor alle procedures, voor alleen de procedures in een module of voor slechts een enkele procedure. U kunt bepalen welk gegevenstype een variabele kan bevatten en u kunt zelfs variabelen maken waarin een combinatie van gegevenstypen kan worden opgeslagen.

In Visual Basic kunt u ook constanten definiëren als u herhaaldelijk naar dezelfde waarde moet verwijzen. Zie hoofdstuk 6, "Werken met Visual Basic-programmacode in procedures", voor meer informatie over variabelen en constanten in Visual Basic.

Werkbladfuncties gebruiken in een procedure

Veel werkbladfuncties kunnen direct worden gebruikt in Visual Basic-procedures. Met de aanduiding **Toepassing**. kunt u gebruik maken van een Microsoft Excel-werkbladfunctie in plaats van een Visual Basic-functie. Zo wordt in de volgende programmacode:

```
X = Bogen(-1)
```

veroorzaakt een fout, terwijl in de volgende programmacode:

```
X = Toepassing.Bogen(-1)
```

gebruik wordt gemaakt van de Microsoft Excel-werkbladfunctie Bogen.

Bestaande macro's gebruiken in een procedure

U kunt uw bestaande macro's in nieuwe Visual Basic-procedures opnemen door gebruik te maken van de methode **Starten**. Als u fouten opspoort en verwijdert in Microsoft Excel 4.0-macro's als onderdeel van uw Visual Basic-procedures, behandelt het foutopsporingsprogramma van Visual Basic deze macro's alsof ze zijn geschreven in Visual Basic. U kunt in uw macro's informatie opvragen en deze aan een procedure doorgeven door gebruik te maken van de macrofunctie TERUG.

Zie *Starten (methode)* in de online Visual Basic Naslaggids voor meer informatie over de methode **Starten** en een gebruiksvoorbeeld van deze methode.

Nieuwe hulpmiddelen voor foutopsporing

Een aantal van de talrijke Visual Basic-hulpmiddelen waarmee u fouten in uw programmacode kunt opsporen zijn:

- Onderbrekingspunten. Een *onderbrekingspunt* geeft een instructie aan of een verzameling voorwaarden waarbij Visual Basic automatisch het programma onderbreekt en de onderbrekingsmodus activeert, zonder de instructie met het onderbrekingspunt uit te voeren. U kunt onderbrekingspunten gebruiken om programmacode en variabelen te onderzoeken op plaatsen waar u vermoedt dat zich een fout voordoet.
- Foutopsporingsknoppen. De foutopsporingsknoppen bevinden zich op de werkbalk Visual Basic, die beschikbaar is als een Visual Basic-module actief is. Met deze knoppen kunt u de instructies of procedures in de programmacode één voor één uitvoeren, onderbrekingspunten instellen en verwijderen, en waarden van variabelen en expressies onderzoeken.

- Het venster Foutopsporing. U geeft dit venster weer met de opdracht **Stap** in het menu **Starten**. Met het venster Foutopsporing kunt u de waarden van variabelen en expressies controleren, terwijl u de programmacode instructie voor instructie uitvoert. Tijdens de uitvoering van programmacode kunt u in dit venster de waarden van variabelen en eigenschappen wijzigen, zodat u kunt zien welk effect verschillende waarden op de programmacode hebben.

Het venster Foutopsporing bevat drie gebieden:

- Het deelvenster Direct. In dit venster wordt het resultaat weergegeven van foutopsporingsinstructies in de programmacode. U kunt ook foutopsporingsinformatie opvragen door opdrachten direct in het deelvenster Direct te typen.
- Het deelvenster Controle. In dit venster worden expressies weergegeven die u wilt controleren terwijl de programmacode wordt uitgevoerd.
- Het codevenster. In dit venster wordt de instructie weergegeven die het meest recent is uitgevoerd. U kunt door de omliggende programmacode bladeren.

Zie hoofdstuk 8, "Fouten in de programmacode opsporen en programmacode testen", voor meer informatie over deze hulpmiddelen en andere aspecten van foutopsporing in Microsoft Excel 5.0.

Visual Basic-equivalenten voor gewone Macrofuncties

De eenvoudigste manier om de Visual Basic-equivalenten van veel gebruikte macrofuncties en opdrachten van Microsoft Excel zichtbaar te maken is macro's op te nemen in Visual Basic met de macrorecorder. U kunt twee vensters in uw werkmap openen, het ene met uw Visual Basic-module en het andere met het werkblad of de grafiek waaraan u werkt tijdens de opname. Terwijl u werkt, worden Visual Basic-instructies toegevoegd in uw module.

Ongeacht de wijze waarop u programma's schrijft in Microsoft Excel, er zullen altijd regelmatig terugkerende taken zijn die u wilt uitvoeren, zoals het selecteren van bereiken, het besturen van de uitvoering van macro's, het zoeken van informatie in werkmappen en het opvragen van informatie over werkmappen en objecten. In de volgende tabel ziet u waar u in dit boek informatie kunt vinden over het uitvoeren van deze taken met Visual Basic.

Informatie over	Vindt u in dit hoofdstuk
Cellen en bereiken selecteren in werkbladen	Hoofdstuk 5, "Werken met objecten in Visual Basic"
Zoeken naar tekst en waarden in werkmappen	
Informatie opvragen over objecten in Microsoft Excel	
Verwijzingen manipuleren	
Bepalen hoe instructies in een procedure worden uitgevoerd	Hoofdstuk 7, "De uitvoering van de programmacode besturen"
Procedures uitvoeren als reactie op gebeurtenissen	Hoofdstuk 13, "Automatische procedures en invoegmacro's maken"

Het Microsoft Excel 5.0-programmapakket bevat ook voorbeelden van programmacode in Visual Basic en de Microsoft Excel 4.0-macrotaal. Als u deze programmacode wilt bekijken, opent u de werkmappen in de directory of map VOORBLDN, die zich bevindt in de directory of map waarin u Microsoft Excel 5.0 hebt geïnstalleerd.

Aangepaste opdrachten en dialoogvensters maken met Visual Basic

Microsoft Excel 5.0 bevat nieuwe, geavanceerde hulpmiddelen voor het maken van aangepaste menu's, opdrachten en dialoogvensters. Zie hoofdstuk 11, "Besturingselementen en dialoogvensters", en hoofdstuk 12, "Menu's en werkbalken", voor meer informatie over het maken van aangepaste opdrachten en dialoogvensters met Visual Basic.

Aangepaste opdrachten maken

Als u met Microsoft Excel versie 4.0 een aangepast menu of een aangepaste opdracht wilt maken, moet u eerst een menu- of opdrachttabel maken. Vervolgens moet u met macrofuncties als MENU.TOEVOEGEN en OPDRACHT.TOEVOEGEN het aangepaste menu of de aangepaste opdracht in een menubalk of menu plaatsen.

Als u dezelfde taak wilt uitvoeren in Microsoft Excel versie 5.0, wijst u met de Menu-editor aangepaste opdrachten en menu's toe aan menubalken. Zie "Het menusysteem wijzigen met de Menu-editor" in hoofdstuk 12 voor meer informatie over het gebruik van de Menu-editor.

Ingebouwde dialoogvensters weergeven

Als u in Microsoft Excel versie 4.0 een ingebouwd dialoogvenster wilt weergeven tijdens het uitvoeren van een macro, gebruikt u de vragende vorm van de macrofunctie die correspondeert met het dialoogvenster. Met de macrofunctie OPMAAKPROFIEL.BEPALEN? geeft u bijvoorbeeld het dialoogvenster weer waarin u het opmaakprofiel van een werkblad definieert.

Wilt u in Microsoft Excel versie 5.0 tijdens de uitvoering van een procedure een ingebouwd dialoogvenster weergeven, dan gebruikt u de methode **Dialogen** met de aanduiding van het dialoogvenster dat u wilt weergeven. Met de volgende instructie geeft u bijvoorbeeld het dialoogvenster van de opdracht **Openen** in het menu **Bestand** weer.

```
Toepassing.Dialogen(xlDlgOpenen).Weergeven
```

Aangepaste dialoogvensters maken en weergeven

Als u in Microsoft Excel versie 4.0 een aangepast dialoogvenster wilt maken, genereert u met de dialoog-editor een definitie voor een dialoogvenster. Deze definitie neemt u op in een macroblad. Vervolgens geeft u het dialoogvenster weer met de macrofunctie DIALOOGVENSTER.

In Microsoft Excel versie 5.0 slaat u aangepaste dialoogvensters op in dialoogbladen in een werkmap. U definieert een aangepast dialoogvenster met de knoppen op de werkbalk Dialoog, waarmee u een dialoogvenster maakt en weergeeft in een dialoogblad van een werkmap. U gebruikt de methode **Weergeven** in een Visual Basic-procedure om het aangepaste dialoogvenster weer te geven.

U kunt een aangepast dialoogvenster dat is gemaakt in Microsoft Excel 4.0, weergeven door in uw Visual Basic-procedures gebruik te maken van de methode **Dialoogvenster**. In het volgende voorbeeld is de methode **Dialoogvenster** gebruikt om een dergelijk dialoogvenster weer te geven en wordt vervolgens het resultaat gecontroleerd. De variabele `DialoogBereik` verwijst naar het bereik in een Microsoft Excel 4.0-macroblad waarin de definitietabel voor het dialoogvenster zich bevindt.

```
Resultaat = DialoogBereik.Dialoogvenster
Indien Niet Resultaat Dan
    ' Gebruiker heeft dialoogvenster geannuleerd.
Anders
    ' Resultaat is het positiegetal van een gekozen besturingselement
Einde Indien
```

Zie hoofdstuk 11, "Besturingselementen en dialoogvensters", voor meer informatie over het gebruik van dialoogvensters bij het programmeren van Microsoft Excel 5.0.

Invoegmacro's maken

Als u in Microsoft Excel versie 4.0 een macro wilt converteren naar een invoegmacro, hoeft u alleen het macroblad op te slaan in de bestandsindeling voor invoegmacro's. De invoegmacro wordt dan verborgen. U kunt Microsoft Excel 4.0-invoegmacro's ook in Microsoft Excel versie 5.0 uitvoeren.

Als u in Microsoft Excel versie 5.0 een invoegmacro wilt maken, gebruikt u de opdracht **Invoegmacro maken** in het menu **Extra**, terwijl een Visual Basic-module of een macroblad of dialoogblad van Microsoft Excel 4.0 actief is. Als u echter nog wilt kunnen werken met de programmacode, moet u deze opslaan in een werkblad met een andere bestandsnaam, voordat u de werkmap met de Visual Basic-programmacode converteert naar een invoegmacro. Hebt u de programmacode eenmaal geconverteerd naar een invoegmacro, dan kunt u de programmacode in de werkmap niet meer bewerken. U kunt de invoegmacro verspreiden onder andere gebruikers van Microsoft Excel 5.0. Andere gebruikers kunnen uw invoegmacro wel uitvoeren, maar kunnen uw programmacode niet bekijken of bewerken.

Zie hoofdstuk 13, "Automatische procedures en invoegmacro's maken", voor meer informatie over invoegmacro's in Microsoft Excel 5.0.

Macro's die Microsoft Excel 4.0 invoegmacro's oproepen

Verschillende van de invoegmacro's van Microsoft Excel, versie 4.0, zijn niet beschikbaar bij Microsoft Excel, versie 5.0. De functionaliteit ervan is rechtstreeks opgenomen in Microsoft Excel. De meeste van uw Microsoft Excel 4.0-macro's zouden zonder verdere aanpassingen moet kunnen worden uitgevoerd in Microsoft Excel 5.0, aangezien Microsoft Excel de functie-oproepen van deze macro's automatisch vertaalt.

Als u macro's hebt waarin zich functies bevinden uit het Analysis ToolPak (ANALYSIS.XLA en ANALYSF.XLA), Scenariobeheerr (SCENARIO.XLA) of Invoegfuncties (INVOEGFN.XLA) dient u echter gebruik te maken van de invoegmacro Koppelingen bijwerken die wordt geleverd bij Microsoft Excel 5.0 voor het bijwerken van verwijzingen naar deze invoegmacro's.

Zie "Koppelingen naar Microsoft Excel 4.0-invoegmacro's bijwerken" in de *Naslaggids Macrofuncties* voor meer informatie over de invoegmacro Koppelingen bijwerken.

BIJLAGE C

Aangepaste Help-onderwerpen weergeven

Met Microsoft Excel kunt u een Help-systeem maken waarin gebruikers informatie kunnen vinden over dialoogvensters, opdrachten, menu's, knoppen en berichten die tijdens de uitvoering van een macro verschijnen.

De Help-onderwerpen kunnen worden gepresenteerd met een aantal krachtige en visueel interessante voorzieningen, bijvoorbeeld voorgrondvensters, grafische voorstellingen, de mogelijkheid van het ene onderwerp naar het andere te springen, *hypergraphics* (grafische voorstellingen met een of meer onzichtbare grafische objecten, zogenaamde *hot spots*) en nog veel meer.

Als u Help-bestanden wilt maken en compileren, hebt u Microsoft Windows Help Compiler versie 3.1 of later nodig. Zie "Meer informatie krijgen over Microsoft Windows Help Compiler" verderop in deze bijlage voor meer informatie over (het verkrijgen van) Help Compiler.

Het Help-compileerprogramma maakt Help-bestanden voor zowel Windows- als Macintosh-toepassingen. Als u Microsoft Excel voor de Macintosh gebruikt, maakt en compileert u de bronbestanden eerst op een Microsoft Windows-compatibele computer en zet u vervolgens de gecompileerde bestanden op de Macintosh.

Microsoft Windows Help Compiler (HC31.EXE) maakt Help-bestanden maar kan ze niet weergeven. Als u Help-bestanden wilt weergeven, moet u Microsoft Windows Help (WINHELP.EXE) of Microsoft Help voor de Macintosh gebruiken, afhankelijk van de omgeving waarin de door u gemaakte Help zal worden gebruikt.

Microsoft Windows Help wordt meegeleverd met Microsoft Windows en staat vaak in de directory WINDOWS.

Microsoft Help voor de Macintosh wordt meegeleverd met Microsoft Excel en staat vaak in de map MICROSOFT binnen de map SYSTEEM. U moet het Help-compileerprogramma afzonderlijk kopen. Zie "Meer informatie krijgen over Microsoft Windows Help Compiler" verderop in deze bijlage als u wilt weten hoe u het Help-compileerprogramma kunt verkrijgen.

Een Help-systeem maken

Wanneer u hebt gelezen hoe u Help-onderwerpen kunt maken, bent u gereed om het Help-systeem te gaan maken voor uw toepassing in Microsoft Excel. In de volgende lijst staan de algemene stappen. Zie de documentatie bij het Help-compileerprogramma voor specifieke informatie over iedere stap. In dit gedeelte wordt aangenomen dat u een Microsoft Windows-compatibele computer gebruikt.

1. Nadat u de aangepaste opdrachten, dialoogvensters en andere elementen van uw toepassing hebt gemaakt, gaat u informatie verzamelen voor bijbehorende Help-onderwerpen. Beslis welke onderdelen aangepaste Help behoeven, wat voor type Help u wilt geven, hoe u de Help-bestanden wilt structureren en hoe u in ieder bestand de Help-onderwerpen wilt structureren.
2. Schrijf de tekst voor de Help-onderwerpen in de bronbestanden, voer alle benodigde stuurcodes in en sla de bronbestanden in RTF-bestandsindeling op.
3. Maak een projectbestand en compileer het Help-bestand.
4. Start Microsoft Excel, start de eigen toepassing en test de weergave van uw aangepaste Help-onderwerpen.

Help-onderwerpen weergeven

Microsoft Windows Help kan alleen bestanden weergeven die zijn gecompileerd door Microsoft Windows Help Compiler. Microsoft Help voor de Macintosh kan zowel gecompileerde bestanden als platte tekstbestanden weergeven.

Kies een van de volgende drie als u een Help-onderwerp in Microsoft Excel wilt weergeven:

- De methode Help van het object Toepassing
- De knop Help in een bericht dat wordt weergegeven door de instructie **BerichtVenster** of **InvoerVenster**
- Een Help-knop in een aangepast dialoogvenster

De methode Help gebruiken

Wanneer u de methode **Help** in uw Visual Basic-programmacode start, geeft Microsoft Excel het opgegeven Help-onderwerp weer. De methode **Help** heeft de volgende syntaxis:

object.**Help** "***HelpBestand***", ***helpContextID***

Hierin is *object* de objectnaam van de toepassing, *HelpBestand* de naam (inclusief het pad) van het bestand dat het onderwerp bevat en *helpContextID* de aanduiding van het onderwerp binnen het Help-bestand.

In Microsoft Excel voor Windows zou u bijvoorbeeld met de volgende Visual Basic-programmacode het Help-onderwerp nummer 103 kunnen weergeven.

```
Toepassing.Help "C:\RAPPORT\RAPPORT.HLP"; 103
```

In Microsoft Excel voor de Macintosh zou u met de volgende Visual Basic-programmacode hetzelfde onderwerp kunnen weergeven.

```
Toepassing.Help "Hard Disk:Rapport:Rapport Help"; 103
```

Aangepaste Help voor dialoogvensters

U kunt de eigenschap **DialoogHelp** van knoppen gebruiken in combinatie met de methode **Help** van het object Toepassing om Help weer te geven voor dialoogvensters.

Elke knop in een dialoogvenster heeft een eigenschap **DialoogHelp**. Als de eigenschap **DialoogHelp** van een knop **Waar** is en de gebruiker op F1 drukt in Microsoft Excel voor Windows (of COMMAND+SHIFT+VRAAGTEKEN in Microsoft Excel voor de Macintosh) wanneer het dialoogvenster wordt weergegeven, reageert Microsoft Excel alsof de gebruiker de knop heeft gekozen. Microsoft Excel start dan de Visual Basic-programmacode die aan de knop is gekoppeld.

Als u wilt dat de gekoppelde Visual Basic-programmacode een Help-onderwerp weergeeft, gebruikt u de methode **Help** van het object Toepassing zoals in het vorige gedeelte is beschreven. Slechts van één knop in een dialoogvenster kan de eigenschap **DialoogHelp** op **Waar** ingesteld zijn, dus u maakt meestal een knop met het knopvlak "Help" die uitsluitend bedoeld is voor het weergeven van Help. Als van geen enkele knop de eigenschap **DialoogHelp** op **Waar** is ingesteld, heeft het indrukken van F1 geen effect.

Meer informatie krijgen over Microsoft Windows Help Compiler

U kunt Microsoft Windows Help Compiler en informatie over het gebruik ervan bij een van de volgende bronnen verkrijgen:

- De *Microsoft Windows Help Authoring Guide* is beschikbaar op CD-ROM via het Microsoft Software Developer Network. In deze veelomvattende handleiding wordt uitgelegd hoe u een compleet Help-systeem kunt ontwerpen en implementeren, en wordt Microsoft Windows Help Compiler beschreven. Het Help-compileerprogramma en voorbeeldcode zijn bij de gids inbegrepen. Neem contact op met uw software-leverancier of Microsoft Corporation als u de *Microsoft Windows Help Authoring Guide* wilt aanschaffen.
- Microsoft Visual Basic, Professional Edition. In deze versie van Visual Basic is het Help-compileerprogramma en bijbehorende documentatie inbegrepen. De documentatie is een beknopte versie van de *Microsoft Windows Help Authoring Guide*, maar is van goede kwaliteit en voldoet waarschijnlijk aan de behoeften van de meeste gebruikers.
- De Microsoft Excel Developer's Kit en de Microsoft Windows Software Development Kit. In deze pakketten is het Help-compileerprogramma inbegrepen. In de bijbehorende documentatie wordt aangenomen dat u weet hoe u Help kunt schrijven met eerdere versies van het Help-compileerprogramma. Neem contact op met uw software-leverancier of Microsoft Corporation als u een van de pakketten wilt aanschaffen.

BIJLAGE D

Werkbalkknoppen in Microsoft Excel

Microsoft Excel heeft meer dan 200 werkbalkknoppen die u kunt weergeven op werkbalken. De meeste ervan worden weergegeven op een van de standaardwerkbalken. In de tabellen in deze bijlage wordt een korte beschrijving gegeven van alle werkbalkknoppen en vakken, ingedeeld per taakcategorie. Bij alle knoppen en vakken staat een identiteitsnummer dat kan worden gebruikt in Visual Basic en de macrotaal van Microsoft Excel 4.0.

Als u meer gedetailleerde informatie wenst over een knop of vak, kiest u de knop Help op de standaardwerkbalk en vervolgens kiest u op de desbetreffende werkbalk het bedoelde element. Als de knop of het vak niet op een werkbalk wordt weergegeven, dubbelklikt u op de knop Help op de standaardwerkbalk en zoekt u vervolgens de naam van het element in de Help-voorziening.

Categorie Controle

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen in de categorie Controle.

Knop		Werking	Nummer
	Notitie	Voegt een notitie toe aan de huidige selectie.	154
	Alle controlepijlen verwijderen	Verwijdert alle pijlen die bij controle worden gebruikt.	153
	Doelcelpijlen verwijderen	Verwijdert één niveau van pijlen die doelcellen aanwijzen.	147
	Broncelpijlen verwijderen	Verwijdert één niveau van pijlen die broncellen aanwijzen.	149
	Informatievenster weergeven	Geeft het Informatievenster weer.	243

Knop		Werking	Nummer
	Doelcellen traceren	Toont de doelcellen van de huidige cel.	148
	Fout traceren	Toont de cellen die de fout in de huidige cel veroorzaken.	174
	Broncellen traceren	Toont de broncellen van de huidige cel.	242

Categorie Grafieken

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen en het vak in de categorie Grafieken.

Knop of vak		Werking	Nummer
	3D-vlakdiagram	Maakt een 3D-vlakdiagram met 3D-markering.	109
	3D-staafdiagram	Maakt een staafdiagram met 3D-markering.	110
	3D-kolomdiagram	Maakt een kolomdiagram met 3D-markering.	111
	3D-lijndiagram	Maakt een 3D-lijndiagram.	113
	3D-kolomdiagram met perspectief	Maakt een 3D-kolomdiagram met een 3D-tekengebied.	112
	3D-cirkeldiagram	Maakt een 3D-cirkeldiagram met gegevenslabels uitgedrukt als percentage.	114
	3D-oppervlakdiagram	Maakt een 3D-oppervlakdiagram.	116
	Vlakdiagram	Maakt een eenvoudig vlakdiagram.	103
	Staafdiagram	Maakt een eenvoudig staafdiagram.	104
	Grafiektypen	Geeft het palet **Grafiektype** weer, zodat u het type van een grafiek kunt wijzigen.	234
	Wizard Grafieken	Start de Wizard Grafieken zodat u de geselecteerde grafiek kunt bewerken of een ingesloten grafiek kunt maken.	121
	Kolomdiagram	Maakt een eenvoudige kolomdiagram.	105

Knop of vak		Werking	Nummer
	Voorkeurgrafiek	Maakt een grafiek met het standaardgrafiektype dat is ingesteld in het tabblad Grafiek van het dialoogvenster **Opties**.	120
	Ringdiagram	Maakt een ringdiagram.	145
	Horizontale rasterlijnen	Voegt primaire rasterlijnen voor de waarde-as toe of verwijdert deze.	122
	Legenda	Voegt een grafieklegenda toe of verwijdert deze.	124
	Lijndiagram	Maakt een lijngrafiek met markering.	107
	Lijn-/kolomdiagram	Maakt een combinatiediagram waarin een lijndiagram over een kolomdiagram ligt.	118
	Cirkeldiagram	Maakt een cirkeldiagram met gegevenslabels uitgedrukt als percentage.	108
	Radardiagram	Maakt een radardiagram met markering.	117
	Stapelkolomdiagram	Maakt een stapelkolomdiagram.	106
	Verticale rasterlijnen	Voegt primaire rasterlijnen toe voor de categorie-as of verwijdert deze.	123
	Volume/hoog/laag/-slot-diagram	Maakt een combinatiegrafiek waarin over een kolomdiagram een lijndiagram ligt met drie gegevensreeksen voor hoge, lage en slotprijzen van aandelen.	119
	Spreidingsdiagram	Maakt een spreidingsdiagram met markering maar zonder lijnen.	115

Categorie Aangepast

De volgende tabel bevat een overzicht van de knoppen met bijbehorende nummers in de categorie Aangepast. Geen van deze knoppen heeft een standaardwerking. U kunt deze knoppen in uw eigen procedures of in macro's van Microsoft Excel 4.0 gebruiken.

Knop	Nummer	Knop	Nummer
Stripballon	219	Mok	218
Bel	226	Noot	210
Blanco	231	Notitieboekj	217

Knop	Nummer	Knop	Nummer
Zakcalculator	237	Openen	206
Camera	228	Palet	97
Klok	213	Opslaan	207
Klaver	224	Jantje lacht	211
Ruiten	222	Patience	201
Envelop	230	Schoppen	223
Vis	214	Luidspreker	209
Jantje huilt	212	Punaise	215
Hand	229	Telefoon	220
Harten	221	Vuilnisbak	225
Integraal	200	Kruis	208

Categorie Gegevens

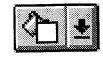
In de volgende tabel wordt de werking beschreven van de werkbalkknoppen in de categorie Gegevens.

Knop		Werking	Nummer
	Gegevens filteren	Geeft alleen de rijen weer met een waarde die overeenkomt met die in de actieve cel en voegt vervolgpijlen in naast iedere kolomnaam in de lijst.	168
	Niveau verlagen	Definieert de geselecteerde detailrijen of kolommen als groep. In een draaitabel worden onderdelen per categorie gegroepeerd om uit meerdere onderdelen één onderdeel te maken.	130
	Details verbergen	Verbergt details van de selectie.	175
	Draaitabelveld	Voegt een subtotaal in voor een rijveld of kolomveld in een bestaande draaitabel. Als in het gegevensgebied een cel is geselecteerd, worden met het klikken op deze knop de samenvattingsfuncties gedefinieerd die in het geselecteerde gegevensveld worden gebruikt.	171
	Draaitabel	Start de Wizard Draaitabel, waarmee u een draaitabel kunt maken of wijzigen.	167

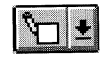
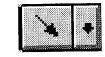
Knop		Werking	Nummer
	Gegevens vernieuwen	Werkt de gegevens in een draaitabel bij nadat de brongegevens zijn gewijzigd.	170
	Details weergeven	Toont details van de selectie.	173
	Pagina's weergeven	Kopieert elke pagina van een paginaveld naar een nieuw werkblad in de huidige werkmap.	172
	Oplopend sorteren	Sorteert de huidige lijst van laagste waarde naar hoogste waarde op de kolom die de actieve cel bevat.	134
	Aflopend sorteren	Sorteert de huidige lijst van hoogste waarde naar laagste waarde op de kolom die de actieve cel bevat.	135
	Niveau verhogen	Verwijdert geselecteerde rijen of kolommen uit hun groep. In een draaitabel wordt een gecombineerde groep onderdelen gescheiden, waarbij de groep overal wordt vervangen door de onderdelen van de groep.	129

Categorie Tekenen

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen en –vakken in de categorie Tekenen.

Knop		Werking	Nummer
	Boog	Geeft een kruiscursor weer waarmee u een boog of cirkelsegment tekent.	85
	Pijl	Geeft een kruiscursor weer waarmee u een boog op het actieve werkblad tekent of voegt een pijl toe aan het actieve grafiekdocument.	77
	Naar voorgrond	Plaatst de geselecteerde objecten voor alle andere objecten.	95
	Kleuren	Geeft het kleurenpalet weer dat u kunt gebruiken om een kleur toe te wijzen aan het geselecteerde object.	233

Knop		Werking	Nummer
	Nieuwe knop	Geeft een kruiscursor weer waarmee u een knop tekent waaraan u een procedure of macro van Microsoft Excel 4.0 toewijst.	80
	Donkere arcering	Brengt donkere arcering aan in geselecteerde cellen of objecten.	49
	Tekenen	Geeft de werkbalk Tekenen weer of verbergt die.	240
	Aanwijzer	Geeft een kruiscursor weer waarmee u tekenobjecten aanwijst.	184
	Schaduw	Plaatst een rechthoek met schaduw rond de geselecteerde cellen of voegt schaduw toe aan de rand van de geselecteerde objecten.	51
	Ovaal	Geeft een kruiscursor weer waarmee u een ovaal of cirkel tekent.	84
	Opgevulde boog	Geeft een kruiscursor weer waarmee u een boog of cirkelsegment tekent, gevuld met het patroon en de kleur van de vensterachtergrond.	90
	Opgevulde ovaal	Geeft een kruiscursor weer waarmee u een ovaal of cirkel tekent, gevuld met het patroon en de kleur van de vensterachtergrond.	89
	Opgevulde vrije-stijl veelhoek	Geeft een kruiscursor weer waarmee u een veelhoek tekent, die een combinatie is van rechte lijnen en lijnen in vrije stijl, gevuld met het patroon en de kleur van de vensterachtergrond.	92
	Opgevulde veelhoek	Geeft een kruiscursor weer waarmee u een veelhoek tekent, gevuld met het patroon en de kleur van de vensterachtergrond.	91
	Opgevulde rechthoek	Geeft een kruiscursor weer waarmee u een rechthoek of vierkant tekent, gevuld met het patroon en de kleur van de vensterachtergrond.	88

Knop		Werking	Nummer
	Vrije-stijl	Geeft een kruiscursor weer waarmee u een doorlopende lijn in vrije stijl tekent.	78
	Vrije-stijl veelhoek	Geeft een kruiscursor weer waarmee u een vorm tekent die een combinatie is van rechte lijnen en lijnen in vrije stijl.	87
	Groeperen	Voegt de geselecteerde grafische objecten samen in één grafisch object.	93
	Lichte arcering	Brengt lichte arcering aan in geselecteerde cellen of objecten.	50
	Lijn	Geeft een kruiscursor weer waarmee u een rechte lijn tekent.	76
	Patroon	Wijzigt het patroon en kleurenpatroon van opgevulde objecten.	232
	Veelhoek	Geeft een kruiscursor weer waarmee u een veelhoek tekent.	86
	Rechthoek	Geeft een kruiscursor weer waarmee u een rechthoek of vierkant tekent.	83
	Vrije vorm wijzigen	Wijzigt de vorm van een veelhoek.	82
	Selectie	Selecteert grafische objecten door er een selectierechthoek rond te tekenen.	81
	Naar achtergrond	Plaatst de geselecteerde objecten rond alle andere objecten.	96
	Figuren	Geeft een palet met tekenvormen weer.	235
	Tekstvak	Geeft een kruiscursor weer waarmee u een tekstvak tekent, waarin u tekst kunt typen.	79
	Groep opheffen	Scheidt gegroepeerde objecten in afzonderlijke objecten.	94

Categorie Bewerken

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen in de categorie Bewerken.

Knop		Werking	Nummer
	Formules wissen	Verwijdert alleen de formules of waarden uit de geselecteerde cellen of wist de geselecteerde objecten.	15
	Opmaak wissen	Verwijdert alleen de opmaak uit de geselecteerde cellen of objecten.	16

Knop		Werking	Nummer
	Kopiëren	Kopieert de selectie naar het Klembord.	13
	Knippen	Knipt de selectie en plaatst deze in het Klembord.	12
	Verwijderen	Verwijdert de geselecteerde cellen of objecten. Als er cellen worden gewist, wordt de opengevallen ruimte ingenomen door omringende cellen.	19
	Kolom verwijderen	Verwijdert de geselecteerde kolom.	21
	Rij verwijderen	Verwijdert de geselecteerde rij.	20
	Omlaag doorvoeren	Kopieert de waarden, formules en opmaak van de cellen in de bovenste rij van de selectie naar de cellen eronder. Als u van beneden naar boven wilt kopiëren, houdt u SHIFT ingedrukt en kiest u de knop.	26
	Rechts doorvoeren	Kopieert de waarden, formules en opmaak van de cellen in de linkerkolom van de selectie naar de cellen ernaast. Als u van rechts naar links wilt kopiëren, houdt u SHIFT ingedrukt en kiest u de knop.	25
	Opmaak kopiëren of plakken	Kopieert opmaak uit een selectie naar een andere.	185
	Invoegen	Voegt lege cellen in op de plaats van de huidige selectie.	22
	Kolom invoegen	Voegt gehele kolommen in op de plaats van de huidige selectie.	24
	Rij invoegen	Voegt gehele rijen in op de plaats van de huidige selectie.	23
	Plakken	Plakt de inhoud van het Klembord.	14
	Opmaak plakken	Plakt alleen de opmaak van de gekopieerde cellen in de geselecteerde cellen.	17
	Waarden plakken	Plakt alleen de waarden in de gekopieerde cellen in de geselecteerde cellen.	18
	Herhalen	Herhaalt de laatste bewerking of opdracht.	11
	Ongedaan maken	Maakt de laatste bewerking of opdracht ongedaan.	10

Categorie Bestand

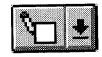
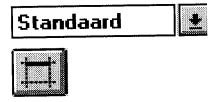
In de volgende tabel wordt de werking beschreven van de werkbalkknoppen in de categorie Bestand.

Knop		Werking	Nummer
	Bestand zoeken	Geeft het dialoogvenster **Bestand zoeken** weer, zodat u bestanden kunt zoeken op kenmerk.	177
	Nieuwe grafiek	Voegt een nieuw grafiekblad in de actieve werkmap in.	8
	Nieuw dialoogblad	Voegt een nieuw dialoogblad links van de actieve werkmap in.	245
	Nieuw macroblad	Voegt een nieuw macroblad in de actieve werkmap in.	6
	Nieuwe module	Voegt een nieuwe module in de actieve werkmap in.	190
	Nieuw werkblad	Voegt een nieuw werkblad in de actieve werkmap in.	7
	Nieuwe werkmap	Maakt een nieuwe werkmap.	9
	Bestand openen	Geeft het dialoogvenster **Openen** weer, zodat u een bestaand bestand kunt openen.	1
	Afdrukken	Drukt het actieve blad af volgens de huidige pagina- en afdrukinstellingen.	3
	Afdrukvoorbeeld	Geeft het actieve blad weer als afdrukvoorbeeld.	4
	Circulatielijst toevoegen of bewerken	Bewerkt de circulatielijst voor een document. Voegt een circulatielijst toe als er geen is.	162
	Bestand opslaan	Slaat wijzigingen in de actieve werkmap op.	2
	Bericht verzenden	Verzendt het huidige document als een elektronisch bericht.	163
	Afdrukbereik bepalen	Definieert het geselecteerde bereik als het af te drukken gebied.	5
	Bestandsstatus	Schakelt een bestand tussen de status Alleen-lezen en Lezen-schrijven.	165
	Bestand bijwerken	Werkt een alleen-lezen-bestand bij tot de laatst opgeslagen versie.	164

Categorie Opmaak

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen en vakken in de categorie Opmaak.

Knop of vak		Werking	Nummer
	Auto-opmaak	Brengt de tabelopmaak aan die het laatst is toegepast met de opdracht **Automatische opmaak** in het menu **Opmaak**. Als u de volgende opmaak in de lijst wilt aanbrengen, houdt u SHIFT ingedrukt en kiest u deze knop.	52
	Randen	Geeft een palet met verschillende typen rand weer.	198
	Onderrand	Voegt een rand toe aan de onderkant van elke geselecteerde cel of verwijdert deze.	47
	Dubbele onderrand	Voegt een dubbele rand toe aan de onderkant van elke geselecteerde cel of verwijdert deze.	48
	Kommanotatie	Brengt de kommanotatie aan in de geselecteerde cellen.	55
	Valutanotatie	Brengt de valutanotatie aan in de geselecteerde cellen.	53
	Donkere schaduw	Brengt een donkere schaduw aan in de geselecteerde cellen of objecten.	49
	Minder decimalen	Geeft een decimaal minder aan als u de knop kiest.	57
	Meer decimalen	Geeft een decimaal meer aan als u de knop kiest.	56
	Linkerrand	Voegt een rand toe aan de linkerkant van elke geselecteerde cel of verwijdert deze.	44
	Lichte schaduw	Brengt een lichte schaduw aan in de geselecteerde cellen of objecten.	50

Knop of vak		Werking	Nummer
	Omtrek	Voegt een rand toe rondom de geselecteerde cellen.	43
	Procentnotatie	Brengt de procentnotatie aan in de geselecteerde cellen.	54
	Rechterrand	Voegt een rand toe aan de rechterkant van elke geselecteerde cel of verwijdert deze.	45
	Opmaakprofiel	Geeft een lijst weer van de gedefinieerde celopmaakprofielen. Selecteer het profiel dat u op de selectie wilt toepassen of typ een nieuwe naam en druk op ENTER als u een opmaak wilt definiëren die gebaseerd is op de geselecteerde cellen.	70
	Bovenrand	Voegt een rand toe aan de bovenkant van elke geselecteerde cel of verwijdert deze.	46
			

Categorie Dialoogvenster

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen in de categorie Dialoogvenster.

Knop		Werking	Nummer
	Aankruisvakje	Maakt een aankruisvakje.	140
	Vervolgkeuze-lijst met Invoervak	Maakt een vervolgkeuzelijst met Invoervak.	197
	Keuzelijst met Invoervak	Maakt een Invoervak/keuzelijst-combinatie.	188

Knop		Werking	Nummer
	Besturingselement-eigenschappen	Wijzigt de eigenschappen van het geselecteerde besturingselement.	238
	Vervolgkeuzelijst	Maakt een vervolgkeuzelijst.	182
	Invoervak	Maakt een Invoervak.	142
	Programmacode bewerken	Bewerkt of maakt programmacode voor het geselecteerde besturingselement.	244
	Groepsvak	Maakt een groepsvak.	181
	Label	Maakt een tekstlabel.	199
	Keuzelijst	Maakt een keuzelijst.	144
	Keuzerondje	Maakt een keuzerondje.	141
	Dialoogvenster starten	Start een dialoogvenster in een afzonderlijk venster.	187
	Schuifbalk	Maakt een schuifbalk.	143
	Ringveld	Maakt een ringveld.	183
	Tabvolgorde	Stelt de tabvolgorde in voor besturingselementen in een dialoogblad.	186
	Rasterlijnen	Geeft de rasterlijnen op een dialoogblad weer of verbergt ze.	239

Categorie Formule

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen in de categorie Formule.

Knop		Werking	Nummer
	Auto-SOM	Voegt de functie SOM in en een voorgesteld sombereik gebaseerd op de gegevens boven of links van de actieve cel.	39
	Dubbele punt	Plaatst een dubbele punt (:) op de invoegpositie op de formulebalk.	35
	Komma	Plaatst een komma (,) op de invoegpositie op de formulebalk.	36

Knop		Werking	Nummer
	Numeriek/inter-punctie	Beperkt handschriftherkenning tot cijfers en interpunctie (bij Microsoft Windows for Pen Computing).	42
	Deelteken	Plaatst een deelteken (/) op de invoegpositie op de formulebalk.	31
	Dollarteken	Plaatst een dollarteken ($) op de invoegpositie op de formulebalk.	38
	Gelijkteken	Plaatst een "is gelijk"-teken (=) op de invoegpositie op de formulebalk.	27
	Exponentteken	Plaatst een exponentteken (^) op de invoegpositie op de formulebalk.	32
	Wizard Functies	Geeft de Wizard Functies weer zodat u een functie kunt invoegen in de geselecteerde cellen.	40
	Haakje openen	Plaatst een haakje openen [(] op de invoegpositie op de formulebalk.	33
	Minteken	Plaatst een minteken (–) op de invoegpositie op de formulebalk.	29
	Vermenigvuldigings-teken	Plaatst een maalteken (*) op de invoegpositie op de formulebalk.	30
	Naam plakken	Geeft het dialoogvenster **Naam plakken** weer zodat u werkbladnamen kunt plakken in de geselecteerde cellen.	41
	Procentteken	Plaatst een procentteken (%) op de invoegpositie op de formulebalk.	37
	Plusteken	Plaatst een plusteken (+) op de invoegpositie op de formulebalk.	28
	Haakje sluiten	Plaatst een haakje sluiten [)] op de invoegpositie op de formulebalk.	34

Categorie Macro

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen en vakken in de categorie Macro.

Knop		Werking	Nummer
	Wizard Functies	Geeft de Wizard Functies weer zodat u een functie kunt invoegen in de geselecteerde cellen.	40
	Nieuwe module	Voegt een nieuwe module in de actieve werkmap in.	190

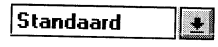
	Directe controle	Geeft de waarde van de geselecteerde expressie weer.	194
	Menu-editor	Geeft het dialoogvenster **Menu-editor** weer zodat u menubalken en opdrachten kunt wijzigen.	192
	Objectenoverzicht	Geeft procedures, objecten, methoden en eigenschappen weer.	191
	Naam plakken	Geeft het dialoogvenster **Naam plakken** weer zodat u werkbladnamen kunt plakken in de geselecteerde cellen.	41
	Macro opnemen	Neemt acties op om een procedure te maken.	98
	Macro hervatten	Hervat een gestopte procedure.	102
	Macro starten	Start de procedure die u bewerkt, vanaf de invoegpositie.	100
	Stap	Voert de volgende instructie uit door in de procedure te stappen.	195
	Macrostap	Voert de procedure die u bewerkt, instructie voor instructie uit vanaf de invoegpositie.	101
	Stap over	Voert de volgende instructie uit door over de procedure te stappen.	196
	Macro-opname stoppen	Stopt de actieve procedure. Als de procedure Macro opnemen wordt uitgevoerd, wordt door het kiezen van deze knop het opnemen gestopt.	99
	Onderbrekingspunt	Stelt onderbrekingspunten in of verwijdert deze.	193

Categorie Tekenopmaak

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen en vakken in de categorie Tekenopmaak.

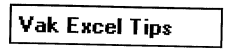
Knop of vak		Werking	Nummer
	Links uitlijnen	Lijnt de inhoud van cel, tekstvak of knop, of geselecteerde tekst in een grafiek links uit.	63
	Rechts uitlijnen	Lijnt de inhoud van cel, tekstvak of knop, of geselecteerde tekst in een grafiek rechts uit.	65
	Vet	Brengt de opmaak Vet aan in de geselecteerde cellen of objecten.	58
	Centreren	Centreert de inhoud van cel, tekstvak of knop, of geselecteerde tekst in een grafiek.	64

Knop of vak		Werking	Nummer
	Centreren over kolommen	Centreert de tekst uit een cel horizontaal over de geselecteerde kolommen.	67
	Tekstkleur wijzigen	Wijzigt de kleur van lettertypen in geselecteerde objecten.	62
	Kleiner lettertype	Vermindert de lettergrootte van de geselecteerde tekst tot het eerstvolgende kleinere formaat in het vak "Lettergrootte".	72
	Dubbel onderstrepen	Brengt dubbele onderstreping aan in de geselecteerde cellen of objecten.	176
	Tekstkleur	Geeft een kleurenpalet weer waarmee u de kleur van het lettertype of de tekst in geselecteerde cellen kunt wijzigen.	236
	Lettertype	Geeft een lijst van de beschikbare lettertypen weer. Selecteer het lettertype dat u wilt aanbrengen in de huidige selectie.	68
	Lettergrootte	Geeft een lijst weer van de beschikbare formaten voor het lettertype in het vak "Lettertype". Selecteer het formaat dat u wilt aanbrengen in de huidige selectie.	69
	Groter lettertype	Vermeerdert de lettergrootte van de geselecteerde tekst tot het eerstvolgende grotere formaat in het vak "Lettergrootte".	71
	Cursief	Brengt cursieve opmaak aan in de geselecteerde cellen of objecten.	59
	Uitvullen	Vergroot de afstand tussen woorden in een tekstvak, knop of cel met teruglopende tekst om elke regel uit te vullen.	66

Knop of vak		Werking	Nummer
	Tekst rechts draaien	Draait tekst een kwartslag zodat deze van boven naar beneden moet worden gelezen.	75
	Tekst links draaien	Draait tekst een kwartslag zodat deze van beneden naar boven moet worden gelezen.	74
	Doorhalen	Brengt de opmaak Doorhalen aan in de geselecteerde cellen of objecten.	61
	Opmaakprofiel	Geeft een lijst weer van de gedefinieerde celopmaken. Selecteer de opmaak die u wilt aanbrengen in de huidige selectie.	70
	Onderstrepen	Brengt onderstreping aan in de geselecteerde cellen of objecten.	60
	Verticale tekst	Plaatst teksttekens boven elkaar, zodat de tekst verticaal moet worden gelezen.	73

Categorie Wizard Tips

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen en het vak in de categorie Wizard Tips.

Knop of vak	Werking		Nummer
	Help bij tips	Geeft een uitgebreide uitleg van de huidige tip.	180
	Vak Wizard Tips	Geeft de huidige tip weer. Met dit vak kunt u door een lijst met tips schuiven.	178
	Excel Tips	Geeft aan of er een tip beschikbaar is en geeft de werkbalk Wizard Tips weer.	179

Categorie Hulpmiddelen

In de volgende tabel wordt de werking beschreven van de werkbalkknoppen en vakken in de categorie Hulpmiddelen.

Knop of vak		Werking	Nummer
	Auto-SOM	Voegt de functie SOM in en een voorgesteld sombereik gebaseerd op de gegevens boven of links van de actieve cel.	39
	Nu berekenen	Berekent de formules in alle geopende documenten of de formule in de formulebalk.	126
	Camera	Kopieert een figuur van het geselecteerde cellenbereik. Klik in het werkblad om de figuur te plakken en deze te koppelen aan de broncellen.	125
	Nieuwe knop	Geeft een kruiscursor weer waarmee u een knop tekent waaraan u een procedure of macro van Microsoft Excel 4.0 toewijst.	80
	Titels blokkeren	Blokkeert de splitsing in het actieve venster of heft de blokkering op. Als er geen splitsing is, wordt het venster links van en boven de actieve cel gesplitst en worden de titels geblokkeerd.	137
	Volledig scherm	Schakelt het weergeven van het volledig scherm aan of uit.	241
	Wizard Functies	Geeft de Wizard Functies weer zodat u een functie kunt invoegen in de geselecteerde cellen.	40
	Help	Voegt een vraagteken toe aan de muisaanwijzer. Wanneer u een opdracht kiest of op een schermgebied klikt, wordt er een toepasselijk Help-onderwerp weergegeven. Kies deze knop nogmaals als u het vraagteken wilt annuleren. Als u dubbelklikt op deze knop, wordt het dialoogvenster voor het zoeken in de Help weergegeven.	128

Knop of vak		Werking	Nummer
	Cel blokkeren	Blokkeert de geselecteerde cellen of objecten, zodat er geen wijzigingen in kunnen worden aangebracht wanneer het document beschermd is.	136
	Microsoft Access	Start Microsoft Access of schakelt ernaar als deze toepassing al gestart is.	216
	Microsoft Fox	Start Microsoft FoxPro of schakelt ernaar als deze toepassing al gestart is. (In Microsoft Excel voor de Macintosh heeft de categorie Hulpmiddelen in plaats hiervan de knop Microsoft FoxBASE+® , die Microsoft FoxBASE+ start of ernaartoe schakelt).	160
	Microsoft Mail	Start Microsoft Mail of schakelt ernaar als deze toepassing al gestart is.	203
	Microsoft PowerPoint	Start Microsoft PowerPoint of schakelt ernaar als deze toepassing al gestart is.	204
	Microsoft Project	Start Microsoft Project of schakelt ernaar als deze toepassing al gestart is.	205
	Microsoft Schedule+	Start Microsoft Schedule+ of schakelt ernaar als deze toepassing al gestart is.	227
	Microsoft Word	Start Microsoft Word of schakelt ernaar als deze toepassing al gestart is.	202
	Naam plakken	Geeft het dialoogvenster **Naam plakken** weer zodat u werkbladnamen kunt plakken in de geselecteerde cellen.	41
	Scenario's	Voegt scenario's toe, geeft ze weer of bewerkt ze.	166
	Huidige gebied selecteren	Selecteert een rechthoekig cellenbereik rond de actieve cel dat door lege rijen en kolommen is omsloten.	133

Knop of vak		Werking	Nummer
	Zichtbare cellen selecteren	Selecteert alleen de zichtbare cellen en niet de verborgen rijen en kolommen.	132
	Overzichts-knoppen weergeven	Geeft de overzichtssymbolen weer op een werkblad dat een overzicht bevat, of verbergt ze. Als er geen overzicht is, wordt u gevraagd of u er een wilt maken.	131
	Oplopend sorteren	Sorteert de huidige lijst van laagste waarde naar hoogste waarde op de kolom die de actieve cel bevat.	134
	Aflopend sorteren	Sorteert de huidige lijst van hoogste waarde naar laagste waarde op de kolom die de actieve cel bevat.	135
	Spelling	Start de spellingcontrole van het actieve document of de tekst in de formulebalk.	127
	Tekstvak	Geeft een kruiscursor weer waarmee u een tekstvak tekent, waarin u tekst kunt typen.	79
	In- en uitzoomen	Wijzigt de vergroting van het actieve werkblad.	189
	Inzoomen	Geeft het werkblad weer in de eerstvolgende hogere vergroting.	138
	Uitzoomen	Geeft het werkblad weer in de eerstvolgende lagere vergroting.	139

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Help oproepen in een Visual Basic®-module

Als u Help wilt oproepen voor een eigenschap, methode, instructie of een ander sleutelwoord, selecteert u het sleutelwoord en drukt u vervolgens op F1 (Microsoft Excel voor Windows™) of ⌘+? (Microsoft Excel voor de Apple Macintosh®).

Objecten-overzicht

Als u wilt bladeren door de objectbibliotheken die beschikbaar zijn voor Visual Basic, klikt u op de knop Objectenoverzicht.

Taken automatiseren met de macrorecorder

U kunt snel taken automatiseren en aangepaste opdrachten maken met behulp van de *Macrorecorder*. Wanneer u de Macrorecorder start, neemt Microsoft Excel automatisch uw toets- en muisbewerkingen op, zodat u deze later opnieuw kunt uitvoeren met een druk op een toets of door te klikken op een knop.

Een macro opnemen

Macro opnemen

1. Kies de opdracht **Macro opnemen** in het menu **Extra** en kies daarna **Nieuwe macro opnemen**. U kunt ook klikken op de knop Macro opnemen op de Visual Basic werkbalk.
2. Typ een naam en een beschrijving van de macro en kies daarna de knop OK.
3. Voer de handelingen uit die u wilt opnemen.

Macro-opname stoppen

4. Klik op de knop Macro-opname stoppen.

Nadat u een macro hebt opgenomen, kunt u deze uitvoeren met de opdracht **Macro** in het menu **Extra** of u kunt de macro toewijzen aan een menu-opdracht, knop of ander object.

Sneltoetsen

Als u	**Drukt u op (Microsoft Windows)**
Een Visual Basic-procedure wilt uitvoeren vanaf de invoegpositie	F5
Een macro wilt onderbreken	ESC
Het dialoogvenster **Objectenoverzicht** wilt weergeven	F2
Het venster Foutopsporing wilt weergeven	CTRL+G
De volgende regel van de programmacode van een Visual Basic-procedure wilt uitvoeren vanuit het venster Foutopsporing	F8

43758NL

Microsoft®