Microsoft Excel VBA voor dummies®

Michael Alexander
John Walkenbach

BBNC uitgevers

Amersfoort, 2019

Inhoud in vogelvlucht

1 Kennismaken met Excel VBA-programmeren

IN DIT DEEL . . .

Leer je Visual Basic for Applications kennen.

Krijg je voorbeelden van dingen die je kunt doen met VBA.

Werk je een echte Excel-programmeersessie door.

Begrijp je hoe Excel omgaat met de beveiliging van macro's.

IN DIT HOOFDSTUK

- **Een conceptueel overzicht van VBA**
- **Ontdekken wat je kunt doen met VBA**
- **De voordelen en nadelen van VBA leren kennen**
- **Het fijne te weten komen over VBA**
- **Compatibel blijven met Excel**

Hoofdstuk 1

Wat is VBA?

Misschien kun je niet wachten om je op het programmeren in VBA te storten, maar maak even pas op de plaats. In dit hoofdstuk vind je geen oefenmateriaal. Maar het bevat wel essentiële achtergrondinformatie die je helpt om een Excel-programmeur te worden. Met andere woorden, dit hoofdstuk effent de weg voor alles wat nog komen gaat en laat je zien waar Excel-programmeren precies thuishoort in het totaalplan van het universum. Het is niet zo saai als je misschien denkt, dus weersta de verleiding om meteen door te bladeren naar hoofdstuk 2.

Oké, wat is VBA?

VBA, dat staat voor *Visual Basic for Applications*, is een programmeertaal die is ontwikkeld door Microsoft – je weet wel, dat bedrijf dat elke paar jaar weer een nieuwe versie van Windows aan je probeert te slijten. Net als bij de andere onderdelen van Microsoft Office is de VBA-taal (gratis) ingebouwd in Excel. Kortgezegd, VBA is het gereedschap dat je gebruikt om programma's te ontwikkelen voor Excel.

Stel je voor dat een intelligente robot alles afweet van Excel. Deze robot kan instructies lezen en ook Excel heel snel en accuraat bedienen. Als je wilt dat de robot iets doet in Excel, schrijf je een aantal robotinstructies door gebruik te maken van speciale codes. Dan draag je de robot op om jouw instructies te volgen terwijl jij achteroverleunt en van je koffie

drinkt. Dat is eigenlijk waar VBA over gaat: een codetaal voor robots. Maar bedenk wel dat Excel niet wordt geleverd met een robot of koffie.


TECHNISCHE INFO

EVEN IETS OVER TERMINOLOGIE

De terminologie van Excel-programmeren kan wat verwarrend zijn. Zo is VBA een programmeertaal, maar het doet ook dienst als een macrotaal. In deze context is een macro een set instructies die Excel uitvoert om toetsaanslagen en muisacties na te bootsen. Hoe noem je iets dat in VBA is geschreven en in Excel wordt uitgevoerd? Is het een *macro* of een *programma*? Het Help-systeem van Excel verwijst vaak naar VBA-procedures als *macro's*, en dus doen wij dat ook in dit boek. Maar je mag het wat ons betreft ook gerust een *programma* noemen.

Overal in dit boek kom je de term *automatiseren* tegen. Deze term betekent dat een reeks stappen automatisch wordt uitgevoerd. Als je bijvoorbeeld een macro schrijft die kleur toevoegt aan bepaalde cellen, het werkblad afdrukt en dan de kleur weer verwijdert, dan heb je deze drie stappen *geautomatiseerd*.

Tussen haakjes, het woord *macro* betekent niet **M**essy **A**nd **C**onfusing **R**epeated **O**peration. Het is afkomstig van het Griekse woord *makros*, dat groot betekent. En dat geldt ook voor de salarisverhoging die jou te wachten staat als je een expert wordt in het programmeren van macro's.

Wat kun je doen met VBA?

Je weet waarschijnlijk wel dat mensen Excel voor duizend-en-een dingen gebruiken. Om maar enkele voorbeelden te noemen:

- wetenschappelijke data analyseren;
- begrotingen en projecties maken;
- facturen en andere formulieren maken;
- grafieken maken van gegevens;
- lijsten maken, zoals klantennamen, cijfers voor leerlingen of cadeau-ideeën;
- enzovoort enzovoort.

De lijst is eindeloos, maar je snapt het wel. Het punt is dat Excel voor de meest uiteenlopende taken wordt gebruikt, en dat iedereen die dit boek leest eigen behoeften en verwachtingen heeft over Excel. Wat vrijwel alle

lezers gemeen hebben, is de *behoefte om een bepaald aspect van Excel te automatiseren*. En daar, beste lezer, hebben we nu VBA voor.

Zo zou je een VBA-programma kunnen maken om bepaalde getallen te importeren en vervolgens je maandelijkse verkoopcijfers op te maken en af te drukken. Nadat je het programma hebt ontwikkeld en getest, kun je de macro uitvoeren met een enkele opdracht, waarna Excel automatisch vele tijdrovende procedures uitvoert. In plaats van dat je je door een hele reeks opdrachten heen moet worstelen, kun je op een knop klikken en iets leuks gaan doen terwijl de macro het werk doet.

In de volgende paragrafen beschrijven we enkele veelgebruikte toepassingen van VBA-macro's. Misschien dat daar iets bij zit waar je blij van wordt.

Een hoop tekst invoegen

Als je vaak je bedrijfsnaam, adres en telefoonnummer moet typen in je werkbladen, kun je een macro maken die dat typen van je overneemt. En je kunt hier zover mee gaan als je wilt. Je zou bijvoorbeeld een macro kunnen maken die automatisch een lijst van al je verkopers invoert.

Een taak automatiseren die je vaak uitvoert

Stel dat je een salesmanager bent en een rapport van de eindemaandcijfers moet voorbereiden om je baas tevreden te houden. Als de taak niet gecompliceerd is, kun je een VBA-programma ontwikkelen om dat voor jou te doen. Je baas zal onder de indruk zijn van de consistente hoge kwaliteit van je rapporten en je wordt gepromoveerd naar een functie waar je absoluut niet voor gekwalificeerd bent.

Terugkerende operaties automatiseren

Als je dezelfde handeling moet uitvoeren op, zeg, twaalf Excel-werkmappen, kun je een macro opnemen terwijl je de taak uitvoert op de eerste werkmap en vervolgens de macro de handeling laten herhalen op de andere werkmappen. Het mooie hiervan is dat Excel nooit zal klagen over verveling. De macrorecorder van Excel is vergelijkbaar met het opnemen van beelden met een camcorder. Maar je hebt er geen camera voor nodig en de batterij hoef je ook nooit op te laden.

Een zelfgemaakte opdracht creëren

Geef je vaak dezelfde reeks Excel-opdrachten? Zo ja, bespaar dan wat tijd door een macro te ontwikkelen die deze opdrachten combineert tot een enkele zelfgemaakte opdracht die je kunt uitvoeren met een enkele toetsaanslag of muisklik. Je zult er waarschijnlijk niet *heel veel* tijd mee besparen, maar je wordt er wel accurater van. En je buurman aan het volgende bureau zal onder de indruk zijn.

Een eigen opdrachtknop creëren

Je kunt de werkbalk Snelle toegang aanpassen met knoppen die jouw eigen macro's uitvoeren. Kantoorpersoneel is vaak erg onder de indruk van knoppen die magische dingen doen. En als je je collega's *echt* wilt verbazen, kun je zelfs nieuwe knoppen toevoegen aan het lint.

Nieuwe werkbladfuncties ontwikkelen

Hoewel Excel beschikt over honderden ingebouwde functies (zoals SOM en GEMIDDELDE), kun je *zelfgemaakte* werkbladfuncties creëren die je formules aanzienlijk kunnen vereenvoudigen. Het zal je verbazen hoe makkelijk dit is. (Hoe dit werkt, lees je in hoofdstuk 20.) En het dialoogvenster Functie invoegen geeft jouw eigen functies weer alsof ze zijn ingebouwd. Mooi spul.

Eigen invoegtoepassingen creëren voor Excel

Waarschijnlijk ben je vertrouwd met enkele invoegtoepassingen die bij Excel worden geleverd. Het Analysis ToolPak, bijvoorbeeld, is een populaire invoegtoepassing. Je kunt VBA gebruiken om je eigen gespecialiseerde invoegtoepassingen te ontwikkelen.

Voordelen en nadelen van VBA

In deze paragraaf schetsen we de goede zaken van VBA, maar ook de minder goede dingen.

Voordelen van VBA

Je kunt vrijwel alles wat je doet in Excel automatiseren. Hiervoor schrijf je instructies die Excel uitvoert. Het heeft meerdere voordelen om een taak met VBA te automatiseren:

- Excel voert de taak altijd op precies dezelfde manier uit. (In de meeste gevallen is consistentie een goede zaak.)
- Excel voert de taak veel sneller uit dan jij met de hand kunt doen.
- Als je een goede macroprogrammeur bent, voert Excel de taken altijd uit zonder fouten (wat we waarschijnlijk niet over jou kunnen zeggen, hoe nauwgezet je ook bent).
- Als je een en ander goed hebt opgezet, kan ook iemand die niets afweet van Excel de taak uitvoeren door de macro te draaien.
- Je kunt in Excel dingen doen die anders onmogelijk zijn. Dat zal je populariteit op kantoor zeker ten goede komen.
- Bij lange en tijdrovende taken hoef je niet verveeld achter je computer te blijven zitten. Excel doet het werk, terwijl jij rondhangt bij de koffieautomaat.

Nadelen van VBA

Natuurlijk moeten we evengoed stilstaan bij de nadelen (of *potentiële* nadelen) van VBA:

- Je moet weten hoe je programma's schrijft in VBA (maar daarom heb je ook dit boek gekocht, nietwaar?). Gelukkig is het niet zo moeilijk als je zou denken.
- Andere mensen die jouw VBA-programma's moeten gebruiken moeten over hun eigen exemplaar van Excel beschikken. Het zou mooi zijn als je op een knop kon drukken die je Excel/VBA-toepassing omtovert in een zelfstandig werkend programma, maar dat is niet mogelijk (en zal dat waarschijnlijk ook nooit zijn).
- Soms gaat er iets fout. Met andere woorden, je kunt er niet blind op vertrouwen dat je VBA-programma altijd foutloos werkt onder alle omstandigheden. Welkom in de wereld van het debuggen en van de technische ondersteuning (als anderen jouw macro's gebruiken).
- VBA is een bewegend doelwit. Zoals je weet, wordt Excel voortdurend door Microsoft geüpdatet. Microsoft spant zich in om versies compatibel te houden, maar het kan zijn dat jouw VBA-code niet correct werkt in oudere of toekomstige versies van Excel.

VBA in een notendop

Om je alvast een voorproefje te geven van wat je te wachten staat, vatten we hier al even wat dingen samen.

- **Je voert acties uit in VBA door het schrijven (of opnemen) van code in een VBA-module.** Je bekijkt en bewerkt VBA-modules in de Visual Basic Editor (VBE).
- **Een VBA-module bestaat uit Sub-procedures.** Een Sub-procedure is een brok computercode die een bepaalde actie uitvoert op of met objecten (komen we zo op). In het volgende voorbeeld zie je een simpele Sub-procedure met de naam AddEmUp. Dit verbazingwekkende programma geeft bij uitvoering het resultaat weer van 1 plus 1:

```
Sub AddEmUp()
    Sum = 1 + 1
    MsgBox "Het antwoord is " & Sum
End Sub
```

- **Een VBA-module kan ook Function-procedures bevatten.** Een Function-procedure geeft een enkele waarde terug. Je kunt de procedure aanroepen vanuit een andere VBA-procedure of deze zelfs gebruiken als een functie in een werkbladformule. Hier zie je een voorbeeld van de Function-procedure AddTwo. Deze functie accepteert twee getallen (argumenten genoemd) en geeft de som van die waarden terug:

```
Function AddTwo(arg1, arg2)
    AddTwo = arg1 + arg2
End Function
```

- **VBA manipuleert objecten.** Excel biedt tientallen objecten die je kunt manipuleren. Voorbeelden van objecten zijn een werkmap, een werkblad, een cellenbereik, een grafiek en een vorm. Er zijn er nog veel meer beschikbaar en je kunt ze manipuleren met VBA-code.
- **Objecten zijn geordend in een hiërarchie.** Objecten kunnen fungeren als *containers* voor andere objecten. Boven aan de hiërarchie staat Excel. Excel is zelf een object met de naam Application. Het Application-object bevat andere objecten, zoals Workbook-objecten (werkmappen) en AddIn-objecten (invoegtoepassingen). Het Workbook-object kan andere objecten bevatten, zoals Worksheet-objecten (werkbladen) en Chart-objecten (grafieken). Een Worksheet-object kan objecten bevatten zoals Range-objecten (celbereiken) en PivotTable-objecten (draaitabellen). De term *objectmodel* verwijst naar de rangschikking van deze objecten. (Objectmodelnerds lezen er meer over in hoofdstuk 4.)

- **Objecten van hetzelfde type vormen een verzameling.** Zo bestaat de verzameling Worksheets uit alle werkbladen in een bepaalde werkmap. De verzameling Charts bestaat uit alle grafiekobjecten in een werkmap. Verzamelingen zijn zelf ook objecten.
- **Je verwijst naar een object door zijn positie in de objectenhiërarchie te specificeren met behulp van een punt als scheidingsteken.** Zo kun je naar de werkmap Map1.xlsx verwijzen als:

  ```
  Application.Workbooks("Map1.xlsx")
  ```

  Dit verwijst naar de werkmap Map1.xlsx in de verzameling Workbooks. De verzameling Workbooks bevindt zich weer in het object Application (ofwel Excel). Als we dit uitbreiden naar een ander niveau, kun je naar Blad1 in Map1.xlsx verwijzen als:

  ```
  Application.Workbooks("Map1.xlsx").Worksheets("Blad1")
  ```

  We kunnen nog verder afdalen en verwijzen naar een specifieke cel (in dit geval cel A1):

  ```
  Application.Workbooks("Map1.xlsx").Worksheets("Blad1").Range("A1")
  ```

- **Als je bepaalde verwijzingen weglaat, gebruikt Excel de actieve objecten.** Als Map1.xlsx de actieve werkmap is, kun je de vorige verwijzing als volgt vereenvoudigen:

  ```
  Worksheets("Blad1").Range("A1")
  ```

  Als je weet dat Blad1 het actieve blad is, kun je de verwijzing nog verder vereenvoudigen:

  ```
  Range("A1")
  ```

- **Objecten hebben eigenschappen.** Je kunt een eigenschap zien als een *instelling* voor een object. Zo heeft een Range-object eigenschappen als Value en Address. Een Chart-object heeft eigenschappen als HasTitle en Type. Je kunt VBA gebruiken om objecteigenschappen vast te stellen en te wijzigen.
- **Je verwijst naar een eigenschap van een object door de objectnaam te combineren met de eigenschapsnaam, gescheiden door een punt.** Zo kun je als volgt verwijzen naar de eigenschap Value in cel A1 op Blad1:

  ```
  Worksheets("Blad1").Range("A1").Value
  ```

- **Je kunt waarden toekennen aan variabelen.** Een *variabele* is een benoemd element dat informatie opslaat. Je kunt variabelen gebruiken in je VBA-code om zaken op te slaan als waarden, tekst en eigenschapsinstellingen. Om de waarde in cel A1 op Blad1 toe te wijzen aan een variabele Rente, gebruik je het volgende VBA-statement:

  ```
  Rente = Worksheets("Blad1").Range("A1").Value
  ```

- **Objecten hebben methoden.** Een *methode* is een actie die Excel uitvoert met een object. Zo is ClearContents een van de methoden voor een Range-object. Deze methode wist de inhoud van een bereik.
- **Je specificeert een methode door het object te combineren met de methode, gescheiden door een punt.** Zo wist het volgende statement de inhoud van cel A1:

```
Worksheets("Blad1").Range("A1").ClearContents
```

- **VBA omvat alle concepten van moderne programmeertalen, waaronder variabelen, arrays en loops (lussen).** Met andere woorden, als je bereid bent enige tijd te besteden aan het leren van de kneepjes, kun je code schrijven die ongelooflijke dingen doet.

Geloof het of niet, maar de vorige opsomming beschrijft VBA in een notendop. Nu hoef je je alleen nog maar te verdiepen in de details. Om deze reden houdt het boek hier nog niet op.

Excel-compatibiliteit


BELANGRIJK

Dit boek is geschreven voor de desktopversies van Excel 2016 en Excel 2019. Als je niet een van deze versies hebt, loop je het risico hier en daar de draad kwijt te raken.

Als je van plan bent om je Excel/VBA-bestanden te delen met andere gebruikers, is het van cruciaal belang dat je weet welke Excel-versie ze gebruiken. Mensen met oudere versies kunnen niet profiteren van functionaliteit die aan latere versies is toegevoegd. Als je bijvoorbeeld VBA-code schrijft die verwijst naar cel XFD1048576 (de laatste cel in een werkmap), krijgen mensen met een oudere versie dan Excel 2007 een foutmelding omdat werkbladen van vóór Excel 2007 slechts 65.536 rijen en 255 kolommen telden (de laatste cel was IV65536).

Excel 2010 en later hebben ook enkele nieuwe objecten, methoden en eigenschappen. Als je deze in je code gebruikt, krijgen gebruikers met een oudere versie van Excel een foutmelding wanneer ze je macro draaien. En daar krijg jij dan de schuld van.