

Einen allgemeinen Überblick über VBA gewinnen

Herausfinden, was mit VBA möglich ist

Die Vorteile und Nachteile bei der Verwendung von VBA kennenlernen

Die Fakten über VBA erfahren

Kompatibilitätsaspekte

Kapitel 1

Was ist VBA?

Falls Sie Angst davor haben, sofort mit der VBA-Programmierung zu beginnen, bleiben Sie ganz ruhig. Dieses Kapitel ist völlig frei von Schulungsmaterial aus der Praxis. Es enthält jedoch wichtige Hintergrundinformationen, die Ihnen dabei helfen, Excel-Programmierer zu werden. Mit anderen Worten, dieses Kapitel ebnet Ihnen den Weg für alles Folgende und verschafft Ihnen ein Gefühl dafür, wie sich die Excel-Programmierung in das Weltbild einfügt. Das ist nicht so langweilig, wie Sie vielleicht denken. Versuchen Sie also, das dringende Verlangen zu unterdrücken, zu Kapitel 2 weiterzublättern.

Was also ist VBA?

VBA steht für *Visual Basic for Applications*. Das ist eine Programmiersprache, die von Microsoft entwickelt wurde – Sie wissen schon, das ist das Unternehmen, das Ihnen alle paar Jahre eine neue Windows-Version verkauft. Excel und die anderen Komponenten von Microsoft Office beinhalten die Sprache VBA (ohne zusätzliche Kosten). Kurz gesagt, VBA ist das Tool, das Menschen verwenden, um Programme zur Steuerung von Excel zu entwickeln.

Stellen Sie sich einen intelligenten Roboter vor, der alles über Excel weiß. Dieser Roboter kann Anweisungen lesen, und er kann Excel sehr schnell und sehr präzise bedienen. Wenn Sie wollen, dass der Roboter in Excel etwas erledigt, schreiben Sie unter Verwendung spezieller Codes Anweisungen für ihn. Anschließend weisen Sie den Roboter an, Ihre Anweisungen abzuarbeiten, während Sie sich zurücklehnen und eine Tasse Kaffee trinken. Mit VBA verhält es sich nicht anders. Letztlich ist es eine Codesprache für Roboter. Beachten Sie jedoch, dass es in Excel keine Roboter und keinen Kaffee gibt.



Ein paar Worte zur Terminologie

Die Terminologie für die Excel-Programmierung kann bisweilen verwirrend sein. Beispielsweise ist VBA eine Programmiersprache, dient aber gleichzeitig als Makro-Sprache. In diesem Kontext ist ein Makro eine Reihe von Anweisungen, die Excel ausführt, um Tastatureingaben und Mausektionen zu simulieren. Wie nennt man etwas, das in VBA geschrieben und in Excel ausgeführt wird? Ist es ein *Makro* oder ist es ein *Programm*? Im Hilfesystem von Excel werden VBA-Prozeduren häufig als *Makros* bezeichnet, deswegen wird in diesem Buch diese Terminologie verwendet. Sie können das Ganze auch ein *Programm* nennen.

Im gesamten Buch werden Sie dem Begriff *automatisieren* begegnen. Dieser Begriff bedeutet, dass mehrere Schritte automatisch ausgeführt werden. Wenn Sie beispielsweise ein Makro schreiben, das mehrere Zellen farblich unterlegt, das Arbeitsblatt ausdruckt und dann die Farbe wieder entfernt, haben Sie diese drei Schritte *automatisiert*.

Das Wort *Makro* ist übrigens kein Produkt des Abkürzungsfimmels, sondern kommt aus dem Griechischen: *Makros* bedeutet *groß* – was auch Ihren Gehaltscheck beschreibt, den Sie erhalten, sobald Sie Profi in der Makro-Programmierung geworden sind.

Was ist mit VBA möglich?

Wahrscheinlich wissen Sie, dass die Benutzer Excel für die unterschiedlichsten Aufgaben einsetzen. Hier einige Beispiele:

- ✓ Analyse wissenschaftlicher Daten
- ✓ Budgetberechnung und Prognosen
- ✓ Erstellen von Rechnungen und anderen Formularen
- ✓ Entwicklung von Diagrammen aus Daten
- ✓ Verwaltung von Listen mit Kundennamen, Studentenbenotungen oder Ideen für Weihnachtsgeschenke (ein hübscher Obstkuchen wäre nicht schlecht)
- ✓ und so weiter und so fort

Diese Liste lässt sich endlos fortsetzen, aber ich glaube, Sie haben das Prinzip verstanden. Der entscheidende Punkt ist, dass Excel für die unterschiedlichsten Aufgaben genutzt werden kann und dass jeder, der dieses Buch liest, unterschiedliche Bedürfnisse und Erwartungen im Hinblick auf Excel hat. Aber fast alle Leser haben gemeinsam, dass sie *irgendeinen Aspekt von Excel automatisieren wollen*. Und genau das leistet VBA.

Beispielsweise können Sie ein VBA-Programm schreiben, das verschiedene Zahlen importiert, diese formatiert und zum Monatsende Ihren Umsatzbericht ausdruckt. Nachdem Sie das Programm entwickelt und getestet haben, können Sie das Makro mit einem einzigen Befehl ausführen, sodass Excel automatisch viele zeitaufwendige Prozeduren für Sie ausführt. Statt sich durch eine mühselige Folge von Befehlen quälen zu müssen, klicken Sie einfach auf eine Schaltfläche und wechseln dann zu Facebook, um sich die Zeit zu vertreiben, während Ihr Makro die Arbeit erledigt.

Die folgenden Abschnitte beschreiben kurz einige gebräuchliche Verwendungszwecke für VBA-Makros. Vielleicht gibt es Beispiele, die genau Ihren Vorstellungen entsprechen.

Text einfügen

Wenn Sie in Ihren Arbeitsblättern häufig Ihren Firmennamen, die Adresse und die Telefonnummer einfügen müssen, können Sie ein Makro erstellen, das Ihnen diese Schreibearbeit abnimmt. Dieses Konzept können Sie beliebig erweitern. Beispielsweise könnten Sie ein Makro entwickeln, das automatisch eine Liste aller Vertriebsmitarbeiter erstellt, die für Ihr Unternehmen arbeiten.

Eine häufig ausgeführte Aufgabe automatisieren

Angenommen, Sie sind Vertriebsleiter und wollen jeweils zum Monatsende einen Umsatzbericht aufbereiten, den Ihr Chef dringend braucht. Wenn diese Aufgabe einfach zu bewerkstelligen ist, können Sie auch ein VBA-Programm schreiben, das diese Aufgabe für Sie erledigt. Ihr Chef wird von der durchgängig hohen Qualität Ihrer Berichte beeindruckt sein, und Sie werden befördert bis zu einer Stelle, für die Sie höchst unqualifiziert sind.

Automatisierung wiederholter Operationen

Wenn Sie dieselbe Operation beispielsweise für zwölf verschiedene Excel-Arbeitsmappen ausführen müssen, können Sie ein Makro aufzeichnen, während Sie die Aufgabe in der ersten Arbeitsmappe ausführen, und es anschließend dem Makro überlassen, diese Aufgabe für die anderen Arbeitsmappen auszuführen. Das Schöne daran ist, dass sich Excel nie beschwert, dass es ihm langweilig wird. Der Makro-Recorder von Excel ist vergleichbar mit einer Videokamera, die Liveaktionen aufzeichnet. Aber man benötigt dafür keine Kamera, und die Batterie ist nie gerade leer.

Einen benutzerdefinierten Befehl anlegen

Führen Sie häufig dieselbe Folge an Excel-Befehlen aus? In diesem Fall sparen Sie sich ein paar Sekunden, indem Sie ein Makro entwickeln, das diese Befehle zu einem einzigen benutzerdefinierten Befehl zusammenfasst, den Sie mit einem einzigen Tastendruck oder über einen Klick aktivieren können. Damit sparen Sie vielleicht nicht so viel Zeit, aber Sie arbeiten möglicherweise präziser. Und Ihr Büronachbar wird beeindruckt sein.

Eine benutzerdefinierte Schaltfläche erstellen

Sie können in der Schnellzugriffsleiste eigene Schaltflächen anlegen, die die von Ihnen geschriebenen Makros ausführen. Büromenschen sind immer sehr beeindruckt von Schaltflächen, die zaubern können. Und wenn Sie Ihre Kollegen *wirklich* beeindrucken wollen, können Sie sogar neue Schaltflächen in das Menüband einfügen.

Neue Funktionen für die Arbeitsmappe entwickeln

Obwohl es in Excel Hunderte eingebauter Funktionen gibt (wie beispielsweise SUMME oder MITTELWERT), können Sie *benutzerdefinierte* Funktionen für Ihre Arbeitsmappen entwickeln, die Ihre Formeln wesentlich vereinfachen. Sie werden überrascht sein, wie einfach das ist. (Wie das geht beschreibt Kapitel 20.) Und noch besser ist, dass im Dialogfeld FUNKTION EINFÜGEN Ihre benutzerdefinierten Funktionen angezeigt werden, sodass es noch mehr danach aussieht, als wären sie eingebaut. Sehr interessant!

Benutzerdefinierte Add-Ins für Excel erstellen

Wahrscheinlich sind Sie schon mit einigen der Add-Ins für Excel vertraut. Beispielsweise ist das Add-In Analysis ToolPak sehr gebräuchlich. Mit VBA können Sie Ihre eigenen, spezifischen Add-Ins erstellen.

Vorteile und Nachteile von VBA

Dieser Abschnitt beschreibt kurz die Vorteile von VBA – aber auch die dunkle Seite wird nicht vergessen.

Vorteile von VBA

In Excel kann fast alles automatisiert werden. Dazu schreiben Sie Anweisungen, die Excel ausführt. Die Automatisierung einer Aufgabe unter Verwendung von VBA bringt mehrere Vorteile mit sich:

- ✓ Excel führt die Aufgabe immer auf dieselbe Weise aus. (Meistens ist diese Konsistenz von Vorteil.)
- ✓ Excel führt die Aufgabe sehr viel schneller aus, als Sie sie manuell erledigen können (es sei denn natürlich, Sie sind Superman).
- ✓ Wenn Sie ein guter Makro-Programmierer sind, führt Excel die Aufgabe immer fehlerfrei aus (was Sie und ich nicht immer von uns behaupten können, unabhängig davon, wie sorgfältig wir arbeiten).
- ✓ Wenn Sie alles korrekt eingerichtet haben, kann auch eine Person, die nicht gut mit Excel umgehen kann, die Aufgabe ausführen.

- ✓ Sie können Dinge in Excel erledigen, die andernfalls unmöglich sind, was Sie im Büro sehr beliebt machen wird.
- ✓ Für lange, zeitaufwendige Arbeiten müssen Sie nicht mehr vor Ihrem Computer sitzen und sich langweilen. Excel erledigt für Sie die Arbeit, während Sie Kaffeetrinken gehen können.

Nachteile von VBA

Es ist nur gerecht, jetzt auch die Nachteile von VBA zu erwähnen (jedenfalls *potenzielle* Nachteile):

- ✓ Sie müssen wissen, wie man in VBA programmiert (aber dafür haben Sie ja jetzt dieses Buch). Glücklicherweise ist das nicht so schwierig, wie Sie vielleicht annehmen.
- ✓ Andere Anwender, die Ihre VBA-Programme benutzen wollen, brauchen ebenfalls Excel. Es wäre praktisch, wenn man eine Schaltfläche drücken könnte, die Ihre Excel-VBA-Anwendung zu einem eigenständigen Programm macht, aber das ist nicht möglich (und wird es wahrscheinlich nie sein).
- ✓ Manchmal läuft etwas schief. Mit anderen Worten, Sie können nicht blind darauf vertrauen, dass Ihr VBA-Programm immer und unter allen Umständen korrekt funktioniert. Willkommen in der Welt des Debuggings und (falls andere Ihre Makros verwenden) des technischen Supports.
- ✓ VBA ist ein bewegliches Ziel. Wie Sie wissen, gibt Microsoft ständig Upgrades für Excel heraus. Und obwohl man sich größte Mühe gibt, Kompatibilität zwischen den Versionen zu gewährleisten, werden Sie immer wieder feststellen, dass der von Ihnen für ältere Versionen geschriebene VBA-Code bei neueren Excel-Versionen nicht mehr fehlerfrei funktioniert.

VBA in Kürze

Hier folgt ein kurzer und grober Überblick über VBA, nur damit Sie wissen, was Sie erwartet.

- ✓ **Sie führen Aktionen in VBA aus, indem Sie Code in einem VBA-Modul schreiben (oder aufzeichnen).** VBA-Module werden im Visual Basic Editor (VBE) angezeigt und bearbeitet.
- ✓ **Ein VBA-Modul besteht aus Sub-Prozeduren.** Eine Sub-Prozedur hat überhaupt nichts mit U-Booten oder belegten Broten zu tun. Stattdessen handelt es sich dabei um Computercode, der irgendwelche Aktionen für irgendwelche Objekte ausführt (wie Sie gleich genauer erfahren werden). Das folgende Beispiel zeigt eine einfache Sub-Prozedur mit dem Namen `ADDEMUP`. Dieses bemerkenswerte Programm zeigt das Ergebnis von 1 plus 1 an.

```
Sub AddEmUp()
    Sum = 1 + 1
    MsgBox "Die Lösung ist " & Sum
End Sub
```

Eine Sub-Prozedur, die nicht fehlerfrei arbeitet, wird als suboptimal bezeichnet.

- ✓ **Ein VBA-Modul kann auch Funktionsprozeduren enthalten.** Eine Funktionsprozedur gibt einen einzigen Wert zurück. Sie können sie von einer anderen VBA-Prozedur aus aufrufen oder sie sogar als Funktion in einer Formel auf einem Arbeitsblatt verwenden. Nachfolgend finden Sie ein Beispiel für eine Funktionsprozedur: `ADDTWO`. Diese Funktion nimmt zwei Zahlen entgegen und gibt die Summe ihrer Werte aus:

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

Eine Funktionsprozedur, die nicht ordnungsgemäß funktioniert, wird als dysfunktional bezeichnet.

- ✓ **VBA manipuliert Objekte.** Excel enthält Dutzende von Objekten, die Sie manipulieren können. Beispiele für Objekte sind unter anderem Arbeitsmappen, Tabellen, Zellenbereiche, Diagramme oder Formen. Möglicherweise stehen Ihnen aber noch sehr viel mehr Objekte zur Verfügung, die Sie mit Ihrem VBA-Code manipulieren können.
- ✓ **Objekte sind hierarchisch angeordnet.** Objekte können als Container für andere Objekte dienen. Ganz oben in der Objekthierarchie steht Excel. Excel selbst ist ein Objekt namens `APPLICATION`. Das Objekt `APPLICATION` enthält weitere Objekte, wie beispielsweise `WORKBOOK`-Objekte und `ADD-IN`-Objekte. Das `WORKBOOK`-Objekt kann weitere Objekte enthalten, beispielsweise `WORKSHEET`-Objekte und `CHART`-Objekte. Ein `WORKSHEET`-Objekt kann beispielsweise `RANGE`-Objekte und `PIVOTTABLE`-Objekte enthalten. Der Begriff *Objektmodell* bezieht sich auf die Anordnung dieser Objekte. (Weitere Informationen über Objektmodelle finden Sie in Kapitel 4.)
- ✓ **Objekte desselben Typs bilden eine Collection.** Die `WORKSHEETS`-Collection beispielsweise besteht aus allen Arbeitsblättern einer bestimmten Arbeitsmappe. Die `CHARTS`-Collection besteht aus allen `CHART`-Objekten in einer Arbeitsmappe. Collections sind ebenfalls Objekte.
- ✓ **Der Verweis auf ein Objekt erfolgt durch die Angabe seiner Position innerhalb der Objekthierarchie und unter Verwendung eines Punkts als Trennzeichen.** Beispielsweise verweisen Sie auf die Arbeitsmappe `ARBEITSMAPPE1.XLSX` wie folgt:

```
Application.Workbooks("Arbeitsmappe1.xlsx")
```

Damit verweisen Sie auf die Arbeitsmappe `ARBEITSMAPPE1.XLSX` in der `WORKBOOKS`-Collection. Die `WORKBOOKS`-Collection ist im `APPLICATION`-Objekt enthalten (das heißt Excel). Durch Erweiterung auf die nächste Ebene können Sie auf Arbeitsblatt 1 in `ARBEITSMAPPE1.XLSX` verweisen:

```
Application.Workbooks("Arbeitsmappe1.xlsx").Worksheets("Tabelle1")
```

Sie können dies auf eine weitere Ebene ausdehnen und auf eine bestimmte Zelle verweisen (in diesem Fall ist das Zelle A1):

```
Application.Workbooks("Arbeitsmappe1.xlsx").Worksheets("Tabelle1").
Range("A1")
```

- ✓ **Falls Sie keine speziellen Verweise (auch als Referenzen bezeichnet) angeben, verwendet Excel die aktiven Objekte.** Wenn BOOK1.XLSX die aktive Arbeitsmappe ist, können Sie den oben gezeigten Verweis wie folgt vereinfachen:

```
Worksheets("Tabelle1").Range("A1")
```

Wenn Sie wissen, dass Arbeitsblatt 1 das aktive Arbeitsblatt ist, können Sie den Verweis noch weiter vereinfachen:

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

- ✓ **Objekte haben Eigenschaften.** Sie können sich eine Eigenschaft als *Einstellung* für ein Objekt vorstellen. Beispielsweise hat ein RANGE-Objekt Eigenschaften wie VALUE (Wert) und ADDRESS (Adresse). Ein CHART-Objekt hat Eigenschaften wie HASTITLE (Hat einen Titel) und TYPE (Typ). Mithilfe von VBA können Sie Objekteigenschaften ermitteln und diese auch ändern.
- ✓ **Der Verweis auf eine Eigenschaft eines Objekts erfolgt durch Kombination des Objekt-namens mit dem Eigenschaftsnamen, durch einen Punkt voneinander getrennt.** Beispielsweise verweisen Sie wie folgt auf die VALUE-Eigenschaft in Zelle A1 auf Blatt 1:

```
Worksheets("Tabelle1").Range("A1").Value
```

- ✓ **Sie können Variablen Werte zuweisen.** Eine Variable ist ein benanntes Element, das Informationen speichert. Sie können Variablen in Ihrem VBA-Code verwenden, um beispielsweise Werte, Text oder Eigenschaftseinstellungen zu speichern. Um der Variablen *Zins* den Wert aus Zelle A1 auf Arbeitsblatt 1 zuzuweisen, verwenden Sie die folgende VBA-Anweisung:

```
Zins = Worksheets("Tabelle1").Range("A1").Value
```

- ✓ **Objekte haben Methoden.** Eine Methode ist eine Aktion, die Excel für ein Objekt ausführt. Ein Beispiel für eine Methode für ein RANGE-Objekt ist CLEARCONTENTS. Wie der Name schon sagt, löscht die Methode den Inhalt des Bereichs.
- ✓ Die Angabe einer Methode erfolgt durch Kombination des Objekts mit der Methode, durch einen Punkt voneinander getrennt. Beispielsweise löscht die folgende Anweisung den Inhalt von Zelle A1:

```
Worksheets("Tabelle1").Range("A1").ClearContents
```

- ✓ **VBA enthält alle Konstrukte einer modernen Programmiersprache, einschließlich Variablen, Arrays und Schleifen.** Mit anderen Worten, wenn Sie sich ein wenig Zeit nehmen, um das Ganze zu sortieren, können Sie Code schreiben, der ungläubliche Dinge erledigt.

Ob Sie es glauben oder nicht, die obige Liste beschreibt VBA in Kurzform komplett. Jetzt sehen wir uns noch die Details an. Deshalb hat dieses Buch mehrere Seiten.

Excel-Kompatibilität



Dieses Buch wurde für die Desktop-Versionen von Excel 2016 und Excel 2019 entwickelt. Wenn Sie andere Versionen einsetzen, kann es sein, dass die hier gezeigten Beschreibungen an manchen Stellen nicht mit Ihrer Benutzeroberfläche übereinstimmen.

Wenn Sie vorhaben, Ihre Excel-VBA-Dateien an andere Benutzer weiterzugeben, müssen Sie unbedingt wissen, welche Version diese benutzen. Anwender mit einer älteren Version können neue Funktionen aus neueren Versionen nicht nutzen. Wenn Sie beispielsweise VBA-Code schreiben, der auf Zelle XFD1048567 verweist (die letzte Zelle in einer Arbeitsmappe), erhalten alle Anwender mit Excel-Versionen vor 2007 eine Fehlermeldung, weil dort die Arbeitsblätter nur 65.536 Zeilen und 255 Spalten unterstützen (die letzte Zelle ist damit IV65536).

In Excel 2010 und später gibt es außerdem einige neue Objekte, Methoden und Eigenschaften. Wenn Sie diese in Ihrem Code verwenden, erhalten Benutzer mit einer älteren Excel-Version eine Fehlermeldung, wenn sie Ihr Makro ausführen, und beschweren sich möglicherweise bei Ihnen.