Excel-VBA

Mit über 1000 Makros für Excel 97 bis 2007

BERND HELD

Markt+Technik

KOMPENDIUM

Einführung Arbeitsbuch Nachschlagewerk

- Über 1000 Makros
- Spezial-Add-In des Autors
- Makrosuchsystem

3 Datentypen, Variablen und Konstanten

Das folgende Kapitel bildet die Voraussetzung für eine strukturierte Programmierung in Excel. Mit Variablen speichern Sie Informationen dauerhaft während der Laufzeit eines Makros, d. h., Sie können somit Variablen auch mehrmals im Makro benutzen, indem Sie Variablen füllen und Werte hochzählen oder subtrahieren. Mit Konstanten legen Sie Informationen fest, die sich selten oder sogar nie ändern. Excel bietet für die Deklaration von Variablen und Konstanten eine ganze Auswahl an Datentypen. Je nach Aufgabe setzten Sie dazu die vorgesehenen Datentypen ein.

Neben den Variablen, die Sie mit Datentypen deklarieren, gibt es auch noch sogenannte Objektvariablen, die einen Verweis auf ein bestimmtes Objekt enthalten.

Sie finden alle Beispiele auf der mitgelieferten CD-ROM in der Datei VARIABLEN.XLS.

3.1 Der Einsatz von Variablen

Sicher werden Sie sich fragen, warum Sie Variablen in der Programmierung brauchen. Variablen werden u. a. dazu benötigt, um Daten zwischenzuspeichern. Wenn Sie beispielsweise das erste Makro ansehen, in dem Sie eine Zelle von einem Tabellenblatt auf ein anderes kopiert haben, werden Sie merken, dass Sie für diese Aufgabe die Zwischenablage als »Variable« missbraucht haben. Wenn es sich nur um eine einzige zu übertragende Information handelt, mag diese Vorgehensweise noch in Ordnung sein. Stellen Sie sich aber einmal vor, Sie müssten mehrere Informationen von einem Blatt auf das andere übertragen. Da kommen Sie um den Einsatz von Variablen nicht herum. Außerdem bietet die Zwischenablage nicht die Beständigkeit wie eine Variable. So ist die Zwischenablage leer, sobald Sie den Inhalt der Zwischenablage einmal z. B. in eine Zelle eingefügt haben. Mit Variablen können Sie dauerhaft arbeiten, d. h., Sie können jederzeit darauf zugreifen, diese abfragen oder verändern und zum Schluss ausgeben.

3.1.1 Regeln für die Syntax von Variablen

Wenn Sie Variablen einsetzen, müssen Sie sich dabei an bestimmte Konventionen für deren Benennung halten:

Regeln für Variablen

- Das erste Zeichen muss aus einem Buchstaben bestehen. Als folgende Zeichen können Sie Buchstaben, Zahlen und einige Sonderzeichen verwenden.
- Sie dürfen keine Leerzeichen in einem Variablennamen verwenden. Wenn Sie einzelne Worte trennen möchten, verwenden Sie dazu das Unterstrichzeichen z. B. `Dim Miete_Januar as Currency`.
- Sonderzeichen wie #, %, &, ! oder ? sind nicht erlaubt.

Wenn Sie Ihre Variablennamen übersichtlich und auch sprechend definieren möchten, empfiehlt sich folgende Schreibweise:

```
Dim TextMeldungFürFehler as String
```

Hier geht aus dem Namen der Variablen klar hervor, wofür diese eingesetzt werden soll. Als zweiter Punkt ist die Variable durch die Schreibweise leicht lesbar.

3.1.2 Variablen am Beginn vom Makro deklarieren

*Variablen werden immer zu Beginn eines Makros deklariert, also nach der Sub-Zeile. Dabei spricht man von **lokalen** Variablen. Diese Variablen können nur in dem Makro verwendet werden, in dem sie deklariert wurden. Nachdem ein Makro durchgelaufen ist, wird diese Variable wieder aus dem Speicher gelöscht.*

Von **globalen** Variablen spricht man, wenn Sie diese allgemeingültig, also in mehreren Makros verwenden möchten. Dann muss die Variablendeklaration vor der `Sub`-Zeile stattfinden.

Globale Variablen können gleich für mehrere Makros verwendet werden. Diese werden nach dem Ende eines Makros auch nicht gelöscht und behalten ihren aktuellen Wert bei. Es gibt Beispiele, bei denen diese Vorgehensweise auch sinnvoll ist. In den meisten Fällen sollten globale Variablen aber weitgehend vermieden werden, da sie wertvollen Speicherplatz auf dem Stapelspeicher belegen, was sich negativ auf das Laufverhalten von Makros auswirken kann.

Eine Variablendeklaration beginnt immer mit der Anweisung `Dim`, gefolgt von einem Variablennamen, den Sie frei wählen können. Danach geben Sie mit dem Schlüsselwort `As` an, welchen Datentyp die Variable erhalten soll. Eine Tabelle mit den gängigsten Datentypen sehen Sie weiter unten im Kapitel.

3.1.3 Variablendeklarationen erzwingen

Sie können Excel so einstellen, dass jede verwendete Variable deklariert sein muss. Vorher läuft dann kein einziges Makro an, sofern es mit Variablen arbeitet, die zuvor nicht deklariert wurden. Um diese wichtige Einstellung vorzunehmen, wechseln Sie in die Entwicklungsumgebung und wählen den Befehl EXTRAS/OPTIONEN. Aktivieren Sie das Blattregister EDITOR, und aktivieren Sie das Kontrollkästchen VARIABLENDEKLARATION ERFORDERLICH.

Was bewirkt diese Einstellung aber genau? Immer wenn Sie ein neues Modul einfügen, wird die Anweisung `Option Explicit` *in die erste Zeile Ihres Modulblattes automatisch eingefügt. Diese Anweisung können Sie selbstverständlich auch von Hand erfassen. Die Anweisung bedeutet nichts anderes, als dass verwendete Variablen im Code vorher deklariert werden müssen. Vorher läuft gar nichts!*

3.1.4 Die wichtigsten Variablentypen

Die beiden wichtigsten Variablentypen sind zum einen die Variable vom Typ `String` und zum anderen die `Integer`-Variable. In einer Variablen von Typ `String` können Sie Texte zwischenspeichern, manipulieren und ausgeben. In einer Variablen vom Typ `Integer` führen Sie mathematische Berechnungen aus. Integer-Variablen werden oft als Zähler in Schleifen verwendet, die Sie im weiteren Verlauf des Buches noch kennenlernen werden.

Entnehmen Sie der Tabelle 3.1 die gängigsten Variablentypen und deren Speicherbedarf:

Variablentyp	Wertebereich/Speicherbedarf
`Byte`	ganze Zahlen zwischen 0 und 255 (1 Byte)
`Boolean`	Wahrheitswert, entweder `True` oder `False` (2 Byte)
`Currency`	Währungs-Datentyp: Festkommazahlen mit 15 Stellen vor und vier Stellen nach dem Komma mit einem Wertebereich von –922.337.203.685.477,5808 bis 922.337.203.685.477,5807 (8 Byte)
`Date`	Datums- und Zeit-Datentyp (8 Byte)
`Double`	Fließkommazahlen mit einer Genauigkeit von 16 Stellen hinter dem Komma (8 Byte)
`Integer`	ganze Zahlen zwischen –32768 und +32767 (2 Byte)
`Long`	ganze Zahlen im Wertebereich von –2.147.483.648 und +2.147.483.647 (4 Byte)
`Object`	Datentyp gibt einen Verweis auf ein Objekt wieder (4 Byte)
`Single`	Fließkommazahlen mit einer Genauigkeit von acht Stellen hinter dem Komma (4 Byte)

Tabelle 3.1: Die Datentypen für die Programmierung

Tabelle 3.1: Die Datentypen für die Programmierung (Forts.)

Variablentyp	Wertebereich/Speicherbedarf
`String`	der Datentyp für alle Texte (10 Byte)
`Variant`	Standarddatentyp, wird automatisch gewählt, wenn kein anderer Datentyp definiert ist (16 Byte)

Im weiteren Verlauf dieses Buches werden Sie die verschiedenen Datentypen im Praxiseinsatz sehen.

Einen weiteren Vorteil, Variablen zu deklarieren, möchte ich Ihnen nicht vorenthalten. Wenn Sie vergessen, Variablen zu deklarieren, und auch nicht die Anweisung `Option explicit` *gesetzt haben, gehen Sie sehr verschwenderisch mit Ihrem Speicher um. Wird für eine Variable kein Datentyp angegeben, wird automatisch der Datentyp* `Variant` *verwendet. Wegen seines hohen Speicherbedarfs von 16 Byte ist er aber nicht zu empfehlen.*

3.1.5 Noch kürzere Deklaration von Variablen

Neben den herkömmlichen Deklarationen für Variablen gibt es auch noch eine etwas verkürzte Form der Variablendeklaration. Sehen Sie sich dazu einmal folgende Tabelle an:

Tabelle 3.2: Datentyp-Deklaration noch kürzer

Ausführlich	Kurzform
`Dim Zähler as Integer`	`Dim Zähler%`
`Dim Zähler Groß as Long`	`Dim ZählerGroß&`
`Dim Betrag as Currency`	`Dim Betrag@`
`Dim Meldung as String`	`Dim Meldung$`

Für jeden oben aufgeführten Datentyp gibt es ein Kurzzeichen, das Sie einsetzen können, um den Programmiercode zu kürzen. Sie sollten aber zumindest am Anfang bei den sprechenden Variablen bleiben.

Wenn Sie möchten, können Sie bei der Benennung von Variablen auch jeweils den Namen der Variablen mit einem Kürzel beginnen lassen, das schon Auskunft über den Datentyp der Variable gibt. So symbolisiert die Variable `str_Meldung` *eindeutig eine Variable vom Datentyp* `String`*, in der Sie Texte zwischenspeichern können. Die Variable* `int_AnzahlGefüllterZellen` *stellt eine Variable vom Typ* `Integer` *dar, in der Sie Zahlenwerte verwalten können.*

3.1.6 Die unterschiedlichen Variablentypen

In Excel haben Sie neben den lokalen und globalen Variablen weitere Möglichkeiten, um Variablen zu deklarieren.

Statische Variablen

Standardmäßig wird der Variableninhalt nach jedem Makroende gelöscht. Sie haben jedoch auch die Möglichkeit, Variablen so definieren, dass deren »Haltbarkeit« nach jedem Makroende erhalten bleibt.

Listing 3.1: Beispiel für statische Variable

```
Sub VariablenInhaltBleibtBestehen()
Static lngAufrufe As Long

lngAufrufe = lngAufrufe + 1
MsgBox "Makro wurde " & lngAufrufe & " Mal ausgeführt!"
End Sub
```

Wenn Sie das letzte Makro mehrmals hintereinander aufrufen, dann werden Sie sehen, dass der Inhalt der Variablen Aufrufe auch nach jedem Makrodurchlauf erhalten bleibt.

Öffentliche Variablen

In der Entwicklungsumgebung von Excel können Sie mehrere Module anlegen. Um Variablen modulübergreifend abfragen oder ändern zu können, müssen Sie eine solche Variable als öffentlich deklarieren. Diese Variablen werden mit der Anweisung `Public` deklariert. Damit haben Sie die Möglichkeit, auf Variablen zuzugreifen, die in anderen Modulen untergebracht sind.

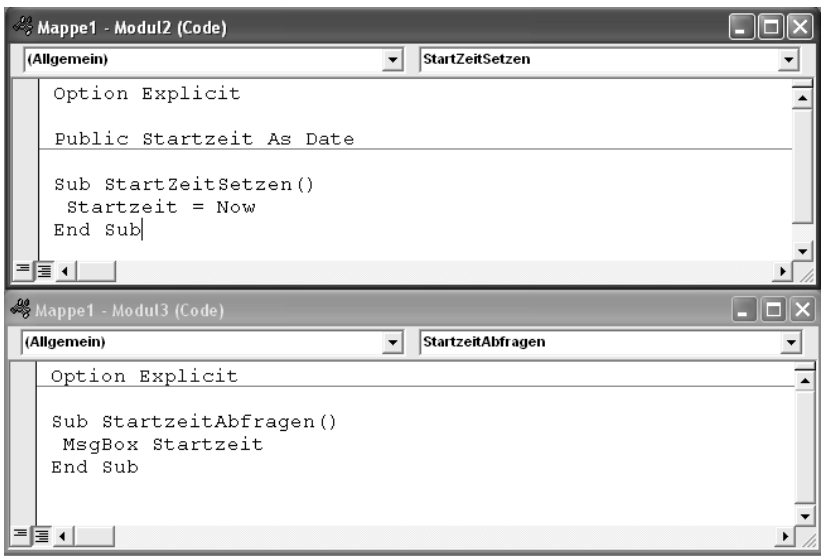

Abbildung 3.1: Der Einsatz einer öffentlichen Variablen

Im folgenden Beispiel soll in einem Modul (MODUL2) die aktuelle Uhrzeit in einer öffentlichen Variablen gespeichert werden. Von einem anderen Modul (MODUL3) aus soll diese Startzeit abgefragt werden. Fügen Sie für diese Aufgabe zwei neue Modulblätter (MODUL2 und MODUL3) ein und erfassen folgende Makros:

Deklarieren Sie in MODUL2 die öffentliche Variable. Danach füllen Sie diese Variable, indem Sie über die Funktionen `Date` und `Time` das aktuelle Tagesdatum sowie die momentane Uhrzeit speichern.

In MODUL3 greifen Sie auf die öffentliche Variable `StartZeit` zu und geben sie in einem Meldungsfenster am Bildschirm aus.

Private Variablen

Möchten Sie die Gültigkeit einer Variablen für alle Makros auf ein bestimmtes Modul beschränken, z. B. MODUL4, dann müssen Sie die Variable mit der Anweisung `Private` deklarieren.

Im folgenden Beispiel soll eine Passwortverwaltung in einem Modul realisiert werden. Dabei soll jeweils die aktive Tabelle über ein Kennwort geschützt werden. Für diesen Zweck speichern Sie das Kennwort in einer privaten Variablen. Es wird dadurch nicht möglich, das Kennwort über Makros aus anderen Modulblättern auszulesen.

Listing 3.2: Beispiel für private Variable

```
Option Explicit

Private strPassw As String

Sub TabelleSchützen()
strPassw = "Test"
ActiveSheet.Protect Password:=strPassw
End Sub

Sub TabelleEntsperren()
 ActiveSheet.Unprotect Password:=strPassw
End Sub
```

Deklarieren Sie zu Beginn im Modulblatt MODUL4 eine private Variable mit dem Namen `StrPassw`. Im Makro `TabelleSchützen` geben Sie das Kennwort bekannt und wenden die Methode `Protect` an, um die aktive Tabelle zu schützen. Das Makro `TabelleEntsperren` hebt den Schutz der aktiven Tabelle auf, indem es die Methode `Unprotect` einsetzt und dabei das Kennwort übergibt. Von einem anderen Modulblatt haben Sie in diesem Beispiel keine Chance, auf die Variable `Passw` zuzugreifen.

3.1.7 Die Objektvariablen

Neben den Variablen, die Sie mit Datentypen deklarieren, gibt es auch noch sogenannte Objektvariablen, die einen Verweis auf ein bestimmtes Objekt enthalten.

Welche Objekte unter anderem zur Verfügung stehen, können Sie ermitteln, wenn Sie in der Entwicklungsumgebung den Objektkatalog aufrufen. Dies geht am schnellsten, indem Sie die Taste F2 drücken.

Alternativ dazu können Sie einfach mit der Deklaration einer Objektvariablen beginnen, indem Sie beispielsweise die Anweisung `Dim Test As` eingeben und dann die Leertaste drücken. Augenblicklich wird Ihnen ein Kontextmenü angeboten, das die zur Verfügung stehenden Objekte anbietet.

Der Einsatz von Objektvariablen wird anhand ein paar typischer Beispiele aufgezeigt.

Die Objektvariable Workbook

Über die Objektvariable `Workbook` können eine bzw. mehrere Arbeitsmappen angesprochen werden. Im folgenden Listing werden die Namen aller momentan geöffneten Arbeitsmappen am Bildschirm angezeigt:

Listing 3.3: Alle geöffneten Arbeitsmappen ermitteln

```
Sub Arbeitsmappe()
Dim Mappe As Workbook

For Each Mappe In Application.Workbooks
 MsgBox Mappe.FullName
Next Mappe
End Sub
```

In einer `For-Each`-Schleife werden alle momentan geöffneten Arbeitsmappen durchlaufen. Diese Mappen sind im Auflistungsobjekt `Workbooks` automatisch verzeichnet. Über die Eigenschaft `FullName` können Sie neben dem eigentlichen Namen der jeweiligen Mappe noch ermitteln, wo diese gespeichert ist.

Die Objektvariable Worksheet

Mithilfe der Objektvariablen `Worksheet` können Sie ein oder auch mehrere Tabellenblätter ansprechen. Im folgenden Listing werden die Namen der aktiven Arbeitsmappe am Bildschirm ausgegeben:

Listing 3.4: Die Tabellennamen der aktiven Arbeitsmappe ermitteln

```
Sub TabellenNamenErmitteln()
Dim Blatt As Worksheet

For Each Blatt In ActiveWorkbook.Worksheets
```

```
 MsgBox Blatt.Name
Next Blatt
End Sub
```

In einer `For-Each`-Schleife werden alle Tabellen der aktiven Arbeitsmappe abgearbeitet. Diese Tabellen sind im Auflistungsobjekt `Worksheets` automatisch verzeichnet. Über die Eigenschaft `Name` ermitteln Sie den Namen der jeweiligen Tabelle.

Die Objektvariable Range

Über die Objektvariable `Range` können Sie einzelne oder auch mehrere Zellen ansprechen. Über das folgende Makro werden alle Zellen innerhalb einer Markierung im Direktbereich ausgelesen.

Listing 3.5: Alle markierten Zellen werden ermittelt.

```
Sub MarkierteZellenErmitteln()
Dim Zelle As Range

For Each Zelle In Selection
 Debug.Print Zelle.Address
Next Zelle
End Sub
```

In einer `For-Each`-Schleife werden alle markierten Zellen der aktiven Tabelle abgearbeitet. Diese Zellen sind im Auflistungsobjekt `Selection` automatisch verzeichnet. Über die Eigenschaft `Address` ermitteln Sie die jeweiligen Koordinaten der markierten Zellen.

3.2 Die Verwendung von Konstanten

Im Gegensatz zu den Variablen ändern die Konstanten ihre Werte nie und bleiben während der Programmausführung immer konstant. Auch hier wird zwischen lokalen und globalen Konstanten unterschieden. Globale Konstanten werden außerhalb der einzelnen Makros definiert und sind damit für alle Makros im Modul verwendbar. Lokale Konstanten hingegen gelten nur in dem Makro, in dem sie definiert wurden. Wie schon bei den Variablen sollten Sie darauf achten, nicht allzu viele globale Konstanten zu verwenden, da sich dies merkbar auf Ihren Speicher auswirkt.

Anbei folgen ein paar typische Deklarationen mit Konstanten:

```
Const Arbeitsmappe = "Mappe1.xls"
Const StartDatum = #1/1/2007#
Const Fehlermeldung1 =  _
 "Fehler beim Drucken aufgetreten!"
Const MWST = 1.19
```

Was kann hier noch verbessert werden? Was für die Variablen gilt, hat auch bei den Konstanten Konsequenzen. In den obigen Beispielen ist noch nicht erklärt worden, welche Datentypen verwendet werden sollen. Momentan wird in allen vier Beispielen der Datentyp Variant eingesetzt. Es geht auch etwas genauer und speichersparender:

```
Const Arbeitsmappe as String = "Mappe1.xls"
Const StartDatum As Date = #1/1/2007#
Const Fehlermeldung1 as String = _
 "Fehler beim Drucken!"
Const MWST as Single = 1.19
```