Einführung in VisualBasic for Applications

Stefan Mahlitz

Inhaltsverzeichnis

1. Grundlagen der VisualBasic for Applications (VBA) Programmierung

Jedes Microsoft Office-Programm (sowie einige andere Programme) stellt eine Objektbibliothek zur Verfügung, die die Funktionen des Programms anbietet. Auf diese Bibliothek greift VBA zu. Programm-Code wird in Modulen gespeichert.

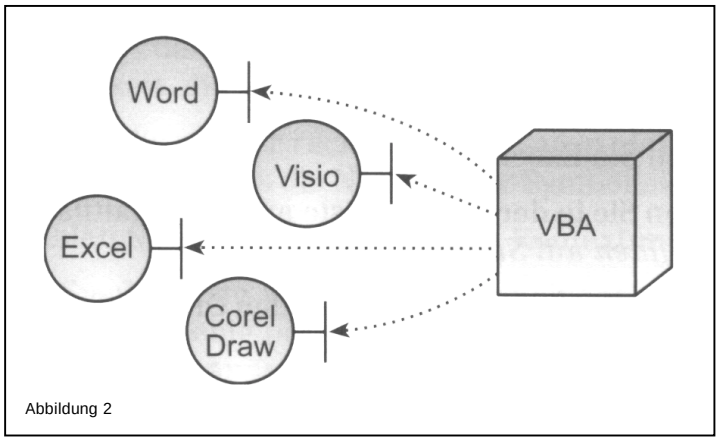

Abbildung 2

1.1 Variablen, Konstanten und Typen

Variablen und Konstanten stellen reservierte Speicherbereiche dar. In ihnen werden Zahlen, Zeichenketten oder ganze Typen gespeichert.

In VBA können Variablen innerhalb von Prozeduren und Funktionen mit der Anweisung `Dim Variablenname As Variablentyp` deklariert werden.

Die Anweisung

```
Dim iSpalte As Integer
```

reserviert einen Speicherplatz von 2 Bytes für die Variable iSpalte.

Der Variablen `iSpalte` kann man jetzt Werte zuweisen.

```
iSpalte = 12345
```

Globale Variablen werden im Kopf des Moduls erzeugt.

```
Public g_iSpalte As Integer
```

Die Variable `g_iSpalte` steht jetzt jeder Prozedur und Funktion zur Verfügung.

Im Modulkopf können auch benutzerdefinierte Datentypen erstellt werden. Dazu wird das Schlüsselwort `Type` verwendet.

```
Type TTesttyp
  Variablenname1 As Integer
  Variablenname2 As String
End Type
```

Variablen können nun diesem Datentyp zugewiesen werden. Sie verhalten sich wie Objekte, `Variablenname1` und `Variablenname2` stellen sozusagen die Eigenschaften des Typs dar.

Wichtige Schlüsselwörter bei der Variablendeklaration sind `Public`, `Private` und `Static`. Sie legen die Verfügbarkeit der Variablen im Programm fest. Mit `Public` deklarierte Variablen sind in jedem Modul des Projektes verfügbar, Als `Private` deklarierte Variablen nur in dem Modul, in dem sie erstellt wurden. Das Schlüsselwort `Static` führt dazu, daß in einer Prozedur oder Funktion deklarierte Variable ihre Werte auch nach Beenden des Unterprogramms beibehält.

1.2 Prozeduren und Funktionen

Prozeduren und Funktionen sind Programme, an die Parameter übergeben werden können. Sie werden verwendet, wenn gleiche Aufgaben an verschiedenen Stellen im Quelltext wiederholt werden müssen, zum Beispiel das Löschen des Bildschirms.

Prozeduren definiert man in VBA mit der `Sub`-Anweisung gefolgt vom Prozedurnamen und der Parameterliste.

Diese Prozedur setzt die Werte der Zellen in Zeile Var1 in den Spalten 2 und 3 auf 0

```
Sub Testen (Var1 As Integer)
  Cells(Var1, 2).Value = 0
  Cells(Var1, 3).Value = 0
End Sub
```

Funktionen können, im Unterschied zu Prozeduren, Werte zurückgeben. Sie werden notwendig, wenn komplexe Algorithmen zur Ermittlung eines Wertes verwendet werden.

Funktionen werden mit der `Function`-Anweisung gefolgt vom Funktionsnamen, der Parameterliste und dem Rückgabedatentyp deklariert.

Diese Funktion berechnet das Produkt aus der Wurzel von Var1 und dem Sinus von Var1 und addiert den Cosinus von Var1 dazu

```
Function Testen2(Var1 As Single) As Single
  Testen = Sqrt(Var1) * Sin(Var1) + Cos(Var1)
End Function
```

Aufgerufen werden Prozeduren und Funktionen mit ihrem Namen und der Parameterliste.

```
Sub Hauptprogramm ()
  Testen (5)
  xyz = testen2 (7)
End Sub
```

Das Hauptprogramm setzt die Werte der Zellen (5, 2) und (5, 3) auf 0 und speichert das Ergebnis von Wurzel (7) * Sin(7) + Cos(7) in die Variable xyz.

Parameter können auf verschiedene Weise übergeben werden. `ByRef` vor dem Variablennamen bewirkt, daß der übergebene Parameter referenziert wird, die Prozedur greift direkt auf

die übergebene Variable zu und kann sie nach außen hin ändern. `ByVal` führt dazu, daß eine Kopie der Variable erstellt wird, Änderungen wirken sich nicht außerhalb des Unterprogramms aus. Wird das Schlüsselwort `Optional` verwendet, müssen die nachfolgend deklarierten Parameter nicht übergeben werden, sie werden dann mit Standardwerten initialisiert.

1.3 Schleifen

Schleifen erlauben die wiederholte Ausführung von Anweisungen. Die verschiedenen von VisualBasic for Applications zur Verfügung gestellten Schleifen unterscheiden sich in Bezug auf die Abbruchbedingung und die minimale Anzahl der Durchläufe.

1.3.1 For ...To ... Next

Eine For-To-Next-Schleife wird solange durchlaufen, bis die Zählvariable den Endwert überschreitet. Bei herunterzählenden Schleifen enden die Durchläufe bei Unterschreiten des Endwertes. Sie wird, wenn die Zählvariable innerhalb der Schleife nicht geändert wird, (Ende – Anfang) / Schritt mal durchlaufen.

```
For Zähler = Anfang To Ende [Step Schritt]
      [Anweisungen]
Next [Zähler]
```

Ein vorzeitiger Abbruch wird mit der Anweisung

```
Exit For
```

erreicht.

1.3.2 For Each ... Next

Eine For-Each-Next-Schleife wird solange durchlaufen, bis alle Elemente der Gruppe abgearbeitet wurden.

```
For Each Element In Gruppe
      [Anweisungen]
Next [Element]
```

Ein vorzeitiger Abbruch wird mit der Anweisung

```
Exit For
```

erreicht.

1.3.3 Do While/Until ... Loop

Eine Do-While-Loop Schleife wird solange ausgeführt, wie die am Schleifenanfang ausgewertete Bedingung wahr ist, also mindestens Null-mal, wenn die Auswertung der Bedingung un-

wahr ergibt. Eine Do-Until-Loop Schleife wird solange ausgeführt, bis die am Schleifenanfang ausgewertete Bedingung wahr ist.

```
Do [{While | Until} Bedingung]
      [Anweisungen]
Loop
```

Ein vorzeitiger Abbruch wird mit der Anweisung

```
Exit Do
```

erreicht.

1.3.4 Do Loop ... While/Until

Eine Do-While-Loop Schleife wird solange ausgeführt, wie die am Schleifenende ausgewertete Bedingung wahr ist, aber mindestens einmal, wenn die Auswertung der Bedingung unwahr ergibt. Eine Do-Until-Loop Schleife wird solange ausgeführt, bis die am Schleifenende ausgewertete Bedingung wahr ist.

```
Do
      [Anweisungen]
Loop [{While | Until} Bedingung]
```

Ein vorzeitiger Abbruch wird mit der Anweisung

```
Exit Do
```

erreicht.

1.4 Verzweigungen

Mit Verzweigungen kann man die Ausführung unterschiedlicher Anweisungen abhängig von verschiedenen Faktoren machen.

1.4.1 If ... Then

Soll zum Beispiel Anweisung a ausgeführt werden, wenn die Variable b = 5, wird das so implementiert:

```
If b = 5 Then Anweisung a
```

Soll für b < > 5 zusätzlich die Anweisung b ausgeführt werden:

```
If b = 5 Then
  Anweisung a
Else
  Anweisung b
End If
```

Die allgemeine Syntax lautet:

```
If Bedingung Then
      [Anweisungen]
      [ElseIf Bedingung-n Then
            [elseifAnweisungen] ...
[Else
      [elseAnweisungen]]
End If
```

1.4.2 Select Case

Mit Select-Case-Anweisungen kann eine Variable mit geringem Aufwand auf mehrere Zustände überprüft weden.

```
Select Case Testausdruck
      [Case Ausdrucksliste-n
            [Anweisungen-n]] ...
      [Case Else
            [elseAnw]]
End Select
```

Soll zum Beispiel Anweisung a ausgeführt werden, wenn Variable z = 1 und Anweisung b ausgeführt werden, wenn z = 2 und Anweisung c ausgeführt werden, wenn z = 3 und ansonsten Anweisung d, kann man folgenden Quelltext verwenden:

```
Select Case z
  Case 1
    Anweisung a
  Case 2
    Anweisung b
  Case 3
    Anweisung c
  Case Else
    Anweisung d
End Select
```

Den gleichen Effekt erreicht man auch mit verschachtelten If-Then aufrufen:

```
If z = 1 Then
  Anweisung a
Else
  If z = 2 Then
    Anweisung b
  Else
    If z = 3 Then
      Anweisung c
    Else
      Anweisung d
    End If
  End If
End if
```

Die Verschachtelung verringert allerdings die Lesbarkeit und die Geschwindigkeit.

1.5 Objekte

Objekte besitzen Eigenschaften (vergleichbar mit Variablen) und Methoden (ähnlich zu Prozeduren). So hat zum Beispiel das Objekt Cells (Zellen) die Eigenschaften Column (Spalte) und die Methode select (auswählen). Eigenschaften können nicht nur Variablen sondern auch Objekte sein, in diesem Beispiel die Eigenschaft Borders (Rahmen), welche auch wieder Methoden und Eigenschaften besitzt. Im Anhang befindet sich eine Liste aller verwendeten Objekte.

Durch den Einsatz von Objekten lassen sich Strukturen und Abhängigkeiten besser darstellen.

1.5.1 Eigenschaften

Auf Eigenschaften greift man wie auf Variablen zu. Allerdings sind einige Eigenschaften nur lesbar, man kann ihnen keine Werte zuweisen. Meistens kann man solche Eigenschaften über Methoden ändern. Der Vorteil besteht darin, daß die Anwendung die Kontrolle über die mit der Eigenschaft verbundenen Zustände behält. So können interne Zähler, Zeiger oder Variablen entsprechend gesetzt werden. Application.Workbooks. Count ist eine nur lesbare Eigenschaft.

```
Application.Workbooks.Count = 5
```

führt zu einer Fehlermeldung, wogegen

```
set Variable = Application.Workbooks.Add
```

fehlerfrei abgearbeitet wird.

Die Methode Add des Objektes Workbooks erstellt eine neue Arbeitsmappe, dadurch erhöht sich die Anzahl in der Anwendung befindlichen Arbeitsmappen.

Greift man mehrmals hintereinander auf Eigenschaften eines Objektes zu, empfiehlt es sich, die `With`-Anweisung zu benutzen.

```
With Application
  .ScreenUpdating = False
  .Statusbar = „Arbeite...“
  .Height = 400
End With
```

ist gleichbedeutend mit

```
Application.ScreenUpdating = False
Application.Statusbar = „Arbeite...“
Application.Height = 400
```

verringert aber den Schreibaufwand und erhöht die Lesbarkeit.

1.5.2 Methoden

Methoden greifen auf Eigenschaften von Objekten zu, ohne daß der Anwender genau weiß, welche Eigenschaften die Methode ändert.

```
Assistant.Move 100, 100
```

bewegt den Assistenten an die Bildschirmkoordinaten 100, 100. Dabei werden offensichtlich die Eigenschaften Left und Top des Assistant Objektes geändert. Es ist durchaus möglich, diese Eigenschaft „per Hand“ auf 100 zu setzen, der Effekt ist derselbe. Nur steigt bei komplizierteren Methoden die Anzahl sich ändernder Eigenschaften und es ist weitaus komfortabler, die entsprechenden Methoden zu benutzen.

1.6 Kommentare

Um die Wartbarkeit des Quelltextes weiter zu erhöhen, bietet es sich an, Kommentare zu verwenden. Kommentare sind Texte, die nicht zur Programmausführung beitragen und deshalb von der Anwendung ignoriert werden. Sie werden mit einem einfachen Hochkomma eingeleitet. Der Kommentar gilt vom Hochkomma (‘) bis zum Zeilenende und ist in der Standardeinstellung grün gefärbt.

2. Anhang

2.1 Quellenanangabe

[1] Microsoft Excel 97 VisualBasic
Reed Jacobson
Microsoft Press Deutschland

[2] Microsoft Excel 97 – Das Handbuch
Reinke Solutions Team
Microsoft Press Deutschland

[3] Microsoft Office 97 – Visual Basic Programmierung
Günter Born
Microsoft Press Deutschland

[4] Newsgroup: microsoft.public.de.excel

[5] Newsgroup: microsoft.public.office.developer.vba

[6] Deja-News: http://www.deja.com

[7] Metager Suchmaschine: http://www.metager.de

[8] Microsoft Excel 97 VisualBasic Online-Hilfe