Workshop 3: VBA-Programmierung mit MS Excel

1 Kontrollstrukturen

In VBA werden zwei wichtige Gruppen von Kontrollstrukturen unterschieden:

- Verzweigungen und
- Schleifen

In Abb. 7 ist dargestellt, wie diese beiden Gruppen weiter unterteilt werden.

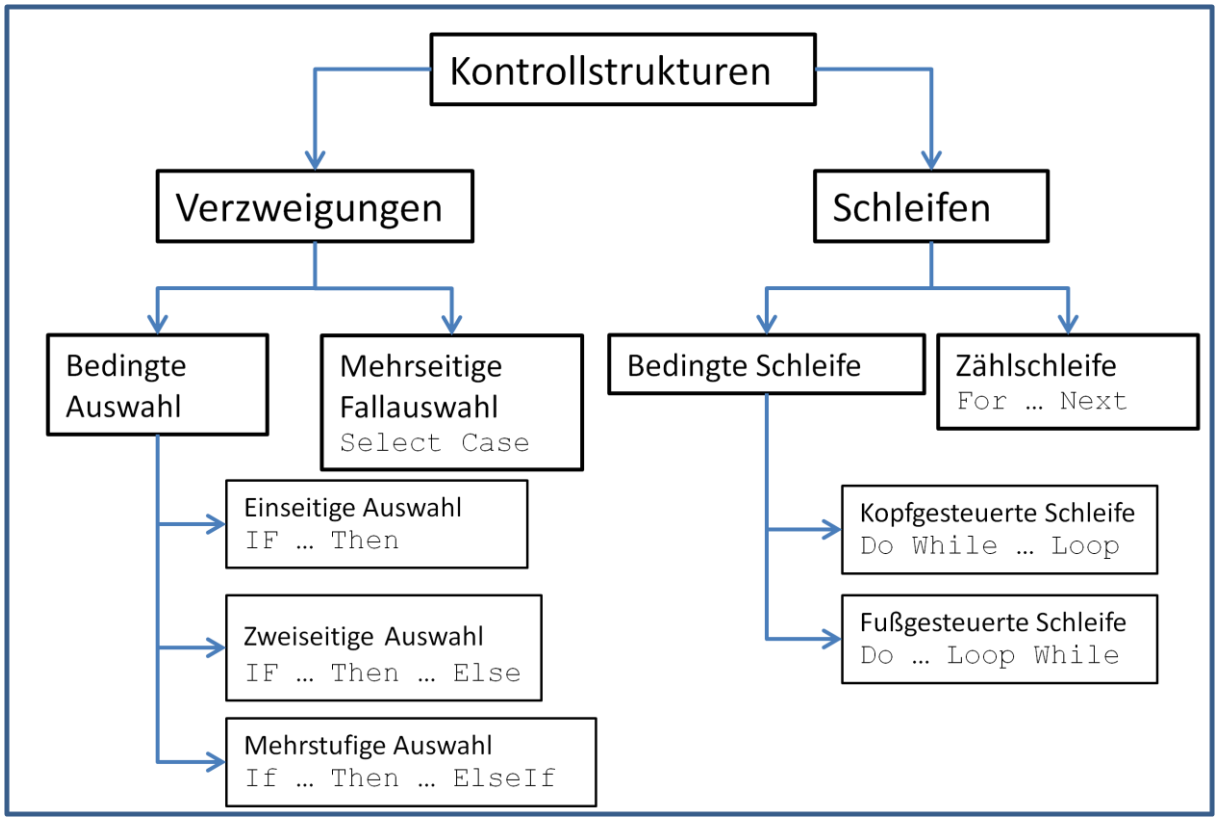


Abb. 7: Kontrollstrukturen in VBA

1.1. Verzweigungen

1.1.1 Einseitige Auswahl

Eine einseitige Auswahl ist wie folgt aufgebaut:

- Sie beginnt mit dem Schlüsselwort `IF`.
- Dahinter wird eine Bedingung (engl. condition) geschrieben.
- Nach der Bedingung folgt das Schüsselwort `Then`.
- Anschließend folgen eine oder mehrere Anweisungen, die ausgeführt werden, wenn die Auswertung der Bedingung den Wert `True` ergibt. Liefert die Auswertung der Bedingung dagegen den Wert `False`, erfolgt keine Aktion.
- Eine einseitige Anwahl wird mit `End If` abgeschlossen. Das ist allerdings nur dann notwendig, wenn mehr als eine einzelne Anweisung hinter `Then` folgen.

Beispiel: Die Prozedur `RabattBerechnen` berechnet einen Rabatt von 5 Prozent auf einen Mindestumsatz von 100 €. Die Prozedur enthält zwei einseitige Bedingungen, die geschachtelt sind. Diese Schachtelung wird im Programmcode durch Einrücken verdeutlicht. Dafür wird häufig der Begriff *Strukturierte Programmierung* verwandt.

- *Zuerst wird die numerische Eingabe des Preises abgefragt.*
- *Danach wird geprüft, ob der Preis* größer oder gleich 100 € ist.

Als Ergebnis der Prozedur wird der Endpreis ausgegeben:

```
Sub RabattBerechnen()
   Const conFmt    As String = "##,###.00 €"   ' Formatierungs-String
   Dim varEingabe As Variant
```

```
    Dim curPreis As Currency, curRabatt As Currency
    varEingabe = InputBox("Bitte geben Sie den Preis ein:", "Rabatt", 100)
    If IsNumeric(varEingabe) Then
        curPreis = CCur(varEingabe)
        If curPreis >= 100 Then
            curRabatt = curPreis * 0.05
            curPreis = curPreis - curRabatt
        End If
        MsgBox "Der Endpreis beträgt: " & Format(curPreis, conFmt)
    End If
End Sub
```

1.1.2 Zweiseitige Auswahl

Die zweiseitige Auswahl

- wird angewendet, wenn es genau zwei Verzweigungsmöglichkeit mit dazugehörigen Anweisungsblöcken gibt.
- führt in Abhängigkeit von einem Bedingungsausdruck nur einen der beiden Bedingungsblöcke aus.

Die Syntax der zweiseitigen Auswahl ergibt sich aus folgendem Pseudo-Code:

```
If Bedingung Then
      Anweisung(en)1
Else
      Anweisung(en)2
End If
```

- Der `Then`-Zweig wird ausgeführt, wenn die Auswertung der Bedingung `True` ergibt.
- Der `Else`-Zweig wird ausgeführt, wenn die Auswertung der Bedingung `False` ergibt.

In die bereits vorgestellte Prozedur `RabattBerechnen` kann beispielweise im `Else`-Zweig folgender Rabatthinweis eingebaut werden:

```
If curPreis >= 100 Then
    curRabatt = curPreis * 0.05
    curPreis = curPreis - curRabatt
    MsgBox "Der Endpreis beträgt: " & Format(curPreis, conFmt)
Else
    MsgBox "Der Endpreis beträgt: " & Format(curPreis, conFmt) & vbCrLf & _
        "Ab einem Betrag von 100 € erhalten Sie 5 % Nachlass."
End If
```

1.1.3 Mehrstufige Auswahl

Die mehrstufige Auswahl wird angewendet, wenn mehr als 2 Auswahlmöglichkeiten vorhanden sind. Die Syntax der mehrseitigen Auswahl ergibt sich aus folgendem Pseudo-Code:

```
If Bedingung1 Then
      Anweisung(en)1
ElseIf Bedingung2 Then
      Anweisung(en)2
ElseIF Bedingung3 Then
```

```
      Anweisung(en)3
Else
      Anweisung(en)4
End If
```

In die bereits vorgestellte Prozedur `RabattBerechnen` kann beispielweise folgende 4-stufige Rabattstaffel {0%, 5%, 7,5%, 10%} in Abhängigkeit von der Preishöhe aufgenommen werden:

```
Dim sngRabatt As Single
If curPreis < 100 Then
   MsgBox "Der Endpreis beträgt: " & _
      Format(curPreis, conFmt) & vbCrLf & _
      "Ab einem Betrag von 100 € erhalten Sie 5 % Nachlass."
ElseIf curPreis < 250 Then
   sngRabatt = 0.05
ElseIf curPreis < 500 Then
   sngRabatt = 0.075
Else
   sngRabatt = 0.1
End If
curRabatt = Round(curPreis * sngRabatt, 2)
curPreis = curPreis - curRabatt
MsgBox "Der Endpreis beträgt: " & Format(curPreis, conFmt)
```

1.1.4 Mehrseitige Auswahl

Bei der mehrseitigen Fallauswahl wird der Wert einer Variablen oder eines Ausdrucks abgefragt. Diese Variable oder dieser Ausdruck wird als Selektor bezeichnet. In Abhängigkeit vom Wert des Selektors können je nach Fall (engl. case) unterschiedliche Anweisungsblöcke durchlaufen werden.

Die folgende Prozedur `JahreszeitBestimmen` demonstriert die Anwendung der mehrseitigen Fallauswahl, auch `Select Case`-Anweisung genannt:

```
Sub JahreszeitBestimmen()
   Dim strSaison As String
   Select Case Month(Date)
      Case 3, 4, 5
         strSaison = "Frühlings"
      Case 6, 7, 8
         strSaison = "Sommer"
      Case 9, 10, 11
         strSaison = "Herbst"
      Case Else
         strSaison = "Winter"
   End Select
   MsgBox "Zur Zeit ist " & strSaison & "-Saison."
End Sub
```

Die Syntax zur Fallauswahl lässt sich wie folgt beschreiben:

- Die Schlüsselwörter `Select Case` leiten die mehrseitige Fallauswahl ein.

- Danach folgt der Selektor. Das kann eine Konstante, eine Variable oder ein Ausdruck sein. Der Wert des Selektors bestimmt, welcher Anweisungsblock ausgeführt wird.
- Hinter dem Schlüsselwort `Case` werden die Vergleichswerte angegeben. Dafür gibt es mehrere Varianten:

Variante		Beispiel:
❶	Ein einzelner Wert	`Case 1`
❷	Eine Liste von Werten	`Case 1, 2, 3`
❸	Ein Wertebereich	`Case 1 To 3`
❸	Ein Vergleichsausdruck	`Case Is > 1`

- Stimmt der Wert des Selektors mit einem der aufgeführten Auswahlwerte überein, werden nur die Anweisungen unmittelbar danach ausgeführt. Alle anderen werden ignoriert.
- Stimmt der Wert des Selektors mit keinem der Auswahlwerte überein, werden die Anweisungen hinter `Case Else` ausgeführt.
- Die Fallauswahl endet mit den Schlüsselwörter `End Select`.

1.2 Schleifen

2.2.1 Schleifenaufbau

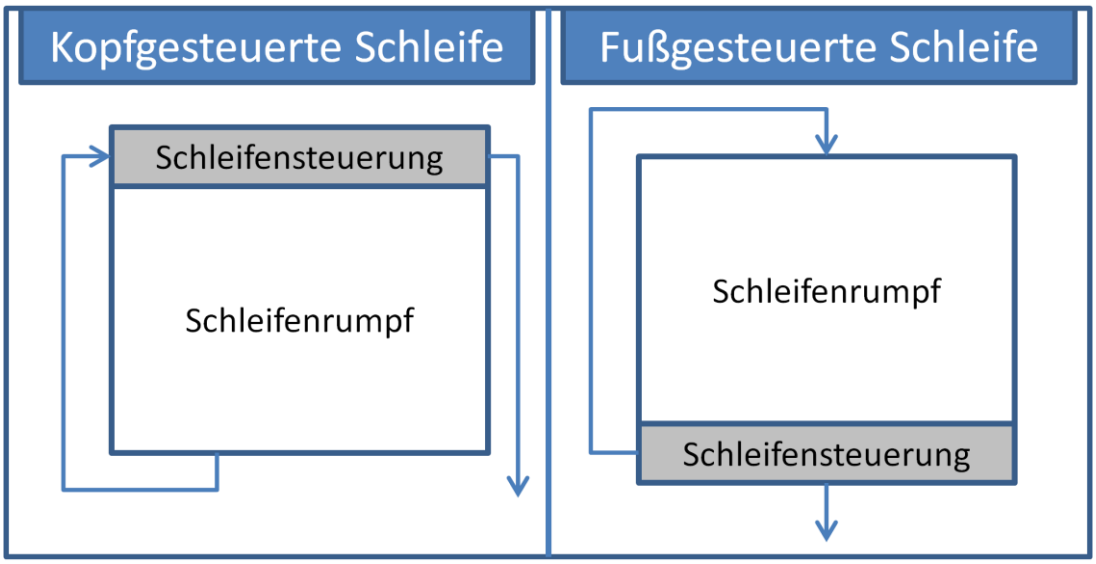
Um einen bestimmten Programmteil mehrmals zu durchlaufen, werden Schleifen benötigt. Eine Schleife besteht im Wesentlich aus zwei Teilen:

- der Schleifensteuerung und
- dem Schleifenrumpf

Die Schleifensteuerung bestimmt, wie oft die Anweisungen des Schleifenrumpfs ausgeführt werden bzw. nach welchen Kriterien entschieden wird, ob eine Wiederholung erforderlich ist.

2.2.2 Schleifenarten

Zwei Arten der Schleifensteuerung werden im Allgemeinen unterschieden:



Kopfgesteuerte Schleife (= abweisende Schleife)	Fußgesteuerte Schleife (= nicht abweisende Schleife)
Die Prüfung, ob die Anweisungen im Schlei-	Die Anweisungen im Schleifenrumpf werden

fenrumpf ausgeführt werden, erfolgt gleich zu Beginn.	mindestens einmal ausgeführt.
Falls das Kriterium zu Beginn nicht erfüllt ist, wird der Schleifenrumpf gar nicht ausgeführt.	Am Ende der Schleife erfolgt die Prüfung, ob einer weiterer Durchlauf erforderlich ist.

Abb. 8 Art der Schleifensteuerung

2.2.3 Die kopfgesteuerte Schleife

Die kopfgesteuerte Schleife ist definitionsgemäß eine abweisende Schleife. Drei Varianten werden unterschieden:

Do While	Do Until	While

`Do [{While} Bedingung]`
`  [Anweisungen]`
`  [Exit Do]`
`  [Anweisungen]`
`Loop`
	`Do [{Until} Bedingung]`
	`  [Anweisungen]`
	`  [Exit Do]`
	`  [Anweisungen]`
	`Loop`
		`While Bedingung`
		`  [Anweisungen]`
		`  [Go To Sprungmarke]`
		`  [Anweisungen]`
		`Wend`

Tab. 8: Syntax der kopfgesteuerten Wiederholung

Mit dem Schlüsselwort `While` wird eine Schliefe durchlaufen, solange die Bedingung zutrifft.
Mit dem Schüsselwort `Until` wird eine Schleife solange durchlaufen, bis die Bedingung erfüllt ist.
Wenn erforderlich, kann mit `Exit Do` bzw. `GoTo Sprungmarke` die Schleife vorzeitig beendet werden.
Die `While...Wend`-Anweisung ist veraltet. Die beiden `Do...Loop`-Anweisungen stellen eine strukturiertere und flexiblere Möglichkeit zur Ausführung von kopfgesteuerten Schleifen dar.

2.2.4 Die fußgesteuerte Schleife

Zwei Varianten werden unterscheiden:

Do...Loop While	Do...Loop Until
Do [Anweisungen] [Exit Do] [Anweisungen] Loop While Bedingung	Do [Anweisungen] [Exit Do] [Anweisungen] Loop Until Bedingung

Tab. 9: Syntax der fußgesteuerten Wiederholung

Das Schlüsselwort `While` bestimmt, dass die Schleife ausgeführt wird, solange die Bedingung.
Das Schüsselwort `Until` bestimmt, dass die Schleife solange durchlaufen wird, bis die Bedingung erfüllt ist.

Das folgende Beispiel demonstriert, wie eine fußgesteuerte Schleife `Do...Loop Until` in Verbindung mit einer bedingten `GoTo`-Anweisung bei der Mehrwertsteuerberechnung angewandt werden kann.

```
Sub MwstBerechnen()
   Const conFmt   As String = "##,###.00 €"  ' Formatierungs-String
   Dim curBrutto  As Variant  ' Bruttobetrag
   Dim avarMwst   As Variant  ' Datenfeld mit MwSt-Sätzen
   Dim varNetto   As Variant  ' Nettobetrag
   Dim varIdx     As Variant  ' Index des Datenfelds
   Dim intCount  As Integer   ' Scheifenzähler
   avarMwst = Array(0, 0.075, 0.19)
   Do
      varNetto = InputBox("Nettobetrag erfassen")
      If Not IsNumeric(varNetto) Then
         intCount = intCount + 1
         If intCount < 2 Then
            MsgBox "Bitte geben Sie einen Zahlenwert ein", vbCritical
         End If
      End If
   Loop Until IsNumeric(varNetto) Or intCount > 2
   If intCount > 2 Then
      MsgBox "Zu viele Fehlversuche. Leider Programmsbbruch!", vbCritical,
"Nettobetrag erfassen"
      Exit Sub
   End If

SchleifenAnfang:
   varIdx = InputBox("Mehrwertsteuersatz eingeben:" & vbNewLine & _
      "0 für 0 %" & vbNewLine & _
      "1 für 7 %" & vbNewLine & _
      "2 für 19.0 %")
   If CInt(varIdx) < 0 Or CInt(varIdx) > 2 Then
      MsgBox "Nummer des MwSt-Satzes liegt nicht zwischen 0 und 2."
      GoTo SchleifenAnfang
   End If
   curBrutto = CCur(varNetto) * (1 + avarMwst(CInt(varIdx)))
   MsgBox "Mit einem Nettobetrag in Höhe von " & Format(CCur(varNetto),
conFmt) & vbNewLine & _
```

```
          "und einem Mehrwertsteuersatz von " & avarMwst(CInt(varIdx)) *
100 & " Prozent " & vbNewLine & _
          "errechnet sich ein Bruttobetrag in Höhe von " & For-
mat(curBrutto, conFmt)
End Sub
```

2.2.5 Exkurs: Verzweigung mit GoTo

Die Prozedur `MwstBerechnen` zeigt u.a. wie mit dem Sprungbefehl `GoTo` eine Scheife programmiert werden kann. Hinter dem Schlüsselwort `GoTo` folgt eine benannte Sprungmarke (engl. Label) die angibt, wohin das Programm gezielt verzweigen soll.

Um mit `GoTo` eine bedingte Verzweigung zu programmieren wird folgendes codiert:

```
If Bedingung Then GoTo Sprungmarke
```

Mit nach diesem Schema gebildeten bedingten Sprüngen lassen sich alle „höheren“ Kontrollstrukturen von VBA nachbilden, z. B. Zählschleifen.

In der folgenden Prozedur `BedingterSprung` wird der Schleifenrumpf genau zehnmal durchlaufen. Bei der Schleifensteuerung wird `GoTo Schleife` angewandt:

```
Sub BedingterSprung()
   Dim intCount As Integer
Schleife:
   Debug.Print "Durchlauf " & intCount
   intCount = intCount + 1
   If intCount < 10 Then GoTo Schleife
End Sub
```

Direktbereich

```
Durchlauf 0
Durchlauf 1
Durchlauf 2
Durchlauf 3
Durchlauf 4
Durchlauf 5
Durchlauf 6
Durchlauf 7
Durchlauf 8
Durchlauf 9
```

2.2.6 Die Zählschleife

Sie können eine zählergesteuerte Schleife verwenden, um einen Block von Anweisungen für eine genau bestimmte Anzahl von Wiederholungen auszuführen. Kennzeichnendes Merkmal einer Zählschleife ist, wie ihr Name bereits verrät, eine Zählervariable. Deren Wert wird mit jedem Schleifendurchlauf erhöht oder vermindert. Sie brauchen sich nicht darum zu kümmern, den Zähler selbst hoch- oder herunterzusetzen. Syntax:

```
For Zähler = Startwert To Endwert [Step Schrittweite]
   [Anweisungen]
   [Exit For]
   [Anweisungen]
Next [Zähler]
```

Argumente der Zählschleife:

Argument	Beschreibung
`Zähler`	Erforderlich. Besteht aus einer numerischen Variablen, die als Schleifenzähler dient.
`Startwert`	Das ist der Startwert vom `Zähler`.

Endwert	Legt den Endwert von Zähler fest.
Schrittweite	Optional. Damit kann der Wert bestimmt werden, um den sich Zähler bei jedem Schleifendurchlauf verändert. Falls kein Wert angegeben wird, ist die Voreinstellung eins.
Anweisungen	Die Anweisungen zwischen For…Next werden so oft ausgeführt wie angegeben.
Exit For	Optional. Innerhalb der Schleife kann eine beliebige Anzahl von Exit For-Anweisungen an beliebiger Stelle als Möglichkeit zum Verlassen der Schleife verwendet werden.

Tab. 10: Argumente der Zählschleife

Im folgenden Code-Beispiel wird die For…Next-Schleife eingesetzt, um eine Suche nach Excel-Dateien in deren Standardverzeichnis durchzuführen. Dabei werden die ersten zehn Dateien des Ordners im Direktfenster ausgegeben. Für den Fall, dass weniger als 10 Dateien gefunden werden, enthält die Prozedur die Abbruchbedingung Exit For.

```
Sub DateienAnzeigen()
   ' Die ersten 10 Excel-Dateien im Standardverzeichnis auslisten
   ' oder vorher abbrechen, falls weniger gefunden werden.
   Dim strVerzeichnis As String  ' Verzeichnisname
   Dim strSuche As String        ' Dateiname
   Dim intCount As Integer       ' Schleifenzähler
   strVerzeichnis = Application.DefaultFilePath & _
      "\Excel_Sheets\*.*"
   strSuche = Dir(strVerzeichnis)
   For intCount = 1 To 10
      Debug.Print strSuche
      strSuche = Dir
      If strSuche = vbNullString Then
         Exit For
      End If
   Next intCount
End Sub
```

2.7 Sonderform der Zählschleife

Eine besondere Form der For-Next-Schleife in VBA ist die For-Each-Next-Schleife. Damit können Sie jedes Element in einem Array oder einer Auflistung (= Liste von Objekten) ansprechen. Die Zahl der Elemente muss vorher nicht bekannt sein.

In der nun folgenden Prozedur TabellenblätterAuslesen wird eine For-Each-Next-Schleife benutzt, um die Namen aller Tabellenblätter in der aktuellen Arbeitsmappe in das Direktfenster zu schreiben.

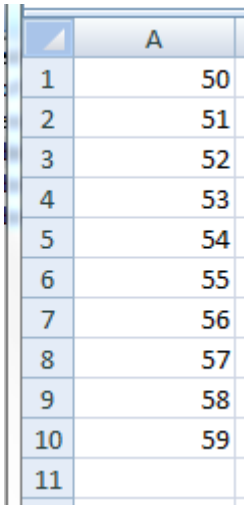
```
Public Sub TabellenblätterAuslesen()
   ' Die Namen aller Tabellenblätter auslesen
   Dim ws As Excel.Worksheet
   For Each ws In Worksheets
      Debug.Print "Name: " & ws.Name
   Next ws
End Sub
```

Workshop 3: VBA-Programmierung mit MS Excel

3 Übungen

3.1 Zellen mit Zählschleife belegen

Erstellen Sie die Prozedur `ZellenBelegen` in einer neuen Arbeitsmappe. Die Prozedur soll die Zahlen von 50 bis 59 in die Zellen A1 bis A10 des Tabellenblatts `Tabelle1` schreiben.



	A
1	50
2	51
3	52
4	53
5	54
6	55
7	56
8	57
9	58
10	59
11	

Verwenden Sie dazu eine Zählschleife.

3.2 Schleifenart erkennen

- Welche Schleifenart wird in der Routine `ZahlenAddieren` benutzt?
- Unter welcher Bedingung kann die Schleife mit `Exit Do` vorzeitig verlassen werden?

```
Sub ZahlenAddieren()
   Dim varInput As Variant
   Dim dblSumme As Double
   Dim bolWeiter As Boolean
   Do
      varInput = InputBox("Bitte geben Sie eine Zahl ein: ", , 1)
      If Not IsNumeric(varInput) Then
         MsgBox "Nur Zahlen sind erlaubt." & vbCrLf & _
                "Geplanter Programmabbruch!", vbExclamation, "Eingabefehler"
         Exit Do
      End If
      dblSumme = dblSumme + CDbl(varInput)
      bolWeiter = MsgBox("Weitere Zahlen erfassen?", vbQuestion + vbYesNo)
   Loop Until bolWeiter = vbNo
   MsgBox "Die Summe der richtig eingegebenen Zahlen lautet: " & dblSumme
End Sub
```

3.3 Gefühlte Temperatur

Sie möchten die gemessene Außentemperatur in einem Wort ausdrücken. Folgende Wörter und Temperaturbereiche sind Ihnen vorgegeben:

Wort	*Temperatur in Grad Celsius*
eisig	*unter 0*
kalt	*0 bis unter 11*
warm	*11 bis unter 21*
heiß	*21 und mehr*

Schreiben Sie dafür die Prozedur `GefuehlteTemperatur`. Die die mehrstufige Auswahl soll zur Anwendung kommen.

3.4 Bedingungen formulieren

Wenn der Wert der Variablen intZahl nicht größer als 100 ist, soll ihr Wert verdoppelt werden, sonst soll er halbiert werden. Wie lautet die benötige `If`-Anweisung?

3.5 Mehrwegverzweigung

Welche Arten von `If`-Anweisungen könnten Sie verwenden, um mehr als zwei verschiedene Fälle zu unterscheiden, beispielsweise zwischen den Werten 10, 20 und 30 der Variablen `intZahl`?

4 Lösungen

4.1 Zellen mit Zählschleife belegen

```
Sub ZellenBelegen()
   Dim intCount As Integer
   For intCount = 50 To 59
      ActiveWorkbook.Worksheets("Tabelle1").Cells(intCount - 49, 1) = intCount
   Next intCount
End Sub
```

4.2 Schleifenart erkennen

- Es handelt sich um eine fußgesteuerte (also eine nicht abweisende) Schleife.
- Ein geplanter Programmabbruch erfolgt bei nicht-numerischer Eingabe.

4.3 Gefühlte Temperatur

```
Sub GefuehlteTemperatur()
   Dim varEingabe As Variant  ' Eingabe
   Dim sngGrad    As Single   ' Temperatur
   Dim strMsg     As String   ' Meldung
   varEingabe = InputBox("Temperatur: ", "Temperatur erfassen")
   If IsNumeric(varEingabe) Then
      sngGrad = CSng(varEingabe)
      ' Mehrstufige Auswahl
      If sngGrad < 0 Then
         strMsg = "eisig"
      ElseIf sngGrad < 11 Then
         strMsg = "kalt"
      ElseIf sngGrad < 21 Then
         strMsg = "warm"
      Else
         strMsg = "heiß"
      End If
      MsgBox "Bei " & sngGrad & " Grad ist es " & strMsg & "!", vbInformation
   End If
End Sub
```

4.4 Bedingungen formulieren

1. Lösung:

```
If intZahl <= 100 Then intZahl = intZahl * 2 Else intZahl = intZahl / 2
```

2. Lösung

```
If intZahl <= 100 Then
   intZahl = intZahl * 2
Else
   intZahl = intZahl / 2
End If
```

Die zweite Lösung ist die besser strukturiert und deshalb vorzuziehen.

4.5 Mehrwegverzweigung

Es gibt mindestens 3 Lösungsmöglichkeiten:

Mehrere einfache `If`-Anweisungen sequenziell anordnen:

```
If intZahl = 10 Then ...
If intZahl = 20 Then ...
If intZahl = 30 Then ...
```

Mehrstufige `If`-Verzweigung anwenden:

```
If intZahl = 10 Then
       ...
ElseIf intZahl = 20 Then
       ...
ElseIf intZahl = 30 Then
       ...
End IF
```

Fallunterscheidung mit `Select Case` vornehmen:

```
Select Case intZahl
   Case 10
       ...
   Case 20
       ...
   Case 30
       ...
End Select
```

Die zweite und dritte Lösung sind übersichtlicher und deshalb vorzuziehen.