

B

Anhang B



Lösungen

Antworten Kapitel 1

Frage 1: VBA bedeutet Visual Basic for Applications.

Frage 2: Ein Programm ist ein Ablaufplan, der in einer bestimmten Programmiersprache geschrieben ist.

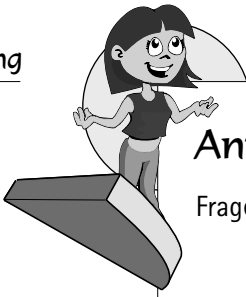
Kapitel 1, Aufgabe 1

```
Sub Ausgabe()  
    MsgBox "Text"  
End Sub
```

Es passiert Folgendes:

- ❖ Eine Fehlermeldung erscheint, wenn du ein Anführungszeichen weglässt.
- ❖ 0 als Ausgabe erscheint, wenn du beide Anführungszeichen weglässt.
- ❖ VBA korrigiert selbstständig in MsgBox, wenn du msgbox klein schreibst.

B



Antworten Kapitel 2

- Frage 1: Variablen sind Speicherplätze. Diese Speicherplätze haben eine Adresse (Variablenname), eine Größe und ein Format (abhängig vom Typ wie z. B. Gleitpunktzahl, String etc.).
- Frage 2: Zur Eingabe am Bildschirm haben wir bisher die `InputBox`-Funktion benutzt.
- Frage 3: Zur Ausgabe am Bildschirm eignet sich die `MsgBox`-Funktion, zur Ausgabe in Excel-Zellen kannst du einer Zelle mit dem `Cells`-Objekt Daten zuweisen: `Cells(1,2)="Hallo"`
- Frage 4: Mit einer bedingten Anweisung ist gemeint, dass nur unter einer bestimmten Bedingung Programmcode abgearbeitet wird. Dazu stellt VBA die `If`-Anweisung sowie die `Select Case`-Anweisung bereit.
- Frage 5: Eine Schleife ist eine Wiederholungsanweisung. Damit steuerst du, wie oft ein Codeblock durchlaufen wird.

Antworten Kapitel 3

- Frage 1: Wenn dir VBA Hilfe anbietet oder du siehst dir den Objektkatalog an, erkennst du vor einer Routine einen fliegenden grünen Klotz, vor einer Eigenschaft ist eine Hand mit Karte angezeigt.
- Frage 2: Im Objektkatalog bekommst du eine umfangreiche Liste der zur Verfügung stehenden Objekte angezeigt.
- Frage 3: Wenn du während des Programmierens den Objektbezeichner samt folgendem Punkt ».« getippt hast, erscheint ein Hilfefeld. Darin kannst du Eigenschaften oder Routinen auswählen.

Kapitel 3, Aufgabe 1

```
Sub Celltest()  
    Cells(1,1).Interior.ColorIndex = 6  
    Cells(2,1).Interior.ColorIndex = 6  
    Cells(3,1).Interior.ColorIndex = 6  
    Cells(1,2).Interior.ColorIndex = 6  
    Cells(2,2).Interior.ColorIndex = 6  
    Cells(3,2).Interior.ColorIndex = 6  
End Sub
```



Kapitel 3, Aufgabe 2

Ausprobieren kannst du die Lösung in der Excel-Mappe `Quartal.xlsm`.

```
Sub QuartalAusfuellen()  
  Sheets("Tabelle1").Activate  
  Cells(1, 1).Value = "Januar"  
  Sheets("Tabelle2").Activate  
  Cells(2, 1).Value = "Februar"  
  Sheets("Tabelle3").Activate  
  Cells(3, 1).Value = "März"  
End Sub
```

Kapitel 3, Aufgabe 3

Ausprobieren kannst du die Lösung in der Excel-Mappe `Regenbogen.xlsm`.

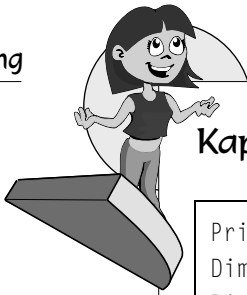
```
Sub Regenboga()  
  Cells(1, 1).Interior.ColorIndex = 1  
  Cells(2, 1).Interior.ColorIndex = 2  
  Cells(3, 1).Interior.ColorIndex = 3  
  Cells(4, 1).Interior.ColorIndex = 4  
  Cells(5, 1).Interior.ColorIndex = 5  
  Cells(6, 1).Interior.ColorIndex = 6  
End Sub
```

Die Version `Regenbogenb`, die in dieser Mappe ebenfalls enthalten ist, zeigt eine Lösung mit der sogenannten `For`-Schleife, die in der ersten Spalte alle Zeilen von 1 bis 56 abläuft und den zugehörigen Farbwert setzt. Gestellt wird die Aufgabe für das Programm mit der Schleifenvariante auf Seite 233 im Kapitel über Schleifen.

Antworten Kapitel 4

- Frage 1: Ein Event ist ein Ereignis. Der Rechner prüft die Aktionen des Anwenders ab. Für bestimmte Aktionen des Anwenders sind Unterprogramme bereitgestellt, die der Programmierer nur noch mit Programmcode zu füllen braucht.
- Frage 2: Bei dem Event `Worksheet_Activate` wird überprüft, ob du das Tabellenblatt gewechselt hast. So, den Rest überlassen wir jetzt dir.

B



Kapitel 4, Aufgabe 1

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    Dim Zeile As Integer
    Dim Spalte As Integer

    'Wichtige Zelle für Hinweis: Zeile=2, Spalte=3
    Zeile = 2
    Spalte = 3
    If Target.Row = Zeile And Target.Column = Spalte Then
        MsgBox "In dieser Zelle dürfen nur Werte zwischen 10 und 100 eingetragen werden."
    End If
End Sub
```

Wichtig ist, dass der Event in dem dazugehörigen Tabellenblatt steht (Projekt-Explorer zum Auswählen des Tabellenblatts, dann an diese Stelle den Code tippen).



In der Mappe EventSinnvoll.xlsm kannst du in Tabelle1 das Programm finden. Du kannst Tabelle2 nutzen, um das Programm selbst zu erstellen und abzuändern.



Die folgenden Beispiele kannst du in der Mappe EventAufgaben.xlsm ausprobieren.

Kapitel 4, Aufgabe 2

Code für Event in Tabelle1:

```
Private Sub Worksheet_Activate()
    MsgBox "Lass mich (Tabelle1) schlafen!"
End Sub
```

während du in Tabelle2 folgenden Event programmierst:

```
Private Sub Worksheet_Activate()
    MsgBox "Lass mich (Tabelle2) schlafen!"
End Sub
```



Je nachdem, welches Tabellenblatt du danach auswählst, kommt es zu der einen oder der anderen Meldung.

Eine Anwendung im Büro wäre z. B., den Event `Worksheet_Activate` einzusetzen, um den Anwender darauf hinzuweisen, dass es sich bei diesem Blatt um veraltete Daten handelt o. Ä.

```
Private Sub Worksheet_Activate()  
    MsgBox "Bitte diese Tabelle nicht mehr ändern, sie ➡  
    wird komplett überarbeitet."  
End Sub
```

Kapitel 4, Aufgabe 3

Der Event, den Excel für das Rechnen im jeweiligen Tabellenblatt bereitstellt, heißt `Worksheet_Calculate()`.

Wenn unser armer Kurt nun wieder einmal über seiner Steuererklärung brütet (oder du über deine Hausaufgaben schmorst), wird ihm das bestimmt nicht gefallen.

```
Private Sub Worksheet_Calculate()  
    MsgBox "Na, wieder eine neue Berechnung fällig, wann ➡  
    sind deine Zahlen endlich komplett?"  
End Sub
```

Eine weitere Ausdrucksmöglichkeit für »hämische PCs«:

Falls sich in Kurts Pizzeria Unbefugte an seinem PC zu schaffen machen, erschreckt er sie mit folgendem Event, der immer dann aktiv wird, wenn wir eine andere Zelle anklicken.

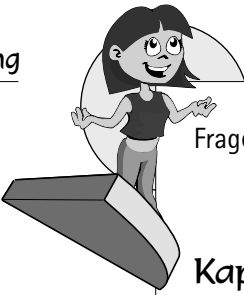
```
Private Sub Worksheet_SelectionChange(ByVal Target As ➡  
Range)  
    MsgBox "Aua, nicht immer so fest klicken"  
End Sub
```

Antworten Kapitel 5

Frage 1: Der Debugger ist ein Hilfsmittel, um Fehler aus dem Programm zu entfernen.

Frage 2: Mit der Taste `[F1]`; wenn du ein Wort wie z. B. ein reserviertes Wort markiert und `[F1]` gedrückt hast, erhältst du Hilfe zu diesem Wort.

B



Frage 3: Wenn du die Aufzeichnung abschließt, musst du unbedingt **AUFZEICHNUNG BEENDEN** (auf der Registerkarte **ENTWICKLERTOOLS**) wählen.

Kapitel 5, Aufgabe 1

```
Sub formatieren2()
'
' formatieren2 Makro
' Wir formatieren die aktive Zelle.
'
With Selection.Interior
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
    .Color = 65535
    .TintAndShade = 0
    .PatternTintAndShade = 0
End With
Selection.Font.Bold = True

End Sub
```

Antworten Kapitel 6

Frage 1: Löcher sind die Bereiche auf dem Zahlenstrahl, die z. B. bei dem Typ **Double** nicht existieren. Es gibt jeweils »daneben« eine Zahl, die als »Ersatz« für die ursprünglich gewünschte Zahl herhalten muss. Bei der Typangabe zu **Double** erkennst du, dass die Null fehlt.

Frage 2: Der Typ **Integer** ist wie **Long** ein Typ für Zahlen ohne Nachkommastellen. **Long** kann größere Zahlen abspeichern als **Integer**.

Frage 3: Bei einem Überlauf ist der Typ zu klein gewählt. Der abzuspeichernde Wert kann nicht mit dem gewählten Typ dargestellt werden.

Kapitel 6, Aufgabe 1

Wir simulieren diesen Vorgang mit einem Programm. Jeder Variablen entspricht ein Glas.

Die beiden ausführbaren Programme findest du in der Mappe **Variablen.xlsm**.



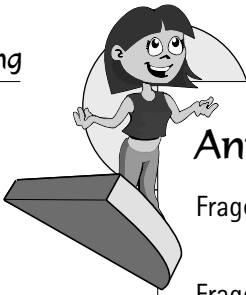
```
Sub Variablentausch1()  
Dim KurtGlas As String  
Dim GastGlas As String  
Dim SektGlas As String  
  
KurtGlas = "Orangensaft"  
GastGlas = "Traubensaft"  
SektGlas = "leer"  
  
'tauschen  
SektGlas = KurtGlas  
KurtGlas = GastGlas  
GastGlas = SektGlas  
  
MsgBox "Kurts Glas: " & KurtGlas 'Traubensaft  
MsgBox "Gast Glas: " & GastGlas 'Orangensaft  
End Sub
```

Alternativ kannst du den Inhalt der Gläser auch so tauschen:

```
Sub Variablentausch2()  
Dim KurtGlas As String  
Dim GastGlas As String  
Dim SektGlas As String  
  
KurtGlas = "Orangensaft"  
GastGlas = "Traubensaft"  
SektGlas = "leer"  
  
'tauschen  
SektGlas = GastGlas  
GastGlas = KurtGlas  
KurtGlas = SektGlas  
  
MsgBox "Kurts Glas: " & KurtGlas 'Traubensaft  
MsgBox "Gast Glas: " & GastGlas 'Orangensaft  
End Sub
```

Im Speicher verhält es sich mit den Variablen wie mit den Gläsern. Wenn wir etwas ausgegossen haben (ersetzt durch etwas anderes, z. B. leer), dann ist es an der Speicherstelle eben weg.

B



Antworten Kapitel 7

Frage 1: Vergleichsoperatoren sind <, <=, =, >= und >. Verwechsle nicht die Zuweisung (=) mit dem Vergleich (=>).

Frage 2: Prinzipiell ist es gleichgültig, ob du & oder + verwendest, um Texte zusammenzuhängen. Probleme handelst du dir ein, wenn du Variableninhalte, die z. B. Zahlen enthalten, mit + verknüpfst.

Kapitel 7, Aufgabe 1

```
Sub Quadrieren()
Dim Zahl As Integer
Dim Quadrat As Long

Zahl = Val(TextBox("Bitte geben Sie eine Zahl ein"))
'Val holt aus der Eingabe die Zahl heraus
Quadrat = Zahl * Zahl
MsgBox Quadrat
End Sub
```

Die Funktion Val() um die Inputbox verhindert, dass mit einer Fehlermeldung abgebrochen wird, wenn der Anwender nichts oder Zeichen eingibt.

In obigem Programm ist es auch wichtig, die Variable Quadrat als Long zu vereinbaren, weil der Wertebereich der quadrierten Zahl sehr groß werden kann.

Kapitel 7, Aufgabe 2

```
Sub AepfelKinder()
Dim Kinder As Integer
Dim Aepfel As Long

Kinder = Val(TextBox("Wie viele Kinder gibt es?"))
Aepfel = Val(TextBox("Wie viele Aepfel gibt es?"))

MsgBox Aepfel \ Kinder & " Apfel / Äpfel bekommt jedes ⚡
Kind, es bleiben " & Aepfel Mod Kinder & " Apfel / Äpfel ⚡
übrig"

End Sub
```




Kapitel 7, Aufgabe 3

```
Sub Nuller()
Dim TelMaier As Long 'Sonst tritt ein Überlauf auf
Dim TelMueller As String

TelMaier = 69123456 'Die führende Null wird schon vom ↗
Excel-Editor entfernt.
TelMueller = "06998765"
MsgBox "Tel. Maier " & TelMaier
MsgBox "Tel. Mueller " & TelMueller
End Sub
```

Die 0 als Vorwahl für Frankfurt (069) fehlt bei der ersten MessageBox.

Der Vorteil des Typs String ist darin zu sehen, dass auch Leerzeichen, Bindestriche und Schrägstriche der Telefonnummer hinzugefügt werden können. Da kaum jemand mit Telefonnummern rechnet, ist diese Wahl naheliegend.

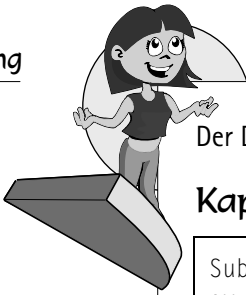
Kapitel 7, Aufgabe 4

In Zellbezügen der Makros ausgedrückt bedeutet das:

```
Sub Notendurchschnitt1()
Dim Zaehler As Integer
Dim Nenner As Integer
Dim Durchschnitt As Double

'Mit dem Unterstrich _ wird eine Zeile
'auf mehrere Zeilen aufteilbar.
Zaehler = Cells(2, 1) * Cells(3, 1) + _
          Cells(2, 2) * Cells(3, 2) + _
          Cells(2, 3) * Cells(3, 3) + _
          Cells(2, 4) * Cells(3, 4) + _
          Cells(2, 5) * Cells(3, 5) + _
          Cells(2, 6) * Cells(3, 6)
Nenner = Cells(3, 1) + Cells(3, 2) + _
          Cells(3, 3) + Cells(3, 4) + _
          Cells(3, 5) + Cells(3, 6)
If Nenner > 0 Then
    Durchschnitt = Zaehler / Nenner
    MsgBox Durchschnitt
End If
End Sub
```

B



Der Durchschnitt beträgt 3,38...

Kapitel 7, Aufgabe 5

```
Sub GitternetzlinienToggle()  
'Wenn die Gitternetzlinien eingeblendet waren, werden ☞  
sie ausgeblendet und umgekehrt.  
'Das nennt man Toggeln.  
ActiveWindow.DisplayGridlines = Not (ActiveWindow.☞  
DisplayGridlines)  
End Sub
```

Kapitel 7, Aufgabe 6

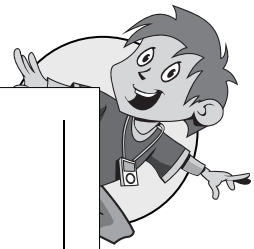
Wir wählen den Event `Worksheet_Activate` für die Tabelle `Gesicht` und füllen dieses leere Unterprogramm mit folgendem Programmcode:

```
Private Sub Worksheet_Activate()  
    MsgBox "Ich bin eine verzauberte Prinzessin, klick mir ☞  
    mit der Maus auf den Mund."  
End Sub
```

Sobald unser Anwender eine Zelle anklickt (nämlich den virtuellen Mund), aktiviert er den nächsten Event (`Worksheet_SelectionChange`), den wir auch noch programmieren müssen. Darin wird abgeprüft, welche Zelle der Anwender angeklickt hat. Waren es die Zellen `Cells(11,8) = H11` oder `Cells(11,9) = I11` oder `Cells(11,10) = J11`, dann wurde der Mund getroffen. Wir schließen bei der Prinzessin die Augen, indem wir die Zellen `Cells(6,6) = F6` und `Cells(6,12) = L6` auf schwarz setzen (Farbindeix schwarz = 1).

Um nun festzustellen, welche Zelle getroffen wurde, stellt uns Excel eine Zielscheibe zur Verfügung, der wir diese Information entnehmen können. `Target` ist diese Zielscheibe. `Target.Row` gibt die Zeile der getroffenen Zelle an, `Target.Column` gibt die Spalte der getroffenen Zelle an.

```
Private Sub Worksheet_SelectionChange(ByVal Target As ☞  
Range)  
If Target.Row = 11 And (Target.Column = 8 Or ☞  
Target.Column = 9 Or Target.Column = 10) Then  
    'Augen schließen:  
    Cells(6, 6).Interior.ColorIndex = 1  
    Cells(6, 12).Interior.ColorIndex = 1
```



```
Else  
    'Augen öffnen  
    Cells(6, 6).Interior.ColorIndex = 2  
    Cells(6, 12).Interior.ColorIndex = 2  
End If  
End Sub
```

Sollte der Anwender nicht auf den Mund geklickt haben, wird der Farbwert der Augen auf weiß (2) gesetzt.

Die Lösung findest du in der Mappe `Prinzessin.xlsm`.

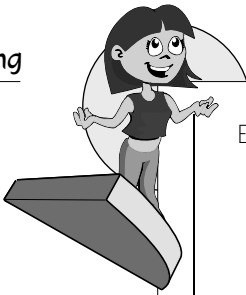
Antworten Kapitel 8

- Frage 1: Du kannst beim Überprüfen von Zeichenfolgen nützliche Tests durchführen, z. B. die Länge überprüfen (`Len`-Funktion, bei der Postleitzahl ist die Länge immer fünf Zeichen). Du kannst überprüfen, ob es Groß- oder Kleinbuchstaben sind, da Namen immer großgeschrieben werden. Dazu kannst du die `Asc`-Funktion verwenden.
- Frage 2: In der Schule werden dir früher oder später die trigonometrischen Funktionen begegnen. Gemeint ist Sinus, Cosinus und Tangens.
- Frage 3: Wenn einer Variablen ein Wert zugewiesen wird, dessen Typ sich von dem der Variablen unterscheidet, dann werden entweder von VBA die Typen automatisch ineinander umgewandelt. (Das haben wir gesehen, wenn wir einer Variablen vom Typ `Integer` den Wert einer `InputBox`-Funktion zugewiesen haben. Bei der leeren Eingabe blieb das Programm stehen.) Oder wir müssen die Umwandlung selbst in die Hand nehmen, wenn die automatische Umwandlung Ärger macht. Mit der Funktion `Val()` haben wir veranlasst, dass aus der leeren Eingabe die 0 wird.

Kapitel 8, Aufgabe 1

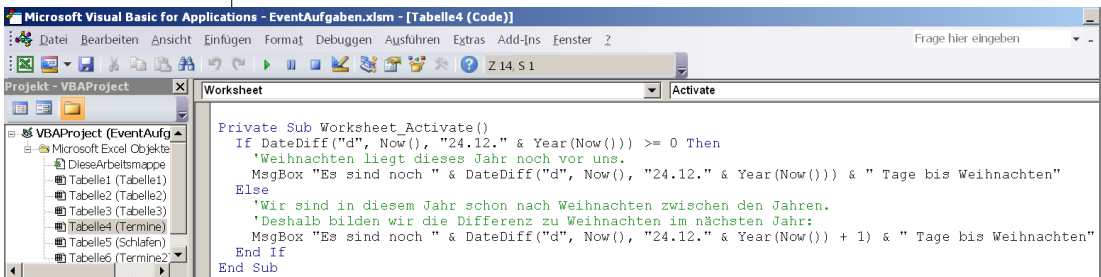
```
Private Sub Worksheet_Activate()  
    If DateDiff("d", Now(), "24.12." & Year(Now())) >= 0 Then  
        'Weihnachten liegt dieses Jahr noch vor uns.  
        MsgBox "Es sind noch " & DateDiff("d", Now(), #  
            "24.12." & Year(Now())) & " Tage bis Weihnachten"
```

B



```
Else
'Wir sind in diesem Jahr schon nach Weihnachten ☹
zwischen den Jahren.
'Deshalb bilden wir die Differenz zu Weihnachten im ☹
nächsten Jahr:
MsgBox "Es sind noch " & DateDiff("d", Now(), ☹
"24.12." & Year(Now()) + 1) & " Tage bis Weihnachten"
End If
End Sub
```

Interessant an diesem Beispiel ist, dass die umbenannte Tabelle im Projekt-Explorer noch als Tabelle4 und nur in Klammern als Termine aufgeführt ist.



Grob gesagt bilden wir die Differenz mit der Datediff-Funktion zwischen dem heutigen Datum (Now()-Funktion) und dem 24.12. von diesem Jahr (Year(Now())). Wenn die Differenz positiv ist, dann liegt Weihnachten noch vor uns; wenn die Differenz negativ ist, dann müssen wir noch fast ein Jahr warten, weil wir am Jahresende nach Weihnachten sind. Deshalb müssen wir zu diesem Jahr noch ein Jahr hinzuaddieren (Year(Now())+1).

Kapitel 8, Aufgabe 2

```
Private Sub Worksheet_Activate()
If Hour(Now()) >= 13 And Hour(Now()) <= 14 Then
MsgBox "Zeit für den Mittagsschlaf"
End If
End Sub
```

Auch hier verweisen wir wieder auf das Kapitel über Funktionen auf Seite 160 für die Datums- und Uhrzeitfunktionen.

Außerdem verweisen wir noch auf das Kapitel mit den logischen Operatoren, in denen And erklärt wird (siehe Seite 139).

Hour(Now()) findet die Stunde des heutigen Tages heraus. Now() liefert uns außer dem Datum auch noch die Uhrzeit.



Wenn nun besagte Stundenzahl zwischen 13 und 14 Uhr liegt, dann sollte unser Kurt eine wohlverdiente Pause einlegen.

Kapitel 8, Aufgabe 3

In dieser Aufgabe soll beim Starten des Rechners herausgefunden werden, ob Kurt Geburtstag hat; anschließend gratuliert der Rechner ihm höflich.

Programmiere einen Event, der mit der `Day()`- und `Month()`-Funktion genau das herausfindet. Du kannst ein Tabellenblatt `Termine2` dazu einfügen, indem du den Event `Activate` programmierst.

```
Private Sub Worksheet_Activate()  
    If Month(Now()) = 4 And Day(Now()) = 1 Then  
        MsgBox "Lieber Kurt, herzlichen Glückwunsch zu ☺  
        deinem heutigen Geburtstag!"  
    End If  
End Sub
```

Eine sinnvolle Anwendung im Geschäftsbereich wäre es, auf wichtige Fristen und Termine auf diese Weise hinzuweisen.

Kapitel 8, Aufgabe 4

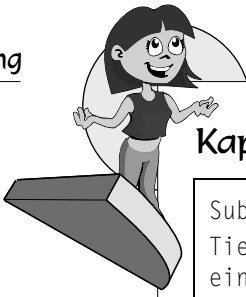
Programmiere folgende Zeilen aus dem Programm `Wuerfel.xlsm`:

```
Sub Wuerfel()  
    MsgBox CInt(Rnd() * 6 + 1)  
End Sub
```

Antworten Kapitel 9

- Frage 1: Die `MsgBox` ... liefert keinen Wert zurück, wenn sie am Bildschirm etwas ausgibt. Die `MsgBox(...)` liefert einen Wert zurück, welche Schaltfläche vom Anwender gedrückt wurde.
- Frage 2: Der ASCII-Code ist der »American Standard Code for Information Interchange«, mit dem international Zeichen mit einer Nummer versehen sind.

B



Kapitel 9, Aufgabe 1

```
Sub Tiernamprogramm()  
Tiernamen = InputBox("Bitte geben Sie einen Tiernamen ein")  
MsgBox "Hallo, wie geht es " & Tiernamen & "?"  
End Sub
```

Ausprobieren kannst du die Lösung in der Excel-Mappe *Aufgaben.xlsm*.

Antworten Kapitel 10

Frage 1: Ein reserviertes Wort wird von der Programmiersprache selbst verwendet. Du darfst dieses Wort für nichts anderes gebrauchen. In VBA werden solche Wörter blau markiert.

Frage 2: Denk an die Löcher im Zahlenbereich von Gleitpunktzahlen. Liegt der tatsächliche Wert, mit dem du vergleichen möchtest, auf einem Loch, dann vergleichst du nur den daneben liegenden Wert, was in der Regel falsch ist.

Frage 3: Das Programm bricht mit einer Fehlermeldung ab.

Kapitel 10, Aufgabe 1

Nun aber los, unsere Variable für die Postleitzahl definieren wir als PLZ vom Typ `String`.

Dies rührt daher, dass wir z. B. bei `Integer` nach 32767 keine Zahlen mehr zur Verfügung haben. Erinnerst du dich? 32768 wäre ein Überlauf. Die Postleitzahl 54321 ist z. B. mit `Integer` nicht darstellbar. Wir müssten auf den Typ `Long` ausweichen.

Dann haben wir sowohl bei Postleitzahlen als auch bei Telefonnummern das Problem mit führenden Nullen. Würden wir die Postleitzahl 01234 als eine Zahl wie z. B. `Integer` oder `Long` vereinbaren, würde sie zu einer fehlerhaften, weil vierstelligen Postleitzahl (1234) umgewandelt werden.

Solange wir weder sortieren noch rechnen, können wir getrost die Ziffern als Zeichenfolge (nämlich als den Typ `String`) abspeichern. Bei dem Typ `String` haben wir diese Probleme vom Tisch, denn die Ziffern werden als Zeichenfolge interpretiert und nicht als Zahl.

Für den Typ `String` können wir die Länge der eingegebenen Postleitzahl mit der `Len`-Funktion herausfinden. `Len(PLZ)` muss gleich 5 sein, sonst ist es ein Fehler. Diese Funktion `Len`, wie auch noch die folgenden Funktionen,



haben wir auf Seite 162 (in dem Kapitel über Funktionen) bereits besprochen. Am besten, du siehst dir einmal das unten angegebene Programm `Plausibel1` an und verfolgst den hier stehenden Text mit den Anweisungen.

Die Variable `Fehler` ist ein sogenanntes *Flag* (eine Flagge), das wir setzen oder hissen, wenn ein bestimmtes Ereignis eingetreten ist. Wir gehen erst einmal optimistisch davon aus, dass die eingegebene Postleitzahl in Ordnung sein wird, und setzen die Flagge `Fehler` (eine boolesche Variable) auf `False`. Dann werden der Reihe nach sechs Kriterien (genau fünf Zeichen und jedes ist eine Ziffer) überprüft, wobei jedes Kriterium dazu führen kann, dass `Fehler` auf `True` gesetzt (und damit die Flagge gehisst) wird, wenn der Test nicht erfolgreich bestanden wurde. Eine Postleitzahl, die »ohne Schrammen durchkommt«, hinterlässt eine Variable `Fehler` mit dem Wert `False`. Sobald mindestens ein `Fehler` aufgetreten ist, ist die Flagge gehisst worden und `Fehler` ist `True`.

Bleibt noch herauszufinden, ob ein Zeichen eine Ziffer ist. Dazu bemühen wir den sogenannten ASCII-Code im Anhang A auf Seite 261.

Jedes Zeichen der Tastatur hat dort eine Nummer. Zum Beispiel hat das Zeichen `A` den ASCII-Code 65. Analog hat die Null `0` den ASCII-Code 48, die `1` den ASCII-Code 49 usw. bis `9`, die den ASCII-Code 57 hat.

Wenn wir nun wissen möchten, ob ein Zeichen eine Ziffer ist, müssen wir herausfinden, ob der ASCII-Code dieses Zeichens zwischen 48 und 57 liegt. Wenn ja, ist es eine Ziffer, sonst nicht.

Allerdings ist unsere Postleitzahl fünfstellig. Damit haben wir auch fünf Zeichen. Mit dem `Mid`-Befehl holen wir das jeweilige Zeichen aus der Zeichenfolge heraus.

Zur Erinnerung bei der `Mid`-Funktion:

- ❖ Nach der öffnenden Klammer der `Mid`-Funktion steht die Zeichenfolge, aus der Zeichen entnommen werden sollen. Das ist unsere Variable `PLZ`.
- ❖ Danach steht die Position, ab der Zeichen entnommen werden sollen. In dem Programm sind das der Reihe nach die Positionen 1 bis 5.
- ❖ Danach steht die Anzahl der zu entnehmenden Zeichen. Da wir immer nur ein Zeichen überprüfen, steht da während des ganzen Programms die `1`.

Und nun formulieren wir mit besagter `Mid`-Funktion, ob die Zeichenfolge im String `PLZ` ab der ersten Stelle der Länge 1 (`Mid(PLZ, 1, 1)`) eine Ziffer ist:

```
If Asc(Mid(PLZ, 1, 1)) < 48 Or Asc(Mid(PLZ, 1, 1)) > 57
```

B



Diese oben beschriebene Programmzeile erledigt das für die erste Ziffer erkennbar an der Mid-Funktion.

Für die zweite Ziffer lautet die Funktion dann (Mid(PLZ, 2, 1)).

```
Sub Plausibel1()
    Dim PLZ As String
    Dim Fehler As Boolean

    PLZ = InputBox("Bitte PLZ eingeben:")
    Fehler = False
    If Len(PLZ) <> 5 Then
        MsgBox "Fehler, PLZ besteht nicht aus genau 5 Zeichen"
        Fehler = True
    Else
        If Asc(Mid(PLZ, 1, 1)) < 48 Or Asc(Mid(PLZ, 1, 1)) > 57 Then
            MsgBox "Fehler in erstem Zeichen"
            Fehler = True
        End If
        If Asc(Mid(PLZ, 2, 1)) < 48 Or Asc(Mid(PLZ, 2, 1)) > 57 Then
            MsgBox "Fehler in zweitem Zeichen"
            Fehler = True
        End If
        If Asc(Mid(PLZ, 3, 1)) < 48 Or Asc(Mid(PLZ, 3, 1)) > 57 Then
            MsgBox "Fehler in drittem Zeichen"
            Fehler = True
        End If
        If Asc(Mid(PLZ, 4, 1)) < 48 Or Asc(Mid(PLZ, 4, 1)) > 57 Then
            MsgBox "Fehler in viertem Zeichen"
            Fehler = True
        End If
        If Asc(Mid(PLZ, 5, 1)) < 48 Or Asc(Mid(PLZ, 5, 1)) > 57 Then
            MsgBox "Fehler in fünftem Zeichen"
            Fehler = True
        End If
    End If
    If Fehler = False Then
        MsgBox "Die Postleitzahl ist OK."
    Else
        MsgBox "Die Postleitzahl ist falsch."
    End If
End Sub
```



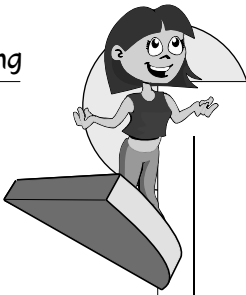

Jetzt überlegen wir einmal weiter: Warum sollen wir denn, wenn wir im ersten Zeichen schon eine Nichtziffer gefunden haben, überhaupt noch die restlichen Zeichen überprüfen? Eine Nichtziffer reicht schon als K.o.-Kriterium aus.

Deshalb schachteln wir unser Programm nun so, dass wir nur so lange weiterprüfen, wie die Postleitzahl korrekt gebildet ist.

Unser Flag `Fehler` wird umgekehrt gesetzt. Wir gehen diesmal davon aus, dass wir eine fehlerhafte Postleitzahl vor uns haben und nur wenn wir in das Innerste unserer Verschachtelung vorgedrungen sind, setzen wir unser Flag `Fehler` auf `False`. Sollten wir aufgrund eines Fehlers vorher aussteigen, bleibt es bei der Annahme, dass die Postleitzahl falsch war.

```
Sub Plausibel2()  
    Dim PLZ As String  
    Dim Fehler As Boolean  
  
    Fehler = True  
    PLZ = InputBox("Bitte PLZ eingeben:")  
    If Len(PLZ) <> 5 Then  
        MsgBox "Fehler, PLZ besteht nicht aus genau 5 Zeichen"  
    Else  
        If Asc(Mid(PLZ, 1, 1)) < 48 Or Asc(Mid(PLZ, 1, 1)) > 57 Then  
            MsgBox "Fehler in erstem Zeichen"  
        Else  
            If Asc(Mid(PLZ, 2, 1)) < 48 Or Asc(Mid(PLZ, 2, 1)) > 57 Then  
                MsgBox "Fehler in zweitem Zeichen"  
            Else  
                If Asc(Mid(PLZ, 3, 1)) < 48 Or Asc(Mid(PLZ, 3, 1)) > 57 Then  
                    MsgBox "Fehler in drittem Zeichen"  
                Else  
                    If Asc(Mid(PLZ, 4, 1)) < 48 Or Asc(Mid(PLZ, 4, 1)) > 57 Then  
                        MsgBox "Fehler in viertem Zeichen"  
                    Else  
                        If Asc(Mid(PLZ, 5, 1)) < 48 Or Asc(Mid(PLZ, 5, 1)) > 57 Then  
                            MsgBox "Fehler in fünftem Zeichen"  
                        Else  
                            MsgBox "OK"  
                            Fehler = False  
                        End If  
                    End If  
                End If  
            End If  
        End If  
    End If  
End Sub
```

B



```

        End If
    End If
End If
End If
End If
End If
If Fehler = False Then
    MsgBox "Die Postleitzahl ist OK."
Else
    MsgBox "Die Postleitzahl ist falsch."
End If

End Sub

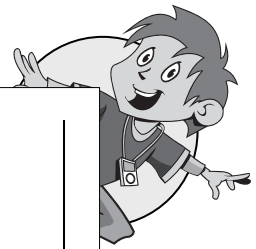
```

Kapitel 10, Aufgabe 2

```

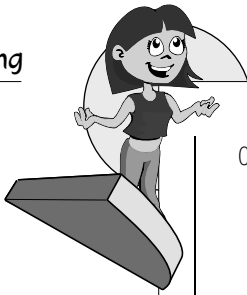
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    Select Case Target.Column 'Welche Spalte wurde getroffen?
    Case 6 'F
        Select Case Target.Row
        Case 11
            MsgBox "5 Punkte" 'Zelle F,11 ist blau, 5 Punkte
        Case 12
            MsgBox "5 Punkte" 'Zelle F,12 ist blau, 5 Punkte
        Case 13
            MsgBox "5 Punkte" 'Zelle F,13 ist blau, 5 Punkte
        End Select
    Case 7 'G
        Select Case Target.Row
        Case 10
            MsgBox "5 Punkte" 'Zelle G,10 ist blau, 5 Punkte
        Case 11
            MsgBox "5 Punkte" 'Zelle G,11 ist blau, 5 Punkte
        Case 12
            MsgBox "5 Punkte" 'Zelle G,12 ist blau, 5 Punkte
        Case 13
            MsgBox "5 Punkte" 'Zelle G,13 ist blau, 5 Punkte
        Case 14
            MsgBox "5 Punkte" 'Zelle G,14 ist blau, 5 Punkte
        Case Else
        End Select
    End Select

```

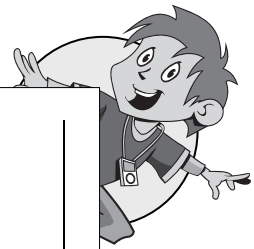


```
Case 8 'H
    Select Case Target.Row
    Case 9
        MsgBox "5 Punkte" 'Zelle H,9 ist blau, 5 Punkte
    Case 10
        MsgBox "5 Punkte" 'Zelle H,10 ist blau, 5 Punkte
    Case 11
        MsgBox "10 Punkte" 'Zelle H,11 ist gelb, 10 Punkte
    Case 12
        MsgBox "10 Punkte" 'Zelle H,12 ist gelb, 10 Punkte
    Case 13
        MsgBox "5 Punkte" 'Zelle H,13 ist gelb, 10 Punkte
    Case 14
        MsgBox "5 Punkte" 'Zelle H,14 ist blau, 5 Punkte
    Case 15
        MsgBox "5 Punkte" 'Zelle H,15 ist blau, 5 Punkte
    Case Else
    End Select
Case 9 'I
    Select Case Target.Row
    Case 8
        MsgBox "5 Punkte" 'Zelle I,8 ist blau, 5 Punkte
    Case 9
        MsgBox "5 Punkte" 'Zelle I,9 ist blau, 5 Punkte
    Case 10
        MsgBox "10 Punkte" 'Zelle I,10 ist gelb, 10 Punkte
    Case 11
        MsgBox "10 Punkte" 'Zelle I,11 ist gelb, 10 Punkte
    Case 12
        MsgBox "15 Punkte" 'Zelle I,12 ist grün, 15 Punkte
    Case 13
        MsgBox "10 Punkte" 'Zelle I,13 ist gelb, 10 Punkte
    Case 14
        MsgBox "10 Punkte" 'Zelle I,14 ist gelb, 10 Punkte
    Case 15
        MsgBox "5 Punkte" 'Zelle I,15 ist blau, 5 Punkte
    Case 16
        MsgBox "5 Punkte" 'Zelle I,16 ist blau, 5 Punkte
    Case Else
    End Select
```

B



```
Case 10 'J
  Select Case Target.Row
    Case 8
      MsgBox "5 Punkte" 'Zelle J,8 ist blau, 5 Punkte
    Case 9
      MsgBox "5 Punkte" 'Zelle J,9 ist blau, 5 Punkte
    Case 10
      MsgBox "10 Punkte" 'Zelle J,10 ist gelb, 10 Punkte
    Case 11
      MsgBox "15 Punkte" 'Zelle J,11 ist grün, 15 Punkte
    Case 12
      MsgBox "20 Punkte" 'Zelle J,12 ist schwarz, 20 Punkte
    Case 13
      MsgBox "15 Punkte" 'Zelle J,13 ist grün, 15 Punkte
    Case 14
      MsgBox "10 Punkte" 'Zelle J,14 ist gelb, 10 Punkte
    Case 15
      MsgBox "5 Punkte" 'Zelle J,15 ist blau, 5 Punkte
    Case 16
      MsgBox "5 Punkte" 'Zelle J,16 ist blau, 5 Punkte
    Case Else
  End Select
Case 11 'K
  Select Case Target.Row
    Case 8
      MsgBox "5 Punkte" 'Zelle K,8 ist blau, 5 Punkte
    Case 9
      MsgBox "5 Punkte" 'Zelle K,9 ist blau, 5 Punkte
    Case 10
      MsgBox "10 Punkte" 'Zelle K,10 ist gelb, 10 Punkte
    Case 11
      MsgBox "10 Punkte" 'Zelle K,11 ist gelb, 10 Punkte
    Case 12
      MsgBox "15 Punkte" 'Zelle K,12 ist grün, 15 Punkte
    Case 13
      MsgBox "10 Punkte" 'Zelle K,13 ist gelb, 10 Punkte
    Case 14
      MsgBox "10 Punkte" 'Zelle K,14 ist gelb, 10 Punkte
    Case 15
      MsgBox "5 Punkte" 'Zelle K,15 ist blau, 5 Punkte
```



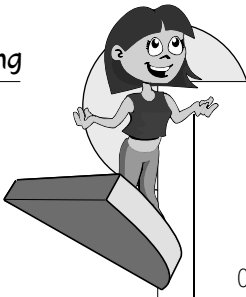
```
Case 16
    MsgBox "5 Punkte" 'Zelle K,16 ist blau, 5 Punkte
Case Else
End Select

Case 12 'L
    Select Case Target.Row
    Case 9
        MsgBox "5 Punkte" 'Zelle L,9 ist blau, 5 Punkte
    Case 10
        MsgBox "5 Punkte" 'Zelle L,10 ist blau, 5 Punkte
    Case 11
        MsgBox "10 Punkte" 'Zelle L,11 ist gelb, 10 Punkte
    Case 12
        MsgBox "10 Punkte" 'Zelle L,12 ist gelb, 10 Punkte
    Case 13
        MsgBox "10 Punkte" 'Zelle L,13 ist gelb, 10 Punkte
    Case 14
        MsgBox "5 Punkte" 'Zelle L,14 ist blau, 5 Punkte
    Case 15
        MsgBox "5 Punkte" 'Zelle L,15 ist blau, 5 Punkte
    Case Else
    End Select

Case 13 'M
    Select Case Target.Row
    Case 10
        MsgBox "5 Punkte" 'Zelle M,10 ist blau, 5 Punkte
    Case 11
        MsgBox "5 Punkte" 'Zelle M,11 ist blau, 5 Punkte
    Case 12
        MsgBox "5 Punkte" 'Zelle M,12 ist blau, 5 Punkte
    Case 13
        MsgBox "5 Punkte" 'Zelle M,13 ist blau, 5 Punkte
    Case 14
        MsgBox "5 Punkte" 'Zelle M,14 ist blau, 5 Punkte
    Case Else
    End Select

Case 14 'N
    Select Case Target.Row
    Case 11
        MsgBox "5 Punkte" 'Zelle N,11 ist blau, 5 Punkte
    Case 12
        MsgBox "5 Punkte" 'Zelle N,12 ist blau, 5 Punkte
```

B



```
Case 13
    MsgBox "5 Punkte" 'Zelle N,13 ist blau, 5 Punkte
Case Else
End Select
Case Else
End Select
End Sub
```

Antworten Kapitel 11

Frage 1: Die kopfgesteuerte Schleife entscheidet vor dem ersten Durchlauf, ob die Schleife überhaupt durchlaufen werden soll. Die fußgesteuerte Schleife lässt einen Durchlauf zu und entscheidet danach, ob mehrere Schleifendurchläufe daraus werden.

Frage 2: Die For-Schleife scheitert an komplexen Bedingungen. Die For-Schleife kann als einfache Bedingung nur testen, ob ein bestimmter Wert von der Laufvariablen überschritten wird.

Kapitel 11, Aufgabe 1

```
Sub Regenbogenb()
Dim Zeile As Integer
For Zeile = 1 To 56
    Cells(Zeile, 1).Interior.ColorIndex = Zeile
Next Zeile
End Sub
```

	A
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

Ausprobieren kannst du die Lösung in der Excel-Mappe Regenbogen.xlsm.



Kapitel 11, Aufgabe 2

Mit einer For ... Next-Schleife und drei Messageboxen könnten wir einen solchen »Kampf« zehn Runden lang simulieren.

```
Sub Boxkampf()  
Dim i As Integer  
For i = 1 To 10  
    MsgBox "Gong zum Anfang der Runde " & i  
    MsgBox "Schiedsrichter passt auf "  
    MsgBox "Gong zum Ende der Runde " & i  
Next i  
End Sub
```

Allerdings musst du beachten, dass der Boxkampf auch vorzeitig beendet werden könnte. Unsere For ... Next-Schleife ist dafür zu unflexibel, es sei denn, wir springen beim Abbruch des Kampfes auch mit EXIT FOR aus der Schleife heraus, was wir hier nicht machen möchten, weil dieser Programmierstil wenig elegant ist.

Eine andere Form der Schleife ist die While-Schleife, die flexibler entscheidet, wie lange »geschleift« wird. Das zugehörige Programm findest du unter AufgabenKap11.xlsm.

```
Sub BoxkampfMitAbbruch()  
Dim i As Integer  
Dim abgebrochen As Boolean  
Dim Eingabe As String  
  
abgebrochen = False  
i = 1  
While i <= 10 And abgebrochen = False  
    MsgBox "Gong zum Anfang der Runde " & i  
    MsgBox "Schiedsrichter passt auf "  
  
    Eingabe = InputBox("KO, Handtuch oder nichts eingeben")  
    If Eingabe = "KO" Or Eingabe = "Handtuch" Then  
        abgebrochen = True  
    MsgBox "Gong zum Ende der Runde " & i  
    i = i + 1  
Wend  
End Sub
```

B



Kapitel 11, Aufgabe 3

Die Lösung findest du auch in der gleichnamigen Mappe Notendurchschnitt.xlsm.

Mit einer Schleife summierst du `Note*Anzahl` Schüler mit dieser Note auf und merkst dir das Ergebnis in der Variablen `Zaehler`. In der zweiten Variablen `Nenner` merkst du dir die Anzahl der Schüler (ebenfalls mit einer Schleife).

Das Resultat erhältst du wie üblich durch eine Division.

```
Sub Notendurchschnitt3()
Dim i As Integer 'unsere Laufvariable
Dim Zaehler As Integer
Dim Nenner As Integer

Zaehler = 0
For i = 1 To 6
    Zaehler = Zaehler + Cells(2, i) * Cells(3, i)
Next i

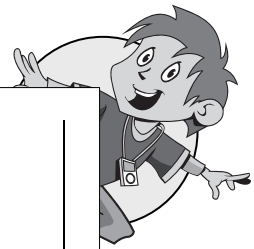
Nenner = 0
For i = 1 To 6
    Nenner = Nenner + Cells(3, i)
Next i

If Nenner > 0 Then
    Durchschnitt = Zaehler / Nenner
    Cells(2, 7) = "Durchschnitt"
    Cells(3, 7) = Durchschnitt
End If
End Sub
```

Eleganter wird das Programm, wenn du nur eine Schleife benutzt:

```
Sub Notendurchschnitt4()
Dim i As Integer 'unsere Laufvariable
Dim Zaehler As Integer
Dim Nenner As Integer

Zaehler = 0
Nenner = 0
For i = 1 To 6
```

```
Zaehler = Zaehler + Cells(2, i) * Cells(3, i)
Nenner = Nenner + Cells(3, i)
Next i

If Nenner > 0 Then
    Durchschnitt = Zaehler / Nenner
    Cells(2, 7) = "Durchschnitt"
    Cells(3, 7) = Durchschnitt
End If
End Sub
```

Kapitel 11, Aufgabe 4

Ausprobieren kannst du die Programme in der Mappe
Postleitzahlen.xlsm.

In der ersten Version mit der For-Schleife läuft das Programm alle Stellen ab, auch wenn bereits eine fehlerhafte Ziffer am Anfang aufgetreten ist. Im Schleifenrumpf wird pro Zeichen im Prinzip das Gleiche abgeprüft. Es ändert sich nur die Laufvariable und damit die Stelle in der Zeichenfolge.

```
Sub Plausibel3()
    Dim PLZ As String
    Dim Fehler As Boolean
    Dim i As Integer

    PLZ = InputBox("Bitte PLZ eingeben:")
    Fehler = False
    If Len(PLZ) <> 5 Then
        MsgBox "Fehler, PLZ besteht nicht aus genau 5 Zeichen"
        Fehler = True
    Else
        For i = 1 To 5
            If Asc(Mid(PLZ, i, 1)) < 48 Or Asc(Mid(PLZ, i, 1)) > 57 Then
                MsgBox "Fehler, an " & i & ". Zeichen"
                Fehler = True
            End If
        Next i
    End If
    If Fehler = False Then MsgBox "OK"
End Sub
```

B



Auch hier stellt sich wie bei den bedingten Anweisungen die Frage, warum wir überprüfen sollen, ob das dritte Zeichen der Postleitzahl eine Ziffer ist, wenn schon das erste Zeichen keine Ziffer war? Wir ändern den Schleifentyp um in eine While-Schleife sowie das Abbruchkriterium.

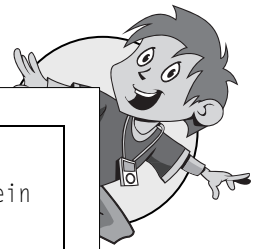
Damit kann die While-Schleife abbrechen, sobald der erste Fehler aufgetreten ist.

```
Sub Plausibel4()
    Dim PLZ As String
    Dim Fehler As Boolean
    Dim i As Integer

    PLZ = InputBox("Bitte PLZ eingeben:")
    Fehler = False
    If Len(PLZ) <> 5 Then
        MsgBox "Fehler, PLZ besteht nicht aus genau 5 Zeichen"
        Fehler = True
    Else
        i = 1
        While (i <= 5) And (Fehler = False)
            If Asc(Mid(PLZ, i, 1)) < 48 Or Asc(Mid(PLZ, i, 1)) > 57 Then
                MsgBox "Fehler, an " & i & ". Zeichen"
                Fehler = True
            End If
            i = i + 1
        Wend
    End If
    If Fehler = False Then MsgBox "OK"
End Sub
```

Kapitel 11, Aufgabe 5

Die in dieser Aufgabe vorgestellten Programme kannst du in der Mappe Bienen aussehen.xlsm testen, doch ist es besser, wenn du zuerst einmal versuchst, dieses Problem selbst zu lösen. Mitunter ist es hilfreich, auf einem Blatt Papier mit Stift (ja, so was gibt es noch) aufzuschreiben, in welcher Reihenfolge welche Zellen verarbeitet werden sollen. Wenn du Klarheit über die Indizes hast, kannst du es programmieren.



```
Sub Bienenaussehen()  
Dim Zeile As Long 'Es könnten auch erheblich mehr Zeilen sein  
  
Zeile = 1  
While Cells(Zeile, 1) <> ""  
    If Zeile Mod 2 = 0 Then  
        Cells(Zeile, 1).Interior.ColorIndex = 1 'schwarzer  
        Hintergrund  
        Cells(Zeile, 1).Font.ColorIndex = 6 'gelbe Farbe  
        Hintergrund  
    Else  
        Cells(Zeile, 1).Interior.ColorIndex = 6 'gelber  
        Hintergrund  
        Cells(Zeile, 1).Font.ColorIndex = 1 'schwarze Farbe  
        Hintergrund  
    End If  
    Zeile = Zeile + 1  
Wend  
End Sub
```

Alternativ kannst du auch mit dem Range-Objekt diese Formatierung vornehmen. Dazu das folgende Beispiel:

```
Sub RangeBienenaussehen()  
Dim Zeile As Long 'Es könnten auch erheblich mehr Zeilen sein  
  
Zeile = 1  
While Cells(Zeile, 1) <> ""  
    If Zeile Mod 2 = 0 Then  
        Range("A" & Zeile & ":A" & Zeile).Interior.ColorIndex  
        = 1 'schwarzer Hintergrund  
        Range("A" & Zeile & ":A" & Zeile).Font.ColorIndex  
        = 6 'gelbe Farbe Hintergrund  
    Else  
        Range("A" & Zeile & ":A" & Zeile).Interior.ColorIndex  
        = 6 'gelber Hintergrund  
        Range("A" & Zeile & ":A" & Zeile).Font.ColorIndex  
        = 1 'schwarze Farbe Hintergrund  
    End If  
    Zeile = Zeile + 1  
Wend  
End Sub
```

B



In diesem Beispiel bringt das `Range`-Objekt keinen Vorteil gegenüber dem `Cells`-Objekt.

Vorteile bietet ein `Range`-Objekt gegenüber dem `Cells`-Objekt, wenn ein ganzer Block formatiert werden soll. Wenn unser Kurt seinen Bienenlook noch um die Spalten 2 und 3 erweitern sollte. Deshalb solltest du dir darüber ein paar Gedanken machen.

Eine Zeile wie beispielsweise

```
Range("A" & Zeile & ":C" & Zeile).Formatangabe = Wert
```

würde diese Aufgabe eleganter lösen als die `Cells`-Objekte.

Nun greifen wir eine Problematik auf, die zum Glück in den neueren Excel-Versionen beseitigt wurde. Trotzdem solltest du wissen, zu welchen Problemen es kommen kann:

Das, was in den Klammern des `Range`-Objekts steht (Parameter oder Argumente), ist die Angabe des Blocks, mit dem etwas passiert.

Die Argumente werden mit `&`-Zeichen verklebt. Das nennen wir zusammenbauen.


In früheren Excel-VBA-Versionen gab es Probleme mit diesem Zusammenbau, weil das nicht vorhandene Vorzeichen als Leerzeichen mit eingeklebt wurde.

Genauer: Wenn in der Variablen `Zeile` die Zahl 1 steht, dann steht darin genau genommen die +1. Statt der +1 wird das + Zeichen weggelassen und stattdessen ein Leerzeichen vor die 1 gestellt.

Statt `A 1:B 1` erwartete VBA diese Zeichenfolge ohne Leerzeichen (`A1:B1`), sodass es in früheren Excel-Versionen zu einer Fehlermeldung kam. Solltest du noch mit einer der älteren Excel-Versionen arbeiten, dann kannst du mithilfe der `Trim`-Funktion, die wir bereits kennengelernt haben, bei den Funktionen für Zeichenfolgen die Leerzeichen entfernen. Die Funktion `Str(Zeile)` (ebenfalls eine Funktion für Zeichenfolgen) wandelt die Variable `Zeile` in eine Zeichenfolge um. Dadurch kann zu Beginn der Zeichenfolge ein Leerzeichen stehen, dass mit der `Trim`-Funktion beseitigt wird. Nähere Angaben zu den beiden Funktionen `Str` und `Trim` findest du in dem Kapitel über Funktionen auf Seite 162.

Das unten angegebene Programm `Zusammenbauen` erledigt dasselbe wie in unserem `Bienenaussehen`-Programm die Zeilen der `Range`-Objekte.



```
Sub Zusammenbauen()  
Dim Zeile As Long  
Zeile = 1  
Range("A" & Trim(Str(Zeile)) & ":A" &   
Trim(Str(Zeile))).Interior.ColorIndex = 6  
End Sub
```

Kapitel 11, Aufgabe 6

Das folgende Programm gibt die errechneten 15 Pizzen in der zweiten Spalte aus:

```
Sub Summenformel()  
Dim Zeile As Long  
  
Zeile = 2  
While Cells(Zeile, 1) <> ""  
    Zeile = Zeile + 1  
Wend  
If Zeile > 2 Then 'Mehr als die Kopfzeile  
    Cells(Zeile, 1) = "Summe"  
    Cells(Zeile, 2).Formula = "=SUM(A2:A" & Zeile & ")"  
End If  
End Sub
```

Das Programm findest du in der Mappe Kolonne.xlsm.

Kapitel 11, Aufgabe 7

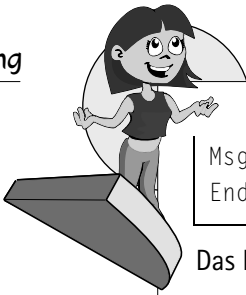
Die in dieser Aufgabe vorgestellten Programme findest du in der Mappe Apfelwaage.xlsm.

Dann können wir eine Waage mit folgendem Programm simulieren:

Die Summe entspricht dabei der Variablen Waagschale, die der Reihe nach die Äpfel aufnimmt.

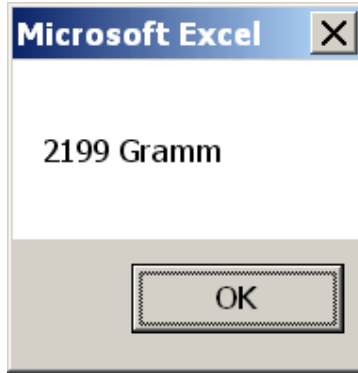
```
Sub Apfelwaage1()  
Dim Waagschale As Double  
Dim Zeile As Integer  
For Zeile = 2 To 16 'Bei 15 Äpfeln  
    Waagschale = Waagschale + Cells(Zeile, 2)  
Next Zeile
```

B



```
MsgBox Waagschale & " Gramm"
End Sub
```

Das Ergebnis ist die MessageBox-Ausgabe:



Wir hätten das Problem auch mit der Excel-Summenfunktion lösen können, wie in der vorherigen Aufgabe, als wir die Anzahl Pizzen errechnet haben.

Hier sollst du lernen, dass es eleganter ist, Programme ein Problem komplett lösen zu lassen, ohne dass noch Zwischenrechnungen an Excel übergeben werden, um damit dann auch noch etwas zu rechnen.

Unsere Waagschalenproblematik formulieren wir in eine `While`-Schleife um:

```
Sub Apfelwaage2()
Dim Waagschale As Double
Dim Zeile As Integer
Zeile = 2
While Zeile <= 16 'Bei 15 Äpfeln
    Waagschale = Waagschale + Cells(Zeile, 2)
    Zeile = Zeile + 1
Wend
MsgBox Waagschale & " Gramm"
End Sub
```

Das Ergebnis muss dasselbe sein wie bei der `For`-Schleife. Wir beginnen wieder in `Zeile = 2` und lassen die Schleife laufen, bis `Zeile = 11` gilt. Danach ist Schleifenende.

Das Weiterschalten der Schleife müssen wir leider selbst übernehmen mit der Zeile `Zeile = Zeile + 1`.



Zu deiner Aufgabe gehört es nun, eine Trace-Tabelle von den Variablen `Zeile` und `Waagschale` anzufertigen.

Zeile	Waagschale
2	122
3	275
4	422
5	534
6	651
7	796
8	952
9	1119
10	1274
11	1427
12	1575
13	1714
14	1878
15	2048
16	2199

Kapitel 11, Aufgabe 8

Die in dieser Aufgabe vorgestellten Programme findest du in Mappe `Grossschreibung.xlsm`.

Vorgehensweise:

Wenn wir die Anzahl an Zeilen bestimmt haben, können wir eine `For-Schleife` programmieren. Dazu stellen wir mittels `Aufzeichnen` fest, wo sich die letzte Zeile befindet.

Die maximale Spalte bekommen wir heraus, indem wir ein Makro aufzeichnen, das uns die Tastenkombination von `Strg` + `Ende` in Excel protokolliert. Von dem selektierten Bereich nehmen wir dann die Eigenschaft `.Row` und speichern das Ergebnis als `AnzahlZeilen` ab.

B



Die zugehörige Programmzeile sieht dann so aus:

```
Dim AnzahlZeilen As Long
ActiveCell.SpecialCells(xlLastCell).Select
AnzahlZeilen = Selection.Row
```

Um die Zeilen und zwei Spalten abzulaufen, benötigst du zwei ineinander geschachtelte Schleifen. In unserem Beispiel läuft die äußere Schleife die Zeilen und die innere Schleife die beiden Spalten ab.

Die Zelle `Cells(Zeile, Spalte)` enthält den jeweiligen Namen.

Für jeden Namen muss getestet werden, ob das erste Zeichen ein Großbuchstabe ist.

Umgekehrt, wenn das erste Zeichen kein Großbuchstabe ist, müssen wir handeln.

Feststellen können wir das mit den Zeilen:

```
If Cells(Zeile, Spalte) <> "" Then
    If Not ((Asc(Mid(Cells(Zeile, Spalte), 1, 1)) >= 65)_
        And (Asc(Mid(Cells(Zeile, Spalte), 1, 1))) <= 90_
        ) Then
        'Kein Großbuchstabe!
    End If
End If
```

Abprüfen, ob der ASCII-Wert des ersten Zeichens zwischen 65 (entspricht A) und 90 (entspricht Z) liegt.

Prinzip Beispiel: Zeile=5, Spalte = 1:

❖ `Asc(Mid(Cells(Zeile, 1), 1, 1))`

❖ `Asc(Mid("gärtner", 1, 1))`

❖ `Asc("g")` ergibt 103

103 liegt nicht zwischen 65 und 90, sodass `Cells(5,1)` korrigiert werden muss.

Das komplette Programm lautet:



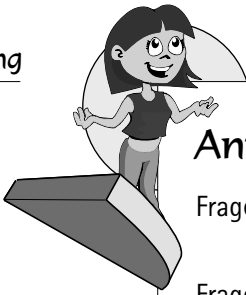
```

Sub Grossschreibung()
    Dim Zeile As Long 'Zeile läuft Zeilen ab
    Dim AnzahlZeilen As Long
    Dim Spalte As Integer 'Wir haben zwei Spalten auf ↗
    dieselbe Weise zu beackern.

    ActiveCell.SpecialCells(xlLastCell).Select
    AnzahlZeilen = Selection.Row
    For Zeile = 2 To AnzahlZeilen
        For Spalte = 1 To 2 'Erst die Nachnamenspalte ↗
            verbessern, dann die Vornamenspalte.
            If Cells(Zeile, Spalte) <> "" Then
                If Not ((Asc(Mid(Cells(Zeile, Spalte), 1, 1)) >= 65) And (Asc(Mid(Cells(Zeile, Spalte), 1, 1)) <= 90) Then
                    'Fehler, das erste Zeichen ist kein ↗
                    Großbuchstabe, die Großbuchstaben liegen ↗
                    im ASCII-Code
                    'zwischen 65 und 90
                    'Korrektur, wenn es ein Kleinbuchstabe ist:
                    If (Asc(Mid(Cells(Zeile, Spalte), 1, 1)) >= 97) &
                    And (Asc(Mid(Cells(Zeile, Spalte), 1, ↗
                    1))) <= 124 Then
                        'Hier dürfen wir den Kleinbuchstaben ↗
                        umwandeln in einen Großbuchstaben.
                        'Die Kleinbuchstaben liegen im ASCII-Code
                        'zwischen 97 und 124
                        Cells(Zeile, Spalte) = UCase(Mid(Cells(Zeile, ↗
                        Spalte), 1, 1)) & Right(Cells(Zeile, Spalte), ↗
                        Len(Cells(Zeile, Spalte)) - 1)
                    Else
                        'Fehler, hier handelt es sich um ein ↗
                        Sonderzeichen.
                        MsgBox "Fehler " & Cells(Zeile, Spalte)
                    End If
                End If
            End If
        Next Spalte
    Next Zeile
End Sub

```

B



Antworten Kapitel 12

- Frage 1: Mittels UCase (Upper Case) wandelst du Zeichen in die zugehörigen Großbuchstaben um.
- Frage 2: Ein Event ist ein Ereignis. Das Ereignis ist eine Aktion des Anwenders, die der Rechner während des Programmablaufs registriert. Wie auf diese Anwenderaktionen reagiert werden soll, regelst du mit einem Programm.

Literaturverzeichnis

Build Your Own PC Game in Seven Easy Steps Using Visual Basic; Scott Palmer; Addison Wesley; 1996; ISBN 0-201-48911-2

Microsoft Excel 97 Visual Basic; Reed Jacobson; Microsoft Press; 1998; ISBN 3-86063-737-1

Integrierte Lösungen mit Office 97; Gerhard Brosius, Christian Benkwitz, Norbert Böker, Tatjana Tegel; Addison-Wesley; 1. Auflage 1997; ISBN 3-8273-1188-8

Microsoft Excel 2002 für Windows – Automatisierung, Programmierung; Wolfgang Muschner; 3. Auflage: Februar 2002; HERDT Verlag, Nackenheim (Germany)

Algorithmen und Datenstrukturen mit Modula-2; Niklaus Wirth; 4., neu-
bearb. u. erw. Aufl.; B.G. Teubner Stuttgart, 1986; ISBN 3-519-02260-5