Kapitel 10

Eigene Funktionen programmieren

In diesem Kapitel lernen Sie

- wie Sie Ihre eigenen Funktionen erstellen können
- welche Typen von Funktionen es gibt
- auf welche Weise Funktionen eingesetzt werden können

10.1 Funktionen verstehen

Excel stellt bereits von Haus aus eine sehr große Anzahl vordefinierter Funktionen zur Verfügung, die zur Berechnung der unterschiedlichsten Werte eingesetzt werden können. Deshalb mag es auf den ersten Blick überflüssig erscheinen, zusätzlich eigene Funktionen in VBA zu programmieren. Trotz allem decken die Excel-eigenen Funktionen nicht immer die Vielzahl der in der Praxis benötigten Berechnungen ab. Hinzu kommt, dass sich wiederholt verwendete Funktionen per VBA verkürzen lassen. In diesem Kapitel werden Sie erfahren, wie per VBA eigene Funktionen programmiert werden können.

Im Unterschied zu den bis jetzt behandelten Prozeduren wird eine Funktion nicht mit `Sub` eingeleitet, sondern mit `Function`. Eine Funktion gibt immer einen Wert zurück. Sie kann in einem Tabellenblatt als Formel verwendet oder aus einer Prozedur heraus aufgerufen werden.

Hinweis: Funktionen können nicht wie Sub-Prozeduren ausgeführt werden

Wenn Sie VBA-Funktionen programmieren, müssen Sie wissen, dass diese nicht wie Prozeduren eingesetzt werden können. Es ist nicht möglich, mittels einer Funktion Veränderungen an Bereichen oder anderen Objekten in Ihrem Tabellenblatt vorzunehmen. Eine Funktion kann beispielsweise nicht die Farbe einer Zelle verändern, auch wenn dies in gewissen Fällen sinnvoll erscheinen mag. Funktionen sind lediglich dazu da, um Werte zurückzugeben.

Die Namen der in Excel bereits eingebauten VBA-Funktionen sind immer in Englisch. Wenn man also eine Tabellenfunktion in VBA verwenden möchte, muss man erst den englischen Namen herausfinden. Sie haben die Möglichkeit, die englischen Funktionsnamen unter Zuhilfenahme des Makrorekorders zu ermitteln. Nehmen wir als Beispiel die Funktion *RUNDEN()*. Führen Sie folgende Schritte aus, um den englischen Funktionsnamen zu ermitteln:

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

1. Geben Sie in die Zelle *A1* den Wert *700,75* ein.
2. Geben Sie in die Zelle *B1* die Formel *=RUNDEN(A1;0)* ein. Die Formel rundet auf eine Ganzzahl auf oder ab.
3. Schließen Sie die Formeleingabe mit der ↵-Taste ab.
4. Aktivieren Sie die Zelle *B1*.
5. Starten Sie den Makrorekorder.
6. Drücken Sie während der Aufzeichnung die Taste F2 und anschließend die ↵-Taste.
7. Beenden Sie die Aufzeichnung.

Im VBA-Editor können Sie nun dem aufgezeichneten Makro die Codezeile `ActiveCell.FormulaR1C1 = "=ROUND(RC[-1],0)"` entnehmen. Die Übersetzung für die Funktion *RUNDEN()* lautet in VBA somit `ROUND`.

Die aufgezeichnete Codezeile sieht auf den ersten Blick etwas verwirrend aus, deshalb hier einige Erläuterungen dazu. Die Eigenschaft `FormulaR1C1` gibt einen Z1S1-Bezug zurück. `R1` (bzw. Z1) steht für *R*ow1 (oder *Z*eile1) und `C1` (bzw. S1) steht für *C*olumn1 (oder *S*palte1). Immer wenn mit dem Makrorekorder eine Funktion aufgezeichnet wird, wird diese Bezugsart ausgegeben.

In Anführungszeichen steht ein Gleichheitszeichen, gefolgt vom Funktionsnamen und dem Zellenbezug. Der Zellbezug wird in diesem Beispiel als `RC[-1]` angegeben. Dies bedeutet, dass sich die Zelle, auf die sich die Berechnung bezieht, in derselben Zeile (`Row`) befindet wie die Formel. Deshalb ist hinter dem `R` kein Index in eckigen Klammern erforderlich. Die zu berechnende Zahl liegt eine Spalte links von der Formel, deshalb die Angabe `[-1]`.

Würde die Formel in der Zelle *C3* stehen, würde der Bezug `R[-2]C[-2]` lauten. Von der Formel aus gesehen würde die zu berechnende Zahl somit *zwei* Zellen links und *zwei* Spalten oberhalb liegen.

In der VBA-Programmierung sind solche Bezüge kaum anzutreffen. Es wird vielmehr auf das Objekt Bezug genommen, wie Sie gleich erfahren werden.

Eine weitere Möglichkeit, den englischen Funktionsnamen einer aktiven Zelle zu ermitteln, besteht darin, eine kleine Prozedur zu verwenden. Die Eigenschaft dazu lautet `Formula`. Wenn Sie `FormulaLocal` benutzen, wird der deutsche Funktionsname angezeigt.

Um die folgende Prozedur zu verwenden, aktivieren Sie eine Zelle, die eine Formel enthält, und führen den Code aus.

```
Sub EnglischerName()
   MsgBox ActiveCell.Formula
End Sub
```

Wenn in der Zelle eine Funktion vorhanden ist, wird diese im Meldungsfeld angezeigt.

Eine einfache Funktion erstellen

Alle Prozeduren, die wir bisher verwendet haben, wurden durch die Anweisung `Sub` eingeleitet und abgeschlossen. Um eine Funktion zu erstellen, wird die Anweisung `Function` benutzt. Ihr folgt der Name, der später im Tabellenblatt als Formel eingegeben werden kann. In den runden Klammern kann ein Argument eingetragen werden, beispielsweise wenn Bezug auf eine Zelle genommen werden soll.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Beim Funktionsnamen und bei den Argumenten handelt es sich um Variablen, die mit einem entsprechenden Datentyp deklariert werden sollten.

Bleiben wir beim Ermitteln von englischen Funktionsnamen. Wir werden nun eine benutzerdefinierte Funktion erstellen, die es ermöglicht, die Formel einer beliebigen Zelle zu ermitteln und deren englischen Namen auszugeben.

```
Function strEnglisch(rngZelle As Range) As String
   strEnglisch = rngZelle.Formula
End Function
```

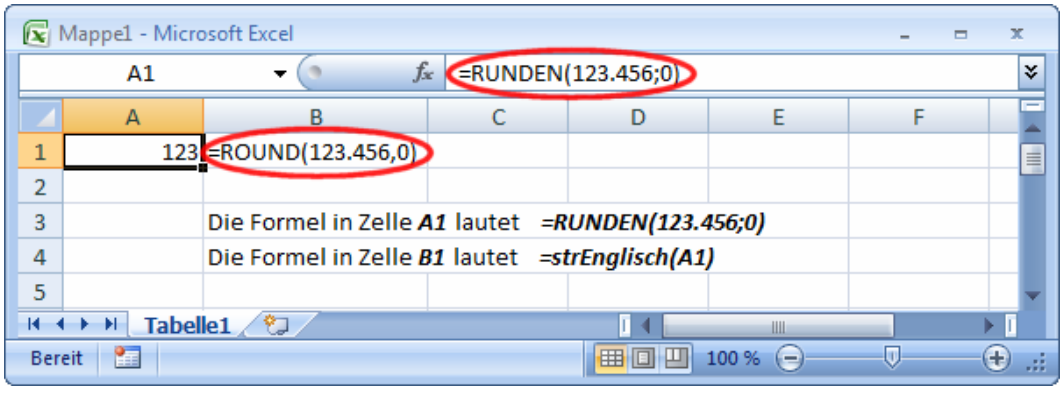

Abbildung 10.1: Den englischen Funktionsnamen ermitteln

Anhand der folgenden Grafik können Sie erkennen, wie die Variablen innerhalb der Funktion verwendet werden.

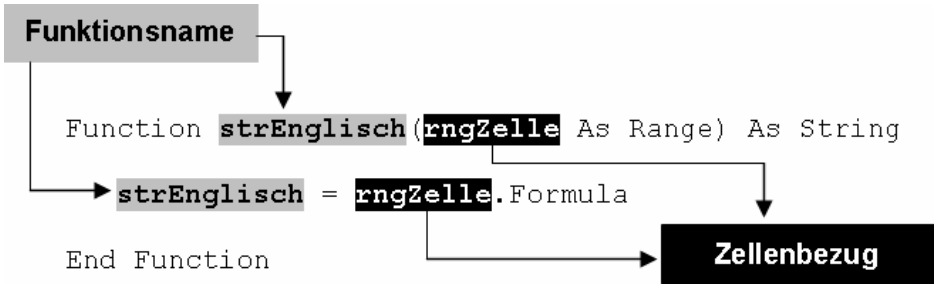

Abbildung 10.2: Variablenabhängigkeiten

Sehen wir uns nun Schritt für Schritt an, was in der Funktion genau geschieht:

1. Im Kopf der Funktion wurde die Variable `strEnglisch` deklariert. Diesen können Sie später im Tabellenblatt als Funktionsnamen eingeben.
2. Als Argument wurde in den rundern Klammern die Variable `rngZelle` als `Range` deklariert. Diese stellt innerhalb der Formel den Bezug zu einer Zelle her.
3. Im Körper der Prozedur wurde dem Funktionsnamen `strEnglisch` der Bereichsname `rngZelle` übergeben. Dabei wurde die Eigenschaft `Formula` mitgeliefert, die ja bekanntlich die englische Bezeichnung einer Funktion zurückgibt.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Die Funktion verwenden

Nun stellen Sie sich vermutlich die Frage, wie Sie die selbst gebaute Funktion im Tabellenblatt verwenden können. Die Antwort ist sehr einfach: So wie jede andere Formel auch.

1. Geben Sie ein Gleichheitszeichen (=) gefolgt vom Funktionsnamen ein: *=strEnglisch*
2. Ergänzen Sie die runden Klammern, in denen Sie den Zellenbezug eingeben.

   ```
   =strEnglisch(A1)
   ```

   Die Zelle, auf die Sie Bezug nehmen, sollte eine Formel enthalten.
3. Drücken Sie die ↵-Taste, um die Funktion abzuschließen.

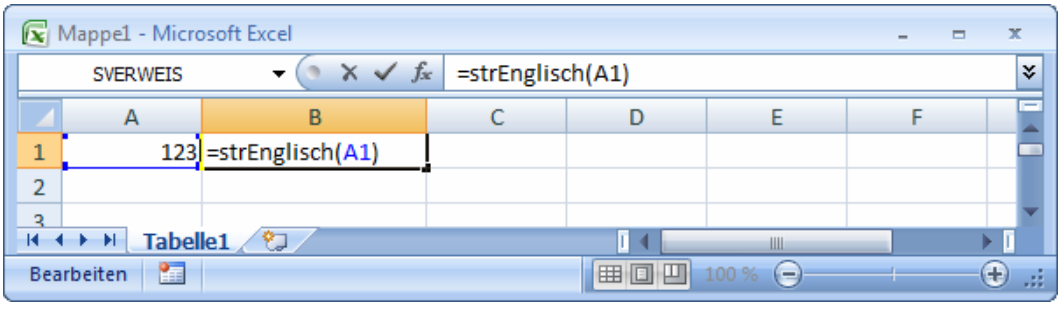

Abbildung 10.3: Der Funktionsname der selbst erstellten Funktion

Wo befindet sich die Funktion?

Eine eigene Funktion können Sie direkt in die Zelle oder in die Bearbeitungsleiste eintippen; außerdem haben Sie die Möglichkeit, zum Einfügen der Funktion den Funktions-Assistenten zu verwenden.

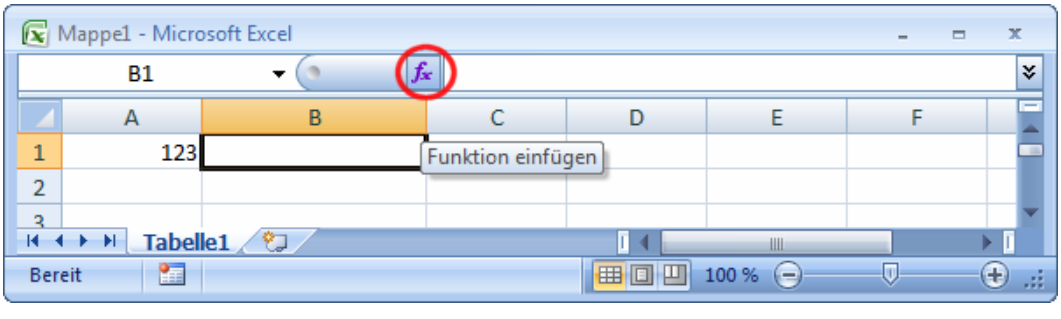

Abbildung 10.4: Die Funktion suchen

1. Rufen Sie den Funktions-Assistenten auf.
2. Wählen Sie im Dropdown-Listenfeld *Kategorie auswählen* die Kategorie *Benutzerdefiniert* aus. Darin befinden sich alle selbst erstellten Funktionen.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

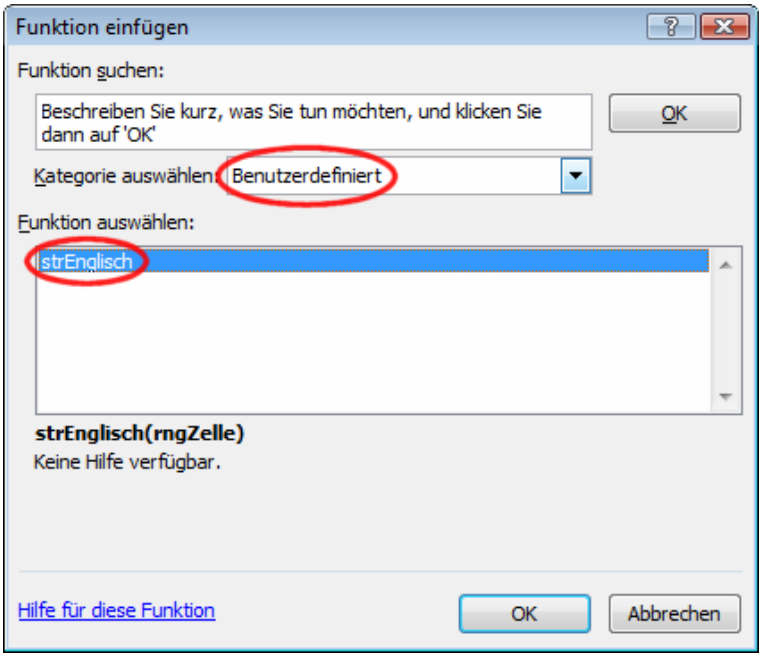

Abbildung 10.5: Die Funktion im Funktions-Assistenten

3. Markieren Sie Ihre Funktion und klicken Sie dann auf die Schaltfläche *OK*. Excel öffnet das Dialogfeld *Funktionsargumente*.

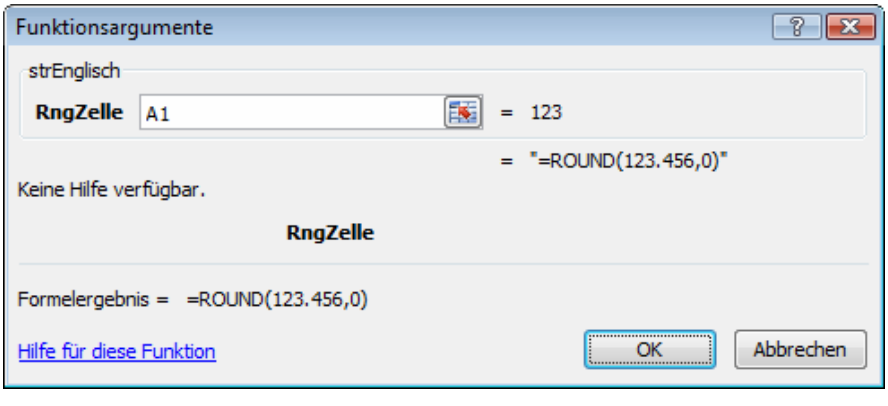

Abbildung 10.6: Die Funktionsargumente aufrufen

4. Geben Sie hier den Zellenbezug ein. Zum Beispiel *A1*.
5. Klicken Sie auf die Schaltfläche *OK*.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Die Kategorie wechseln

Die benutzerdefinierten Formeln befinden sich also in der Kategorie *Benutzerdefiniert*. Dies muss nicht unbedingt sein. Sie können auf Wunsch die Funktion einer anderen Kategorie zuweisen. Es stehen insgesamt 15 Kategorien zur Verfügung. Einige davon werden erst erstellt, wenn Sie ihnen eine Funktion zuweisen. Diese Kategorien sind in der nachfolgenden Tabelle mit einem Sternchen (*) gekennzeichnet.

Tabelle 10.1: Funktionskategorien

Index	Kategoriename
1	Finanzmathematik
2	Datum & Zeit
3	Math. & Trigonom.
4	Statistik
5	Matrix
6	Datenbank
7	Text
8	Logik
9	Information
10	Menübefehle *
11	Benutzerorientiert *
12	Makrosteuerung *
13	DDE/Extern *
14	Benutzerdefiniert *
15 bis 32	Benutzerdefinierte Kategorien *

Hinweis: Add-Ins

Sollten Sie Add-Ins installiert haben, kann es sein, dass weitere Kategorien angezeigt werden.

Um eine Funktion in eine andere Kategorie zu verschieben, müssen wir eine Standardprozedur schreiben. Die Anweisung, um eine neue Kategorie zu erstellen, lautet: `Application.MacroOptions`. Ihr können verschiedene Argumente übergeben werden.

- Dem Argument `Macro` weisen wir den Funktionsnamen `strEnglisch` zu.
- Mittels `Description` können Sie eine Beschreibung zur Funktion hinterlegen.
- Dem Argument `Category` weisen wir den Index oder den Namen der Kategorie zu, in die wir die Funktion verschieben wollen (siehe obige Tabelle).

```
Sub KategorieWechseln()
   Application.MacroOptions _
      Macro:="strEnglisch", _
      Description:="Englischen Funktionsnamen ermitteln.", _
      Category:=9    ' Oder "Information"
End Sub
```


Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Wenn Sie nun den Funktions-Assistenten aufrufen, finden Sie die von Ihnen erstellte Funktion in der Kategorie *Information*.

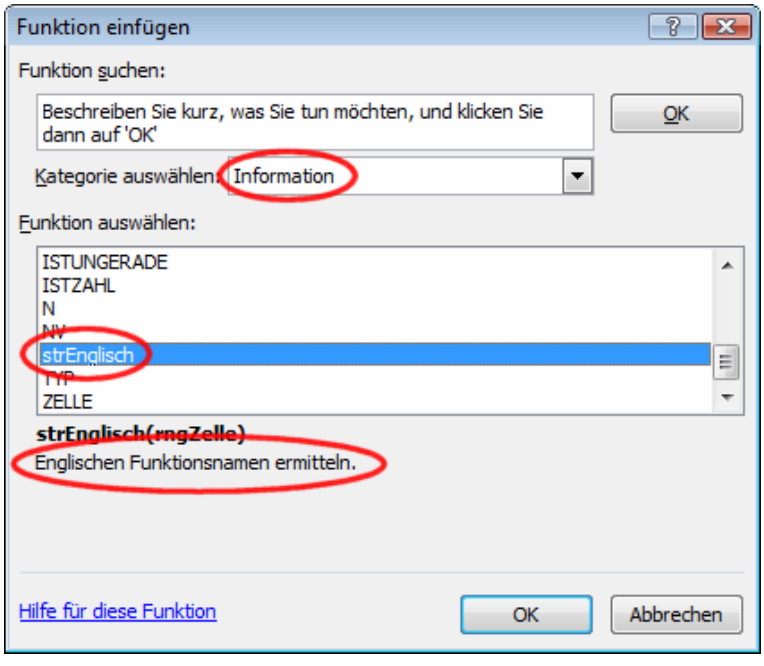

Abbildung 10.7: Die Kategorie wechseln

Eine benutzerdefinierte Kategorie erstellen

Wie Sie Tabelle auf der vorigen Seite entnehmen können, gibt es noch die Kategorien mit dem Index 15 bis 32. Es gibt sie erst seit der Version 2003. Um genauer zu sein, gab es in der Version 2002 noch die Kategorie 15 mit dem Namen *Technisch*. Diese wurde mit der Version 2003 entfernt.

Im Bereich von 15 bis 32 können Sie eigene Kategorien erstellen. Somit stehen 18 benutzerdefinierte Kategorien zur Verfügung.

Um eine eigene Kategorie zu erstellen, verwenden Sie für das Argument `Category` den gewünschten Namen.

```
Sub BenutzerdefKategorie()
   Application.MacroOptions _
      Macro:="strEnglisch", _
      Description:="Englischen Funktionsnamen ermitteln.", _
      Category:="Meine Kategorie"
End Sub
```


Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

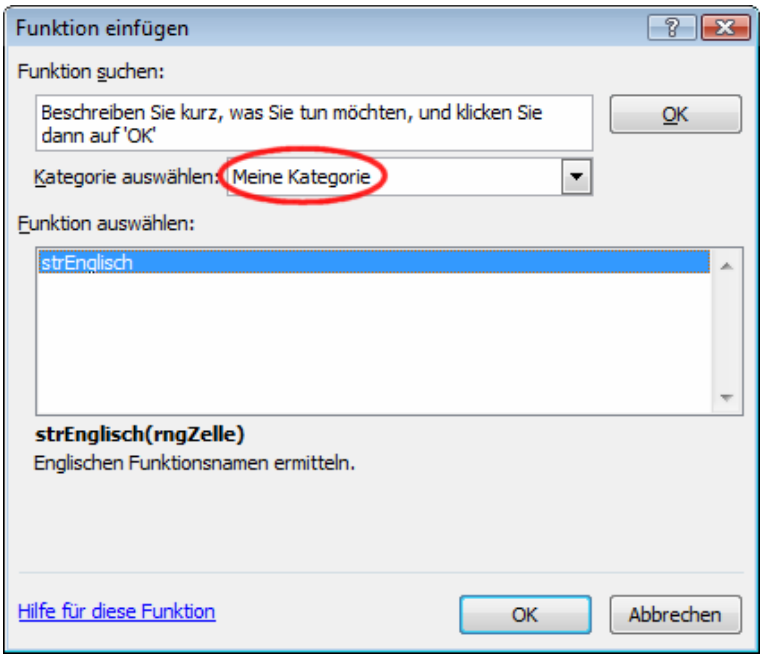

Abbildung 10.8: Eine eigene Kategorie anlegen

Sobald eine benutzerdefinierte Kategorie einmal erstellt wurde, kann sie neben dem Namen auch über den Index angesprochen werden.

10.2 Erweiterte Funktionen

Bis jetzt haben wir erst eine Formel mit einem Argument erstellt. Es gibt aber noch weit mehr Möglichkeiten.

Eine Funktion ohne Argumente

Sie müssen einer Funktion nicht zwingend ein Argument übergeben. Es gibt Situationen, in denen der Wert, der in der Zelle erscheinen soll, berechnet werden kann, ohne dass dazu Parameter übergeben werden müssen.

Die nachfolgende Funktion ist so ein Beispiel. Es wird mittels der Funktion `strWSName` der Name des aktiven Tabellenblattes ermittelt. Das Klammernpaar nach dem Funktionsnamen bleibt leer.

```
Function strWSName() As String
   strWSName = ActiveSheet.Name
End Function
```


Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Eine Funktion mit mehreren Argumenten

Sie können Funktionen erstellen, die mehrere Übergabewerte erwarten. Dies ist beispielsweise dann der Fall, wenn Sie Werte berechnen möchten, die aus verschiedenen Zellen stammen.

1. Wir werden nun eine Funktion erstellen, die Werte aus drei Zellen entnimmt. Dementsprechend bereiten wir drei Argumente in den runden Klammern der Funktion `dblNetto` auf.
2. Es sollen die Nettokosten unter Berücksichtigung der Anzahl, des Preises und des Rabatts berechnet werden. Die Berechnung für das Ergebnis, das die Funktion `dblNetto` ausgeben soll, findet im Körper der Funktion statt.

```
Function dblNetto(rngAnzahl As Range, _
                  rngPreis As Range, _
                  rngRabatt As Range) As Double
   Dim dblZwischensumme As Double

   dblZwischensumme = rngAnzahl * rngPreis
   dblNetto = dblZwischensumme - dblZwischensumme * rngRabatt
End Function
```

3. Wenn Sie den Funktions-Assistenten aufrufen, müssen Sie nur noch die drei Zellen eingeben, die berechnet werden sollen. Da die Berechnung bereits in der Funktion hinterlegt ist, sind innerhalb der Funktion keine Operatoren erforderlich.

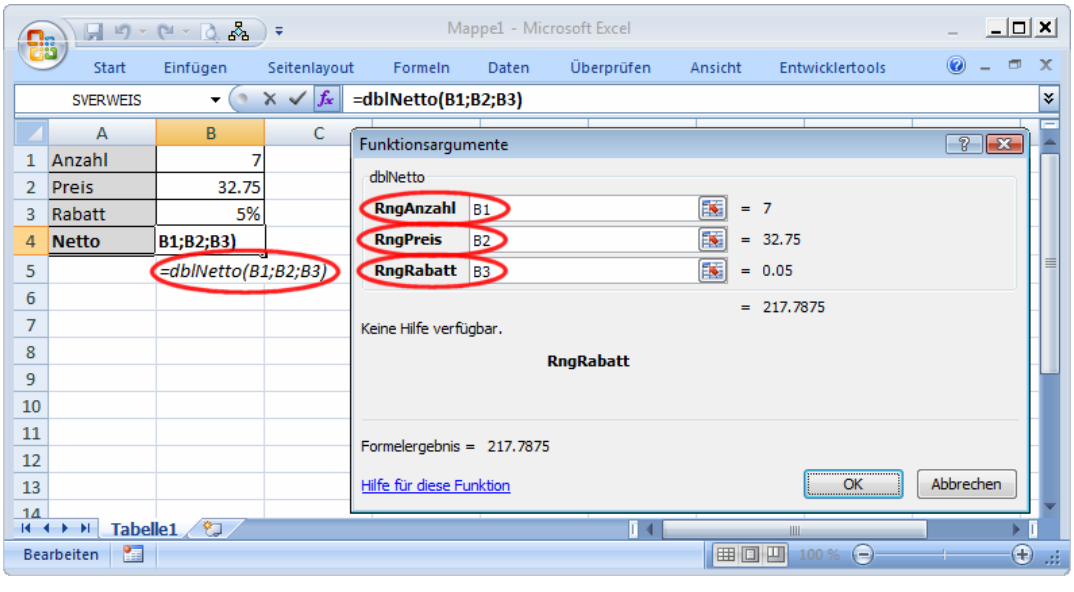

Abbildung 10.9: Mehrere Argumente in einer Funktion

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Unbekannte Anzahl an Argumenten (Parameter-Array)

Bei gewissen Berechnungen kann die Anzahl an Argumenten variieren. Es kann vorkommen, dass die Funktion einmal drei Übergabewerte berechnen muss und ein anderes Mal fünf, beispielsweise wenn Sie eine Summe über vereinzelte Zellen bilden möchten, die sich an verschiedenen Stellen des Tabellenblatt befinden.

Es wäre sinnlos, fünf Argumente in der Funktion zur Verfügung zu stellen, wenn letztendlich nur drei davon gebraucht werden. Zudem würden fünf Argumente nicht ausreichen, wenn später zehn Zellen addiert werden müssen. Eine Funktion mit einer festen Anzahl an Argumenten würde dabei früher oder später an Grenzen stoßen.

VBA stellt für solche Fälle eine sogenannte Parameter-Array-Funktion zur Verfügung. Dem Argument, das als Parameter-Array dienen soll, wird das Schlüsselwort `ParamArray` vorangestellt.

```
Function dblAddition(ParamArray varListe() As Variant) As Double
```

Es ist zwingend erforderlich, die Variable für das Argument als Datentyp `Variant` zu deklarieren.

Wir erstellen nun eine Funktion, mit der Sie die Summe beliebig vieler Zellen berechnen können.

1. Hinterlegen Sie im Kopf der Funktion ein Argument, dem das Schlüsselwort `ParamArray` vorangeht.
2. In der Codezeile unter dem Kopf deklarieren Sie die Variable `varArgument`. Diese benötigen Sie danach in der `For Each`-Schleife. Da damit die Werte des Parameter-Arrays durchlaufen werden, muss die Variable ebenfalls mit dem Datentyp `Variant` deklariert werden.
3. In der `For Each`-Schleife werden sämtliche Werte, die das Parameter-Array enthält, durchlaufen. Innerhalb der Schleife können Sie die Addition (+) der Werte vornehmen. Die Summe wird in der Variablen `dblAddition` gespeichert, die unser Funktionsname ist.

```
Function dblAddition(ParamArray varListe() As Variant) As Double
   Dim varArgument As Variant

   For Each varArgument In varListe()
      dblAddition = dblAddition + varArgument
   Next varArgument
End Function
```

4. Sobald Sie die Funktion fertig programmiert haben, können Sie zum Tabellenblatt wechseln und dort die Funktion aufrufen. Verwenden Sie dazu den Funktions-Assistenten.
5. Aktivieren Sie im Funktions-Assistenten die benutzerdefinierte Funktion `dblAddition`.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

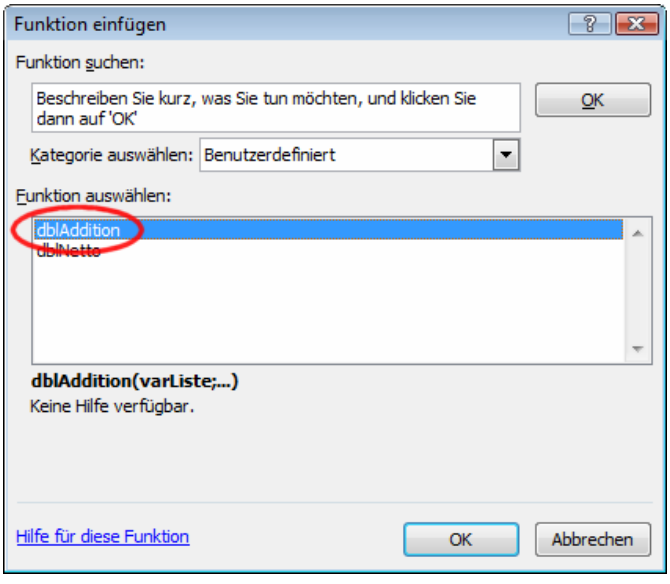

Abbildung 10.10: Die Funktion *dblAddition* aufrufen

6. Klicken Sie auf *OK*, um die das Dialogfeld *Funktionsargumente* zu öffnen. Dort sehen Sie das Parameter-Array.

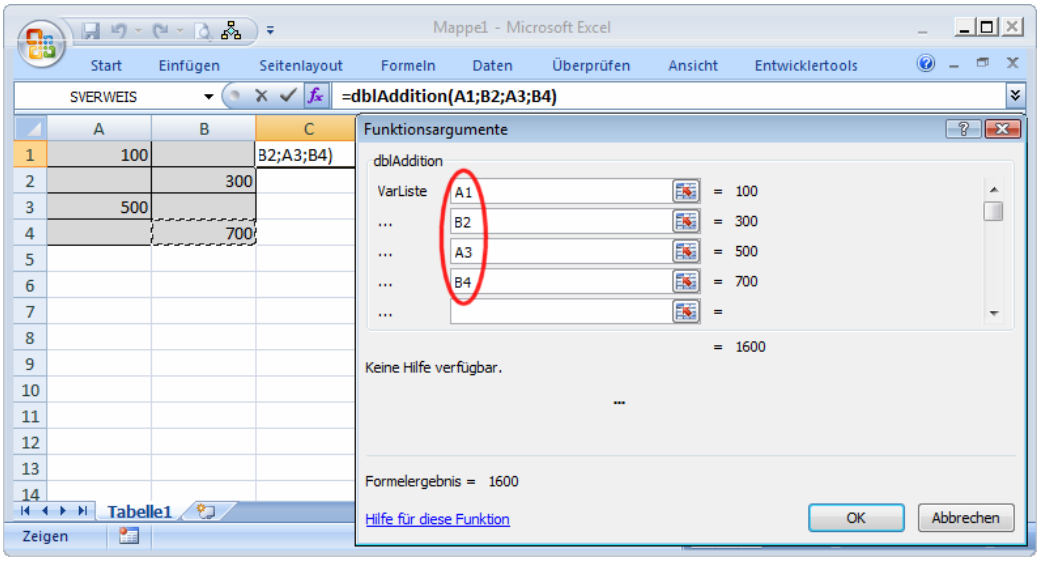

Abbildung 10.11: Einzelne Zellen zur Addition übergeben

7. Sie können nun beliebig viele Werte oder Zellbezüge eingeben. Das Array wird mit jeder Eingabe erweitert.
8. Nach vollständiger Eingabe klicken Sie auf die Schaltfläche *OK*. Die Formel mit dem Ergebnis ist nun in der Zelle vorhanden.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Bereichsfunktionen

Sie können auch Funktionen entwerfen, die Bereichseingaben (*A1:B5*) ermöglichen. Beispielsweise dann, wenn Sie eine Summe bilden möchten. Sie müssen lediglich das Argument in den runden Klammern mit dem Datentyp `Range` hinterlegen.

Um die Excel-eigene Summenfunktion nachzubilden, verwenden Sie das Objekt `Application` gefolgt von der Eigenschaft `WorksheetFunction`. Nach diesen beiden Angaben folgt die Arbeitsblattfunktion `Sum`.

```
Function dblSumme(rngBereich As Range) As Double
   dblSumme = Application.WorksheetFunction.Sum(rngBereich)
End Function
```

Die Funktion können Sie nun wie die Standardsummenfunktion von Excel verwenden.

Neuberechnungen *(Application.Volatile)*

Benutzerdefinierte VBA-Funktionen werden in der Regel, genau wie Tabellenfunktionen, nur dann neu berechnet, wenn es erforderlich ist, beispielsweise wenn sich der Inhalt einer Zelle ändert, die das Ergebnis der Funktion beeinflusst. Wenn die VBA-Funktion keinen Übergabewert erwartet, wird keine Neuberechnung vorgenommen, da ja kein Bezug zu einer Zelle besteht.

Nehmen wir als Beispiel die Tabellenfunktion *=ZUFALLSZAHL()*. Wenn Sie diese Formel in eine Zelle eintragen, wird jeweils eine neue Zufallszahl generiert, sobald sich in einer Zelle etwas ändert. Sie können auch die Taste F9 drücken, um die Neuberechnung manuell anzustoßen.

Schauen wir uns nun an, was passiert, wenn wir die Funktion per VBA nachstellen.

1. Geben Sie die folgenden Codezeilen in ein Standardmodul ein:

   ```
   Function dblZufall() As Double
      dblZufall = Rnd()
   End Function
   ```

2. Wechseln Sie zum Tabellenblatt und geben Sie in eine beliebige Zelle die Formel *=ZUFALLSZAHL()* ein.
3. Geben Sie in eine andere Zelle die Formel *=dblZufall()* ein.
4. Drücken Sie nun die Taste F9. Sie werden feststellen, dass die Tabellenfunktion *ZUFALLSZAHL* eine neue Zahl generiert hat. Unsere selbst erstellte VBA-Funktion hat den alten Wert beibehalten.
5. Damit in Zukunft auch die VBA-Funktion auf das Drücken der Taste F9 oder eine Zellenänderung reagiert, müssen Sie den Code um die Anweisung `Application.Volatile` erweitern. *Volatile* bedeutet in diesem Zusammenhang *veränderlich*.

   Damit der Unterschied deutlich wird, erstellen Sie am besten eine Funktion mit einem neuen Funktionsnamen. Die neue Funktion sieht nun wie folgt aus:

   ```
   Function dblZufallVolatile() As Double
      Application.Volatile
      dblZufallVolatile = Rnd()
   End Function
   ```

6. Wechseln Sie wiederum zu Ihrem Tabellenblatt und geben Sie in eine weitere Zelle die Formel *=dblZufallVolatile()* ein.
7. In Ihrem Tabellenblatt sollten nun drei Formeln vorhanden sein. Drücken Sie erneut die Taste F9.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Sowohl die Tabellenfunktion als auch die zweite VBA-Funktion haben nun eine neue Zufallszahl in der Zelle ausgegeben.

10.3 Funktionen für Prozeduren

Bis jetzt haben wir VBA-Funktionen nur in Tabellenblättern verwendet. Sie können diese jedoch auch in `Sub`-Prozeduren benutzen. Oftmals wird ein und derselbe Codeblock in einer Prozedur mehrmals eingetippt oder eine Reihe an Anweisungen wird in verschiedenen Prozeduren immer und immer wieder benötigt. Damit diese Codezeilen nicht jedes Mal von Neuem eingegeben werden müssen, können Sie diese in einer Funktion auslagern.

Ein typisches Beispiel ist das Auffinden der ersten leeren Zeile in einer Tabelle. Diese Funktion wird häufig benötigt, wenn per VBA Daten untereinander in eine Spalte eingetragen werden müssen. Es reicht dann nicht aus, nur die erste leere Zelle pro Spalte aufzufinden, denn es könnte vorkommen, dass einzelne Zellen einer Tabelle nicht ausgefüllt sind. Wenn jede einzelne Spalte für sich auf die erste leere Zelle überprüft würde, käme es zu einer Verschiebung ganzer neuer Datensätze.

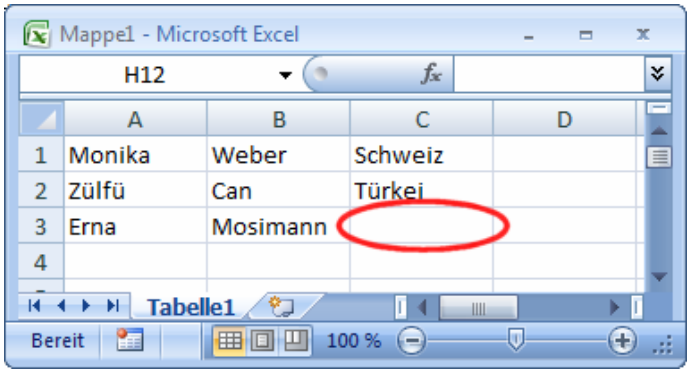

Abbildung 10.12: Die leere Zelle muss berücksichtigt werden

1. Die folgende Funktion überprüft jede einzelne Spalte auf ihre Einträge. In der `For`-Schleife wird der Variablen `lngMin` jeweils der Zeilenindex der letzten benutzten Zelle einer Spalte übergeben.

```
Function lngLetzteZeile() As Long
   Dim i As Integer, lngMin As Long, lngMax As Long

   For i = 1 To 256
      lngMin = Cells(65536, i).End(xlUp).Row

      If lngMax < lngMin Then
         lngMax = lngMin
      End If
   Next i

   lngLetzteZeile = lngMax
End Function
```

2. Die `If`-Abfrage prüft, ob der Inhalt der Variablen `lngMax` kleiner ist als der Inhalt von `lngMin`. Wenn dies zutrifft, wird der größere Wert an die Variable `lngMax` übergeben.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

3. Dadurch erhalten wir am Ende der Prozedur den Index der letzten benutzten Zeile. Diese Zahl übergeben wir dem Funktionsnamen `lngLetzteZeile`.
4. Als Nächstes erstellen wir eine `Sub`-Prozedur, die unsere VBA-Funktion verwendet, um die Tabelle um eine weitere Zeile zu ergänzen.

```
Sub LetzteZeileBefuellen()
   Cells(lngLetzteZeile + 1, 1) = "Robert"
   Cells(lngLetzteZeile, 2) = "Weber"
   Cells(lngLetzteZeile, 3) = "Schweiz"
End Sub
```

5. Den Funktionsnamen tragen wir als Zeilenindex im Objekt `Cells` ein. Der Rückgabewert unserer VBA-Prozedur ist die letzte *benutzte* Zeile der Tabelle. Wir müssen jedoch eine Zeile vorrücken (`lngLetzteZeile + 1`), um die erste *freie* Zeile zu erreichen. Dies muss nur bei der ersten Zuweisung geschehen. Dadurch, dass der erste Eintrag erfolgt ist, hat sich auch der Zeilenindex unserer Funktion um den Wert 1 erhöht. Bei den nächsten Zuweisungen kann somit auf das +1 verzichtet werden. Die Tabelle ist nun um eine Zeile erweitert.

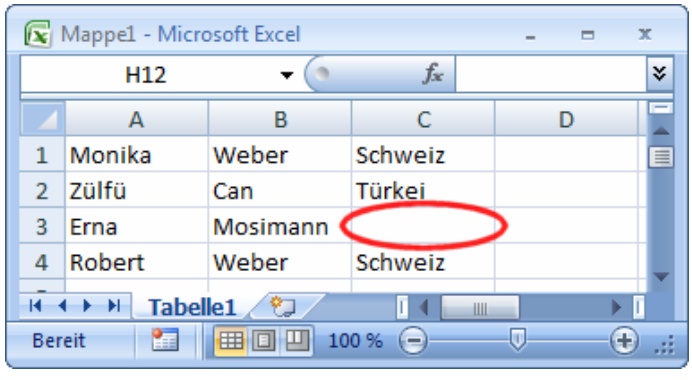

Abbildung 10.13: Die leere Zelle wurde beim Einfügen der neuen Zeile berücksichtigt

10.4 Nützliche Helfer

Nachfolgend finden Sie einige Beispiele, die Ihnen das praktische Arbeiten mit Funktionen näherbringen soll.

Eine Rabattstaffel erstellen

Mittels einer VBA-Funktion können Sie eine benutzerdefinierte Rabattstaffel erstellen. Die Rabattstaffel in unserem Beispiel ist nach folgendem Muster aufgebaut:

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Tabelle 10.2: Eine Rabattstaffel

Betrag	Rabatt
Kleiner als 100	0 %
100 bis 499	5 %
500 bis 999	10 %
Größer oder gleich 1000	15 %

Um die Rabattstaffel zu erstellen, können Sie wahlweise eine `If-` oder eine `Select Case`-Entscheidung verwenden. Wir benutzen hier das `Select Case`:

```
Function dblRabatt(rngBetrag As Range) As Double
   Select Case rngBetrag
      Case Is < 100
         dblRabatt = 0
      Case 100 To 499
         dblRabatt = 0.05
      Case 500 To 999
         dblRabatt = 0.1
      Case Is >= 1000
         dblRabatt = 0.15
   End Select
End Function
```

1. Um die Rabattstaffel anzuwenden, geben Sie in eine Zelle einen Betrag ein. Zum Beispiel in Zelle *A1*.
2. In die Zelle *B1* geben Sie die Formel *=dblRabatt(A1)* ein.
3. Nun müssen Sie der Zelle *B1* noch das Prozentformat zuweisen. Verwenden Sie dazu die Schaltfläche *Prozentformat* in der Gruppe *Format* der Registerkarte *Start*.

Eine Auswahl treffen

Die Funktion `Choose` kann verwendet werden, um eine Auswahl zu treffen. Der Funktion wird eine beliebige Anzahl an Argumenten übergeben. In unserem Beispiel sind es drei.

Tabelle 10.3: Eine Auswahl

Index	Rückgabe
1	Sehr gut
2	Befriedigend
3	Ungenügend

```
Function strNote(byt As Byte) As String
   strNote = Application.WorksheetFunction. _
             Choose(byt, _
             "Sehr gut", "Befriedigend", "Ungenügend")
End Function
```


Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

1. Um die Funktion zu verwenden, geben Sie beispielsweise in die Zelle *A1* den Wert 1 ein.
2. In die Zelle *B1* geben Sie die Formel *=strNote(A1)* ein. Der Rückgabewert in Zelle *B1* ist »Sehr gut«.

   Sie können alternativ den Index 1 direkt in die Formel eingeben, um als Ergebnis »Sehr gut« zu erhalten.

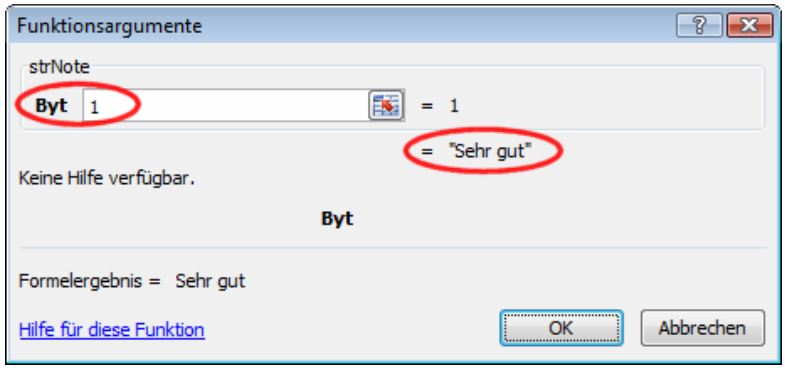

Abbildung 10.14: Eine Benotungsskala

Prüfen, ob eine Zelle eine Formel enthält

Mittels einer VBA-Funktion können Sie ermitteln, ob eine Zelle eine Formel enthält. Es wird dazu die Eigenschaft `HasFormula` verwendet. Der Rückgabewert ist `WAHR`, wenn eine Formel vorhanden ist und `FALSCH`, wenn nicht. Um dies umzusetzen, muss der Funktionsname mit dem Datentyp `Boolean` hinterlegt werden.

```
Function blnFormel(rng As Range) As Boolean
   blnFormel = rng.HasFormula
End Function
```

1. Geben Sie in die Zelle *A1* eine Formel ein, beispielsweise *=500+300*.
2. Geben Sie in die Zelle *B1* die Formel *=blnFormel(A1)* ein.
3. Ändern Sie den Inhalt der Zelle, sodass keine Formel mehr enthalten ist. Geben Sie einfach nur eine Zahl oder einen Text ein.

   Der Rückgabewert ändert sich in `FALSCH`.

Ermitteln, aus welcher Zelle die Funktion aufgerufen wird

Sie können eine VBA-Funktion verwenden, um zu ermitteln, von welcher Zelle aus der Funktionsaufruf erfolgt. Verwenden Sie dazu die Eigenschaft `Caller` des Applikationsobjekts. Die Eigenschaft `Address` gibt ohne Verwendung der Argumente `False, False` den absoluten Zellbezug zurück.

```
Function strAdresse() As String
   strAdresse = Application.Caller.Address
End Function
```

Geben Sie in die Zelle *A1* die Formel *=strAdresse()* ein. Der Rückgabewert ist *A1*.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Farbige Zellen zählen

Sie können mittels einer VBA-Funktion veranlassen, dass Zellen, die mit einer bestimmten Hintergrundfarbe belegt sind, addiert werden.

Im Funktionskopf müssen dazu zwei Argumente hinterlegt werden. Das erste dient dazu, den Bereich aufzunehmen, der die zu zählenden Farben enthält. Das zweite nimmt eine Zelle entgegen, die die zu zählende Farbe enthält.

```
Function intFarbe(rng As Range, rngFarbe As Range) As Integer
```

Innerhalb der `For Each`-Schleife werden die Zellen des angegebenen Bereichs durchlaufen und auf die Farbe überprüft. Wenn eine entsprechende Farbe gefunden wird, wird die Variable, die den Funktionsnamen darstellt, um den Zähler 1 erhöht.

```
Function intFarbe(rng As Range, rngFarbe As Range) As Integer
   Dim c As Range

   For Each c In rng
      If c.Interior.Color = rngFarbe.Interior.Color Then
         intFarbe = intFarbe + 1
      End If
   Next c
End Function
```

1. Formatieren Sie einige Zellen mit roter Hintergrundfarbe.
2. Rufen Sie die Funktion `intFarbe` auf, indem Sie den Funktions-Assistenten verwenden.
3. Geben Sie als erstes Funktionsargument den Bereich ein, in dem die roten Zellen zu finden sind.
4. Im zweiten Feld geben Sie eine Zelle an, die die zu zählende Farbe enthält.
5. Verlassen Sie den Funktions-Assistenten, indem Sie die Schaltfläche *OK* anklicken.

   Die Anzahl an roten Zellen wird nun in der aktiven Zelle ausgegeben.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

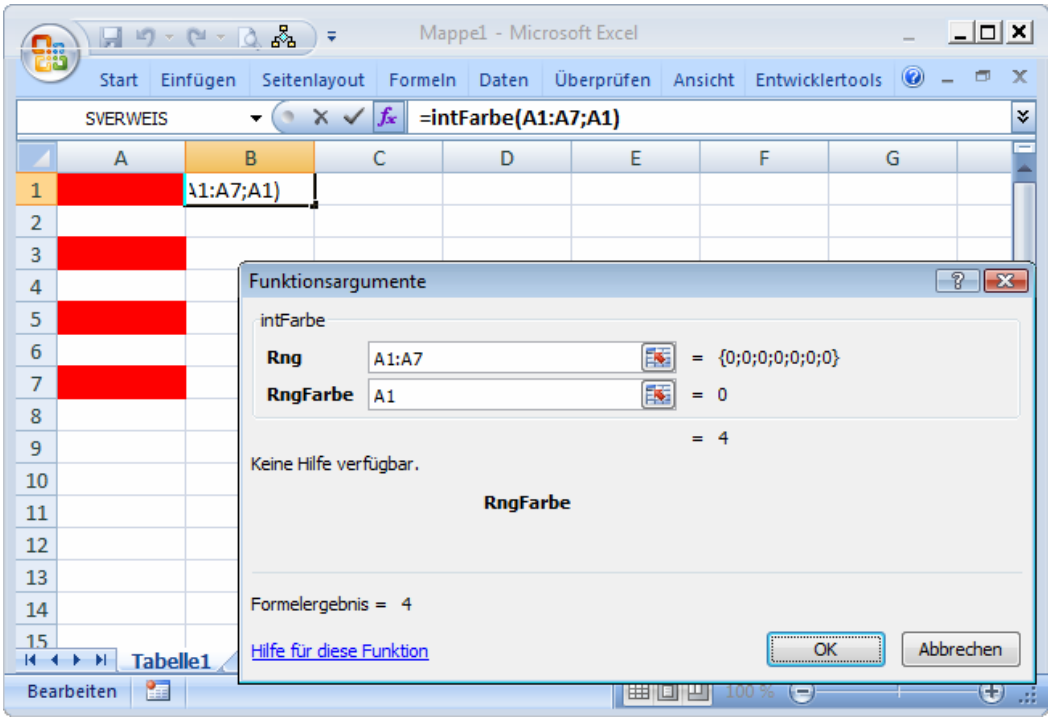

Abbildung 10.15: Farbige Zellen zählen

Hinweis: Manuelle Neuberechnung erforderlich

Wenn Sie mit Farben arbeiten, müssen Sie die Neuberechnung immer manuell anstoßen. Verwenden Sie hierzu die Taste F9.

Der Grund hierfür liegt darin, dass sich bei der Zuweisung von Farben der eigentliche Inhalt der Zelle nicht verändert.

Wenn Sie in eine der farbigen Zellen eine Zahl eingeben oder verändern, erfolgt ebenfalls eine Aktualisierung.

10.5 Übungen zu diesem Kapitel

In diesem Abschnitt finden Sie einige Übungen zu diesem Kapitel. Die richtigen Antworten und Lösungen finden Sie wie immer auf der Website *www.richtig-einsteigen.de.*

Übung 10.1

Wie lautet der englische Name der Funktion *ANZAHL*? Ermitteln Sie ihn per VBA.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)

Übung 10.2

Erstellen Sie eine Funktion, mit der Sie den Namen der aktiven Mappe ermitteln können.

Übung 10.3

Erstellen Sie eine Funktion mit zwei Übergabewerten. Die beiden Werte sollen multipliziert werden.

10.6 Zusammenfassung

Sie haben bereits mehr als die Hälfte des Buches durchgearbeitet und einiges an neuen Erfahrungen sammeln können. Lassen Sie uns einen Rückblick auf dieses Kapitel werfen.

- Sie haben gelernt, was unter Funktionen zu verstehen ist, und wissen, wie Sie auch eigenen Formeln programmieren können.
- Nachdem Sie eigene Funktionen erstellt haben, können Sie diese gezielt der gewünschten Kategorie zuweisen.
- Sie können Funktionen erstellen, die keine oder mehrere Argumente entgegennehmen. Des Weiteren sind Sie in der Lage, Funktionen mit einer unbekannten Anzahl an Argumenten zu generieren. Sie kennen Bereichsfunktionen und können Neuberechnungen bei Bedarf anstoßen.
- Anhand einiger Praxisbeispiele haben Sie erfahren, wie selbst programmierte Funktionen im Alltag verwendet werden können.

Monika Weber: Richtig Einsteigen: Excel 2007 VBA-Programmierung. Microsoft Press 2008 (ISBN 978-3-86645-209-1)