

# Excel 2003 Programmierung mit Visual Basic



Autoren: Andreas Klein, Stefanie Friedrich

1. Auflage: 2005

© Merkwerk

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form ohne Genehmigung der Firma Merkwerk reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Diese Schulungsunterlage wurde mit Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. Für fehlerhafte Angaben und deren Folgen kann daher keine Haftung übernommen werden.

<b>1 Der Makrorecorder.....</b>	<b>5</b>
1.1 Was ist ein Makro?.....	5
1.2 Das Makro ausführen.....	6
1.3 Das Makro mit einem Symbols starten.....	7
1.4 Symbole in Symbolleisten verändern.....	8
1.5 Das Makro mit einer Tastenkombination starten.....	9
<b>2 Makro, Quellcode, Prozedur.....</b>	<b>10</b>
2.1 Der Visual Basic Editor.....	10
2.2 Makros bearbeiten.....	10
2.3 Makros kopieren.....	11
2.4 Kommentare.....	12
2.5 Das Schlüsselwort NOT.....	13
2.6 Das Schlüsselwort With.....	13
<b>3 Erste eigene Programmiererelemente.....</b>	<b>15</b>
3.1 Module und Prozeduren.....	15
3.2 Die Messagebox.....	15
<b>4 Variablen.....</b>	<b>17</b>
4.1 Eine Variable einsetzen.....	17
4.2 Die Variablendeklaration.....	17
4.3 Der Gültigkeitsbereich von Variablen.....	18
4.4 Datentypen.....	19
<b>5 Weitere Programmiererelemente.....</b>	<b>21</b>
5.1 Die Inputbox.....	21
5.2 Eine Excelliste durchsuchen.....	22
5.3 Die relative Aufzeichnung.....	23
5.4 Suchergebnisse übergeben.....	24
5.5 Texte kombinieren.....	25
<b>6 Exkurs: Die Hilfe.....</b>	<b>26</b>
6.1 Rückgabewerte der Messagebox.....	27
<b>7 Verschiedene Funktionen.....</b>	<b>29</b>
7.1 Die if-Anweisung.....	29
7.2 Die mid-Funktion.....	31
<b>8 Debuggen.....</b>	<b>33</b>
<b>9 Ausgabeformat ändern.....</b>	<b>34</b>
<b>10 Programmieren eines Formulars.....</b>	<b>35</b>

<a href="#">10.1 Das Formular erzeugen.....</a>	<a href="#">36</a>
<a href="#">10.2 Steuerelemente einfügen.....</a>	<a href="#">36</a>
<a href="#">10.3 Eigenschaften.....</a>	<a href="#">37</a>
<a href="#">10.4 Steuerelemente verändern.....</a>	<a href="#">40</a>
<a href="#">10.5 Dem Kombinationsfeld Listentext zuordnen.....</a>	<a href="#">40</a>
<a href="#">10.6 Steuerelemente programmieren.....</a>	<a href="#">40</a>
<a href="#">10.7 Gesamtpreis berechnen und ausgeben.....</a>	<a href="#">42</a>
<a href="#">10.8 Das Formular anzeigen.....</a>	<a href="#">44</a>
<b><a href="#">11 Die Fehlerbehandlung.....</a></b>	<b><a href="#">50</a></b>
<b><a href="#">12 Makros speichern.....</a></b>	<b><a href="#">53</a></b>
<a href="#">12.1 Die persönliche Makroarbeitsmappe.....</a>	<a href="#">53</a>
<a href="#">12.2 Eine eigene Makroarbeitsmappe.....</a>	<a href="#">54</a>
<b><a href="#">13 Funktionen.....</a></b>	<b><a href="#">56</a></b>
<a href="#">13.1 Funktionen erstellen.....</a>	<a href="#">56</a>
<a href="#">13.2 Funktionen in Excel anwenden.....</a>	<a href="#">57</a>
<a href="#">13.3 Funktionen im Code anwenden.....</a>	<a href="#">57</a>
<b><a href="#">14 Lösungen zu den Übungen.....</a></b>	<b><a href="#">60</a></b>
<b><a href="#">15 Index</a></b>	<b><a href="#">70</a></b>

Zur Handhabung des Skripts:

Der Aufbau des vorliegenden Skriptes verläuft anhand zweier großer Beispiele (Grundlage sind die Dateien Gasherd.xls und Park.xls), die schrittweise in den einzelnen Kapiteln erarbeitet werden. Arbeiten an diesen beiden Programmen werden mit PRAKTIKUM angekündigt.



ÜBUNGEN sind zusätzliche vertiefende Aufgaben, die die verschiedenen Inhalte wiederholen. Die Lösungen finden sich in einem eigenen Kapitel am Ende des Skriptes.



Außerdem sind in dieser Schulungsunterlage viele HINWEISE aufgeführt, die wichtige Informationen sammeln und verschiedenen BEISPIELE, die konkrete Anwendungsfälle für verschiedene Bereiche aufzeigen.



# 1 Der Makrorecorder

In diesem Kapitel lernen Sie

- wann und wie Sie Makros einsetzen
- wie Sie Makros aufzeichnen und abspielen
- wie Sie Makros über Symbole oder Tastenkombinationen ausführen

## 1.1 Was ist ein Makro?

Mit einem Makro lassen sich verschiedene Befehle automatisch hintereinander abarbeiten. Ein Makro kann effizient Routineaufgaben bearbeiten.

Makro werden mit dem Makrorecorder aufgezeichnet. Die einzelnen Schritte eines Makros müssen in der richtigen Reihenfolge aufgenommen werden. Daher ist es sinnvoll, sich die nötigen Schritte vor der Aufzeichnung zu notieren.



### PRAKTIKUM

Ein Makro soll für alle Spalten eines zusammenhängenden Bereichs die Spaltenbreite optimal anpassen.

Die einzelnen Schritte des Makros:

1. Zelle A1 anklicken
2. STRG UMSCHALT \*
3. FORMAT / SPALTEN / OPTIMALE BREITE FESTLEGEN

Starten Sie den Makrorecorder mit der Befehlsfolge

EXTRAS / MAKRO / AUFZEICHNEN

Makro aufzeichnen

Makroname:  
Makro1

Tastenkombination:    Makro speichern in:  
Strg+    Diese Arbeitsmappe

Beschreibung:  
Makro am 03.01.2005 von Merkwerk aufgezeichnet


OK    Abbrechen

Ersetzen Sie den eingetragenen Makronamen, geben Sie „Spalte“ ein und bestätigen Sie mit OK.



Nun erscheint die Symbolleiste „Aufzeichnung beenden“. Die Aufzeichnung läuft bereits, alle Aktionen werden nun aufgenommen.

Klicken Sie in die Zelle A1 der Tabelle. Markieren Sie den aktuellen Bereich mit der Tastenkombination STRG UMSCHALT \*. Wählen Sie den Befehl FORMAT / SPALTE / OPTIMALE BREITE BESTIMMEN.

Beenden Sie die Makroaufzeichnung mit einem Klick auf das Symbol „Aufzeichnung beenden“  oder über die Befehlsfolge

EXTRAS / MAKRO / AUFZEICHNUNG BEENDEN



**HINWEISE:** Da der Makrorecorder alle Befehle aufzeichnet, also auch Befehle, die Sie nur aus Versehen gewählt und widerrufen haben, werden diese beim Ausführen des Makros wiederholt. Sie sollten deshalb vor der Aufzeichnung schon genau wissen, welche Befehle Sie ausführen lassen wollen.

Regeln für Makronamen:

- Sonderzeichen (also auch Leerzeichen) sind nicht erlaubt
- Zellbezüge (z. B. „A1“) sind nicht erlaubt
- das erste Zeichen muss ein Buchstabe sein
- Zur Abgrenzung mehrerer Wörter verwenden Sie den Unterstrich

## 1.2 Das Makro ausführen

Wählen Sie die Befehlsfolge

EXTRAS / MAKRO / MAKROS

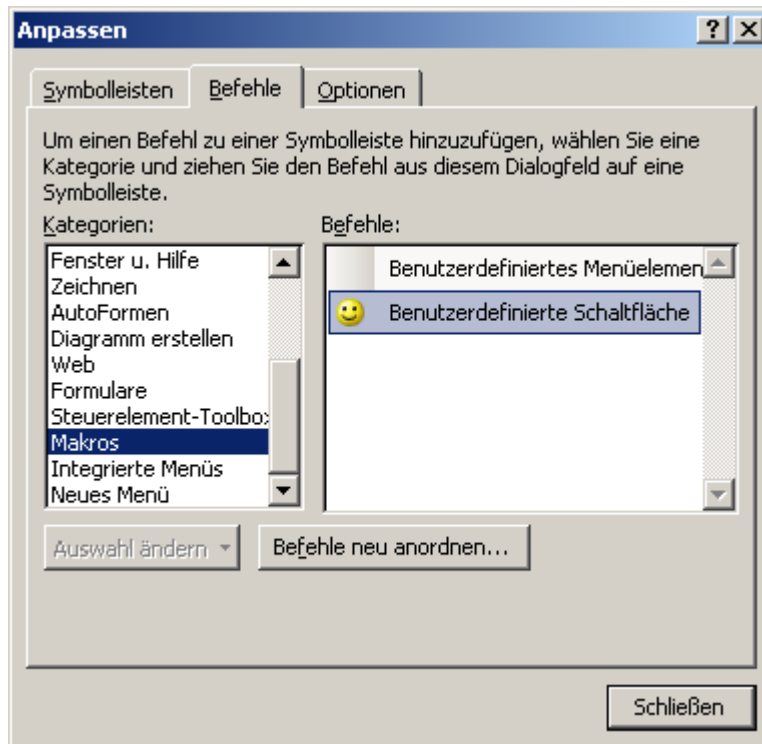
Markieren Sie das Makro „Spalte“ und anschließend auf die Schaltfläche AUSFÜHREN.

### 1.3 Das Makro mit einem Symbols starten

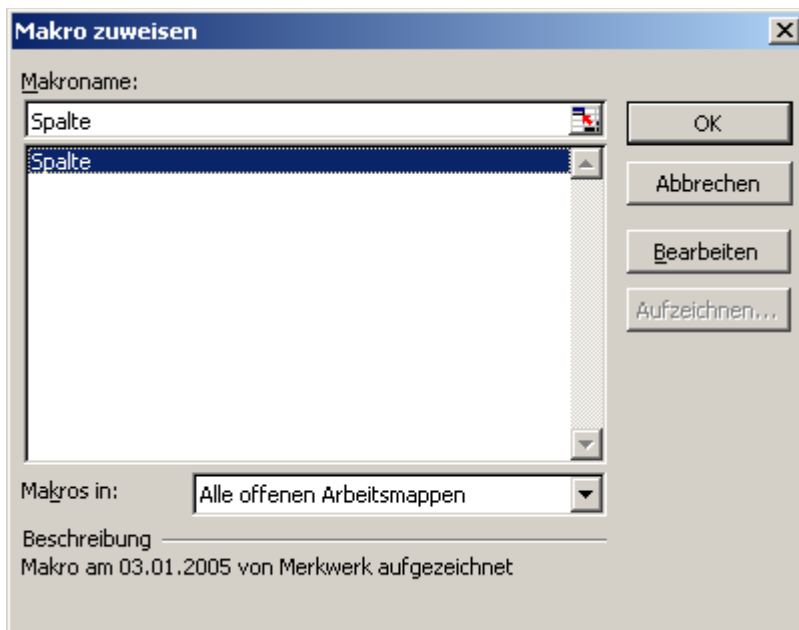
Um das neue Makro schnell über ein Symbol in der Symbolleiste starten zu können, wählen Sie den Befehl

EXTRAS / ANPASSEN

und im Register BEFEHLE die Kategorie MAKROS aus. Ziehen Sie nun den Smiley in eine Symbolleiste.

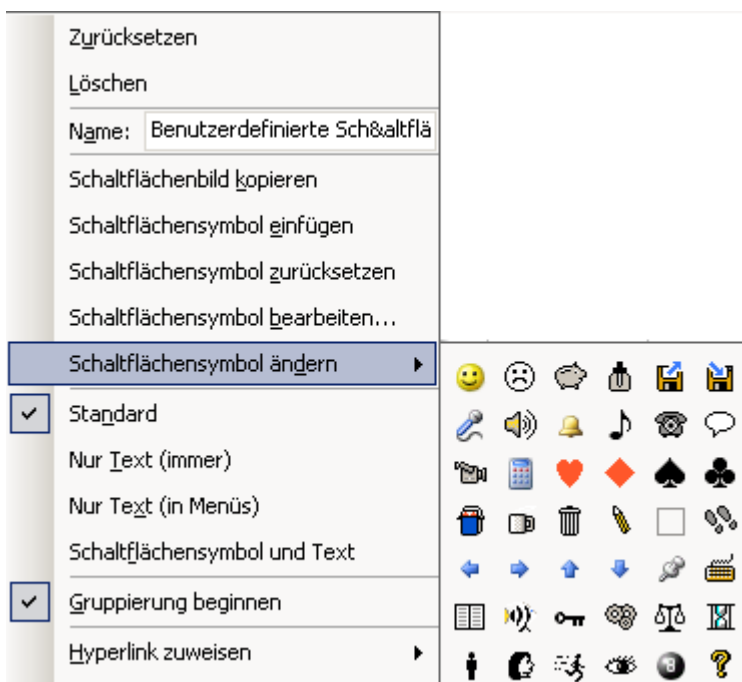


Danach schließen Sie den Dialog. Um ein Makro zuzuweisen, klicken Sie auf den Smiley in der Symbolleiste, wählen anschließend das passende Makro aus und bestätigen mit OK.



## 1.4 Symbole in Symbolleisten verändern

Um ein Symbol zu verändern, muss das Dialogfenster EXTRAS / ANPASSEN geöffnet sein. Klicken Sie dann mit der rechten Maustaste auf das zu verändernde Symbol. Im Kontextmenü können Sie über SCHALTFLÄCHENSYMBOL ÄNDERN ein neues Symbol auswählen.



Sie können auch nur einen Text anzeigen, geben Sie dazu den gewünschten Text bei NAME ein. Oder Sie zeichnen Ihr Symbol selbst über SCHALTFLÄCHENSYMBOL BEARBEITEN... .

## 1.5 Das Makro mit einer Tastenkombination starten

Möchten Sie das Makro mit einer Tastenkombination belegen, können Sie das entweder bei der Aufzeichnung angeben oder später über den Befehl

EXTRAS / MAKRO / MAKROS

bei den OPTIONEN.



**HINWEISE:** Belegte Tastenkombinationen wie STRG p (Drucken) werden damit deaktiviert. Suchen Sie sich also Kombinationen aus, die von Excel standardmäßig noch nicht belegt sind.

Um das Doppelbelegen von Tasten in Kombination mit der STRG Taste zu vermeiden, haben Sie die Möglichkeit einen Großbuchstaben einzugeben.



# 2 Makro, Quellcode, Prozedur

In diesem Kapitel lernen Sie

- wo Sie den Quellcode eines Makros finden
- wie der Visual Basic Editor aufgebaut ist
- wie Sie ein Makro über den Quellcode bearbeiten können

## 2.1 Der Visual Basic Editor

Ein Makro, das mit dem Makrorecorder aufgezeichnet wurde, wird in Visual Basic übersetzt. Den Code für ein Makro ansehen kann man im Visual Basic Editor. Dorthin gelangt man über die Befehlsfolge

EXTRAS / MAKRO / MAKROS

und den Schalter BEARBEITEN.

Der Text, den man dort findet, entspricht den einzelnen Schritten, die vom Makrorecorder aufgezeichnet wurde. Die Makrosprache ist englisch. Der Code für das Makro SPALTE sieht folgendermaßen aus:

```
Sub Spalte()  
Range("A1").Select  
Selection.CurrentRegion.Select  
Selection.Columns.AutoFit  
End Sub
```

Makros werden in Modulen abgelegt. Man findet Sie im Projektfexplorer auf der linken Seite. In einem Modul können sich mehrere Makros, auch Prozeduren genannt, befinden. Eine Prozedur beginnt immer mit `sub` und endet mit `end sub`.

Grüner Text kennzeichnet Kommentare. Kommentare erläutern den Code. Weitere Informationen dazu finden Sie im Kapitel Kommentare.

## 2.2 Makros bearbeiten



**BEISPIEL:** Nach Ausführung des Makros SPALTE ist der gesamte Bereich markiert. Das soll geändert werden. Das Makros soll so angepasst werden, dass nach dem Ausführen des Makros wieder wie zu Anfang die Zelle A1 markiert ist.

Für das Markieren der Zelle A1 ist die Codezeile

```
Range("A1").Select
```

zuständig. Kopieren Sie diese Zeile an das Ende des Makros.

```
Sub Spalte()  
    Range("A1").Select  
    Selection.CurrentRegion.Select  
    Selection.Columns.AutoFit  
    Range("A1").Select  
End Sub
```

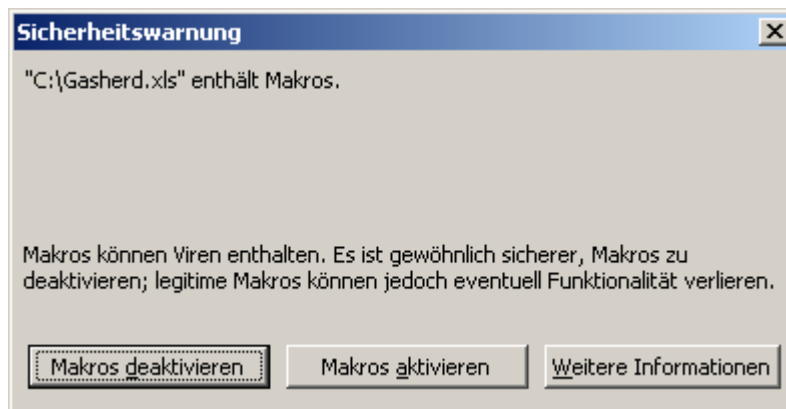
Führen Sie das Makro erneut aus.



ÜBUNGEN:

1. Im Makro Spalte soll die Zelle G8 markiert werden. Passen Sie den Code entsprechend an.
2. Zeichnen Sie ein neues Makro auf, das in einer beliebigen Zelle die Schriftgröße 20, die Schriftfarbe rot und den Schriftschnitt fett einstellt.
3. Testen Sie ihre Makros, schauen Sie diese im Editor an. Speichern und schließen Sie die Excelarbeitsmappe.

Wenn Sie eine Arbeitsmappe öffnen, die Makros enthält, erscheint folgender Dialog:



**HINWEIS:** Da Makros potentiell auch Viren enthalten können, sollten Sie diese nur aktivieren, wenn Sie sicher sind, dass die Datei virenfrei ist. Dann wählen Sie die Schaltfläche **MAKROS AKTIVIEREN**.

## 2.3 Makros kopieren

Zeichnen Sie ein neues Makro auf, das die Zelle B1 markiert. Wenn Sie sich das neue Makro im Visual Basic Editor anschauen, werden Sie

feststellen, dass die beiden Makros aus den vorherigen Übungen nicht angezeigt werden.

Diese befinden sich im Modul 1, das neue Makro im Modul 2. Excel legt beim Aufzeichnen von Makros in jeder Sitzung ein neues Modul an.

Öffnen kann man die Module mit einem Doppelklick im Projektextplorer.



ÜBUNGEN:

1. Kopieren Sie das Makro von Modul 2 in Modul 1 (aber nicht in eine vorhandene Prozedur). Löschen Sie anschließend Modul 2.
2. Erstellen Sie ein neues Makro, das die Zelle G8 grau färbt. Kopieren Sie dieses in das Makro Spalte, so dass dieses Makro zusätzlich die neue Funktion übernimmt.

## 2.4 Kommentare

Kommentare erläutern den Quellcode. Ohne eine eigene Funktion zu haben dienen Kommentare dazu, ein Programm nachvollziehbar zu machen, so dass ein Programmierer stets leicht erkennen kann, was hinter dem Code steckt. Kommentare beginnen immer mit einem Hochkomma vorn und werden im Visual Basic Editor grün dargestellt.



ÜBUNG: Kommentieren Sie den Quellcode des Makros „Spalte“.



PRAKTIKUM

Löschen Sie in der Exceldatei Gasherd.xls in der Zelle F5 den Listenpreis. Da sich in der Zelle G5 eine Formel auf diese Zelle bezieht und der Wert jetzt Null ist, wird in der Zelle G5 auch eine Null angezeigt. In der Praxis passiert das, wenn Sie eine Formel über mehrere Zeilen oder Spalten kopieren und sich die Formel auf eine nicht ausgefüllte Zelle bezieht.

Deshalb soll ein Makro erstellt werden, dass diese Nullwerte ausblendet:

Starten Sie den Makrorecorder. Blenden Sie die Nullwerte über

EXTRAS / OPTIONEN

im Register Ansicht. Beenden Sie die Aufnahme des Makrorecorders. Weisen Sie dem Makro eine Schaltfläche zu. Lassen Sie sich die Nullwerte wieder anzeigen (Befehle wie oben) und testen Sie das neue Makro NULLWERTE.

Schauen Sie sich den Quellcode an.

```
Sub Nullwerte()  
    ActiveWindow.DisplayZeros = False  
End Sub
```



ÜBUNG: Manipulieren Sie den Code so, dass die Nullwerte wieder angezeigt werden.

## 2.5 Das Schlüsselwort NOT

Um das Makro so zu ändern, dass die Nullwerte sowohl ein- als auch ausgeschaltet werden können, gibt es die Anweisung Not.

Ändern Sie den Quellcode wie unten dargestellt.

```
Sub Nullwerte()  
    ActiveWindow.DisplayZeros = Not ActiveWindow.DisplayZeros  
End Sub
```

Testen Sie das Makro.

Das Schlüsselwort Not wandelt eine Aussage von wahr in falsch oder von falsch in wahr um. Das Symbol in der Symbolleiste kann nun durch einen Klick die Nullwerte ein- und anschließend wieder ausschalten.

## 2.6 Das Schlüsselwort With

ActiveWindow legt das zu verändernde Objekt fest, in diesem Fall das aktive Fenster. Wollen Sie für dieses Objekt weitere Anweisungen programmieren, können Sie diese mit dem Schlüsselwort With zusammenfassen.

In unserem Beispiel sollen zusätzlich zu den Nullwerten auch die Gitternetzlinien ein- und ausgeblendet werden.

Das Ergebnis könnte so aussehen:

```
Sub Nullwerte()  
    ActiveWindow.DisplayZeros =Not ActiveWindow.DisplayZeros  
    ActiveWindow.DisplayGridlines = Not  
ActiveWindow.DisplayGridlines  
End Sub
```

Kürzer und überschaubarer ist die Lösung mit With:

```
Sub Nullwerte()  
    With ActiveWindow  
        .DisplayZeros = Not .DisplayZeros  
        .DisplayGridlines = Not .DisplayGridlines  
    End With
```

End Sub

Man muss hinter dem Schlüsselwort `With` das Objekt nur einmal angeben. Alle folgenden Anweisungen beziehen sich nun auf das genannte Objekt, solange bis `End With` folgt. Dies verringert den Arbeitsaufwand und macht den Quellcode übersichtlicher.

# 3 Erste eigene Programmierelemente

In diesem Kapitel lernen Sie

- wie Sie eigene Module und Prozeduren erstellen
- die Arbeitsweise einer Messagebox

## 3.1 Module und Prozeduren

In einem Modul werden Prozeduren abgespeichert. Bisher wurden die Module und Prozeduren vom Makrorecorder erstellt. Neue Module kann man im Visual Basic Editor auch selbst erzeugen. Wählen Sie die Befehlsfolge

EXTRAS / MAKRO / VISUAL BASIC EDITOR.

Sie befinden sich im Visual Basic Editor und können hier ein neues Modul erzeugen über die Befehlsfolge

EINFÜGEN / MODUL.

Alle Prozeduren beginnen mit dem Schlüsselwort `Sub` und enden mit `End sub`.

Gibt man in dem neuen Modul `Sub` ein und dahinter den Prozedurnamen (z. B. VERSUCH), wird nach Betätigung der Entertaste das `End sub` automatisch erstellt.

```
Sub versuch()  
End Sub
```

Dieses Programm kann man in Excel ausführen über

EXTRAS /MAKRO/ MAKROS

und den Schalter AUSFÜHREN, nachdem man das Makro VERSUCH gewählt hat. Das Programm wird ausgeführt, auch wenn am Bildschirm nichts sichtbar ist.

## 3.2 Die Messagebox

Eine Messagebox zeigt einen Text oder Wert am Bildschirm an. Die Messagebox wird im Visual Basic Editor folgendermaßen eingesetzt:

```
Sub versuch()
```

```
MsgBox („Hallo“)  
End Sub
```

Führen Sie das Programm in Excel aus. Sie erhalten nun folgende Meldung:

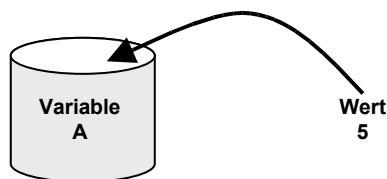


# 4 Variablen

In diesem Kapitel lernen Sie

- was Variablen sind und wozu Sie sie brauchen
- wie man Variablen einsetzt
- die Variablendeklaration

Eine Variable ist ein Platz, an dem das Programm Werte ablegt. Soll sich ein Programm eine Zahl oder einen Text merken, braucht man einen Speicherort, an dem dieser Wert abgelegt wird. Eine Variable ist ein solcher Speicherort, der beliebig befüllt werden kann.



In Visual Basic sieht das so aus:

```
A=5
```

## 4.1 Eine Variable einsetzen

Erstellen Sie eine neue Prozedur und bestimmen Sie für die Variable A den Wert 5. Mit einer Messagebox können Sie sich diesen Wert anzeigen lassen.

```
Sub versuch()  
    A = 5  
    MsgBox (A)  
End Sub
```



**HINWEIS:** A ist ein schlechter Variablenname, man sollten für Variablen immer „sprechende Namen“ verwenden.

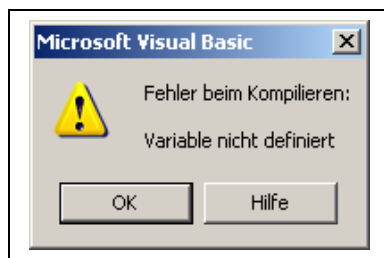
## 4.2 Die Variablendeklaration

Um Fehler zu vermeiden, sollten Variablen beim Programmieren deklariert/dimensioniert. Das sieht folgendermaßen aus:


```
Option Explicit  
Dim kundenummer
```



Der Begriff `Option Explicit` erzwingt die Variablendeklaration. Das heißt, alle Variablen, die hier nicht deklariert werden, dürfen im Programm nicht verwendet werden. Es kommt zu folgender Fehlermeldung, wenn nicht deklarierte Variablen im Programm eingesetzt werden:



`Option Explicit` steht immer in der ersten Zeile des Codes eines Moduls.

Das Programm bleibt an der Stelle stehen, wo der Fehler auftritt (gelber Pfeil), ist aber noch nicht beendet und muss mit dem ZURÜCKSETZEN Button  beendet werden.



**HINWEISE:** Durch das Deklarieren von Variablen kann man Fehler vermeiden. Am besten verwenden Sie bei der Variablendeklaration Großbuchstaben, z. B. `KundenNummer`, wenn Sie diese Variable im Programm verwenden möchten, schreiben Sie alles klein. Haben Sie die Variable richtig geschrieben, wird bei einem Zeilenwechsel die Groß- Kleinschreibung automatisch korrigiert.

Man kann die Variablendeklaration für jedes neue Modul automatisch erforderlich machen, indem man über die Befehlsfolge

EXTRAS / OPTIONEN

im Register EDITOR den Haken bei `VARIABLENDEKLARATION ERFORDERLICH` setzt.

### 4.3 Der Gültigkeitsbereich von Variablen

Wenn Variablen über sämtlichen Prozeduren in einem Objekt dimensioniert werden, sind sie auf Modul- / Formularebene verfügbar. Das heißt, innerhalb eines Formulars oder Moduls können alle Prozeduren auf diese Variablen zugreifen.

Bei jedem Einsatz des Moduls / Formulars wird Speicherplatz für diese Variablen reserviert, auch wenn die Variable bei einer Anwendung nicht gebraucht wird.

Wird eine Variable nur in einer Prozedur eingesetzt, dimensioniert man sie auch nur dort. Sobald diese Prozedur verlassen wird, wird der reservierte Arbeitsspeicher wieder freigegeben.

## 4.4 Datentypen

Bei der Deklaration der Variablen kann durch die Angabe des Datentyps die Art der Information festgelegt werden, die später in dieser Variablen gespeichert wird (z. B. nur Texte oder Ganzzahlen). Die Angabe des Datentyps bestimmt die Größe des Speicherplatzes, die für diese Variable bereitgestellt werden soll. Um nicht unnötigen Speicherplatz zu reservieren, ist es sinnvoll, Datentypen zuzuweisen.

Der Datentyp wird beim Dimensionieren nach dem Variablennamen angegeben:

dim Variablennamen as Datentyp



**HINWEISE:** Wird kein Datentyp angegeben, wird der Variablen der Datentyp Variant zu gewiesen. Variant ist ein Datentyp, der je nach Inhalt unterschiedliche Datentypen annimmt.

Die Hilfe bietet unter dem Stichwort „Datentypen“ eine gute Übersicht über die verfügbaren Datentypen, ihre Einsatzmöglichkeiten und die benötigte Speichergröße:

<b>Byte</b>	1 Byte	0 bis 255
<b>Boolean</b>	2 Bytes	<b>True</b> oder <b>False</b>
<b>Integer</b>	2 Bytes	-32.768 bis 32.767
<b>Long</b> (lange Ganzzahl)	4 Bytes	-2.147.483.648 bis 2.147.483.647
<b>Single</b> (Gleitkommazahl mit einfacher Genauigkeit)	4 Bytes	-3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte.
<b>Double</b> (Gleitkommazahl mit doppelter Genauigkeit)	8 Bytes	-1,79769313486231E308 bis -4,94065645841247E-324 für negative Werte; 4,94065645841247E-324 bis 1,79769313486232E308 für positive Werte.

<b>Currency</b> (skalierte Ganzzahl)	8 Bytes	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807
<b>Decimal</b>	14 Bytes	+/- 79.228.162.514.264.337.593.543.950.335 ohne Dezimalzeichen; +/-7,9228162514264337593543950335 mit 28 Nachkommastellen; die kleinste Zahl ungleich Null ist +/-0,00000000000000000000000000000001.
<b>Date</b>	8 Bytes	1. Januar 100 bis 31. Dezember 9999.
<b>Object</b>	4 Bytes	Beliebiger Verweis auf ein Objekt vom Typ <b>Object</b> .
<b>String</b> (variable Länge)	10 Bytes plus Zeichenfolgenlänge	0 bis ca. 2 Milliarden.
<b>String</b> (feste Länge)	Zeichenfolgenlänge	1 bis ca. 65.400
<b>Variant</b> (mit Zahlen)	16 Bytes	Numerische Werte im Bereich des Datentyps <b>Double</b> .
<b>Variant</b> (mit Zeichen)	22 Bytes plus Zeichenfolgenlänge	Wie bei <b>String</b> mit variabler Länge.

# 5 Weitere Programmierelemente

In diesem Kapitel lernen Sie

- den Einsatz der Inputbox kennen
- wie man relative Zellbezüge aufzeichnen kann
- wie man eine Excelliste nach bestimmten Werten durchsucht und die Suchergebnisse einer Variablen übergibt

## 5.1 Die Inputbox

Die Variable soll künftig nicht fix angegeben ( $a=5$ ), sondern flexibel ausfüllbar sein. Eine Inputbox ermöglicht es, dass ein Anwender in Excel eine Eingabe vornehmen kann. Der Wert einer Inputbox muss einer Variablen übergeben werden. Eine Inputbox wird in Visual Basic folgendermaßen eingesetzt:

```
Sub versuch()  
    Nummer=InputBox("bitte geben Sie eine Nummer ein")  
End Sub
```



**BEISPIEL:** Der Anwender soll seinen Nachnamen eingeben können. Dieser wird dann in einer Messagebox angezeigt.

```
Option Explicit  
Dim NachName  
  
Sub versuch()  
    NachName = InputBox("bitte geben Sie Ihren Namen ein")  
    MsgBox (NachName)  
End Sub
```



### PRAKTIKUM

Das bisher Gelernte soll nun Beispiel mit der Gasherd-Datei geübt und vertieft werden.

Ziel wird es sein, ein Programm zu schreiben, bei dem ein Anwender über eine Kundennummer die passende Firma findet und bei Bedarf diese Daten in einen Brief schreiben kann. Den Kunden sollen zusätzlich zu ihrem jeweiligen Rabatt nochmals 3 % Nachlass gewährt werden. Der Sachbearbeiter bestimmt eine Kundennummer, aufgrund der Excel die Firma, den Ansprechpartner und den aktuellen Endpreis aus der Liste heraussucht. Auf Wunsch kann der Sachbearbeiter dann einen Brief an den Kunden vom Programm erzeugen lassen. In diesem Brief sollen die Firma, der Ansprechpartner und der Rabattpreis automatisch aufgenommen werden.

In einem ersten Schritt gibt der Sachbearbeiter die Kundennummer in eine Inputbox ein.

```
KundenNummer = InputBox("Bitte Kundennummer eingeben")
```

Der Wert soll anschließend in einer Messagebox angezeigt werden.

```
Option Explicit
'option explicit erzwingt die Variablendeklaration, d.h.
'bevor eine Variable benutzt werden kann, muss sie dem System
'bekannt gemacht werden
'bekannt machen tut man mit dem Schlüsselwort Dim, siehe Kommentar
'nächste Zeile

Dim KundenNummer
'ab jetzt darf die Variable Kundennummer verwendet werden
'hier wurde bewusst mit Groß- und Kleinschreibung gearbeitet,
'das dient als Rechtschreibprüfung, wenn die Variable später
'auftaucht

Sub finden()

    KundenNummer = InputBox("Bitte Kundennummer eingeben")
    'Der Variablen KundenNummer wird der Wert der Inputbox
übergeben
    'Den Wert der Inputbox schreibt der Benutzer rein

    MsgBox (KundenNummer)
    'Der Benutzer bekommt eine Meldung am Bildschirm mit dem
'Wert der Variablen KundenNummer

End Sub
```

Das Programm soll über ein Symbol in der Symbolleiste gestartet werden können.

## 5.2 Eine Excelliste durchsuchen

Die Werte, die der Sachbearbeiter eingegeben hat, sollen in der Excelliste gesucht werden. Der Zellzeiger soll in die entsprechenden Zelle springen.

In Excel findet man Zelleninhalte über den Befehl

BEARBEITEN / SUCHEN

Suchen Sie nach „1006“ und lassen dabei den Makrorecorder mitlaufen. Es wird folgender Code erzeugt:

```
Sub Makro1()
```

```

Cells.Find(What:="1006", After:=ActiveCell,
LookIn:=xlFormulas, _
    LookAt:=xlPart, SearchOrder:=xlByRows, _
    SearchDirection:=xlNext, MatchCase:=_
    False, SearchFormat:=False).Activate
End Sub

```

Der Code ist sehr lange, da alle Optionen des Suchen-Dialogs übernommen werden, auch wenn diese Optionen gar nicht genutzt wurden. Überflüssige Informationen kann man herauslöschen und die Zeile in die Prozedur „finden“ kopieren.

```
Cells.Find("1006").Activate
```

Nun immer allerdings wird nur die Nummer 1006 in Excel gesucht. Deshalb muss die Nummer in der Klammer durch die Variable KundenNummer ersetzt werden:

```

Sub finden()

    KundenNummer = InputBox("Bitte Kundennummer eingeben")

    Cells.Find(KundenNummer).Activate
    'diese Zeile stammt vom Makrorecorder, mit Bearbeiten/Suchen
    'wurde nach 1006 gesucht;
    'der Makrorecorder hat alle Optionen aus dem Suchen-Dialog
    'übernommen,
    'durch großzügiges Löschen und Ausprobieren wurde der Code
    'ausgedünnt.
    'die Zahl muss durch die Variable KundenNummer ersetzt werden

    MsgBox (KundenNummer)

End Sub

```

### 5.3 Die relative Aufzeichnung


Nun soll im vorliegenden Beispiel nach der Eingabe der Kundennummer der Firmenname, der Ansprechpartner und der Endpreis ausgegeben werden. In einem ersten Schritt lautet die Frage: Was muss man tun, um von der Kundennummer zur Firma zu gelangen?

Man muss eine Zelle weiter nach rechts, und genau dies zeichnet man nun mit dem Makrorecorder auf. Heraus kommt dabei dies:

```
Range("B6").Select
```

Bei einem Test stellt man fest, dass das Programm immer die gleiche Zelle anspringt, in diesem Beispiel die B6. Das liegt daran, dass im Code die absolute Zelladresse steht. Der Verweis muss jedoch relativ

sein, je nach Kundennummer immer eine Zelle innerhalb der Zeile nach rechts.

Einen relativen Verweis bestimmt man noch vor der Aufzeichnung in der Symbolleiste des Makrorecorders mit dem Schalter RELATIVER VERWEIS .

Das Ergebnis sieht nun so aus:

```
ActiveCell.Offset(0, 1).Range("A1").Select
```

Die erste Zahl in der Klammer besagt, wie viele Zeile man nach unten gehen soll, die zweite Zahl, wie viele Zeilen nach rechts.

Kopiert man diese Zeile in die Prozedur „finden“, ergibt das folgendes Programm:

```
Sub finden()  
  
    KundenNummer = InputBox("Bitte Kundennummer eingeben")  
  
    Cells.Find(KundenNummer).Activate  
  
    ActiveCell.Offset(0, 1).Select  
    'Der Curser springt von der ausgewählten Kundennummer 0 Zeilen  
    'nach unten und eine Zelle nach rechts  
  
    MsgBox (KundenNummer)  
  
End Sub
```

## 5.4 Suchergebnisse übergeben

Excel soll sich nun auch noch merken, welcher Inhalt in der angesprochenen Zelle steht. Dazu bedarf es einer Variablen, die zuerst dimensioniert wird.

Der neuen Variablen FiRma wird das Suchergebnis zugewiesen über:

```
FiRma = Selection  
'Der Variablen FiRma wird der Inhalt der markierten Zelle  
'zugewiesen
```

Eine Messagebox soll nun noch die Variable FiRma ausgeben.

```
MsgBox (FiRma)
```



**ÜBUNG:** Nun soll auch noch der Ansprechpartner herausgefunden und in einer Messagebox ausgegeben werden. (Den Code muss man dafür nicht nochmals aufzeichnen, sondern nur entsprechend kopieren und anpassen.)

## 5.5 Texte kombinieren

Man kann in einer MessageBox verschiedene Variablen und Texte zusammensetzen. Texte stehen dabei immer in Anführungszeichen, jedes einzelne Element muss mit einem & verknüpft sein, vor und hinter dem ein Leerzeichen eingefügt wurde.

Soll in der MessageBox stehen „Herr Geiger von der Firma Elektro Fritz“, muss das folgendermaßen programmiert werden:

```
MsgBox (AnsprechPartner & " von der Firma " & Firma)
```



**ÜBUNG:** Nun soll auch der Preis aus der Excelliste gesucht und in der MessageBox angegeben werden.



# 6 Exkurs: Die Hilfe

In diesem Kapitel lernen Sie

- wie Sie die Visual Basic Hilfe sinnvoll einsetzen können
- wie man einen Zeilenumbruch in einer Messagebox erzeugen kann

In der Messagebox soll ein Zeilenumbruch eingefügt werden. Um herauszubekommen, wie das funktioniert, kann man sich der Visual Basic Hilfe bedienen.

Zur Hilfe im Visual Basic Editor gelangt man über die Funktionstaste F1. Stellt man den Zellzeiger auf ein bestimmtes Schlüsselwort (z. B. `msgBox`) im Quellcode, bekommt man Informationen zu genau diesem Thema.

So erfährt man unter anderem, wie man mehrere Zeilen in der Messagebox erzeugt:

<i>prompt</i>	Erforderlich. Ein <a href="#">Zeichenfolgenausdruck</a> , der als Meldung im Dialogfeld erscheint. Die Maximallänge von <i>prompt</i> ist - je nach Breite der verwendeten Zeichen - etwa 1024 Zeichen. <u>Wenn <i>prompt</i> aus mehreren Zeilen besteht, müssen Sie die Zeilen mit einem Wagenrücklaufzeichen (<b>Chr(13)</b>), einem Zeilenvorschubzeichen (<b>Chr(10)</b>) oder einer Kombination aus Wagenrücklaufzeichen und Zeilenvorschubzeichen (<b>Chr(13) &amp; Chr(10)</b>) trennen.</u>
---------------	--

Demnach kann man durch Einfügen von `& Chr(13) & Chr(10) &` einen Zeilenumbruch in einer Messagebox erzeugen.



**HINWEIS:** Üblicher und kürzer ist allerdings der Code `& vbCrLf & . vbCrLf` erzeugt das gleiche wie oben, nämlich ein Wagenrücklaufzeichen (Carriage Return) und ein Zeilenvorschubzeichen (Line Feed)

```
MsgBox ("Ansprechpartner ist " & AnsprechPartner & vbCrLf & _  
....."von der Firma " & Firma)  
' vbCrLf erzeugt einen Zeilenumbruch in der Messagebox  
' carriage return (Wagenrücklauf) line feed (Zeilenvorschub)
```



**ÜBUNG:** Die Messagebox der Prozedur „finden“ soll wie unten aussehen:



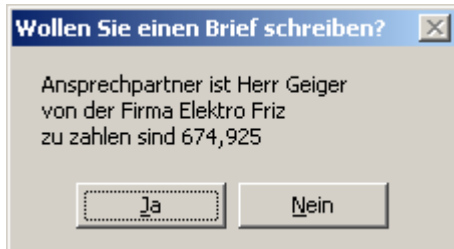


**HINWEIS:** Wird eine Codezeile zu lang und daher unübersichtlich, kann man einen Zeilenumbruch erzeugen, indem man am Ende der Zeile ein Leerzeichen und einen Unterstrich einfügt. Für Visual Basic steht der Code dann weiterhin in einer Zeile.

```
MsgBox("Ansprechpartner ist " & AnsprechPartner _  
      & vbCrLf & "von der Firma " & Firma)
```

## 6.1 Rückgabewerte der Messagebox

In der Messagebox soll es künftig zwei Schalter geben, JA zum Bestätigen, dass ein Brief erstellt werden soll und NEIN, um keinen Brief zu schreiben. Außerdem soll in der Titelleiste stehen: *Wollen Sie einen Brief schreiben?*



Die Syntax und die Werte der verschiedenen Schalter werden in der Hilfe erklärt:

MsgBox(prompt[, buttons] [, title] [, helpfile, context])

<b>vbOKOnly</b>	0	Nur die Schaltfläche <b>OK</b> anzeigen.
<b>VbOKCancel</b>	1	Schaltflächen <b>OK</b> und <b>Abbrechen</b> anzeigen.
<b>VbAbortRetryIgnore</b>	2	Schaltflächen <b>Abbruch</b> , <b>Wiederholen</b> und <b>Ignorieren</b> anzeigen.
<b>VbYesNoCancel</b>	3	Schaltflächen <b>Ja</b> , <b>Nein</b> und <b>Abbrechen</b> anzeigen.

<b>VbYesNo</b>	4	Schaltflächen <b>Ja</b> und <b>Nein</b> anzeigen.
<b>VbRetryCancel</b>	5	Schaltflächen <b>Wiederholen</b> und <b>Abbrechen</b> anzeigen

Beim Klick auf einen der beiden Buttons JA oder NEIN gibt es unterschiedliche Rückgabewerte, die in eine Variable geschrieben werden.

<b>vbOK</b>	1	<b>OK</b>
<b>vbCancel</b>	2	<b>Abbrechen</b>
<b>vbAbort</b>	3	<b>Abbruch</b>
<b>vbRetry</b>	4	<b>Wiederholen</b>
<b>vbIgnore</b>	5	<b>Ignorieren</b>
<b>vbYes</b>	6	<b>Ja</b>
<b>vbNo</b>	7	<b>Nein</b>

Die Umsetzung in Code sieht folgendermaßen aus:

```
RetVal = MsgBox("Ansprechpartner ist " & AnsprechPartner _
    & vbCrLf & "von der Firma " & FiRma _
    & vbCrLf & "zu zahlen sind " & EndPreis _
    , 4, "Wollen Sie einen Brief schreiben?")
```

Statt der Werte kann man auch die Konstante verwenden.

```
RetVal = MsgBox("Ansprechpartner ist " & AnsprechPartner _
    & vbCrLf & "von der Firma " & FiRma _
    & vbCrLf & "zu zahlen sind " & EndPreis _
    , vbYesNo, "Wollen Sie einen Brief schreiben?")
```

Um den Code lesbarer zu machen, sollten den Konstanten der Vorzug vor den Werten gegeben werden.

# 7 Verschiedene Funktionen

In diesem Kapitel lernen Sie

- die if-Anweisung kennen
- die mid-Funktion kennen

## 7.1 Die if-Anweisung

Die neue Messagebox hat nun zwei Schalter, bei denen jeweils andere Aktionen ausgeführt werden sollen. Klickt man auf JA, soll eine neue Prozedur aufgerufen werden, die das Briefformular befüllt, klickt man auf NEIN, soll nichts passieren.

Eine solche Fallunterscheidung realisiert man mit einer if-Anweisung. Mit ihrer Hilfe können zwei oder mehr Bedingungen mit verschiedenen Anweisungen versehen werden.

Es gibt zwei Arten der Syntax für die if-Anweisung, die beide in der Hilfe zu finden sind. Bei der Ersten kann man nur zwei Fälle unterscheiden, die Zweite ermöglicht die Unterscheidung mehrerer Bedingungen:

```
If Bedingung Then [Anweisungen] [Else elseAnweisungen]
```

Alternativ können Sie die Block-Syntax verwenden:

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

Das sieht dann entweder so aus:

```
IfRetVal = vbYes Then Call BriefSchreiben
```

oder mit der Block-Syntax:

```
IfRetVal = vbYes Then  
    Call BriefSchreiben  
End If
```

Um die Prozedur ausprobieren zu können, braucht es eine zweite Prozedur BriefSchreiben. Zum Testen erscheint dort lediglich eine MessageBox.

```
Sub BriefSchreiben()  
    MsgBox ("Hallo")  
End Sub
```



#### PRAKTIKUM

Nun soll die Prozedur BriefSchreiben erstellt werden. Um die einzelnen Schritte analytisch zu erfassen, schreibt man am besten so genannten Pseudocode, die einzelnen Schritte werden als Kommentar geschrieben. Das erleichtert später das Programmieren.

```
Sub BriefSchreiben()  
  
    'zum Tabellenblatt Brief wechseln  
  
    'die Zelle B1 markieren  
  
    'den Wert der Variablen FiRma eintragen  
  
    'die Zelle B2 markieren  
  
    'den Wert der Variablen AnsprechPartner eintragen  
  
    'die Zelle B10 markieren  
  
    'den Wert von EndPreis eintragen  
  
    'die Zelle B11 markieren  
  
    'EndPreis * 0.97 einfügen  
  
End Sub
```

Sofern der Makrorecorder diese Aufgaben lösen kann, werden die Schritte einzeln aufgezeichnet und der Code an die entsprechende Stelle kopiert. Wertzuweisungen und Berechnungen werden von Hand programmiert:

```
Sub BriefSchreiben()  
  
    'zum Tabellenblatt Brief wechseln  
    Sheets("Brief").Select  
    'die Zelle B1 markieren  
    Range("B1").Select  
    'den Wert der Variablen FiRma eintragen  
    Selection = FiRma  
    'die Zelle B2 markieren  
    Range("B2").Select  
    'den Wert der Variablen AnsprechPartner eintragen  
    Selection = AnsprechPartner  
    'die Zelle B10 markieren  
    Range("B10").Select
```

```
'den Wert von EndPreis eintragen
Selection = EndPreis
'die Zelle B11 markieren
Range("B11").Select
'EndPreis * 0.97 einfügen
Selection = EndPreis * 0.97
```

```
End Sub
```

Wenn man das Makro „finden“ vom Arbeitsblatt Brief aus startet, bekommt man eine Fehlermeldung. Deshalb sollte man dafür sorgen, dass immer zuerst in das Arbeitsblatt Gasherd gewechselt wird.

```
Sub finden()
Sheets("Gasherd").Select
'Beim Klick auf den Makroschalter wird immer das Tabellenblatt
'Gasherd ausgewählt
```

## 7.2 Die mid-Funktion

Nun soll in einem weiteren Feld eine Anrede der Ansprechpartner eingetragen werden. Je nach Geschlecht soll in der Zelle A5 „Sehr geehrte Frau“ und der Name oder „Sehr geehrter Herr“ und der Name stehen.

Um diese Unterscheidung treffen zu können, muss man in der Variablen Ansprechpartner prüfen, ob dort Frau oder Herr steht. Dies leistet die mid-Funktion. Sie prüft Text und gibt die Ergebnisse einer Variablen zurück (siehe dazu die Hilfe).

```
GeSchlecht = Mid(AnsprechPartner, 1, 1)

'In die Variable GeSchlecht wird das Ergebnis der Mid-Funktion
'geschrieben
'In der Klammer steht der zu prüfende Text, `hier die Variable
'AnsprechPartner
'Das erste 1 besagt, dass das Auslesen beim ersten Buchstaben
'des Textes beginnt
'Die zweite 1 prüft die Buchstaben in einer Länge von 1, also
'nur den ersten Buchstaben
```

Nun muss noch die Zelle angegeben werden, in die die Anrede eingetragen werden soll. Vor dem Eintrag muss mit einer if-Anweisung geprüft werden, ob der Ansprechpartner mit einem „F“ oder einem „H“ beginnt. (Die Groß- und Kleinschreibung ist wichtig.)

```
Range("a5").Select

If GeSchlecht = "F" Then
    Selection = "Sehr geehrte " & AnsprechPartner & ", "
```

```
Else
    Selection = "Sehr geehrter " & AnsprechPartner & ", "
```

```
End If
```



ÜBUNG:

Wenn nun in einem Anredefeld Firma statt Frau steht, würde eine falsche Anrede im Brief stehen. In einem solchen Fall soll dort künftig „*Sehr geehrte Damen und Herren*“ stehen.

# 8 Debuggen

In diesem Kapitel lernen Sie

- wie ein Programm den Code durchläuft
- wie man Fehler im Code aufspürt

Wenn man ein Programm ausführt, sieht man nur das Ergebnis am Bildschirm. Man kann die einzelnen Schritte eines Programms aber auch einzeln durchlaufen/debuggen. Dabei sieht man genau, welche Codezeilen das Programm wann durchläuft, wann es welche Werte annimmt und wo eventuelle Fehler auftreten.

Dazu verwendet man die Funktionstaste F8 im VB-Editor. Damit wird das Programm, in welchem sich der Cursor befindet, im Einzelschrittmodus Zeile für Zeile durchlaufen. Stellt man den Mauszeiger auf die Variablen, sieht man, welcher aktuelle Wert dieser Variablen übergeben wurde.

Will man das Programm an einer bestimmten Stelle prüfen, setzt man dort mit F9 einen Haltepunkt. Läuft das Programm während der Ausführung in diese Zeile, wird es an dieser Stelle unterbrochen und kann bei Bedarf mit F8 im Einzelschrittmodus weitergeführt werden. Nochmaliges Betätigen der Taste F9 entfernt den Haltepunkt wieder.



**ÜBUNG:** Debuggen Sie in Gasherd.xls die Routine „Finden“. Wann wird eine Codezeile ausgeführt, wann werden den Variablen die Werte übergeben? Prüfen Sie nach jedem Schritt mit F8 die Auswirkungen in der Exceltabelle.



# 9 Ausgabeformat ändern

In diesem Kapitel lernen Sie

→ wie man Zahlenformate in der MessageBox anpassen kann

Künftig soll der Betrag in der MessageBox mit zwei Nachkommastellen dargestellt und mit einem Eurozeichen versehen werden. Dies kann man über die Funktion Format einstellen:

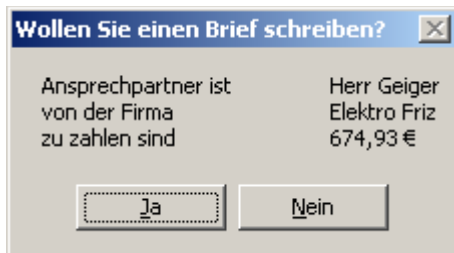
```
RetVal = MsgBox("Ansprechpartner ist " & AnsprechPartner _  
    & vbCrLf & "von der Firma " & FiRma _  
    & vbCrLf & "zu zahlen sind " & _  
    Format(EndPreis, "0.00") & " €" _  
    , vbYesNo, "Wollen Sie einen Brief  
schreiben?")
```

oder auch:

```
...Format(EndPreis, "currency")
```



**ÜBUNG:** Die Konstante vbTab fügt einen Tab ein.  
Gestalten Sie die Dialogbox folgendermaßen:



# 10 Programmieren eines Formulars

In diesem Kapitel lernen Sie

- benutzerdefinierte Formulare kennen
- Steuerelemente wie Drehfelder, Textfelder, Optionsschaltflächen und Kombinationsfelder kennen
- wie man diese Steuerelemente mit Prozeduren verknüpft

Im folgenden Beispiel soll ein Formular entwickelt werden, das dem Benutzer Eingaben ermöglicht.

Konkret soll für ein Reisebüro ein Formular erstellt werden (Park.xls). Dabei sollen mehrere Optionen wählbar sein.

- die Anzahl der Personen
- die Art des Eintritts: ob der Kunde von vorn herein alle Attraktionen buchen will, nur den Eintritt oder Eintritt mit Verpflegung
- ob ein Hotel mitgebucht werden soll

The screenshot shows a window titled "Bestellformular" with a close button in the top right corner. The window contains a form with the following elements:

- A section titled "Hotel buchen?" with two radio buttons labeled "Ja" and "Nein".
- Two buttons labeled "Bestellen" and "Abbrechen" positioned to the right of the radio buttons.
- A section titled "Auswahl" with a dropdown menu.
- A section titled "Anzahl der Personen" with a numeric spinner control.

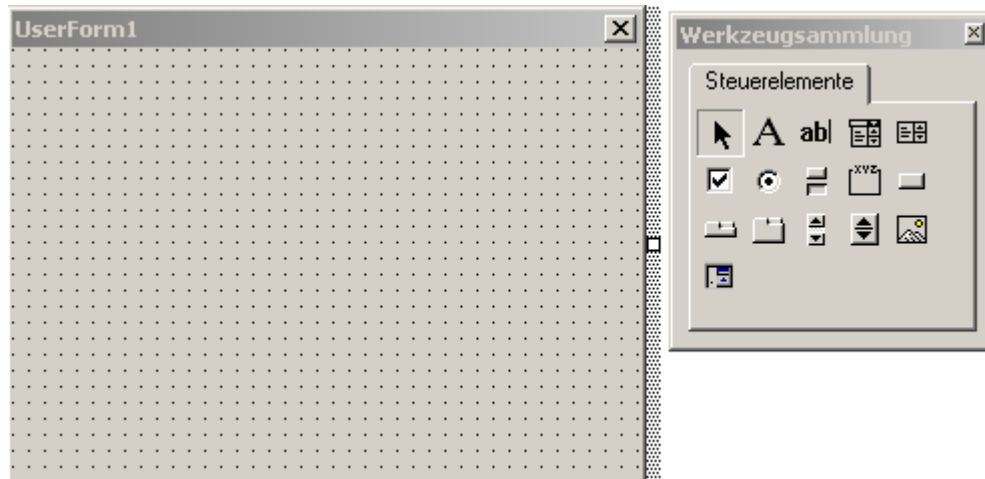
In Abhängigkeit der verschiedenen Optionen verändert sich der Gesamtpreis der Bestellung.

Nach der Auswahl der verschiedenen Optionen und der Anzahl der Personen soll eine Briefvorlage mit dem zu zahlenden Betrag gefüllt werden.

## 10.1 Das Formular erzeugen

Um ein Formular einzufügen, wählen Sie im VB-Editor den Befehl

EINFÜGEN / USERFORM



## 10.2 Steuerelemente einfügen

Ist das Formular markiert, befindet sich daneben das Fenster Werkzeugsammlung (oder ANSICHT / WERKZEUGSAMMLUNG).



Darin befinden sich unter anderem folgende Symbole.

### Steuerelement

Befehlsschaltfläche (*commandbutton*)

Optionsschaltfläche (*optionbutton*)

Rahmen (*frame*)

Kombinationsfeld (*combobox*)

### Bezeichnung im Code

cmdName

optName

fraName

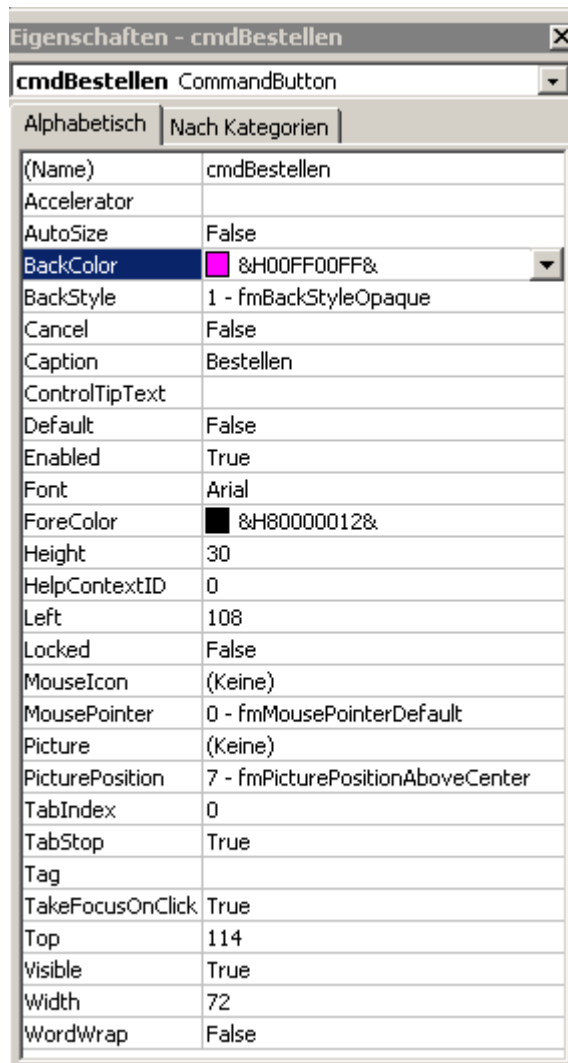
cboName

Textfeld (*textbox*)  
Bezeichnungsfeld (*label*)  
Drehfeld (*spinbutton*)  
Kontrollkästchen (*checkbox*)

txtName  
lblName  
spiName  
chkName

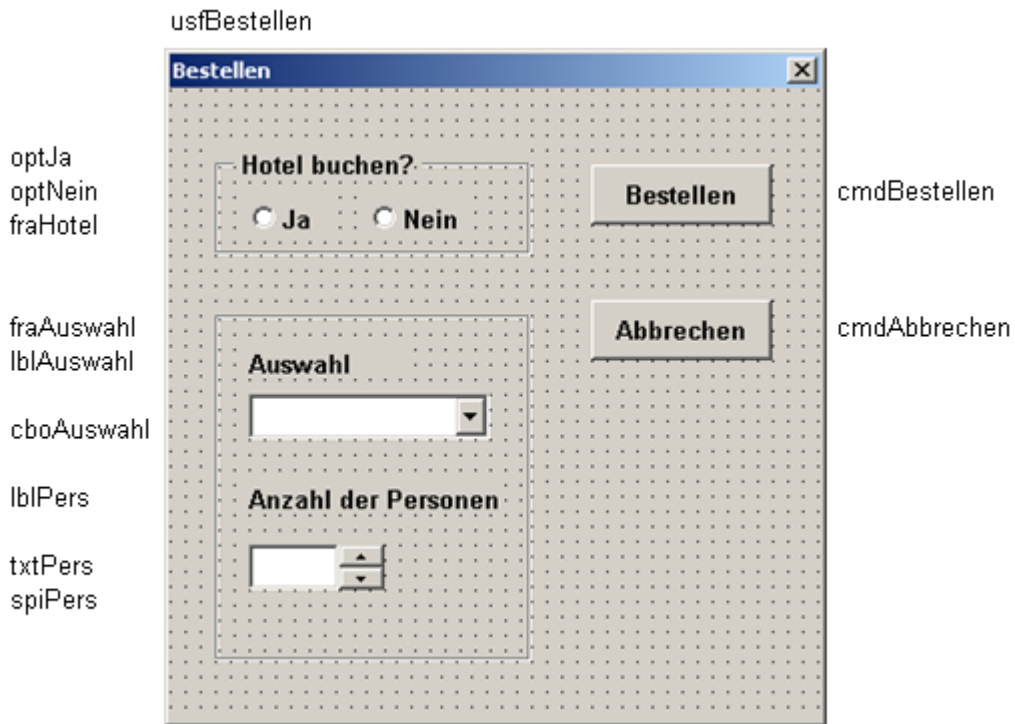
## 10.3Eigenschaften

Jedes Steuerelement hat verschiedene Eigenschaften, die man im Eigenschaftfenster einstellen kann. Dazu gehören unter anderem die Farbe, Schriftart, Größe oder der Name eines Objektes.



### PRAKTIKUM

Die einzelnen Steuerelemente sollen wie unten abgebildet erzeugt und benannt werden:



### 10.3.1 Rahmen einfügen

Ein Rahmen mit der Beschriftung „Hotel buchen?“ soll die beiden Optionsschaltflächen JA und NEIN umfassen und zusammenfassen. Dazu wählen Sie das Symbol „Rahmen“



und ziehen dann einen Rahmen im Formular auf.

Die Eigenschaft CAPTION ist der Text, den der Benutzer im Formular sieht (Hotel buchen?). Über die Eigenschaft NAME wird das Objekt im Code angesprochen (usfHotel).

### 10.3.2 Optionfelder einfügen

Mit zwei Schaltflächen soll unterschieden werden, ob ein Hotel gebucht werden soll oder nicht. Um ein Optionfeld einzufügen, klicken Sie auf folgendes Symbol



und ziehen ein Rechteck im Rahmen „fraHotel“ auf. Kopieren Sie die Schaltfläche indem Sie sie mit gehaltener STRG - Taste am Rahmen

ziehen. Vergeben Sie die Bezeichnungen „Ja“ und „Nein“ und die Namen optJa und optNein.

### 10.3.3 Bezeichnungsfeld einfügen

Erstellen Sie ein Bezeichnungsfeld mit dem Titel „Auswahl“ über den Schalter



### 10.3.4 Kombinationsfeld einfügen

Der Preis soll von der gewählten Auswahl abhängen. Die Auswahl wird mit einem Kombinationsfeld getätigt, das Sie über folgendes Symbol erstellen:



### 10.3.5 Drehfeld erstellen

Die Anzahl der Personen soll über ein so genanntes Drehfeld eingegeben werden. Mit den beiden Pfeilen am rechten Rand wird die Anzahl mit der Maus nach oben bzw. unten verändert. Das Drehfeld wird mit folgendem Symbol eingefügt:



Um zu sehen, welche Zahl eingestellt wurde, bedarf es eines weiteren Steuerelementes, dem Textfeld.

### 10.3.6 Textfeld erstellen

Ein Textfeld erstellt man über das Symbol



Als Beschriftung für die beiden Elemente Textfeld und Drehfeld soll darüber ein Bezeichnungsfeld eingefügt werden, das die Benennung „Anzahl der Personen“ bekommt.

## 10.4 Steuerelemente verändern

Wenn Sie ein Steuerelement nachträglich in Größe und Position verändern wollen, so müssen Sie es zuerst markieren. Das gewählte Objekt wird mit einem Rahmen und Markierungspunkten versehen. Am Rahmen können Sie die Position, an den Markierungspunkten die Größe des Objekts verändern.

Schriftarten, Farben und weitere Eigenschaften können Sie für jedes einzelne Steuerelement im Eigenschaftenfenster einstellen.

## 10.5 Dem Kombinationsfeld Listentext zuordnen

Beim Klick auf den Pfeil des Kombinationsfeldes sollen die drei möglichen Auswahlmöglichkeiten (Alle Attraktionen, Nur Eintritt, Eintritt und Verpflegung) erscheinen. Diese weist man über die Eigenschaft „RowSource“ zu. Dort gibt man an, in welchem Tabellenblatt und in welchen Zellen die entsprechenden Texte stehen.

## 10.6 Steuerelemente programmieren

Mit einem Doppelklick auf die Steuerelemente des Formulars im VB-Editor gelangt man in das Codefenster. Der Code wird im entsprechenden Formular abgelegt.

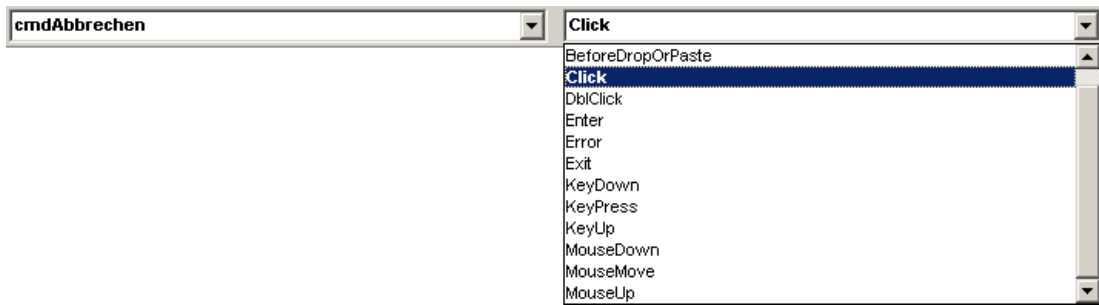


**HINWEIS:** Man kann schnell zwischen dem Code und dem Objektfenster wechseln, indem man die beiden Schalter **CODE ANZEIGEN** und **OBJEKT ANZEIGEN** oben links nutzt:



### 10.6.1 Ereignisse

Es gibt verschiedene Ereignisse, die das Programm starten. Ereignisse können Mausklicks oder Mausbewegungen oder das Betätigen von Tasten sein. Im Praktikum soll das Programm des Schalters Abbrechen bei einem Klick auf den Schalter ausgelöst werden.



## 10.6.2 Den Abbrechenschalter programmieren

Der Befehl `Unload Me` schließt ein Formular.

```
Private Sub cmdAbbrechen_Click()
    Unload Me
End Sub
```

## 10.6.3 Verknüpfung des Wertes von Drehfeld und Textfeld

Jedes Steuerelement hat verschiedene Eigenschaften. Die gewünschte Eigenschaft wird im Code direkt nach dem Elementnamen angegeben. Werte werden in der Eigenschaft `.value` abgelegt.

```
Private Sub spiPers_Change()
    txtPers.Value = spiPers.Value
End Sub
```



**HINWEIS:** Mit der Tastenkombination STRG + Leertaste kann man den angefangenen Text einer Variable oder einer Eigenschaft vervollständigen lassen. So lassen sich Schreibfehler vermeiden.

## 10.6.4 Standardeinstellungen des Dialogs

Beim Starten des Programms sollen im Formular bestimmte Werte voreingestellt sein. Zum Beispiel soll die Anzahl der Personen standardmäßig auf 2 gesetzt sein.

Das Ereignis, das bei jedem Start als Erstes ausgeführt wird, heißt `Initialize`.

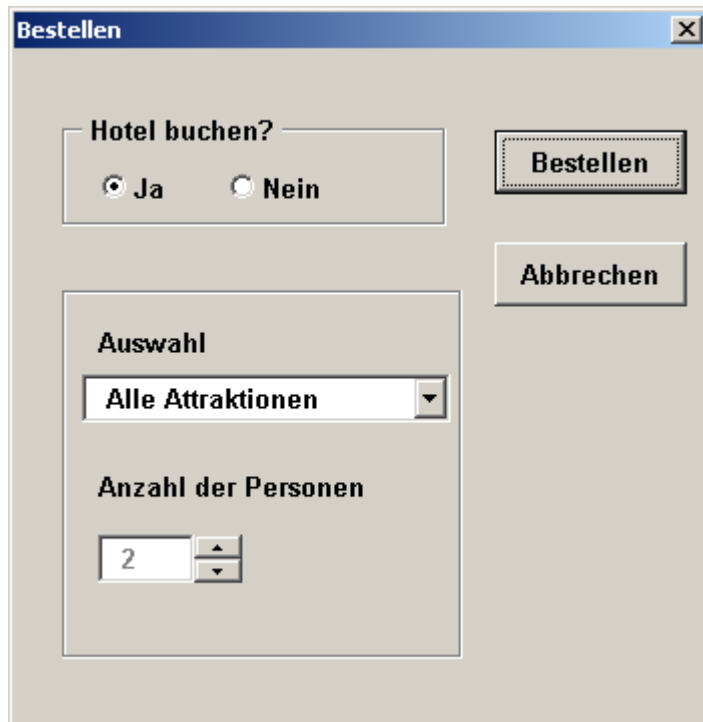
```
Private Sub UserForm_Initialize()
    spiPers.Value = 2
End Sub
```



**ÜBUNG:** Ändern Sie die Standeinstellungen für das Dialogfenster wie unten dargestellt. Den Wert des Textfeldes soll man nur



über das Drehfeld verändern können. Er soll die 1 nicht unterschreiten und die 20 nicht überschreiten können.



## 10.7 Gesamtpreis berechnen und ausgeben

Für die Berechnung sollen im Beispiel folgende Preise gelten:

Hotel	180
Alle Attraktionen	120
Nur Eintritt	25
Eintritt mit Verpflegung	80

Der Gesamtpreis berechnet sich aus:

$(Hotel + Auswahl\ der\ Attraktion) * Anzahl\ der\ Personen$

Eine neue Prozedur „Berechnen“ soll nun prüfen, welche Optionen gewählt wurden und welcher Gesamtpreis sich daraus ergibt.

Zuerst wird der Pseudocode geschrieben:

```
Sub Berechnen()  
  
    'Der Gesamtpreis ergibt sich aus (Hotel+Auswahl)*Anzahl der  
    'Personen  
  
    'Die Variable Hotel hängt ab von optJa.  
    'Wenn der Wert .Value=true ist, ergibt sich 180, sonst 0  
  
    'Die Variable Auswahl ergibt sich aus dem Listindex von
```

```

'cboAuswahl
'Ist der Listindex=0, dann ist der Wert 120, bei 1 wird er 25,
'bei 2 wird er 80

'Die Anzahl der Personen ist gleich dem Wert von spiPers

'Der Gesamtpreis ergibt sich aus (Hotel+Auswahl)*Anzahl der
'Personen

'Der Gesamtpreis soll in einer Messagebox erscheinen

End Sub

```

So fällt es leichter, das Programm zu codieren:

```

Sub Berechnen()

    If optJa.Value = True Then
        Hotel = 180
    Else
        Hotel = 0
    End If
    'Die Variable Hotel hängt ab von optJa.
    'Wenn der Wert .Value=true ist, ergibt sich 180, sonst 0

    If cboAuswahl.ListIndex = 0 Then
        Auswahl = 120
    ElseIf cboAuswahl.ListIndex = 1 Then
        Auswahl = 25
    Else
        Auswahl = 80
    End If
    'Die Variable Auswahl ergibt sich aus dem Listindex von
    'cboAuswahl
    'Ist der Listindex=0, dann ist der Wert 120, bei 1 wird er 25,
    'bei 2 wirds 80

    AnzahlPersonen = spiPers.Value
    'Die Anzahl der Personen ist gleich dem Wert von spiPers

    Gesamt = (Hotel + Auswahl) * AnzahlPersonen
    'Der Gesamtpreis ergibt sich aus (Hotel+Auswahl)*Anzahl der
    'Personen

    MsgBox (Gesamt)
    'Der Gesamtpreis soll in einer Messagebox erscheinen

End Sub

```

Bei der Fallunterscheidung kann man statt einer If-Funktion auch die Anweisung `select case` verwenden:

```

Select Case cboAuswahl.ListIndex
Case 0
    Auswahl = 120
Case 1
    Auswahl = 25
Case 2
    Auswahl = 80
End Select

```

Der Gesamtpreis soll nun in der Zelle D22 ausgegeben werden.

```
Range("C22").Select  
Selection = Gesamt
```

Anschließend soll das Formular geschlossen werden.

```
Unload Me
```



**ÜBUNG:** Die Anzahl der Personen soll in die Zelle C19 geschrieben werden. Wenn Hotel gewählt wird, soll in der Zelle C20 „Ja“ stehen, sonst „Nein“. Bei der Auswahl weiterer möglicher Optionen sollen Kreuze in die entsprechenden Zellen gesetzt werden.



**HINWEIS:** Mit der Tastenkombination ALT + F11 kann man zwischen dem Codefenster und der Exceltabelle hin- und herwechseln.

### 10.7.1 Die Schaltfläche „Bestellen“ programmieren

Bei Betätigen der Schaltfläche „Bestellen“ soll der Preis berechnet werden. Die Prozedur „Bestellen“ öffnet die Prozedur „Berechnen“.

```
Private Sub cmdBestellen_Click()  
    Call Berechnen  
End Sub
```



**HINWEIS:** Man kann markierte Bereiche über den Schalter BLOCK AUSKOMMENTIEREN schnell in Kommentare verwandeln. Der Schalter AUSKOMMENTIERUNG DES BLOCKS AUFHEBEN macht dies wieder rückgängig.



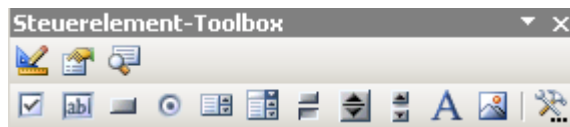
## 10.8 Das Formular anzeigen

Bisher kann man das Formular „Bestellen“ nur aus dem VB-Editor starten.

Da man künftig das Formular nur in dieser einen Tabelle angezeigt bekommen soll, arbeitet man mit einem Steuerelement (ein Schalter), das direkt in das Tabellenblatt gezeichnet wird.

## 10.8.1 Steuerelemente im Tabellenblatt

Steuerelemente findet man in der Symbolleiste STEUERELEMENT-TOOLBOX (über ANSICHT / SYMBOLLEISTE)



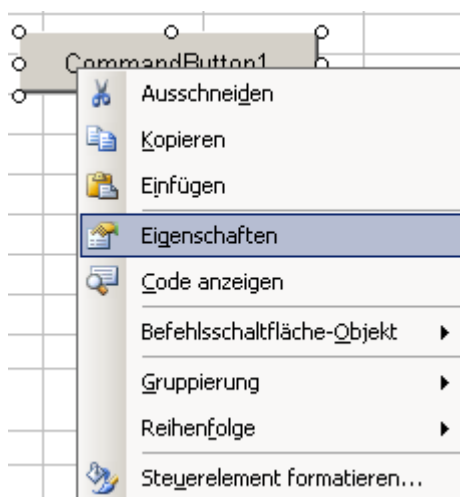
Dort wählt man per Mausklick ein Element aus und zeichnet es mit gedrückter linker Maustaste in die Exceltabelle.



Befindet man sich im Entwurfsmodus, lässt sich der Button verändern, verlässt man den Entwurfsmodus, kann man den Schalter betätigen. Man wechselt die beiden Modi über folgenden Schalter:



Die Befehlsschaltfläche umbenennen kann man über die Eigenschaften im Eigenschaftensfenster. Zu diesem gelangt man über die rechte Maustaste auf dem Steuerelement (im Entwurfsmodus).



Nun fehlt noch die Prozedur für den neuen Schalter, die bei Klick das Formular „Bestellen“ startet. Mit einem Doppelklick auf dem Schalter gelangt man in den VB-Editor. Der dazugehörige Code wird im Tabellenblatt abgelegt (bisher lag der Code entweder in einem Modul oder einem Formular).

```

Private Sub cmdDialogAnzeigen_Click()
    usfBestellen.Show
End Sub

```



**ÜBUNG:** Das Formular „Bestellen“ soll weitere Textfelder bekommen, in denen man eine Rechnungsadresse eintragen kann. Diese soll in der Exceltabelle in die Zellen A7 bis A9 eingetragen werden (*Barverkauf* wird überschrieben).

Reisebüro Heinz-Horst Müller Waldstr. 12 23432 Burgenhausen

Sandra Wolf  
Hasenstr. 12  
23434 Horsthausen

## 10.8.2 Kontrollkästchen programmieren

Die Felder der Rechnungsadresse sollen ein- und ausblendbar sein, je nachdem, ob eine Adresse angegeben werden soll oder nicht. Dazu kann man Kontrollkästchen verwenden. Diese Checkbox soll bei einem Klick den Rahmen *Rechnungsadresse* ausblenden.

```
fraAdresse.Visible = Not fraAdresse.Visible
```

The screenshot shows a Windows-style dialog box titled "Bestellen". It contains several controls:

- A group box "Hotel buchen?" with two radio buttons: "Ja" (selected) and "Nein".
- Buttons "Bestellen" and "Abbrechen" on the right side.
- A group box "Auswahl" containing a dropdown menu with "Alle Attraktionen" selected.
- A group box "Anzahl der Personen" containing a numeric spinner box with the value "2".
- A checkbox labeled "Rechnungsadresse angeben" which is checked.
- A group box "Rechnungsadresse" containing four text input fields: "Vorname", "Nachname", "Straße", "Postleitzahl", and "Ort".



#### PRAKTIKUM

Ist der Rahmen Rechnungsadresse ausgeblendet, soll das Formular entsprechend verkleinert sein.

```
Private Sub chkRechnung_Click()
    If chkRechnung.Value = True Then
        usfBestellen.Height = 445
    Else
        usfBestellen.Height = 283
    End If
End Sub
```

Initialisierung:

```
usfBestellen.Height = 445
chkRechnung.Value = True
```

oder

```
usfBestellen.Top = 100
usfBestellen.Left = 250
usfBestellen.Height = 283
chkRechnung.Value = False
```



ÜBUNG: Nehmen Sie die Option „Anzahl der Tage“ mit auf. Verändern Sie den Quellcode so, dass die Anzahl der gebuchten Tage mit in die Berechnung eingeht.

**Bestellen** [X]

Hotel buchen?  
 Ja     Nein

**Bestellen**  
**Abbrechen**

**Auswahl**  
 Alle Attraktionen ▾

Anzahl der Personen    Anzahl der Tage  
 2    2

Rechnungsadresse angeben

12	Zur Verfügung stehen folgende Optionen:	
13		gewählt:
14	Alle Attraktionen	X
15	Nur Eintritt	
16	Eintritt und Verpflegung	
17		
18	Tage	4
19	Personen	2
20	Hotel gebucht	Ja
21		
22	<b>Gesamtpreis:</b>	<b>2.400,00 €</b>



# 11 Die Fehlerbehandlung

In diesem Kapitel lernen Sie

→ wie Sie Fehler mit einer Fehleroutine abfangen können

In der Regel müssen Fehler vom Programmierer abgefangen werden, so dass ungültige Eingaben gar nicht erst vorkommen können. Als letzte Sicherung kann man zusätzlich eine Fehleroutine einbauen. Diese ist immer folgendermaßen aufgebaut:

```
Sub Fehler()  
....On Error GoTo localError  
  
....PROZEDUR  
  
Exit Sub  
....localError:  
....MsgBox (err.Description)  
End Sub
```



**BEISPIEL:** Für die Berechnung  $\text{Ergebnis} = \text{Zähler} / \text{Nenner}$  sollen mögliche Fehler abgefangen werden, zum Beispiel das Teilen durch Null oder die Eingabe eines Buchstaben.

```
Sub Fehler()  
  
    Zähler = 1  
    Nenner = 0  
    Ergebnis = Zähler / Nenner  
    MsgBox (Ergebnis)  
  
End Sub
```

Mit `On Error Resume Next` geht es nach einem Programmfehler in der nächsten Zeile weiter, anstatt in Fehlerzeile hängen zu bleiben.

```
Sub Fehler()  
On Error Resume Next  
  
    Zähler = 1  
    Nenner = 0  
    Ergebnis = Zähler / Nenner  
    MsgBox (Ergebnis)  
  
End Sub
```

Dabei wird allerdings eine leere Messagebox ohne Fehlermeldung angezeigt, die Rechnung wird übersprungen.

Mit `GoTo Name` und `Name:` kann man Teile der Prozedur überspringen.

```

Sub Fehler()
GoTo localError

    Zähler = 1
    Nenner = 0
    Ergebnis = Zähler / Nenner
    MsgBox (Ergebnis)

localError:
End Sub

```

Dann wird aber gar nicht gerechnet (wie man beim Debuggen sieht). Der Sprungbefehl soll nur ausgeführt werden, wenn ein Fehler auftritt. Dafür kann man die beiden Befehle kombinieren:

```

Sub Fehler()
On Error GoTo localError

    Zähler = 1
    Nenner = 0
    Ergebnis = Zähler / Nenner
    MsgBox (Ergebnis)

localError:
'msgBox ("Fehler")
End Sub

```

Nun wird jedoch auch nach korrekter Ergebnisausgabe zusätzlich die Messagebox „Fehler“ angezeigt. Daher muss man bei korrekter Berechnung die Routine verlassen können, ohne sie bis zum Ende durchlaufen zu lassen:

```

Sub Fehler()
On Error GoTo localError

    Zähler = 1
    Nenner = 0
    Ergebnis = Zähler / Nenner
    MsgBox (Ergebnis)
Exit Sub

localError:
'msgBox ("Fehler")
End Sub

```

Künftig soll bei einer falschen Eingabe der Art des Fehlers angezeigt werden.

```

Sub Fehler()
On Error GoTo localError

    Zähler = 1
    Nenner = 0
    Ergebnis = Zähler / Nenner
    MsgBox (Ergebnis)
Exit Sub

localError:

```

```
localError:
msgBox (err.Description)
End Sub
```

Man kann aber auch eigene Fehlerbeschreibungen schreiben, die je nach Fehlerart variieren:

```
Sub Fehler()
On Error GoTo localError

    Zähler = 1
    Nenner = 0
    Ergebnis = Zähler / Nenner
    MsgBox (Ergebnis)
    Exit Sub

localError:

If Err.Number = 11 Then
    MsgBox ("Nicht durch Null teilen")
ElseIf Err.Number = 13 Then
    MsgBox ("Bitte nur Zahlen eingeben")
Else
    MsgBox ("Es ist ein Fehler aufgetreten")
End If

'wenn Fehler 11, dann "Nicht durch Null teilen"
'wenn Fehler 13, dann "Bitte nur Zahlen eingeben"
'sonst "Es ist ein Fehler aufgetreten"
End Sub
```

Statt mit der if-Anweisung könnte man hier auch mit `select case` arbeiten:

```
Sub Fehler()
On Error GoTo localError

    Zähler = 1
    Nenner = 0
    Ergebnis = Zähler / Nenner
    MsgBox (Ergebnis)

Exit Sub

localError:

    Select Case Err.Number
        Case 11
            MsgBox ("Nicht durch Null teilen")
        Case 13
            MsgBox ("Bitte nur Zahlen eingeben")
        Case Else
            MsgBox ("Es ist ein Fehler aufgetreten")
    End Select
End Sub
```

# 12 Makros speichern

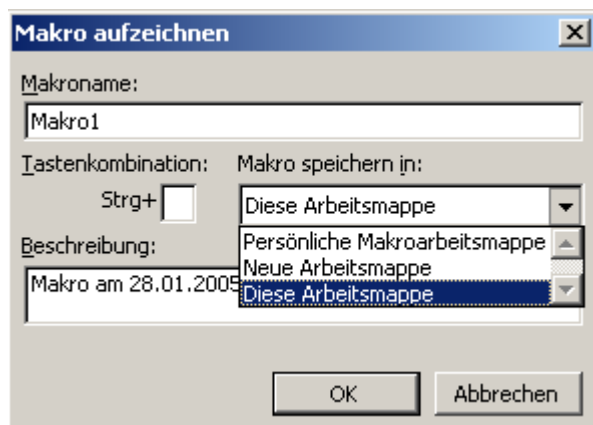
In diesem Kapitel lernen Sie

- wo Ihre Makros gespeichert werden
- wann ein Makro verfügbar ist
- wie Sie einen eigenen Ordner für Makros erstellen und verwenden

## 12.1 Die persönliche Makroarbeitsmappe

Die von Ihnen aufgezeichneten Makros sind bis jetzt in der jeweiligen Arbeitsmappe abgelegt, sind also auch nur dann verfügbar, wenn sich diese Mappe im Arbeitsspeicher befindet. Wollen Sie ein Makro global zur Verfügung stellen, so bietet Ihnen Excel die „persönliche Arbeitsmappe“. Diese befindet sich im Ordner XLSTART im Profil und wird bei jedem Start von Excel „unsichtbar“ geöffnet.

Starten Sie den Makrorecorder und wählen Sie als Speicherort PERSÖNLICHE MAKROARBEITSMAPPE aus.



**ÜBUNG:** Ein neues Makro soll für den markierten Bereich zwei Nachkommastellen und das Tausenderzeichen einstellen. Es soll global abgelegt werden und über ein Symbol in der Symbolleiste zu starten sein.



**HINWEISE:** Sie können sich die persönliche Makroarbeitsmappe über das Menü

FENSTER / EINBLENDEN

anzeigen lassen.

Das nächste Makro, das Sie aufzeichnen, wird wieder in der persönlichen Arbeitsmappe gespeichert. Die Einstellung der Makro-Optionen bleibt solange erhalten, bis Sie sie wieder manuell ändern.

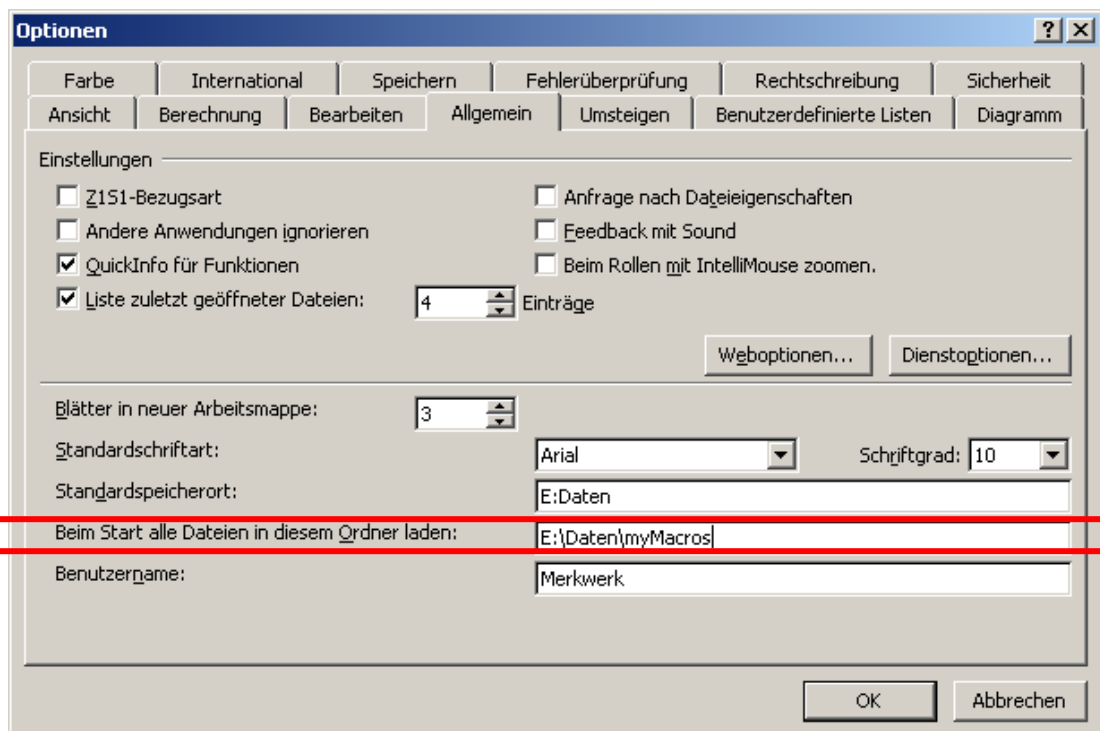
## 12.2 Eine eigene Makroarbeitsmappe

Sie können auch statt der vorgegebenen persönlichen Makroarbeitsmappe im Ordner XLSTART einen eigenen Ordner an einem beliebigen Ort erstellen und dort global ihre Makros abspeichern.

Dazu erstellen Sie einen neuen Ordner. Dann speichern sie eine Arbeitsmappe mit dem Makro, das global verwendet werden soll, in diesen Ordner. Um zukünftig auf dieses Makro zugreifen zu können, stellen sie im Menü unter

EXTRAS / OPTIONEN / ALLGEMEIN

den Ordner ein, der die globalen Makros enthält und der künftig bei jedem Start von Excel im Hintergrund geöffnet werden soll.



**HINWEISE:** Eigene Ordner haben den Vorteil, dass man genau weiß, wo sich wichtige Dateien befinden, um sie gegebenenfalls zu ändern und regelmäßig eine Sicherungskopie zu erstellen.

Wollen Sie ein weiteres Makros global ablegen, kopieren Sie dieses nach der Aufzeichnung in eine Arbeitsmappe im oben angegebenen Ordner.

# 13 Funktionen

In diesem Kapitel lernen Sie

- wie Sie eigene Funktionen erstellen
- wie Sie Funktionen in Excel anwenden können

## 13.1 Funktionen erstellen

Funktionen sind vordefinierte Formeln, wie z.B. die Funktion Summe, die einen angegebenen Bereich aufaddiert.

Eine Funktion erstellen Sie in einem Modul im Visual Basic-Editor.

Funktionen werden im Code nicht mit `Sub`, sondern mit `Function` eingeleitet und mit `End Function` beendet.

Eine Funktion besteht aus dem Funktionsnamen und den Argumenten (Übergabeparametern) und sie hat einen Rückgabewert. Die Argumente sind die Werte, die der Funktion übergeben werden. Der Rückgabewert ist das Ergebnis der Funktion. Dies wird durch den Ausdruck bestimmt (z. B. eine mathematische Operation).

Die Syntax sieht folgendermaßen aus:

```
Function Funktionsname (Argument1 , Argument2 , ....)
    Funktionsname = Ausdruck
End Function
```



**HINWEISE:** Mehrere Argumente werden durch Kommata getrennt.

Im VB-Editor werden im Gegensatz zur Eingabe im Tabellenblatt Dezimalzahlen mit einem Punkt und nicht mit einem Komma angegeben!



**BEISPIEL:**

Es soll eine Funktion erstellt werden, die aus einem Nettobetrag den Bruttobetrag errechnet. Erstellen Sie dafür ein neues Modul. Die Funktion soll *Bruttobetrag* heißen, das Argument ist der *Nettobetrag*.

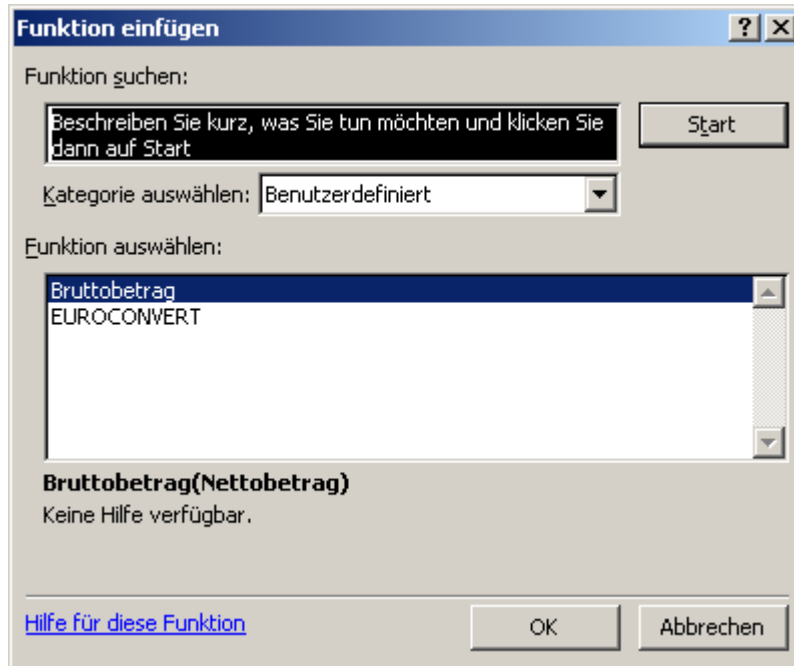
```
Function Bruttobetrag(Nettobetrag)
    Bruttobetrag = Nettobetrag * 1.16
End Function
```

Die Funktion ist nun in der Tabelle verfügbar, allerdings nur solange die Datei, in der Sie die Funktion erstellt haben, geöffnet ist. Wollen Sie

Funktionen global abspeichern, gehen Sie wie bei Prozeduren vor (siehe dazu das Kapitel Makros speichern Seite 52)

## 13.2 Funktionen in Excel anwenden

Selbst erstellte Funktionen können wie alle anderen Funktionen über den Funktionsassistenten aufgerufen werden (EINFÜGEN / FUNKTION). Dort befinden sie sich in der Kategorie BENUTZERDEFINIERT.



### ÜBUNGEN:

1. Erstellen Sie eine Funktion, die den Kreisumfang aus dem Radius errechnet.
2. Erstellen Sie eine Funktion, die aus einem Preis und einem Rabatt in Prozent die Ersparnis herausrechnet.

## 13.3 Funktionen im Code anwenden

Man kann Funktionen nicht nur in Excel direkt anwenden, sondern auch in vorhandenen Prozeduren einsetzen. So kann man z. B. eine Berechnung aus der Prozedur ausgliedern und in einer eigenen Funktion anstellen. Die Prozedur ruft die Funktion auf, übergibt ihr Werte und bekommt ein Ergebnis zurückgeliefert.



### PRAKTIKUM



Im Beispiel Park.xls soll die Preisberechnung mittels einer Funktion erfolgen.

Die Prozedur „Berechnen“ ruft die Funktion „Gesamtpreis“ auf, übergibt ihr Werte, die der Benutzer im Formular eingetragen hat (Hotel, Auswahl, AnzahlPersonen, AnzahlTage) und bekommt das Ergebnis zurückgeliefert.

Dazu wird in einem neuen Modul die Funktion „Gesamtpreis“ erstellt. Die Argumente müssen nicht die gleichen Namen haben, wie die Variablen in der Prozedur „Berechnen“.

```
Function Gesamtpreis(Hotel, Eintritt, Personen, Tage)
    Gesamtpreis = (Hotel + Eintritt) * Personen * Tage
End Function
```

In einem ersten Test kann man in einer Prozedur darunter den Gesamtpreis einer Variablen übergeben, dafür müssen den Argumenten Werte zugewiesen werden. Die Argumente werden im Hilfetext angezeigt.

```
Sub versuch()

    ZuBezahlen = Gesamtpreis()
                    Gesamtpreis(Hotel, Eintritt, Personen, Tage)
End Sub
```

```
Sub versuch()
    ZuBezahlen = Gesamtpreis(180, 100, 3, 2)
End Sub
```

Nun soll die Funktion in der Prozedur „Berechnen“ angewendet werden. Statt wie bisher

```
'Gesamt = (Hotel + Auswahl) * AnzahlPersonen * AnzahlTage
```

wird der Variablen „Gesamt“ der Rückgabewert der Funktion Gesamtpreis übergeben. Dazu muss die Funktion die Werte, die der Benutzer im Formular ausgewählt hat, bekommen. Die Werte sind in den Variablen Hotel, Auswahl, AnzahlPersonen und AnzahlTage gespeichert.

```
Gesamt = Gesamtpreis(Hotel, Auswahl, AnzahlPersonen,
AnzahlTage)
```

Da die Funktion in einem Modul abgelegt ist, ist sie nicht nur im Formular einsetzbar. Die Funktion kann so auch aus Excel oder von anderen Objekten (z.B. Formularen) benutzt werden. Außerdem kann

die Funktion aus einer Prozedur mit rein lokalen Variablen aufgerufen werden.

# 14Lösungen zu den Übungen

## Übung Seite 10

Im Makro Spalte soll die Zelle G8 markiert werden. Passen Sie den Code entsprechend an.

```
Sub Spalte()  
    Range("A1").Select  
    Selection.CurrentRegion.Select  
    Selection.Columns.AutoFit  
    Range("G8").Select  
End Sub
```

Zeichnen Sie ein neues Makro auf, das in einer beliebigen Zelle die Schriftgröße 20, die Schriftfarbe rot und den Schriftschnitt fett einstellt.

```
Sub Makro1()  
    With Selection.Font  
        .Name = "Arial"  
        .FontStyle = "Fett"  
        .Size = 20  
        .Strikethrough = False  
        .Superscript = False  
        .Subscript = False  
        .OutlineFont = False  
        .Shadow = False  
        .Underline = xlUnderlineStyleNone  
        .ColorIndex = 3  
    End With  
End Sub
```

## Übung Seite 11

Erstellen Sie ein neues Makro, das die Zelle G8 grau macht. Kopieren Sie dieses in das Makro Spalte, so dass dieses Makros zusätzlich die neue Funktion übernimmt.

```
Range("A1").Select  
    Selection.CurrentRegion.Select  
    Selection.Columns.AutoFit  
    Range("G8").Select  
    With Selection.Interior  
        .ColorIndex = 15  
        .Pattern = xlSolid  
    End With
```

## Übung Seite 11

Kommentieren Sie den Quellcode des Makros „Spalte“.

```
Range("A1").Select
```

```
'Zelle A1 wird markiert
Selection.CurrentRegion.Select
'Der aktuelle Bereich wird markiert
Selection.Columns.AutoFit
'Die optimale Spaltenbreite wird für den markierten Bereich
eingestellt
Range("G8").Select
'Die Zelle G8 wird markiert
With Selection.Interior
    .ColorIndex = 15
    .Pattern = xlSolid
'Die Zelle bekommt einen grauen Hintergrund
```

## Übung Seite 11

Manipulieren Sie den Code so, dass die Nullwerte wieder angezeigt werden.

```
Sub Nullwerte()
    ActiveWindow.DisplayZeros = True
End Sub
```



## Übung Seite 23

Nun soll auch noch der der Ansprechpartner herausgefunden und in einer MessageBox ausgegeben werden. (Den Code muss man nicht nochmals aufzeichnen.)

```
ActiveCell.Offset(0, 1).Select
AnsprechPartner = Selection
MsgBox (AnsprechPartner)
```

## Übung Seite 24

Nun soll auch der Preis aus der Excelliste gesucht und in der MessageBox angegeben werden.

```
ActiveCell.Offset(0, 4).Select

EndPreis = Selection
MsgBox (EndPreis)
```

## Übung Seite 25

Die MessageBox der Prozedur „finden“ soll wie unten aussehen:



```
MsgBox ("Ansprechpartner ist " & AnsprechPartner _
    & vbCrLf & " von der Firma " & Firma _
    & vbCrLf & "zu zahlen sind " & EndPreis)
```

## Übung Seite 31

Wenn nun in einem Anredefeld Firma statt Frau steht, würde eine falsche Anrede im Brief stehen. In einem solchen Fall soll dort künftig „*Sehr geehrte Damen und Herren*“ stehen.

Dafür muss man die Textprüfung auf weitere Zeichen ausweiten.

```
GeSchlecht = Mid(AnsprechPartner, 1, 4)
Range("a5").Select

If GeSchlecht = "Frau" Then
    Selection = "Sehr geehrte " & AnsprechPartner & ",,"
ElseIf GeSchlecht = "Herr" Then
    Selection = "Sehr geehrter " & AnsprechPartner & ",,"
Else
    Selection = "Sehr geehrte Damen und Herren,"
End If
```

Eleganter wäre folgende Abwandlung des Codes, denn dabei könnte man die Variable AnRede eventuell später nochmals nutzen:

```
Geschlecht = Mid(AnsprechPartner, 1, 4)

If Geschlecht = "Frau" Then
    AnRede = "Sehr geehrte " & AnsprechPartner
ElseIf Geschlecht = "Herr" Then
    AnRede = "Sehr geehrter " & AnsprechPartner
Else
    AnRede = "Sehr geehrte Damen und Herren"
End If

Range("a5").Select
Selection = AnRede
```

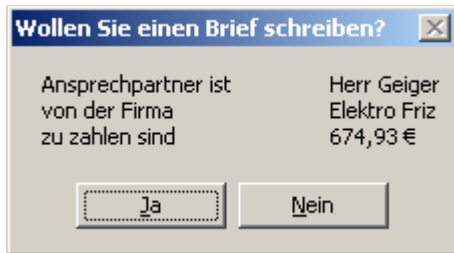
## Übung Seite 32

Debuggen Sie in Gasherd.xls die Routine „Finden“. Wann wird eine Codezeile ausgeführt, wann werden den Variablen die Werte übergeben? Prüfen Sie nach jedem Schritt mit F8 die Auswirkungen in der Exceltabelle.

Mit F8 wird die gelb markierte Codezeile im Einzelschrittmodus ausgeführt. In diesem Moment werden den Variablen Werte übergeben.

### Übung Seite 33

Die Konstante vbTab fügt einen Tab ein.  
Gestalten Sie die Dialogbox folgendermaßen:



```
RetVal = MsgBox("Ansprechpartner ist " & vbTab & _  
    AnsprechPartner  
    & vbCrLf & "von der Firma " & vbTab & vbTab & Firma _  
    & vbCrLf & "zu zahlen sind " & vbTab & vbTab & _  
    Format(EndPreis, "0.00") & " €" _  
    , vbYesNo, "Wollen Sie einen Brief schreiben?")
```

### Übung Seite 40

Ändern Sie die Starteinstellungen für das Dialogfenster wie unten dargestellt. Den Wert des Textfeldes soll man nur über das Drehfeld verändern können. Er soll die 1 nicht unterschreiten und die 20 nicht überschreiten können.

```

Private Sub UserForm_Initialize()
    With spiPers
        .Value = 2
        .Max = 20
        .Min = 1
    End With
    'txtPers.Value = 2          braucht man nicht, da der Wert in
    spiPers
    'beim Starten verändert
    'wird und damit laut Private Sub spiPers_Change() automatisch
    'auch txtPers
    optJa.Value = True
    cboAuswahl.RowSource = "Tabelle1!a14:a16"
    cboAuswahl.ListIndex = 0
    txtPers.Enabled = False
    cboAuswahl.Style = fmStyleDropDownList
End Sub

```

## Übung Seite 43

Die Anzahl der Personen soll in die Zelle C19 geschrieben werden. Wenn ein Hotel gewählt wird, soll in der Zelle C20 „Ja“ stehen, sonst „Nein“. Bei der Auswahl weiterer möglicher Optionen sollen Kreuze in die entsprechenden Zellen gesetzt werden.

Anzahl ermitteln:

```

AnzahlPersonen = spiPers.Value
'Die Anzahl der Personen ist gleich dem Wert von spiPers
Range("C19").Select
Selection = AnzahlPersonen
'Die Anzahl wird ins Tabellenblatt geschrieben
,oder Range("C19").Value = AnzahlPersonen

```

## Hat der Benutzer ein Hotel gewählt?

```
Range("C20").Select
If optJa.Value = True Then
    Hotel = 180
    Selection = "Ja"
Else
    Hotel = 0
    Selection = "Nein"
End If
```

## oder mit select case

```
Range("C20").Select
Select Case optJa.Value
    Case True
        Hotel = 180
        Selection = "Ja"
    Case False
        Hotel = 0
        Selection = "Nein"
End Select
```

## Art des Eintritts ermitteln:

```
Range("C14:C16").Value = ""
'wird die Berechnen-Prozedur nochmals aufgerufen, werden alle
'gesetzten X entfernt.

If cboAuswahl.ListIndex = 0 Then
    Auswahl = 120
    Range("C14").Value = "X"
ElseIf cboAuswahl.ListIndex = 1 Then
    Auswahl = 25
    Range("C15").Value = "X"
Else
    Auswahl = 80
    Range("C16").Value = "X"
End If
```

## oder mit select case

```
Select Case cboAuswahl.ListIndex
    Case 0
        Auswahl = 120
        Range("C14").Value = "X"
    Case 1
        Auswahl = 25
        Range("C15").Value = "X"
    Case 2
        Auswahl = 80
        Range("C16").Value = "X"
End Select
```



## Übung Seite 45

Das Formular „Bestellen“ soll weitere Textfelder bekommen, in denen man eine Rechnungsadresse eintragen kann. Diese soll in die Exceltabelle in die Zellen A7 bis A9 eingetragen werden (*Barverkauf* wird überschrieben).

Bestellen

Hotel buchen?

Ja  Nein

Bestellen

Abbrechen

Auswahl

Alle Attraktionen

Anzahl der Personen

2

Rechnungsadresse

Vorname Nachname

Straße

Postleitzahl Ort

Reisebüro Heinz-Horst Müller Waldstr. 12 23432 Burgenhausen

Sandra Wolf	
Hasenstr. 12	
23434 Horsthausen	

Code in Sub Berechnen:

```

'in die Zelle A7 soll Vorname Nachname stehen
Range("A7").Select
Selection = txtVorname.Text & " " & txtNachname.Text
'in der Zelle A8 Straße
Range("A8").Select
Selection = txtStrasse.Text
'in Zelle A9 die Plz und der Ort
Range("A9").Select
Selection = txtPostleitzahl.Text & " " & txtOrt.Text

```

## Übung Seite 47

Nehmen Sie die Option „Anzahl der Tage“ mit auf. Verändern Sie den Quellcode so, dass die Anzahl der gebuchten Tage mit in die Berechnung eingeht.

12	Zur Verfügung stehen folgende Optionen:	
13		gewählt:
14	Alle Attraktionen	X
15	Nur Eintritt	
16	Eintritt und Verpflegung	
17		
18	Tage	4
19	Personen	2
20	Hotel gebucht	Ja
21		
22	<b>Gesamtpreis:</b>	<b>2.400,00 €</b>

## Text- und Drehfeld verbinden

```
Private Sub spiTage_Change()  
    txtTage.Value = spiTage.Value  
End Sub
```

## Initialisierung

```
spiTage.Value = 2  
txtTage.Enabled = False
```

Wert in das Formular schreiben (Code in Sub Berechnen):

```
AnzahlTage = spiTage.Value  
Range("C18").Select  
Selection = AnzahlTage
```

## Berechnung

```
Gesamt = (Hotel + Auswahl) * AnzahlPersonen * AnzahlTage
```

## Übung Seite 52

Ein neues Makro soll für den markierten Bereich das zwei Nachkommastellen und das Tausenderzeichen einstellen. Es soll global abgelegt werden und über ein Symbol in der Symbolleiste zu starten sein.

```
Selection.NumberFormat = "#,##0.00"
```

Dieses Makro muss in der persönlichen Arbeitsmappe abgelegt sein.

## Übung Seite 56

Erstellen Sie eine Funktion, die den Kreisumfang aus dem Radius errechnet.

```
Function Kreisumfang(Radius)  
    Kreisumfang = Radius * 3.1415  
End Function
```

oder

```
Function Kreisumfang(Radius)  
    Kreisumfang = Radius * Application.Pi  
End Function
```

Erstellen Sie eine Funktion, die aus einem Preis und einem Rabatt in Prozent die Ersparnis herausrechnet.

```
Function Ersparnis(Preis, Rabatt)
    Ersparnis = Preis * Rabatt
End Function
```

# 15Index

ActiveWindow.....	13	Option Explicit.....	18
Argumente.....	56, 58	Optionsfelder.....	38
Bezeichnungsfelder.....	39	persönliche Arbeitsmappe.....	53
Datentyp.....	19	Projektextplorer.....	10, 12
Datentypen.....	19	Prozedur.....	10, 15, 29, 65
debuggen.....	33	Pseudocode.....	30, 42
deklarieren.....	17	Quellcode.....	10
dimensionieren.....	17, 19	Rahmen.....	38
Drehfelder.....	39	relativer Verweis.....	24
Eigenschaften.....	37	Rückgabewert.....	27, 28, 56
End sub.....	15	select case.....	43, 52
Entwurfsmodus.....	45	Steuerelemente.....	36, 45
Ereignisse.....	40	Sub.....	15
Fehler.....	33, 50	Textfelder.....	39
Fehlerroutine.....	50	Unload Me.....	41
Formate.....	34	value.....	41
Formular.....	35, 36	Variable.....	17, 24
Funktionen.....	56, 58	Variablendeklaration.....	17, 18
Funktionsassistent.....	57	Variant.....	19
GoTo.....	50	vbTab.....	34
Haltepunkt.....	33	Visual Basic Editor.....	10, 15, 45
Hilfe.....	19, 26, 27	With.....	13, 14
if-Anweisung.....	29, 31	Zeilenumbruch.....	26, 27
Initialize.....	41	Übergabeparameter.....	56
Inputbox.....	21, 22		
Kombinationsfelder.....	39		
Kommentare.....	10, 12, 44		
Kontrollkästchen.....	47		
Makro.....			
aufzeichnen.....	5		
ausführen.....	6		
global speichern.....	53, 55		
mit Tastenkombination			
ausführen.....	9		
über Symbol starten.....	7		
Makroarbeitsmappe.....	54		
Makrorecorder....	5, 6, 22, 23, 30		
Messagebox... ..	15, 17, 24, 26, 27		
mid-Funktion.....	31		
Modul.....	10, 12, 15		
Not.....	13		
Not“ .....	47		
On Error.....	50		