

Workshop 10: VBA-Programmierung mit MS Excel

1 Programmieren eines Excel-Formulars	1
1.1 Lehrmethode	1
1.2 Szenario	2
1.3 Aufgabenstellung für den Programmierer.....	2
1.4 Stammdaten	2
1.5 Formulargestaltung	3
2 Das Formular erzeugen	4
2.1 Formulargröße bestimmen.....	5
2.2 Eigenschaften des Formulars	5
2.3 Formular anzeigen	5
2.4 Formular	5
2.5 Wichtige Ereignisse eines Formulars	5
3 Steuerelemente einfügen	7
3.1 Verwendung der Steuerelemente	7
3.2 Eigenschaften der Steuerelemente	8
3.3 Steuerelemente verändern	9
3.4 Dem Kombinationsfeld Listenfeldeinträge zuordnen.....	9
3.5 Steuerelemente programmieren	9
4 Auf Ereignisse reagieren.....	9
4.1 Den Abbrechenschalter programmieren.....	9
4.2 Die Verknüpfung des Wertes von Drehfeld und Textfeld	9
4.3 Gesamtpreis berechnen und ausgeben	10
4.4 Kontrollkästchen für die Rechnungsadresse programmieren	12
5 Steuerelemente im Tabellenblatt	12
6 Grundlagen der Fehlerbehandlung	13
7 Anhang: Schnellübersicht zum VBA-Editor	15
8 Übungen	15
8.1 Fallunterscheidung in Prozedur <i>Berechnen ändern</i>	15
8.2 Hilfetext für drei Controls in die UserForm einbauen	16
8.3 Darstellung von drei Controls in der UserForm ändern.....	16

1 Programmieren eines Excel-Formulars

In dieser Lektion lernen Sie

- benutzerdefinierte Formulare kennen,
- was Steuerelemente wie Drehfelder, Textfelder, Optionsschaltflächen und Kombinationsfelder bedeuten,
- wie diese Steuerelemente mit Prozeduren verknüpft werden.

1.1 Lehrmethode

Das isolierte Pauken von Befehlswörtern einer Programmiersprache führt zu nichts. Das Programmieren lernt man am besten an vielen praktischen Beispielen. Dort können die Sprachelemente im logischen Zusammenhang kennengelernt werden.

Workshop 10: VBA-Programmierung mit MS Excel

Im Folgenden wird eine funktionsfähige Excel-Lösung zunächst mit allen Einzelheiten beschrieben. Im Rahmen des dazugehörigen Excel-Workshops wird

- die Lösung vorgestellt
- die Funktionsweise erläutert.
- der Quellcode analysiert

Im Text erkennen Sie bestimmte Programmelemente an der Formatierung:

<i>Arial, GROSSBUCHSTABEN</i>	verweisen auf Menüpunkte
Times New Roman, fett	kennzeichnet alle vom Programm vorgegebenen Namen
<i>Arial, kursiv</i>	kennzeichnet vom Anwender vergebene Namen
Courier New	kennzeichnet VBA-Quellcode

1.2 Szenario

Ein Reisekaufmann vermittelt u. a. Angebote eines Vergnügungsparks. Als Werkzeug für die Berechnung des Gesamtpreises ausgewählter Leistungen wird Excel eingesetzt, allerdings ohne Bildschirmformulare. Deshalb entstehen gar nicht selten Fehler, die beispielsweise auf Folgendem beruhen:

- Basisdaten werden versehentlich gelöscht oder überschrieben, weil direkt mit den Excel-Tabellenblättern gearbeitet wird.
- Die Berechnung des Gesamtpreises fehlerhaft ist, obwohl die Basisdaten in Ordnung sind.
- Unzulässige Bestellmengen akzeptiert werden, weil dafür keine Prüfungen auf Plausibilität vorgenommen werden.

Um in Zukunft solche Pannen zu vermeiden, entschließt sich der Kleinunternehmer, mit dem Excel-Chaos ein für allemal aufräumen. Von einem Geschäftsfreund hat er erfahren, dass Excel eine integrierte Entwicklungsumgebung besitzt, die u.a. die Erstellung von komfortablen Eingabe-Formularen (sogen. **UserForm**) ermöglicht. Deshalb beauftragt er einen ihm empfohlenen Programmierer mit sehr guten VBA-Kenntnissen¹, seine Anwendungen komfortabler und sicherer zu gestalten.

1.3 Aufgabenstellung für den Programmierer

Ein Formular für eine einfache Reisebüro-Anwendung mit Excel soll entwickelt werden. Im Mittelpunkt steht dabei die Vermittlung von Leistungen eines Vergnügungsparks durch das Reisebüro. Bei der Auftragserfassung sollen folgende Optionen wählbar sein:

- ob ein Hotel mit gebucht wird oder nicht
- die Art des Eintritts, nämlich
 1. Buchung aller Attraktionen von vornherein
 2. Eintritt und Verzehrbon
 3. Nur Eintritt
- die Anzahl der Personen
- die Anzahl der gebuchten Tage

Außerdem soll es möglich sein, die Rechnungsadresse des Bestellers zu erfassen.

1.4 Stammdaten

Die Stammdaten für die geplante Anwendung sind in einem Excel-Tabellenblatt namens *Basisdaten* enthalten. Den festen Inhalt dieses Tabellenblatts zeigt Abb.1:

¹ VBA bedeutet *Visual Basic for Applications* und stellt im Kern die gemeinsame Programmiersprache für alle Komponenten der Office-Suite von Microsoft dar. Für die Erstellung und Bearbeitung von VB-Code wird eine integrierte Entwicklungsumgebung mitgeliefert, der *Visual Basic-Editor*. Kurz: *VB-Editor*.

Workshop 10: VBA-Programmierung mit MS Excel

	A	B	C	D	E
1	Preise je Person und Tag		Preis		
2	Hotel		180,00 €		
3	Alle Attraktionen		120,00 €		
4	Eintritt und Verzehrbon		80,00 €		
5	Nur Eintritt		25,00 €		
6			Rechnungsadresse		
7	Vorname, Nachname				
8	Strasse				
9	PLZ, Ort				
10					
11					
12					
13	Zur Verfügung stehende Optionen		gewählt		
14	Alle Attraktionen				
15	Eintritt und Verzehrbon				
16	Nur Eintritt				
17					
18	Anzahl Tage				
19	Anzahl Personen				
20	Hotel gebucht?				
21					
22	Gesamtpreis				Kalkulieren
23					
24					

Abb. 1: Tabellenblatt *Basisdaten*

Die im Tabellenblatt *Basisdaten* enthaltene Befehlsschaltfläche *Kalkulieren* dient dazu, ein Eingabeformular aufzurufen. Die im Eingabeformular enthaltene Programmier-Logik sorgt dafür, dass die im obigen Tabellenblatt erkennbaren Ziel-Zellen² gefüllt werden einschließlich des jeweils dazugehörigen Gesamtpreises. In Abhängigkeit der verschiedenen Optionen verändert sich der Gesamtpreis einer Bestellung.

1.5 Formulargestaltung

Zwecks Übersichtlichkeit besteht das gewünschte Formular aus zwei Teilbereichen. Der obere Bereich ist immer sichtbar und dient der Erfassung der bereits beschriebenen Wahlmöglichkeiten. Der untere Bereich ist nur auf Anforderung sichtbar und lässt sich mittels Kontrollkästchen³ wieder ausblenden. Abb. 2 zeigt den Entwurf für eine **UserForm** mit den

- darauf gelegten Steuerelementen
- Namen dieser Steuerelemente
- Präfixen⁴ für diese Namen.

² Die Ziel-Zellen sind die eingerahmten leeren Zellen.

³ Definition in Tab. 2.

⁴ Die gewählten Präfixe beruhen auf einer verbreiteten Konvention der weltweiten Excel-Community. Sie erleichtert die Lesbarkeit und Pflege umfangreicher Excel-Programme erheblich.

Workshop 10: VBA-Programmierung mit MS Excel

Steuerelemente des Formulars *frmBestellung*

The screenshot shows a form titled 'Bestellung' with the following elements and their labels:

- optJa** and **optNein**: Radio buttons for 'Hotel buchen?'.
- cmdBestellen** and **cmdAbbrechen**: Buttons for 'Bestellen' and 'Abbrechen'.
- fraHotel**: The frame containing the radio buttons.
- fraAuswahl**: The frame containing the 'Bitte auswählen!' section.
- cboAuswahl**: A dropdown menu for 'Art der Eintritts' with 'Alle Attraktionen' selected.
- txtPers** and **spnPers**: A text box and a spinner for 'Anzahl Personen' (value: 2).
- txtTage** and **spnTage**: A text box and a spinner for 'Anzahl Tage' (value: 1).
- chkAdresse**: A checked checkbox for 'Rechnungsadresse erfassen?'.
- fraAdresse**: The frame containing the address fields.
- txtVorname** and **txtNachname**: Text boxes for 'Vorname' and 'Nachname'.
- txtStrasse**: A text box for 'Strasse'.
- txtPostleitzahl** and **txtOrt**: Text boxes for 'Postleitzahl' and 'Ort'.

Bedeutung der Präfixe

- cbo**: Kombinationsfeld (ComboBox)
- chk**: Kontrollkästchen (CheckBox)
- cmd**: Befehlsschaltfläche (CommandButton)
- fra**: Rahmen (Frame)
- spn**: Drehfeld (SpinButton)
- txt**: Textfeld (TextBox)

Abb. 2: Formular *frmBestellung*

Quelle: In Anlehnung an: Andreas Klein, Stefanie Friedrich, Excel 2003 Programmierung mit Visual Basic, 1. Auflage: 2005 (<http://www.merkwerk.de/files/exvba.pdf>)

2 Das Formular erzeugen

Um eine **UserForm** zu erstellen wechseln Sie zunächst in die Entwicklungsumgebung (beispielsweise mit der Tastenkombination Alt + F11).

Ihnen wird eine leere **UserForm** angezeigt, auf der Sie nun die oben beschriebenen Steuerelemente anordnen können. Der **TabIndex**⁵ der Steuerelemente richtet sich nach der Reihenfolge ihrer Erstellung.

⁵ Vgl. dazu Tab. 3

Workshop 10: VBA-Programmierung mit MS Excel

Die **UserForm** kann weitgehend frei benannt werden. Hier wird sie *frmBestellung* genannt.

2.1 Formulargröße bestimmen

Die Größe des Formulars wird entweder über das zugehörige **Eigenschaftenfenster** bestimmt, oder indem der Maus-Cursor auf den Rand des Formulars gestellt wird und an diesem gezogen wird.

2.2 Eigenschaften des Formulars

Mit der Funktionstaste F4 oder der Schaltfläche **Eigenschaften** wird das **Eigenschaftenfenster** aktiviert. Hier kann u. a. festgelegt werden:

- Name des Formulars (engl. *Name*)
- Bezeichnung des Formulars (engl. *Caption*)
- Größe
- Hintergrundfarbe
- Position
- und einiges mehr

2.3 Formular anzeigen

Um das jeweilige Formular anzuzeigen, wird die Methode **Show** verwendet. Von der Schaltfläche *cmdAuswahlAnzeigen*, die auf den Tabellenblatt *Basisdaten* liegt, wird das Formular *frmBestellung* wie folgt aufgerufen:

```
Sub cmdAuswahlAnzeigen()  
    frmBestellung.Show  
End Sub
```

2.4 Formular schließen

Es gibt zwei Möglichkeiten, ein Formular zu schließen:

- die Methode **Hide** oder
- die Anweisung **Unload**.

Hide macht das Formular unsichtbar, aber alle Einstellungen bleiben im Speicher erhalten. Wird das Formular mit *Show* erneut angezeigt, erscheint es genau so, wie es vorher aussah.

Die Anweisung **Unload** schließt das Formular und entfernt es aus dem Speicher. Dabei werden auch die Variablen gelöscht, die zu einer Prozedur des Formulars gehören. Im vorliegenden Beispiel erfolgt das Schließen des Formulars *frmBestellung* mit folgender Prozedur:

```
Private Sub cmdAbbrechen_Click()  
    ' Bei Betätigung der Befehlsschaltfläche "Abbrechen"  
    ' das aktuelle Formular schließen und entladen  
    Unload frmBestellung ' hier würde 'Unload Me' genügen  
End Sub
```

2.5 Wichtige Ereignisse eines Formulars

Die wichtigsten Ereignisse eines Formulars sind **Initialize** und **Terminate**.

- **Initialize** tritt ein, wenn das Formular geladen wird, also vor dem ersten Anzeigen.
- **Terminate** wird ausgelöst, wenn das Formular geschlossen wird. Dabei kann das Formular sowohl über das Kreuz in der Titelleiste als auch mit der Anweisung **Unload** geschlossen werden.

Das Ereignis, das bei jedem Start des Formulars als Erstes ausgeführt wird, heißt **Initialize**. Im vorliegenden Beispiel ist die Prozedur **Initialize** wie folgt aufgebaut:

Workshop 10: VBA-Programmierung mit MS Excel

```
Private Sub UserForm_Initialize()  
    ' Diese Prozedur wird von Excel zu allererst ausgeführt.  
    ' Hier werden u. a. Standardwerte gesetzt.  
  
    ' Konstante setzen. 1 Punkt ist etwa 0,35 mm (1/72 Zoll) groß.  
    Const conAbstand As Integer = 20 ' Punkte  
  
    intGanz = Me.Height ' Gesamte Formularhöhe als ganzzahliger Wert  
  
    ' Beginn der unteren Formularhälfte bestimmen  
    With Me!chkAdresse ' Mit Kontrollkästchen für Rechnungsadresse  
        intHalb = .Top + .Height + conAbstand  
    End With  
  
    ' Ergebnisse im Direktfenster ausgeben  
    Debug.Print "Halbe Formularhöhe ermittelt mit chkAdresse: " & _  
        intHalb & vbNewLine & _  
        "Halbe Formularhöhe ermittelt mit Funktion 'HalbeHoehe': " & _  
        (HalbeHoehe + conAbstand)  
  
    With frmBestellung ' mit aktuellem Formular ...  
        .Height = intHalb ' Formularhöhe setzen  
        .Caption = "Bestellung"  
        With !spnPers ' mit Feld Anzahl Personen ...  
            .Value = 2 ' Startwert setzen  
            .Min = 1 ' Minimalwert setzen  
            .Max = 10 ' Maximalwert setzen  
        End With  
  
        With !spnTage ' mit Feld für Anzahl Tage ...  
            .Value = 1 ' Startwert setzen  
            .Min = 1 ' Minimalwert setzen  
            .Max = 14 ' Maximalwert setzen  
        End With  
  
        With !cboAuswahl ' mit Kombinationsfeld ...  
            ' Datenherkunft bestimmen mit Arbeitsblatt und Zellbereich  
            .RowSource = "Basisdaten!A14:A16"  
            ' 1. Eintrag markieren  
            .ListIndex = 0  
        End With  
  
        ' Andere Startwerte setzen  
        !chkAdresse.Value = False  
        !optJa = True  
    End With  
End Sub
```

Beim Starten des Programms sollen in der **UserForm** bestimmte Werte voreingestellt werden. Zum Beispiel soll die Anzahl der Personen *snpPers* standardmäßig auf 2 gesetzt sein (siehe oben im Quellcode).

Wenn eine Programmzeile mit einem Hochkomma beginnt, dann handelt es sich um einen Kommentar, der die Verständlichkeit und Pflege des Quellcodes verbessern soll.

Verständlichkeit und Lesbarkeit werden auch dadurch gefördert, indem die Programmzeilen im Quellcode (wie oben ersichtlich) systematisch eingerückt werden. Feste Konventionen gibt es dafür nicht. Der Programmierer muss sich eigene Regeln setzen und konsequent einhalten, mit anderen Worten, für seinen Programmierstil ist jeder selbst verantwortlich.

Workshop 10: VBA-Programmierung mit MS Excel

3 Steuerelemente einfügen

In diesem Abschnitt wird beschrieben, wie ausgewählte Steuerelemente auf dem Formular angeordnet und weiter bearbeitet werden können. Der in Abb. 2 gezeigte Formularentwurf enthält folgende 26 Steuerelemente (engl. **Controls**):

Steuerelement	Anzahl
Befehlsschaltflächen	2
Bezeichnungsfelder	8
Drehfelder	2
Kombinationsfelder	1
Kontrollkästchen	1
Optionsfelder	2
Rahmen	3
Textfelder	7
Summe Controls	26

Tab 1: Anzahl Steuerelemente im Benutzerformular

3.1 Verwendung der Steuerelemente

Die in Tab. 1 genannten Steuerelemente werden im Allgemeinen wie folgt verwendet:

Steuerelement	Kürzel	Verwendung
Befehlsschaltfläche	cmd	Werden meistens dafür verwendet, bestimmte Aktionen auszulösen. Typisches Beispiel: Programme starten
Bezeichnungsfeld	lbl	Damit kann Text angezeigt werden. Ein Bezeichnungsfeld ist ein reines Ausgabefeld, der Anwender kann darin nichts ändern.
Drehfeld	spn	Durch Klicken auf eines der Dreiecke im Drehfeld kann ein Wert verkleinert oder vergrößert werden
Kombinationsfeld	cbo	Ist eine Mischung aus Eingabefeld und Listefeld. Listenfelder erlauben die Auswahl eines oder mehrerer vorgegebener Einträge.
Kontrollkästchen	chk	Sie erlauben die Auswahl zweier Zustände wie wahr/falsch, ja/nein oder an/aus.
Optionsfeld	opt	Optionsfelder erlauben die Auswahl zweier Zustände, allerdings werden häufig mehrere Optionsfelder zu einer sogen. <i>Optionsgruppe</i> zusammengefasst. Innerhalb einer <i>Optionsgruppe</i> kann nur ein Optionsfeld aktiv sein.
Rahmen	fra	Sind ein rein optisches Element. Damit kann man z. B. die Gruppen der Optionsfelder hervorheben.
Textfeld	txt	Darin lassen sich beliebige Texte anzeigen oder erfassen. Ebenso können Formeln eingefügt werden.

Tab. 2: Verwendung der Steuerelemente im Benutzerformular

Ist das Formular markiert, befindet sich daneben das Fenster **Werkzeugsammlung**:

Workshop 10: VBA-Programmierung mit MS Excel

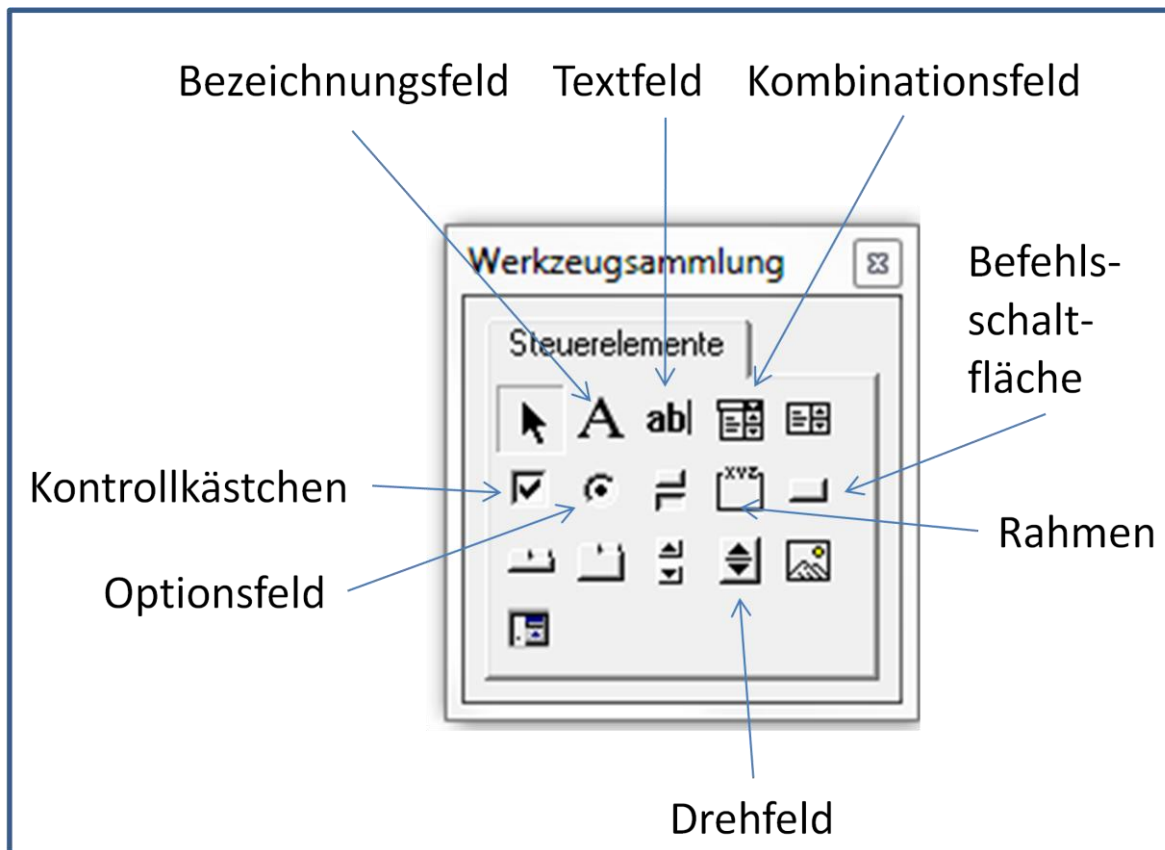


Abb. 3: **Werkzeugsammlung** zur Auswahl von Steuerelementen

3.2 Eigenschaften der Steuerelemente

Jedes Steuerelement hat verschiedene **Eigenschaften**, die man im jeweiligen **Eigenschaftenfenster** einstellen kann. Dazu gehören unter anderem die Farbe, Schriftart, Größe und der Name eines Objektes. Beispielsweise beinhaltet Abb. 4 einen Ausschnitt des Eigenschaftsfensters für eine Befehlsschaltfläche. Ihr **Name** ist `cmdBestellen`, ihre Beschriftung (siehe Eigenschaft **Caption**) lautet *Bestellen*.

Eigenschaften - cmdBestellen	
cmdBestellen CommandButton	
Alphabetisch Nach Kategorien	
(Name)	cmdBestellen
Accelerator	
AutoSize	False
BackColor	<input type="checkbox"/> &H8000000F&
BackStyle	1 - fmBackStyleOpaque
Cancel	False
Caption	Bestellen
ControlTipText	
Default	False
Enabled	True
Font	Tahoma
ForeColor	<input checked="" type="checkbox"/> &H80000012&
Height	21,75
HelpContextID	0
Left	132

Abb. 4: **Eigenschaftsfenster** zu `cmdBestellen`

Workshop 10: VBA-Programmierung mit MS Excel

Es gibt einige Eigenschaften, die für fast alle Steuerelemente gleichermaßen gelten, siehe Tab. 3.

Eigenschaft	Bedeutung
BackColor	bestimmt die Farbe des Hintergrunds eines Steuerelements
ControllTipText	definiert den optionalen Hilfetext zu einem Steuerelement
Enabled	legt fest, ob das Steuerelement ausgewählt werden kann oder nicht
Height	gibt die Höhe eines Steuerelements an
Left	legt die Position des linken Rands eines Steuerelements fest
Name	legt den Namen fest, der im Quellcode verwendet wird, um auf das Steuerelement zuzugreifen
TabIndex	definiert die Reihenfolge der Auswahl mittels der Tab-Taste.
Tag	ist vorgesehen für optionale Zusatzinformationen
Top	legt die Position des oberen Rands eines Steuerelements fest
Visible	legt fest, ob das Steuerelement sichtbar oder unsichtbar ist
Width	definiert die Breite eines Steuerelements

Tab .3: Allgemeingültige Eigenschaften von Steuerelementen

3.3 Steuerelemente verändern

Wenn ein Steuerelement nachträglich in Größe und Position verändert werden soll, muss es zuerst markiert werden. Das ausgewählte Objekt wird mit einem Rahmen und mit 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.

3.4 Dem Kombinationsfeld Listenfeldeinträge zuordnen

Beim Klick auf den Pfeil des Kombinationsfeldes sollen die drei möglichen Optionen (*Alle Attraktionen, Eintritt und Verzehrbon, Nur Eintritt*) erscheinen. Die Auswahlmöglichkeiten werden über die Eigenschaft **RowSource** des Kombinationsfeldes zugeordnet. Dort wird angegeben, in welchem Tabellenblatt und in welchen Zellenbereich die entsprechenden Einträge (hier Optionstexte) zu finden sind.

3.5 Steuerelemente programmieren

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

4 Auf Ereignisse reagieren

Es gibt verschiedene Ereignisse, auf die das Programm reagieren soll. Ereignisse können Mausklicks, Mausbewegungen oder das Betätigen von Tasten sein.

4.1 Den Abbrechenschalter programmieren

Der Befehl **Unload Me** schließt ein Formular.

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

4.2 Die Verknüpfung des Wertes von Drehfeld und Textfeld

Jedes Steuerelement hat verschiedene Eigenschaften. Die gewünschte Eigenschaft wird im Quellcode direkt nach dem Objektnamen angegeben. Werte werden in der Eigenschaft **Value** abgelegt.

Workshop 10: VBA-Programmierung mit MS Excel

```
Private Sub spnPers_Change()  
    txtPers.Value = spnPers.Value  
End Sub
```

Hinweis: Mit der Tastenkombination Strg + Leertaste kann man den angefangenen Text einer Variablen oder einer Eigenschaft vom VBA-Editor vervollständigen lassen. So lassen sich Schreibfehler vermeiden und gleichzeitig das Vorhandensein des jeweiligen Objekts prüfen.

4.3 Gesamtpreis berechnen und ausgeben

Bei Betätigen der Schaltfläche *Bestellen* soll der Gesamtpreis berechnet werden. Die Prozedur *Bestellen* ruft die Prozedur *Berechnen* auf.

```
Private Sub cmdBestellen_Click()  
    ' Die Befehlsschaltfläche 'Bestellen' wurde angeklickt  
    Call Berechnen ' Gesamtpreis berechnen in der Prozedur 'Berechnen'  
End Sub
```

Die Prozedur *Berechnen* stelle den Kern der Anwendung dar. Sie soll prüfen, welche Optionen gewählt wurden und welcher Gesamtpreis sich daraus ergibt.

Aufgrund der Einstellung **Option Explicit** des VBA-Editors⁶ müssen alle Variablen vor ihrer Verwendung im Quellcode ausdrücklich deklariert werden. Hier erfolgt die Deklaration aller Variablen im Kopf der Prozedur.

```
Sub Berechnen()  
    ' Die Prozedur berechnet den Gesamtpreis  
  
    ' Variablen deklarieren:  
    ' Weil "Option Explicit" in diesem Beispiel vorgegeben ist,  
    ' müssen die benutzen Variablen ausdrücklich deklariert werden.  
    ' Die gewählten Präfixe (cur, int) der Variablennamen  
    ' geben deren Datentyp (Currency, Integer) an  
    Dim curHotel, curAuswahl, curGesamt As Currency  
    Dim intAuswahl, intPersonen, intTage As Integer  
  
    ' Möglichen Laufzeitfehler auffangen mit ...  
    On Error GoTo Programmfehler  
  
    Call ZielZellenLeeren      ' Ziel-Zellen leeren  
  
    ' Der Hotelpreis hängt ab von einer Ja-Nein-Entscheidung  
    If optJa.Value = True Then ' Wenn ja, ...  
        curHotel = Range("C2") ' dann Hotelpreis zwischenspeichern  
    Else                       ' Wenn nein, ...  
        curHotel = 0          ' dann ist der Hotelpreis null  
    End If  
    Range("C20") = IIf(curHotel = 0, "nein", "ja")  
  
    ' Die Variable intAuswahl ergibt sich aus dem ListIndex von cboAuswahl  
    intAuswahl = cboAuswahl.ListIndex ' ListIndex ist nullbasiert!  
  
    ' Den jeweiligen Fall der Auswahl bearbeiten:  
    Select Case intAuswahl  
        Case 0  
            curAuswahl = Range("C3") ' 1. Fall: Alle Attraktionen  
            Range("C14").Value = "X" ' Preis aus Zelle C3 übernehmen  
                                     ' Zelle C14 ankreuzen  
        Case 1  
            curAuswahl = Range("C4") ' 2. Fall: Eintritt und Verzehrbon  
                                     ' Preis aus Zelle C4 übernehmen
```

⁶ Rufen Sie den Menüpunkt EXTRAS OPTIONEN auf, um das Dialogfeld OPTIONEN zu öffnen. Im Register EDITOR kann gesetzt werden, dass Variablendefinition erforderlich ist.

Workshop 10: VBA-Programmierung mit MS Excel

```
        Range("C15").Value = "X" ' Zelle C15 ankreuzen
    Case 2
        curAuswahl = Range("C5") ' 3. Fall: Nur Eintritt
        Range("C16").Value = "X" ' Preis aus Zelle C5 übernehmen
        Range("C16").Value = "X" ' Zelle C16 ankreuzen
End Select

' Die Anzahl der Tage ist gleich dem Wert von spnTage
intTage = spnTage.Value
Range("C18") = intTage

' Die Anzahl der Personen ist gleich dem Wert von spnPers
intPersonen = spnPers.Value
Range("C19") = intPersonen

' Der Gesamtpreis ergibt sich aus (Hotel+Auswahl)*AnzahlPersonen*AnzahlTage
curGesamt = (curHotel + curAuswahl) * intPersonen * intTage

' Der bisherige Gesamtpreis "curGesamt" wird zur Sicherheit
' mit dem Rückgabewert der Funktion "GesamtPreis" verglichen.
If curGesamt <> GesamtPreis(curHotel, curAuswahl, intPersonen, intTage) Then
    GoTo Formelfehler
End If
'Gesamtpreis in Zelle C22 einsetzen
Range("C22").Value = curGesamt

' Wenn Vorname und Nachname erfasst wurden, dann ...
If Not IsNull(txtVorname) And Not IsNull(txtNachname) Then
    ' Die Zellen C7 bis C9 der Excel-Tabelle mit der Rechnungsadresse füllen

    ' In Zelle C7 den Vor- und Nachnamen einsetzen
    Range("C7") = txtVorname & " " & txtNachname

    ' In Zelle C8 die Strasse einsetzen
    Range("C8") = txtStrasse

    ' In Zelle C9 die Plz und den Ort einsetzen,
    ' mit anderen Sprachelementen.
    Range("C9").Select
    Selection = txtPostleitzahl & " " & txtOrt
Else
    ' Zellen C7 bis C9 nicht aktualisieren
End If
GoTo Ausgang

Formelfehler:
MsgBox Prompt:="Ein Rechenfehler ist aufgetreten!" & vbCrLf & _
        "Die Prozedur wird abgebochen!", _
        Buttons:=vbExclamation + vbOKOnly, _
        Title:="Formelfehler"
' Wegen Programmabbruch werden die Ziel-Zellen zurückgesetzt.
Call ZielZellenLeeren

Ausgang:
Exit Sub ' Prozedur verlassen

Programmfehler:
MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
        Buttons:=vbCritical, _
        Title:="Berechnen"
Resume Ausgang
End Sub
```

Der Gesamtpreis wird in Zelle C22 ausgegeben, siehe Quellcode!

Die Anzahl der Personen wird in Zelle C19 geschrieben (siehe Quellcode). Wenn *Hotel* gewählt wird, soll in Zelle C20 „ja“ stehen, sonst „nein“ (siehe Quellcode). Bei der Auswahl anderer Optionen werden Kreuze in die dafür vorgesehenen Zellen gesetzt.

Workshop 10: VBA-Programmierung mit MS Excel

4.4 Kontrollkästchen für die Rechnungsadresse programmieren

Die fünf Textfelder der Rechnungsadresse sollen ein- und ausblendbar sein, je nachdem, ob eine Adresse erfasst werden soll oder nicht. Dazu kann ein Kontrollkästchen verwendet werden. Diese **CheckBox** soll beim Klick-Ereignis den gesamten Rahmen *Rechnungsadresse* ausblenden bzw. wieder einblenden.

Ist der Rahmen *Rechnungsadresse* ein- bzw. ausgeblendet, soll das Formular entsprechend vergrößert bzw. verkleinert werden. Das bewirkt folgende Ereignis-Prozedur:

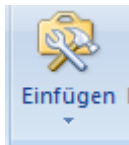
```
Private Sub chkAdresse_Click()  
    ' Formularhöhe mit einfacher If-Abfrage einstellen  
    ' in Abhängigkeit von der CheckBox für die Rechnungsadresse  
    If chkAdresse.Value = True Then  
        frmBestellung.Height = intGanz      ' ganzes Formular anzeigen  
    Else  
        frmBestellung.Height = intHalb     ' nur den oberen Teil des Formulars zeigen  
    End If  
End Sub
```

Die Initialisierung für diesen Vorgang erfolgt typischerweise in der Prozedur **Initialize**. Dort lauten die entsprechenden Anweisungen wie folgt:

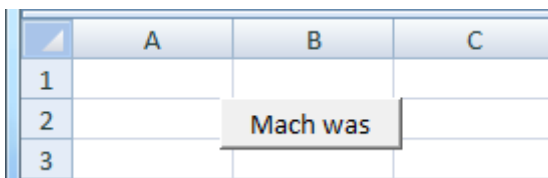
```
intGanz = Me.Height      ' Gesamte Formularhöhe als ganzzahliger Wert  
  
' Beginn der unteren Formularhälfte bestimmen  
With Me!chkAdresse     ' Mit Kontrollkästchen für Rechnungsadresse  
    intHalb = .Top + .Height + conAbstand  
End With
```

5 Steuerelemente im Tabellenblatt

Ausgehend von der Normalansicht des Tabellenblatts kann über den Menüpfad **ENTWICKLERTOOLS - EINFÜGEN** auf die sogen. **Formularsteuerelemente** zugegriffen werden:



Dort wählt man ein Steuerelement aus und fügt es an der gewünschten Stelle in die Excel-Tabelle ein. Die folgende Abbildung zeigt eine auf diese Weise eingefügte Befehlsschaltfläche mit der Bezeichnung (**Caption**) *Mach was*.



Mittels rechter Maustaste lässt sich dieser Befehlsschaltfläche ein passendes Makro zuweisen.

Workshop 10: VBA-Programmierung mit MS Excel

6 Grundlagen der Fehlerbehandlung

In diesem Abschnitt lernen Sie

- wie Sie Fehler mit einer Fehlerroutine 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 Fehlerroutine einbauen. Diese ist im Prinzip immer folgendermaßen aufgebaut:

```
Sub Fehler()  
    On Error GoTo Sprungmarke  
    Anweisungen der Prozedur ausführen  
    Exit Sub  
Sprungmarke:  
    MsgBox (Err.Description)  
End Sub
```

Mit dem Befehl

```
On Error Goto Sprungmarke
```

wird die eigene Fehlerbehandlung eingeleitet. Tritt ein Laufzeitfehler auf, verzweigt das Programm zur Sprungmarke und arbeitet die dort aufgeführten Befehle ab. Die drei Befehle `Resume`, `Resume Sprungmarke` und `Resume Next` werden eingesetzt, um nach einem Laufzeitfehler den Programmfluss weiter abzuwickeln. Abb. 5 veranschaulicht die Folgen der Fehlerbehandlung mit den `Resume`-Befehlen:

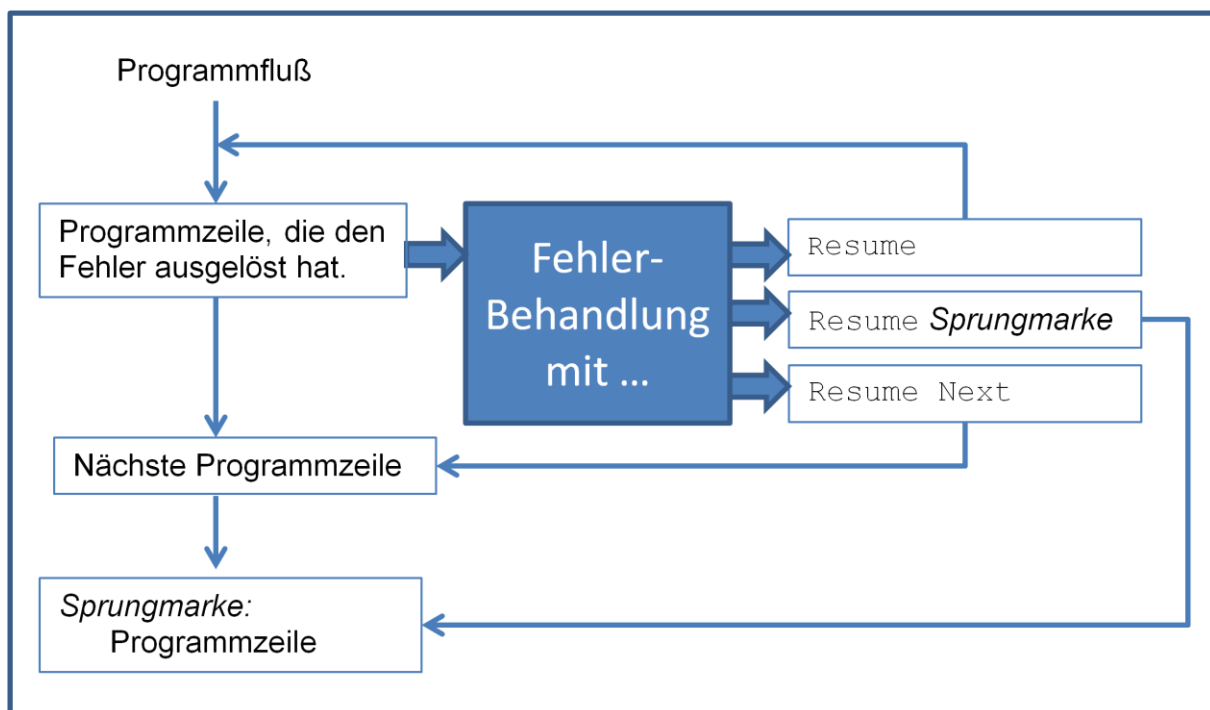


Abb. 4: Fehlerbehandlung mit `Resume`-Befehlen

Dementsprechend ist in die Prozedur *Berechnen* folgende Logik zur Fehlerbehandlung eingebaut:

Workshop 10: VBA-Programmierung mit MS Excel

```
Formelfehler:
    MsgBox Prompt:="Ein Rechenfehler ist aufgetreten!" & vbCrLf & _
        "Die Prozedur wird abgebochen!", _
        Buttons:=vbExclamation + vbOKOnly, _
        Title:="Rechenfehler"
    ' Wegen Programmabbruch werden die Ziel-Zellen zurückgesetzt.
    Call ZielZellenLeeren

Ausgang:
    Exit Sub ' Prozedur verlassen

Programmfehler:
    MsgBox Prompt:="Fehler # " & Err.Number & ": " & Err.Description, _
        Buttons:=vbCritical, _
        Title:="Berechnen"
    Resume Ausgang
End Sub
```

In dieser Beispiel kann auf *Programmfehler* und *Formelfehler* (siehe oben) spezifisch reagiert werden. Der Befehl **Resume Ausgang**, der sich unter der Sprungmarke *Programmfehler* befindet, bewirkt, dass die Prozedur trotz Laufzeitfehler über den normalen *Ausgang* verlassen wird.

Soll beim Auftreten eines Laufzeitfehlers direkt mit der Programmzeile weitergearbeitet werden, die auf die fehlerhafte Zeile folgt, wird folgender Befehl verwendet:

```
On Error Resume Next
```

Dieser Befehl hat die gleiche Gültigkeit und Lebensdauer wie die Prozedur oder Funktion, in der er verwendet wird.

Um die eigene Fehlerbehandlung innerhalb einer Prozedur oder Funktion wieder anzuschalten, wird folgender Befehl eingesetzt:

```
On Error GoTo 0
```

Die Sprungmarke 0 ist eine *Visual Basic*-interne Adresse. Nach Ausführung dieses Befehls erhält *das Visual Basic* die Kontrolle über die Fehlerbehandlung zurück.

Jeder Fehler wird von VBA in einem *Err*-Objekt abgelegt, dessen Eigenschaften in Programm abgerufen bzw. dessen Methoden im Programm verwendet werden können. Die wichtigsten Eigenschaften und Methoden des Fehlerobjekts **Err** sind in Tab. 4 aufgeführt:

Eigenschaft/Methode	Beschreibung
Number	Gibt die Nummer des Fehlers zurück
Description	Gibt die eine Fehlerbeschreibung zurück
Source	Gibt den Namen der Anwendung zurück, die den Fehler ausgelöst hat
Raise	Erzeugt einen Fehler
Clear	Setzt das Fehlerobjekt zurück

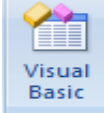



Tab. 4: Eigenschaften und Methoden des **Err**-Objekts

Tipp: Klicken Sie auf die Schaltfläche **MICROSOFT OFFICE**  und anschließend auf **EXCEL-OPTIONEN**. Klicken Sie auf die Kategorie **FORMELN**. Aktivieren Sie unter **FEHLERÜBERPRÜFUNG** das Kontrollkästchen **FEHLERÜBERPRÜFUNG IM HINTERGRUND AKTIVIEREN**. Damit wird die eigene Fehlerbehandlung ausgeschaltet, d. h.,

Workshop 10: VBA-Programmierung mit MS Excel

Visual Basic behandelt jeden Fehler selbst. Allerdings ist diese Option nur während des Testens einer Anwendung sinnvoll. Später sollte sie jedoch wieder deaktiviert werden.

7 Anhang: Schnellübersicht zum VBA-Editor

Sie möchten ...	
Den VBA-Editor starten	ENTWICKLERTOOLS 
den Projekt-Explorer öffnen	ANSICHT PROJEKT-EXPLORER oder Strg + R oder 
ein Code-Fenster öffnen	ANSICHT CODE oder F7 oder 
das Eigenschaftsfenster öffnen	ANSICHT-EIGENSCHAFTSFENSTER oder F4 oder 
Symbolleisten anpassen oder neue erzeugen	ANSICHT SYMBOLLISTEN ANPASSEN ...
die Grundeinstellungen des VBA-Editors ändern	EXTRAS OPTIONEN
Kontextmenüs des VBA-Editors anpassen	ANSICHT SYMBOLLEISTEN ANAPSSEN, Register Symbolleisten, Kontrollfeld KONTEXTMENÜ aktivieren, Befehl aus dem Dialogfenster in das Kontextmenü ziehen

8 Übungen

8.1 Fallunterscheidung in Prozedur *Berechnen ändern*

Ersetzen Sie die Fallunterscheidung *Select Case intAuswahl ... End Select* durch folgenden IF ... Else ... End If Block:

```
If intAuswahl = 0 Then          ' 1. Fall: Alle Attraktionen
    ' Anweisungen für Fall 1
ElseIf intAuswahl = 1 Then     ' 2. Fall: Eintritt und Verzehrbon
    ' Anweisung für Fall 2
ElseIf intAuswahl = 2 Then     ' 3. Fall: Nur Eintritt
    ' Anweisungen für Fall 3
Else                          ' unvorhergesehene Auswahl
    GoTo FallFehler
End If

' Neue Sprungmarke
FallFehler:
    ' Meldung ausgeben
    GoTo Ausgang
```

Workshop 10: VBA-Programmierung mit MS Excel

8.2 Hilfetext für drei Controls in die UserForm einbauen

Ändern Sie die **ControlTipText** Eigenschaften folgender **Controls**:

TypeName	Name	ControlTipText (InfoText)
CommandButton	<i>cmdBestellen</i>	Gesamtpreis berechnen
CommandButton	<i>cmdAbbrechen</i>	Prozedur abbrechen
ComboBox	<i>cboAuswahl</i>	Wählen Sie bitte die Art der Eintrittskarte!

8.3 Darstellung von drei Controls in der UserForm ändern

Ändern Sie die **SpecialEffect** Eigenschaft folgender **Controls**:

TypeName	Name	Alte Darst. (graviert)	Neue Darst. (erhöht)
Frame	<i>fraHotel</i>	fmSpecialEffectEtched	fmSpecialEffectRaised
Frame	<i>fraAuswahl</i>	fmSpecialEffectEtched	fmSpecialEffectRaised
Frame	<i>fraAdresse</i>	fmSpecialEffectEtched	fmSpecialEffectRaised