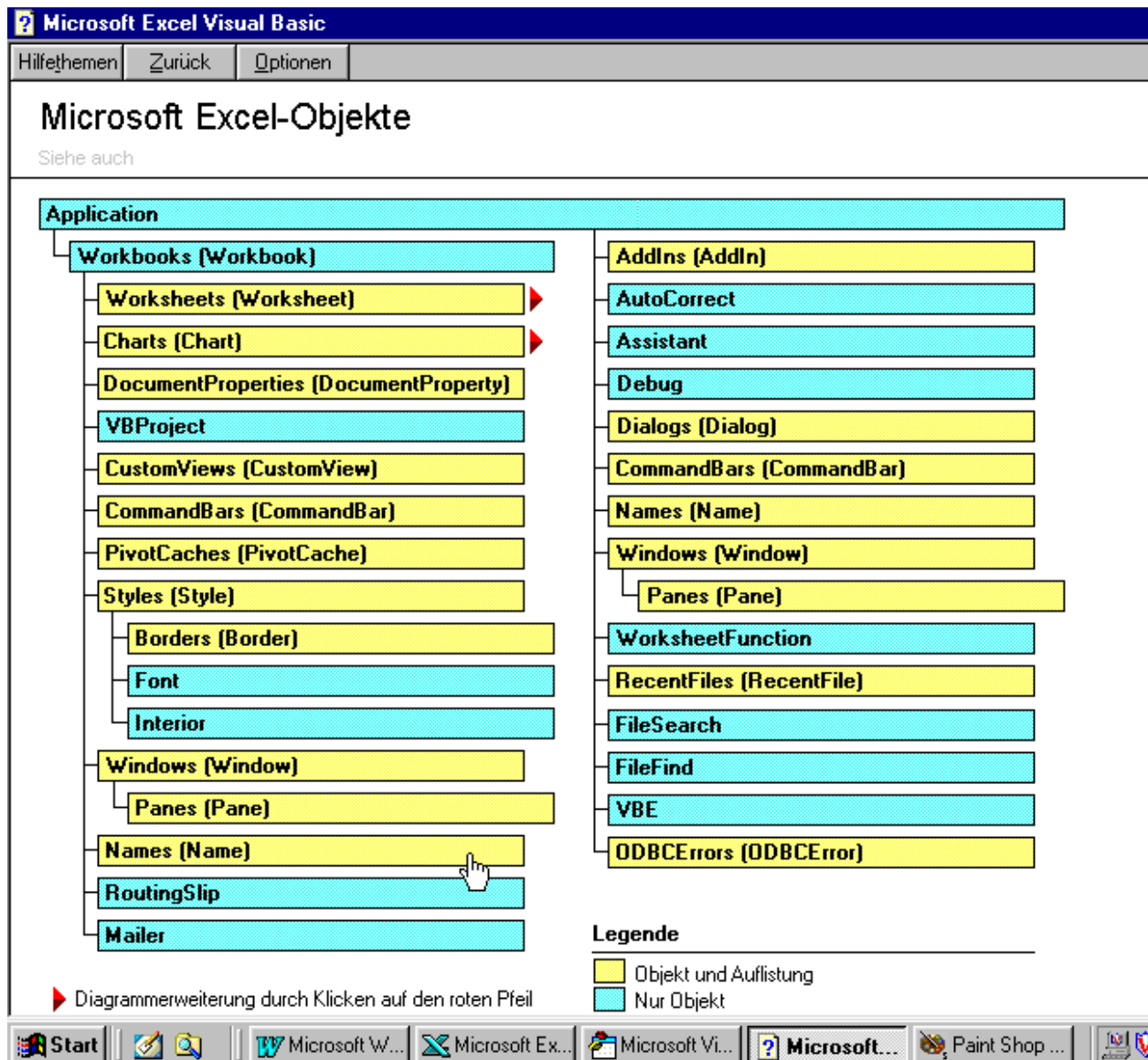


Die Objekt-Hierarchie



Durch Anklicken des Objektes oder der Auflistung erhält man weitere Informationen.

Beispiele/Erläuterungen zum Objektmodell⁴

Das Application-Objekt

Stellt die gesamte Microsoft Excel-Anwendung dar.

Verwenden des Application-Objekts

Verwenden Sie die Application-Eigenschaft, um das Application-Objekt zurückzugeben. Im folgenden Beispiel wird die Windows-Eigenschaft auf das Application-Objekt angewendet.

⁴ vgl. Objektkatalog

```
Application.Windows("mappe1.xls").Activate
```

Anmerkungen

Viele Eigenschaften und Methoden, die gängige Objekte der Benutzeroberfläche, wie z. B. die aktive Zelle (ActiveCell-Eigenschaft), zurückgeben, können ohne den Objektkennzeichner Application verwendet werden. Anstelle von Application.ActiveCell.Font.Bold = True können Sie beispielsweise folgendes eingeben: ActiveCell.Font.Bold = True.

Workbook-Objekt

Stellt eine Microsoft Excel-Arbeitsmappe dar. Das Workbook-Objekt ist ein Element der Workbooks-Auflistung. Die Workbooks-Auflistung enthält alle derzeit in Microsoft Excel geöffneten Workbook-Objekte.

Verwenden des Workbook-Objekts

In diesem Abschnitt werden die folgenden Eigenschaften zur Rückgabe eines Workbook-Objekts beschrieben:

- Workbooks-Eigenschaft
- ActiveWorkbook-Eigenschaft
- ThisWorkbook-Eigenschaft

Workbooks-Eigenschaft

Verwenden Sie Workbooks(Index), wobei Index der Name oder die Indexnummer der Arbeitsmappe ist, um ein einzelnes Workbook-Objekt zurückzugeben. Im folgenden Beispiel wird die erste Arbeitsmappe aktiviert.

```
Workbooks(1).Activate
```

Die Indexnummer gibt die Reihenfolge an, in der die Arbeitsmappen geöffnet oder erstellt wurden. Workbooks(1) ist die zuerst erstellte Arbeitsmappe, Workbooks(Workbooks.Count) die zuletzt erstellte. Durch das Aktivieren einer Arbeitsmappe wird ihre Indexnummer nicht geändert. Alle Arbeitsmappen sind in der Numerierung enthalten, auch dann, wenn sie ausgeblendet sind.

Die Name-Eigenschaft gibt den Arbeitsmappennamen zurück. Es ist nicht möglich, den Namen mittels dieser Eigenschaft festzulegen. Wenn Sie den Namen ändern müssen, speichern Sie die Arbeitsmappe mit der SaveAs-Methode unter einem anderen Namen. Im folgenden Beispiel wird das Tabellenblatt "Tabelle1" in der Arbeitsmappe "Zahnrad.xls" aktiviert (die Arbeitsmappe muß bereits in Microsoft Excel geöffnet sein).

```
Workbooks("Zahnrad.xls").Worksheets("Tabelle1").Activate
```

ActiveWorkbook-Eigenschaft

Die ActiveWorkbook-Eigenschaft gibt die derzeit aktive Arbeitsmappe zurück. Im folgenden Beispiel wird der Name des Autors für die aktive Arbeitsmappe festgelegt.

```
ActiveWorkbook.Author = "Birgit Neureuter"
```

ThisWorkbook-Eigenschaft

Die ThisWorkbook-Eigenschaft gibt die Arbeitsmappe zurück, in der der Visual Basic-Code ausgeführt wird. Meist ist diese mit der aktiven Arbeitsmappe identisch.

Ausgaben formatieren

Format-Funktion

Syntax (Kurzform)

Format(Ausdruck[, Format])

Beispiele:

In diesem Beispiel werden verschiedene Einsatzmöglichkeiten der Format-Funktion zur Formatierung von Werten unter Verwendung von integrierten und benutzerdefinierten Formaten dargestellt. Die tatsächliche Ausgabe bei der Verwendung des Datum-Trennzeichens (/), Zeit-Trennzeichens (:) und der AM-/PM-Zeichenfolge hängt von dem aktuellen Gebietsschema Ihres Systems ab.

Dim Zeit1, Datum1, ZF1

Zeit1 = #5:04:23 PM#

Datum1 = #1/27/93#

ZF1 = Format(Time, "Long Time") Aktuelle Systemzeit im langen Zeitformat

ZF1 = Format(Date, "Long Date") Aktuelles Systemdatum im langen Datumsformat.

ZF1 = Format(Zeit1, "h:m:s") Liefert "17:4:23".

ZF1 = Format(Zeit1, "hh:mm:ss AMPM") Liefert "17:04:23".

ZF1 = Format(Datum1, "dddd, mmm d yyyy") Liefert "Mittwoch, 27. Januar 1993".

Benutzerdefinierte Formate

Wichtig:

, liefert den Tausenderpunkt

. steht für Dezimalkomma (vgl. Gebietsschema)

ZF1 = Format(5459.4, "##,##0.00") Liefert "5.459,40".

ZF1 = Format(334.9, "###0.00") Liefert "334,90".

ZF1 = Format("BANANE", "<") Liefert "banane".

ZF1 = Format("birne", ">") Liefert "BIRNE".

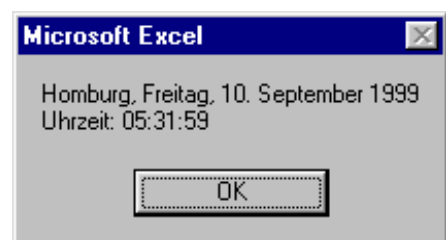
Stringverknüpfung

Bei der Stringverknüpfung handelt es um eine einfache Addition von Teilstrings mit Hilfe des Verknüpfungsoperators „&“. Es können Variablen, Konstanten und beide auch beliebig gemischt zusammengefügt werden.⁵

```
Sub formate2()
Dim datum, ort
    datum = Format(Date, "long date")
    ort = "Homburg"
    MsgBox ort & ", " & datum
End Sub
```



```
Sub formate3()
Dim datum, zeit, ort
    datum = Format(Date, "long date")
    zeit = Format(Time, "long time")
    ort = "Homburg"
    MsgBox ort & ", " & datum & Chr(13) _
        & "Uhrzeit: " & zeit
End Sub
```



Hinweise:

chr(13) erzeugt einen Zeilenumbruch. chr(10) erzeugt einen Zeilenvorschub. Mit der Funktion chr() können auch Zeichen ausgegeben werden (chr(37) -> liefert z. B. das Zeichen %).

Der Unterstrich (_) mit führendem Leerzeichen erlaubt einen Zeilenumbruch langer Programmzeilen.

Weitere Stringoperationen

Ausgabe eines Teilstrings (auch Leerzeichen sind Zeichen!!!)

```
Left(String, Länge)
Right(String, Länge)
Mid(String, Start, Länge des Rückgabestrings)
```

Beispiele

```
Sub teilstring()
    MsgBox Left("Das ist ein schöner Tag", 5)
    MsgBox Right("Das ist ein schöner Tag", 5)
    MsgBox Mid("Das ist ein schöner Tag", 5, 2)
End Sub
```



⁵ Dieter Staas, Excel 97 für Anwendungsprogrammierer, Hanser-Verlag, 1997, S. 129

Zeichenfolgeausdrücke trimmen

Ltrim(Zeichenfolgeausdruck) -> entfernt führende Leerzeichen

Rtrim(Zeichenfolgeausdruck) -> entfernt nachfolgende Leerzeichen

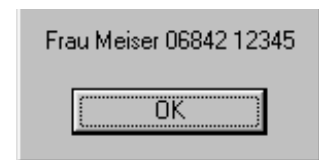
Trim(Zeichenfolgeausdruck) -> entfernt führende und nachfolgende Leerzeichen

Beispiel:

In Tabelle1 seien in den Zellen A1 bis C1 Daten enthalten, die führende Leerzeichen enthalten. Über die Ltrim- bzw. Trim-Funktion können diese entfernt werden.

```
Sub trimmen()
    Dim Anrede As Object
    Dim Name As Object
    Dim Telefon As Object

    Set Anrede = Worksheets("Tabelle1").Range("A1")
    Set Name = Worksheets("Tabelle1").Range("B1")
    Set Telefon = Worksheets("Tabelle1").Range("C1")
    MsgBox Anrede & " " & Name & " " & Telefon
    MsgBox Trim(Anrede) & " " & Trim(Name) & " " & Trim(Telefon)
End Sub
```



Umwandlung in Gross- und Kleinbuchstaben

Diese Funktionen sind insbesondere hilfreich, um bei unterschiedliche Anwendung der Gross- und Kleinschreibung im Rahmen von Stringvergleichen Fehler auszuschliessen.

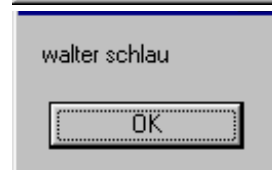
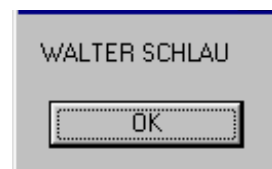
Ucase(Zeichenfolgeausdruck) -> Umwandlung in Grossbuchstaben

Lcase(Zeichenfolgeausdruck) -> Umwandlung in Kleinbuchstaben

Beispiel:

```
Sub umwandlung()
    Dim Vorname As Object
    Dim Name As Object
    Dim Gesamtname As String

    Set Vorname = Worksheets("Tabelle1").Range("A2")
    Set Name = Worksheets("Tabelle1").Range("B2")
    Gesamtname = Vorname & " " & Name
    MsgBox UCase(Gesamtname)
    MsgBox LCase(Gesamtname)
End Sub
```



Kontrollstrukturen

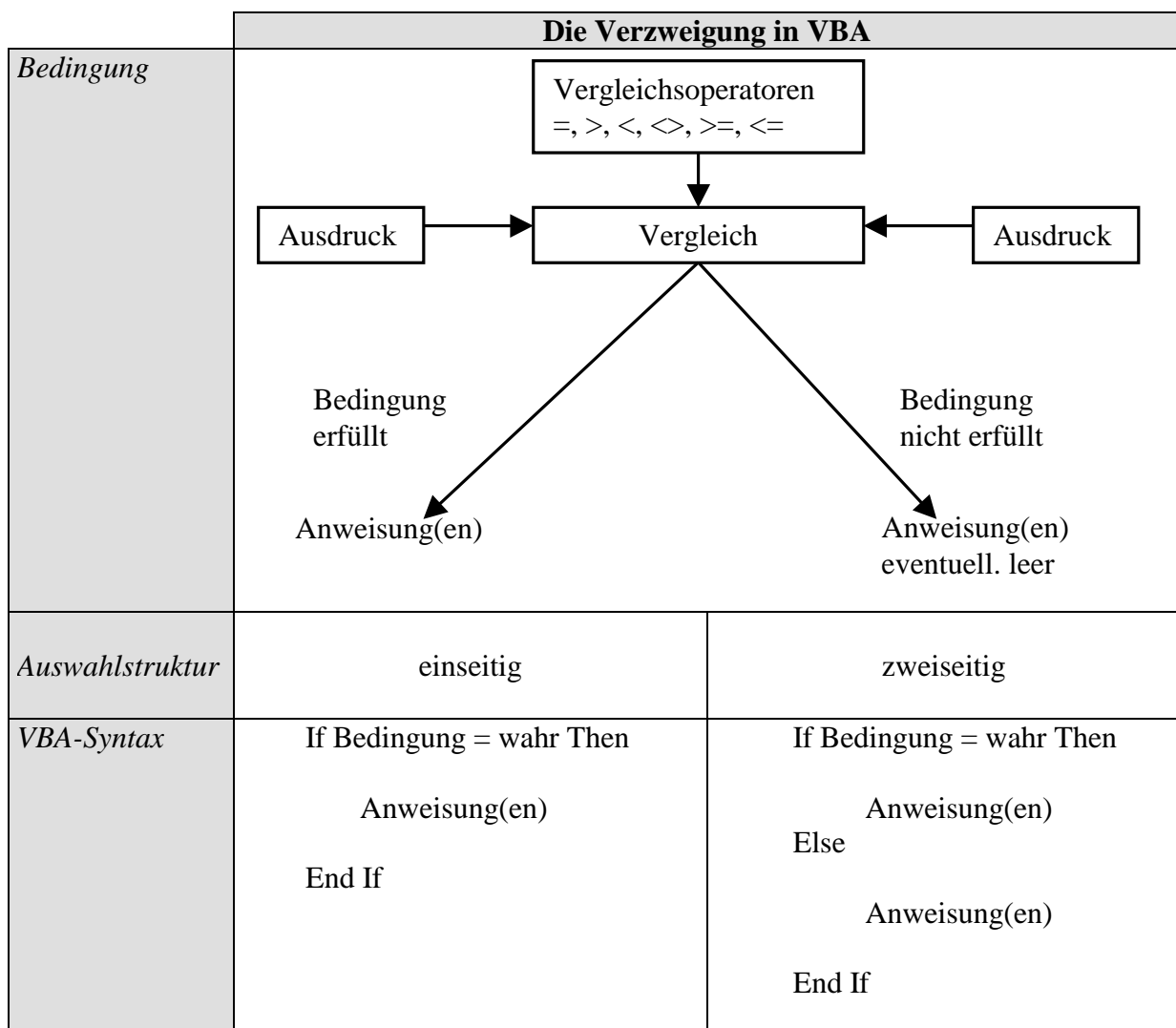
Der Programmablauf wird über verschiedene Kontrollstrukturen gesteuert. Dadurch können alternative Abläufe sowie kontrollierte Wiederholungen eingebaut werden. Die Kontrollstrukturen unter VBA sind denen in anderen Programmiersprachen (z. B. Basic, Pascal, C) ähnlich.

Grundtypen:

- Sequenz (wir hier nicht explizit behandelt)
- Verzweigung
- Mehrfachauswahl
- Schleife

Die Verzweigung

Struktur



Mit dem If-Konstrukt kann insbesondere auch der Rückgabewert der InputBox ausgewertet werden (vgl. S. 12).

Komplexere Verzweigungen können mit einer Variante von If konstruiert werden.

```
If Bedingung1 = wahr Then
    Anweisung(en)
ElseIf Bedingung2 = Wahr Then
    Anweisung(en)
ElseIf Bedingung3 = Wahr Then
    Anweisung(en)
.
.
.
Else
    Anweisung(en)
End If
```

Beispiel:

```
Sub noten()
    Dim note As Integer
    note = InputBox("Geben Sie eine Note ein!", "Noten")
    If note = 1 Then
        MsgBox ("sehr gut")
    ElseIf note = 2 Then
        MsgBox ("gut")
    .
    .
    .
    ElseIf note = 5 Then
        MsgBox ("mangelhaft")
    Else
        MsgBox ("ungenügend")
    End If
End Sub
```

Merke: Nach der Abarbeitung des If-Konstruktes wird der Programmablauf hinter End If fortgesetzt.

Die Mehrfachauswahl

Alternativ zur If-Abfrage kennt VBA eine weitere Verzweigungsstruktur. Insbesondere bei mehrseitiger Auswahl läßt sie sich häufig einfacher handhaben und übersichtlicher einsetzen als IF. Die mehrseitige Auswahl ist davon abhängig, welchen Wert eine Variable besitzt. Die Variable, deren Wert geprüft werden soll, steht in *Ausdruck*.

Syntax:

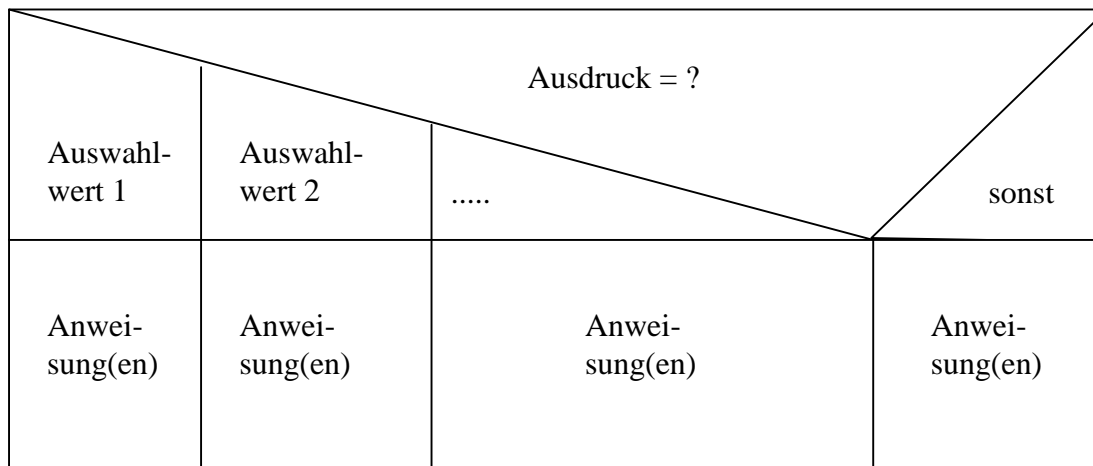
```
Select Case Ausdruck
    Case Auswahlwert 1
        Anweisung(en)
    Case Auswahlwert 2
        Anweisung(en)
    .....
    Case Auswahlwert n
        Anweisung(en)
    Case else
        Anweisung(en)
End Select
```

Ausdruck kann eine Variable, eine Konstante oder ein arithmetischer Ausdruck sein.

Jeder Block beginnt mit einem Auswahlwert. Dabei kann es sich um einen konkreten Wert, einen Wertebereich oder um einen Vergleichsausdruck handeln.

Case select wird bearbeitet, wenn keiner der Auswahlwerte mit dem Ausdruck übereingestimmt hat

Strukturdiagramm



Handelt es sich beim Ausdruck um Auswahlwertemengen, so müssen für den Auswahlwert die Ober- und Untergrenzen angegeben werden.

Select Case Ausdruck

```

Case Anfang 1 To Ende 1
    Anweisung(en)
Case Anfang 2 To Ende 2
    Anweisung(en)
....
Case else
    Anweisung(en)
End Select

```

Beispiel:

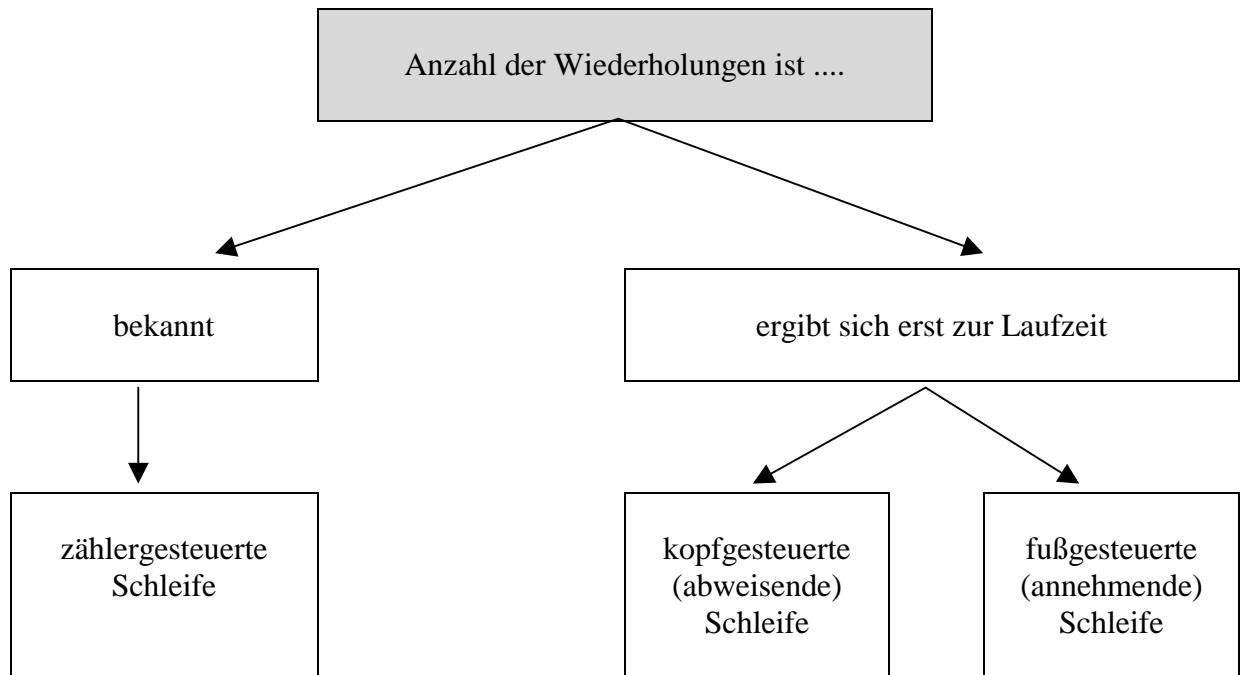
```

Sub Bereiche()
    Dim umsatz As Double
    umsatz = InputBox("Geben Sie einen Umsatz ein!")
    Select Case umsatz
        Case Is < 0
            MsgBox ("Das war nicht viel")
        Case 0 To 10000
            MsgBox ("Naja")
        Case 1000 To 40000
            MsgBox ("Donnerwetter")
        Case Is > 40000
            MsgBox ("Gratulation")
    End Select
End Sub

```

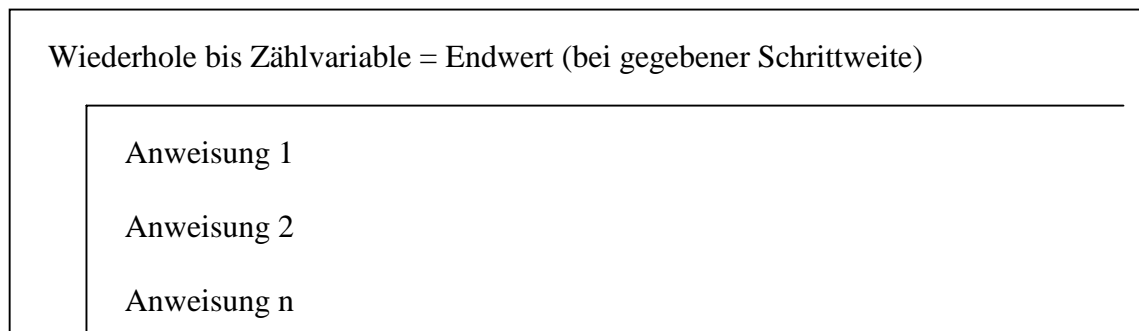
Die Wiederholung (Schleife)

Schleifen dienen dazu, Anweisungen wiederholt auszuführen. VBA kennt eine Vielzahl von Schleifentypen, von denen im Folgenden jedoch nur die wichtigsten angesprochen werden (Zählschleife; kopf- bzw. fustgesteuerte Schleife).



Die zählergesteuerte Schleife

Strukturdiagramm:



Syntax:

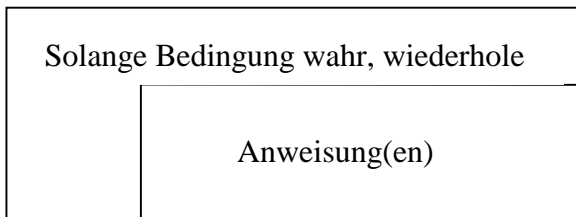
For Zähler = Startwert To Endwert

Anweisung(en)

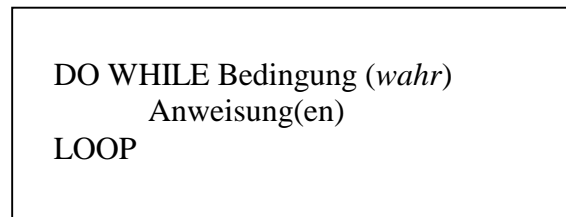
Next

Kopfgesteuerte Schleife

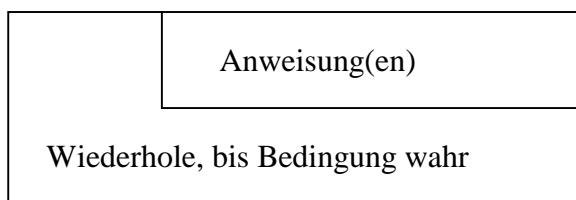
Strukturdiagramm:



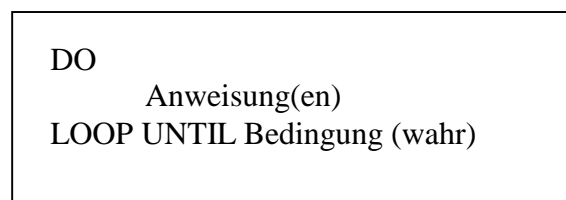
Syntax:

*Fußgesteuerte Schleife*

Strukturdiagramm:



Syntax:



Zusammenfassung:

Merkmale unterschiedlicher Schleifentypen

Schleifentyp	zählergesteuert	kopfgesteuert	fußgesteuert
Festlegung des Anfangswertes			
Überprüfung der Abbruchbedingung			
Änderung der Laufvariablen			

Sonderfall:

Diese Schleife ist besonders für die Abarbeitung von Feldern bzw. von Aufzählungsmethoden geeignet (z. B. Zugriff auf alle nicht leeren Felder eines Tabellenblattes; Zugriff auf eine Gruppe zusammengehörender Objekte usw.)

For Each Element in Gruppe

Anweisung(en)

Next Element

Beispiel:

```
Sub Blätter_zeigen()
  Dim Blattname As Object
  For Each Blattname In ActiveWorkbook.Sheets
    MsgBox Blattname.Name
  Next Blattname
End Sub
```

Ergebnis: Zeigt nacheinander die Blattnamen der aktiven Arbeitsmappe in Meldeboxen an.

Alternativ:

```
Sub Blätter_zeigen2()
  Dim Blattname As Variant
  For Blattname = 1 To Application.Sheets.Count
    MsgBox Application.Sheets(Blattname).Name
  Next Blattname
End Sub
```

Sprunganweisungen (Auszug)

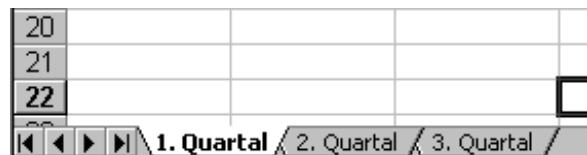
Mit diesen Anweisungen kann eine Kontrollstruktur oder eine Prozedur verlassen werden, wenn eine bestimmte Bedingung erfüllt ist.

Exit Sub -> Prozedur wird verlassen

Exit For -> Schleife wird verlassen

Beispiel:

```
Sub Sprung_Abbruch()
  Dim Zähler As Integer
  For Zähler = 1 To Worksheets.Count
    Worksheets(Zähler).Name = Zähler & ". Quartal"
    If Zähler = 4 Then
      Exit For
    End If
  Next Zähler
End Sub
```

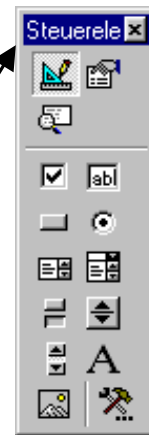
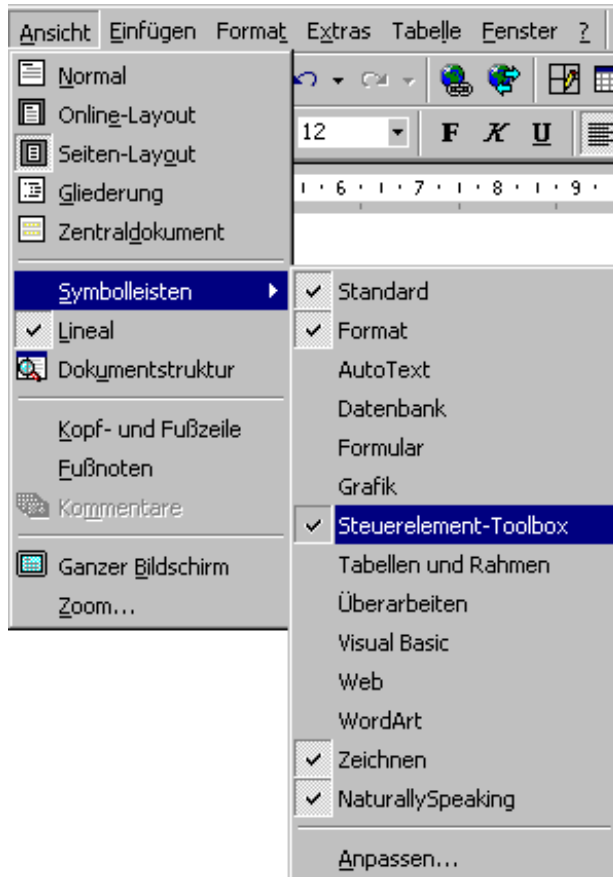


Arbeiten mit Dialogen

Dialoge können auf der Tabellenebene oder aber in VBA eingesetzt werden. Mit Hilfe der einzelnen Elemente eines Dialoges können z. B. Eingaben vorgenommen werden (etwa über Listenfelder) oder Steuerfunktionen ausgeführt werden (etwa über Schaltflächen). Generell gilt, dass das Erstellen von Dialogen auf VBA-Ebene – nicht zuletzt wegen der Vielzahl an Eigenschaften - dem Programmentwickler erheblich mehr Möglichkeiten bietet.


Dialoge auf Tabellenebene

Die einzelnen Elemente erreicht man über die Symbolleiste Steuerelement-Toolbox (die Symbolleiste Formulare stammt noch aus der Zeit von Excel 5/7 und bietet im Vergleich zur Steuerelement-Toolbox nur eingeschränkte Funktionalität).

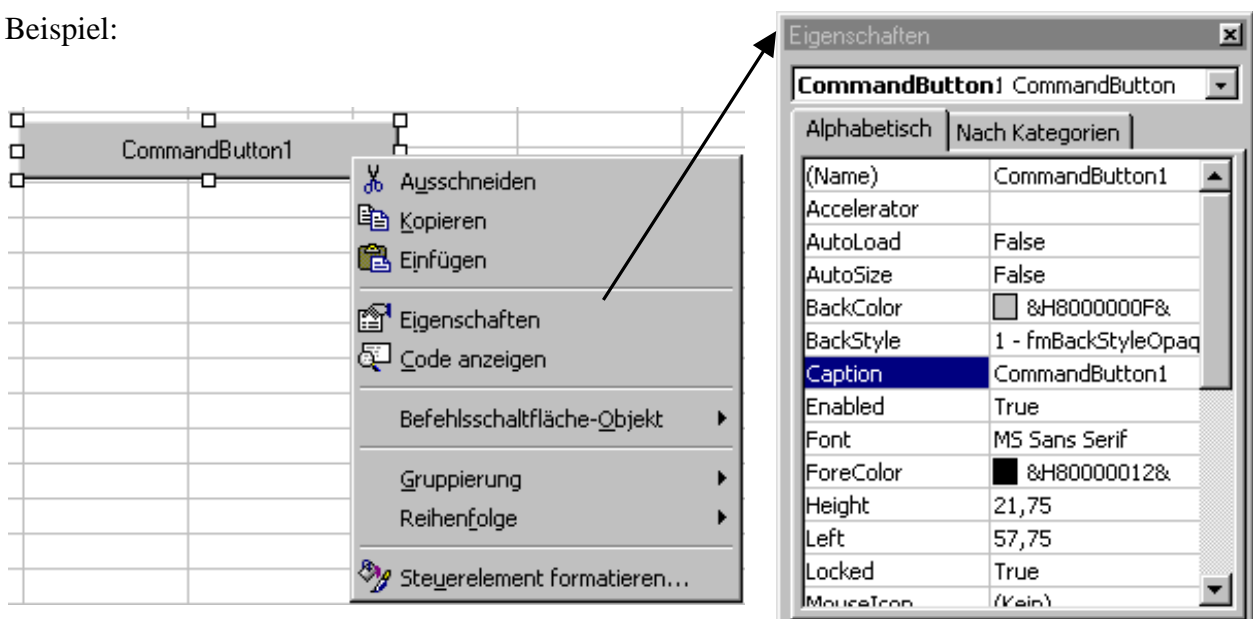


Als Element eines Dialogs bieten sich z. B. an:

- Kontroll- und Optionsfelder
- Text- und Bezeichnungsfelder
- Schaltflächen
- Listen- und Kombinationsfelder

Die Elemente der Steuerelement-Toolbox verfügen über eine Reihe von Eigenschaften, die im Eigenschaftsfenster eingestellt werden können (Aufruf des Kontextmenüs durch rechten Mausklick auf das Element bei aktiviertem Entwurfsmodus ).

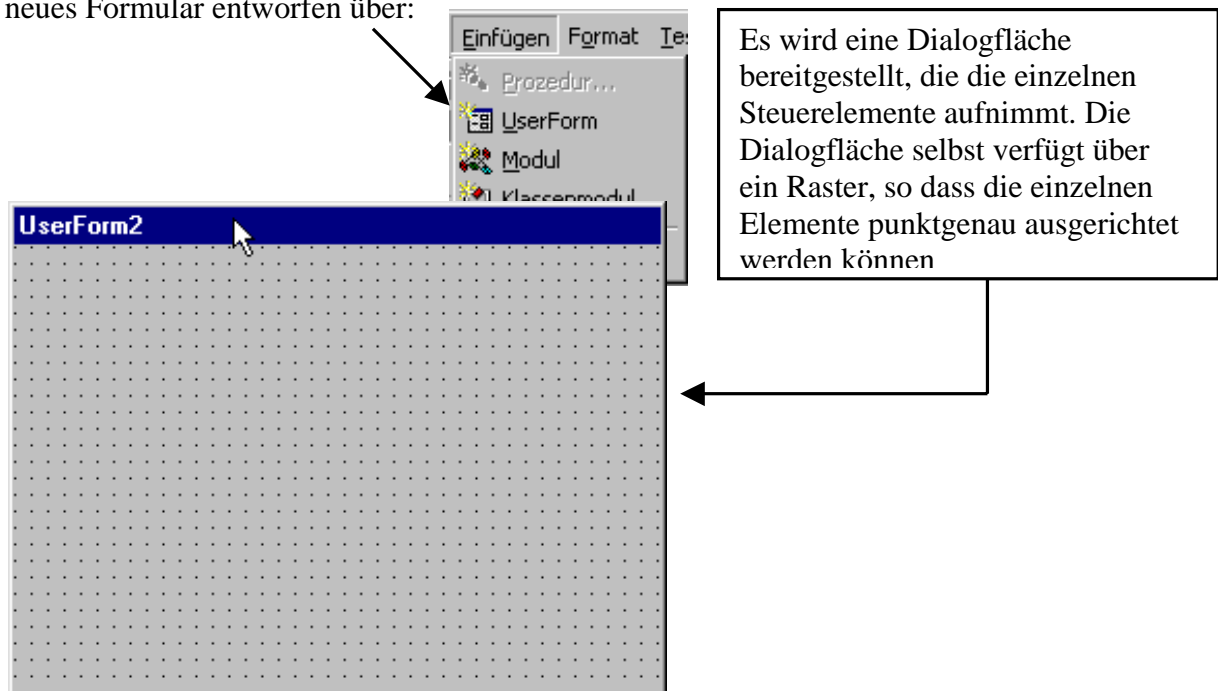
Beispiel:



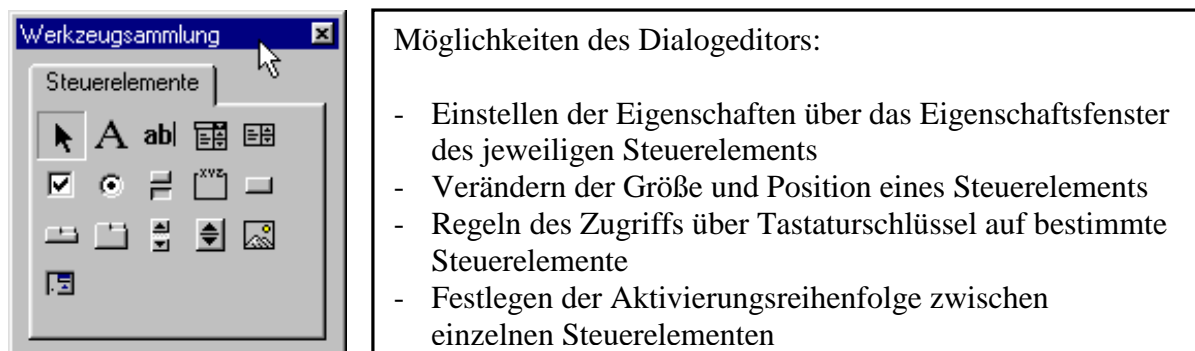
Die meisten Eigenschaften dienen dazu, das gewählte Steuerelement zu formatieren bzw. seine Position am Bildschirm festzulegen. Interessant sind die Möglichkeiten, über die Eigenschaft *Name* den voreingestellten Namen zu ändern und über *Caption* ein Steuerelement zu beschriften. Die letztgenannte Eigenschaft kann beispielsweise auch zur Laufzeit ihren Wert ändern.

Dialoge über UserForm

Mit dem Dialogeditor, der ein Bestandteil von VBA ist, können auf komfortable Weise benutzerdefinierte Formulare entwickelt werden. In der Entwicklungsumgebung wird ein neues Formular entworfen über:

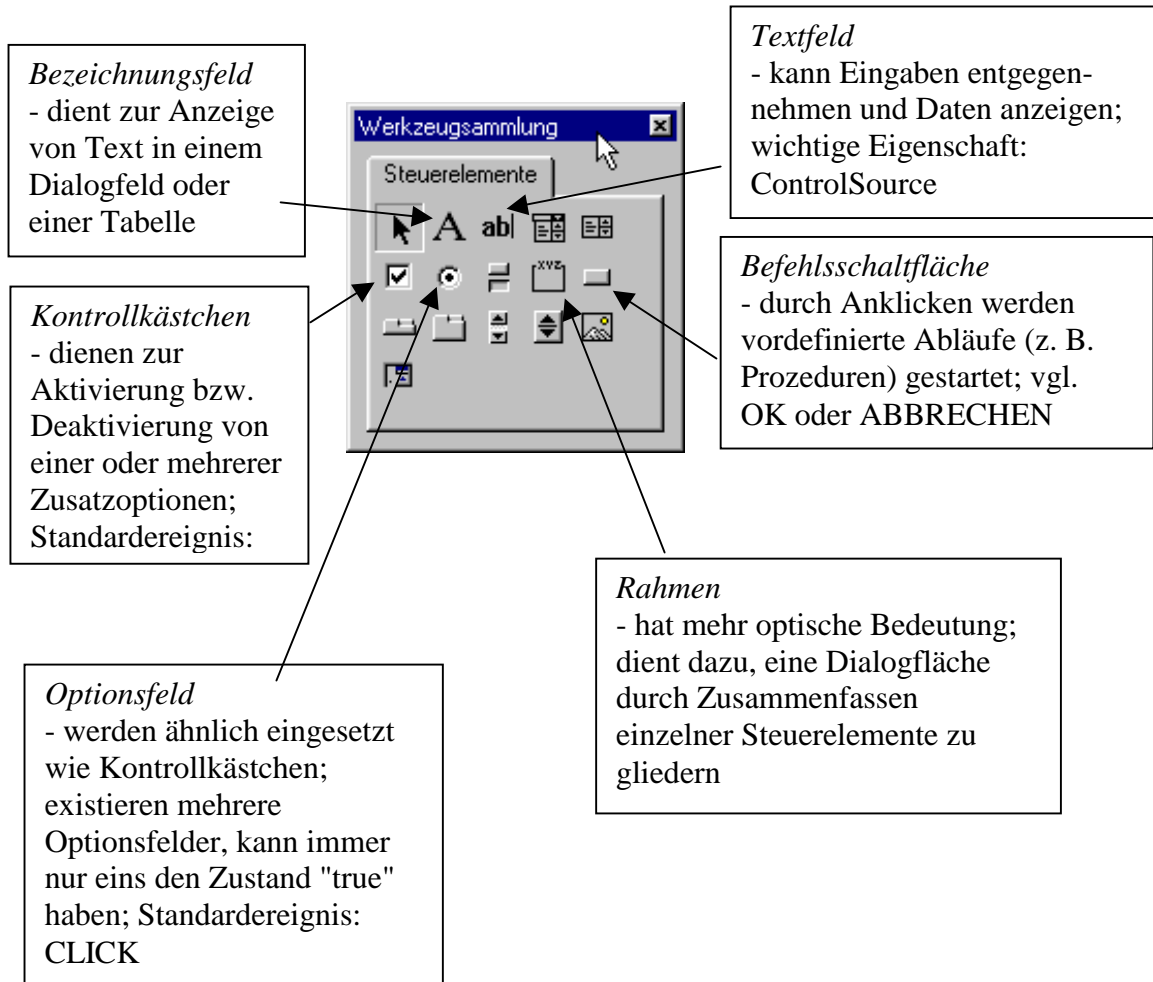


Gleichzeitig mit der Dialogfläche wird eine sogenannte „Werkzeugsammlung“ eingeblendet. Sie bietet alle Steuerelemente an, die für den Entwurf des Formulars zur Verfügung stehen.



Darüber hinaus gibt es verschiedene Möglichkeiten der optischen Gestaltung, wie z. B.:

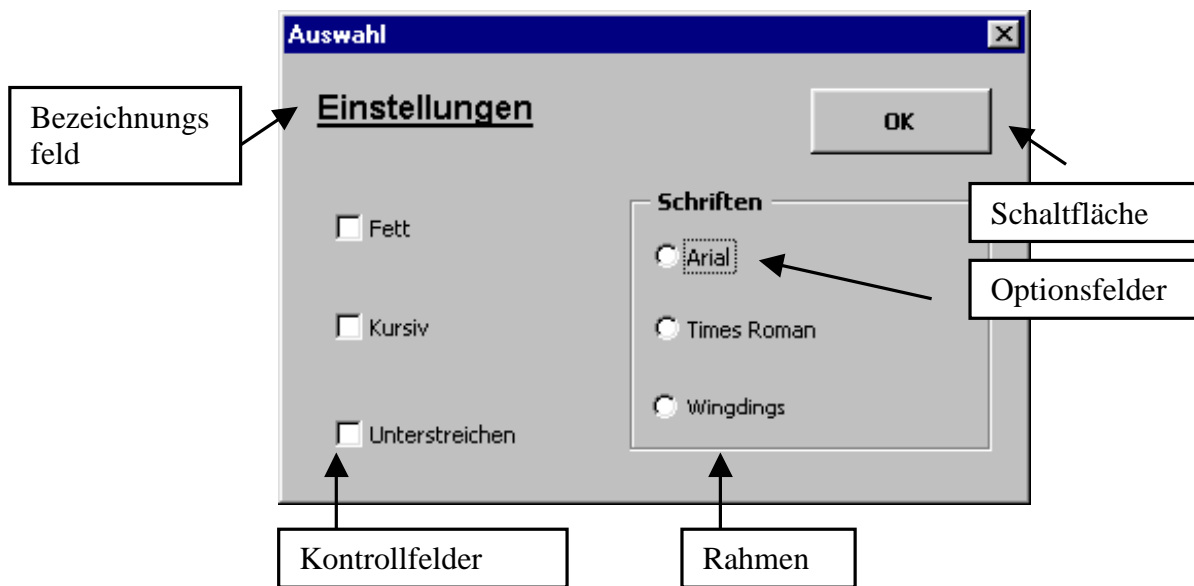
- farbliche Gestaltung eines Steuerelements
- Gruppieren einzelner Steuerelemente, die sachlogisch zusammengehören



Beispiele/Steuerelemente	Code, der erzeugt wird (Standardereignis)
<input type="text"/>	Private Sub TextBox1_Change() End Sub
<input type="checkbox"/> CheckBox1	Private Sub Check_Box1_Click() End Sub
<input type="radio"/> OptionButton1	Private Sub OptionButton1_Click() End Sub
<input type="button" value="CommandButton1"/>	Private Sub CommandButton1_Click() End Sub
<input type="spinbutton"/>	Private Sub SpinButton1_Change() End Sub

Eine interessante Eigenschaft ist *ControlSource*. Über sie kann der Inhalt eines Steuerelementes mit einer Zelle verbunden werden (z. B. um ein Textfeld mit dem Wert aus einer Tabelle vorzubelegen).

Beispiele:



Auszug aus dem Quellcode:

```
Private Sub CheckBox1_Click()
    Range("A1").Select
    If CheckBox1 = True Then
        Selection.Font.Bold = True
    Else
        Selection.Font.Bold = False
    End If
End Sub
```

```
Private Sub CheckBox2_Click()
    Range("A1").Select
    If CheckBox2 = True Then
        Selection.Font.Italic = True
    Else
        Selection.Font.Italic = False
    End If
End Sub
```

usw.

```
Private Sub OptionButton1_Click()
    Range("A1").Select
    If OptionButton1 = True Then
        With Selection.Font
            .Name = "Arial"
            .Size = 10
        End With
    End If
End Sub
```

usw.

Beenden des Dialogs über Schaltfläche OK:

```
Private Sub CommandButton1_Click()
    Unload Me
End Sub
```

Hinweis:

Die Betrachtung von Listenfeldern, Kombinationsfeldern, Drehfeldern und Multipages ist nicht Gegenstand dieses Skripts!

Makro einem Symbol zuweisen

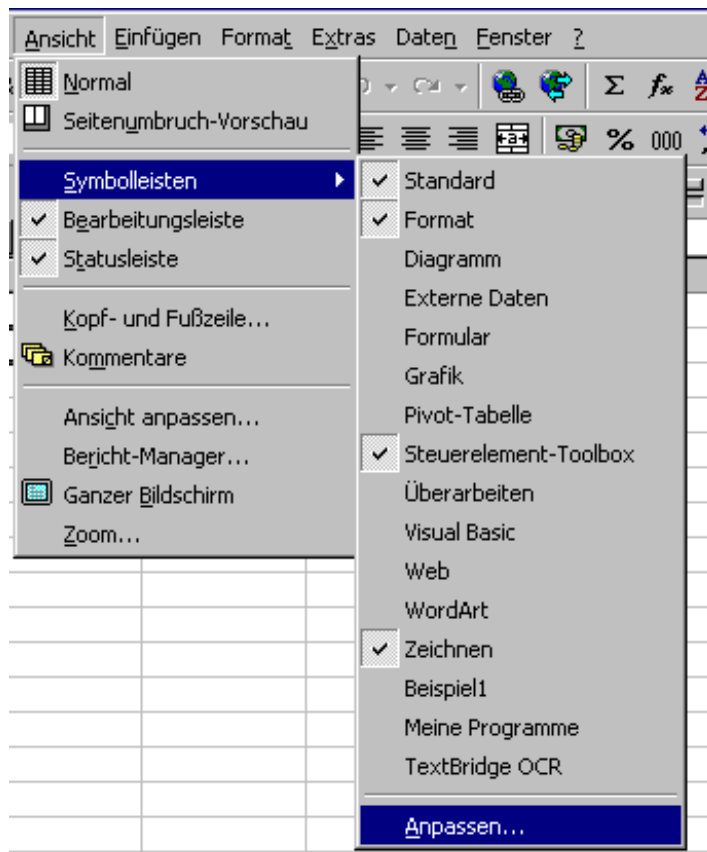
Mitunter kann es sehr interessant sein, ein Makro über ein Symbol aus einer Symbolleiste zu starten.

Angenommen, Sie haben ein Makro aufgezeichnet, das den Dateinamen einschließlich des kompletten Pfads in den Fußbereich eines Dokuments druckt (vgl. Aufgabe 17). Dieses Makro wollen Sie einem neuen Symbol zuweisen und es in die Standardsymbolleiste einfügen.



neues Symbol, das die Pfadeintragung bewirkt

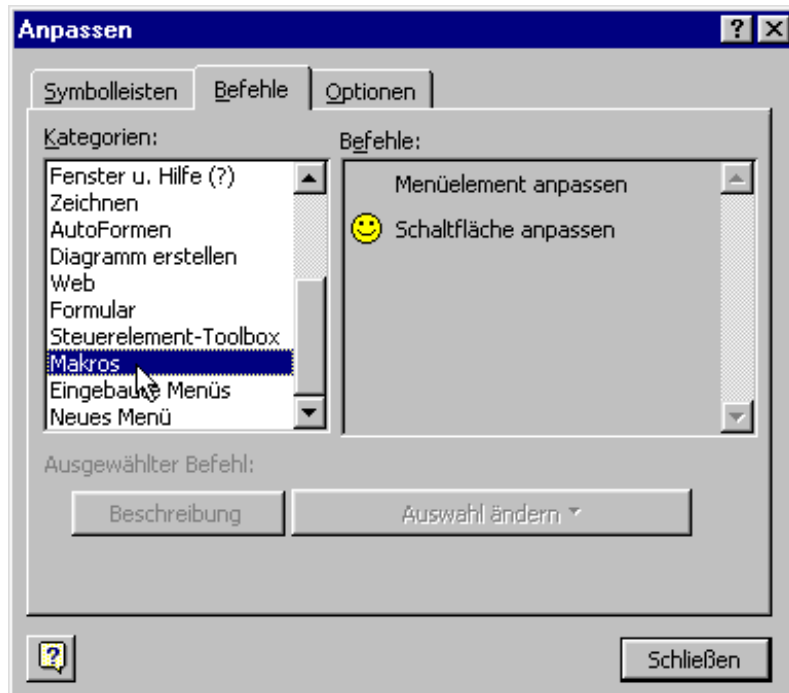
Schritt 1:



Hinweis:

Über die Auswahl *Ansicht/Symbolleisten/Anpassen* können auch neue Symbolleisten entworfen bzw. bestehende Symbolleisten geändert werden.

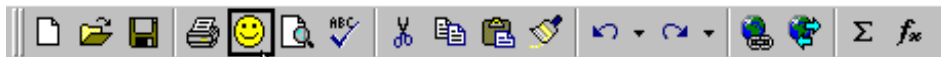
Schritt 2:



Wählen Sie nun das Registerblatt *Befehle* und aus dem Fenster *Kategorien* den Punkt *Makros*. Rechts erscheint nun ein „Standard-symbol“, dem ein Makro zugewiesen werden kann.

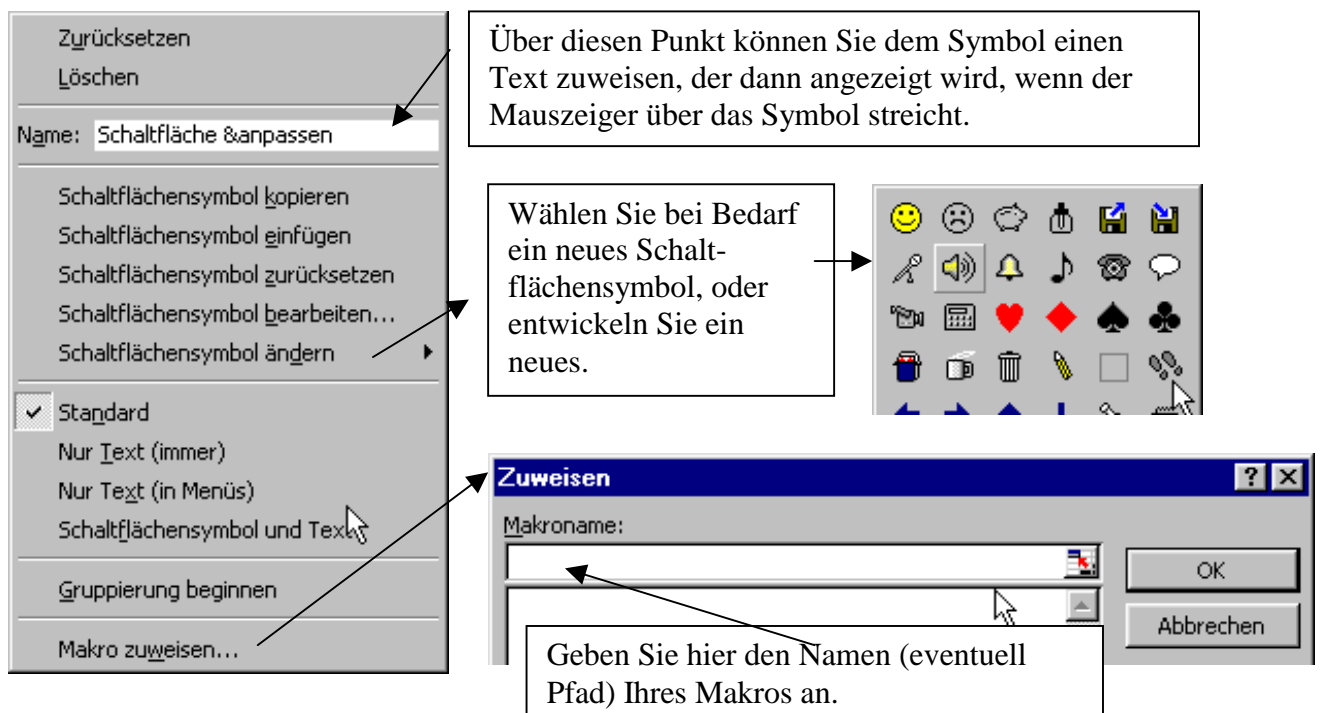
Schritt 3:

Markieren Sie das Symbol und schieben Sie es bei gedrückter linker Maustaste an die gewünschte Position in der Symbolleiste.



Schritt 4:

Klicken Sie das Symbol mit der rechten Maustaste an (alternativ: Schaltfläche *Auswahl ändern* aus dem Fenster *Anpassen* anklicken).



Ein Symbol kann gelöscht werden, indem Sie wiederum über *Ansicht/Symboleisten/Anpassen* gehen, anschließend das betreffende Symbol markieren und es im letzten Schritt löschen.

Erstellen von benutzerdefinierten Funktionen

VBA bietet die Möglichkeit, eigene Funktionen zu erstellen. Funktionen sind in ihrem Aufbau den Prozeduren sehr ähnlich, geben aber im Unterschied zu diesen einen Wert zurück. Daher können Funktionen auch in Ausdrücken verwendet werden.

Beispiel:

Es ist eine Funktion zu erstellen, die die Höhe der Provision in Abhängigkeit vom Umsatz berechnet. Wie jede Prozedur, so werden auch die Funktionen in der Entwicklungsumgebung erstellt.

Function provision(umsatz)

 If umsatz <= 0 Then

 provision = 0

 ElseIf umsatz > 0 And umsatz < 10000 Then

 provision = umsatz * 0.05

 Else

 provision = umsatz * 0.1

 End If

End Function

So erstellte Funktionen können ohne weiteres auch auf Tabellenebene über den Befehl *Einfügen/Funktion* genutzt werden.

