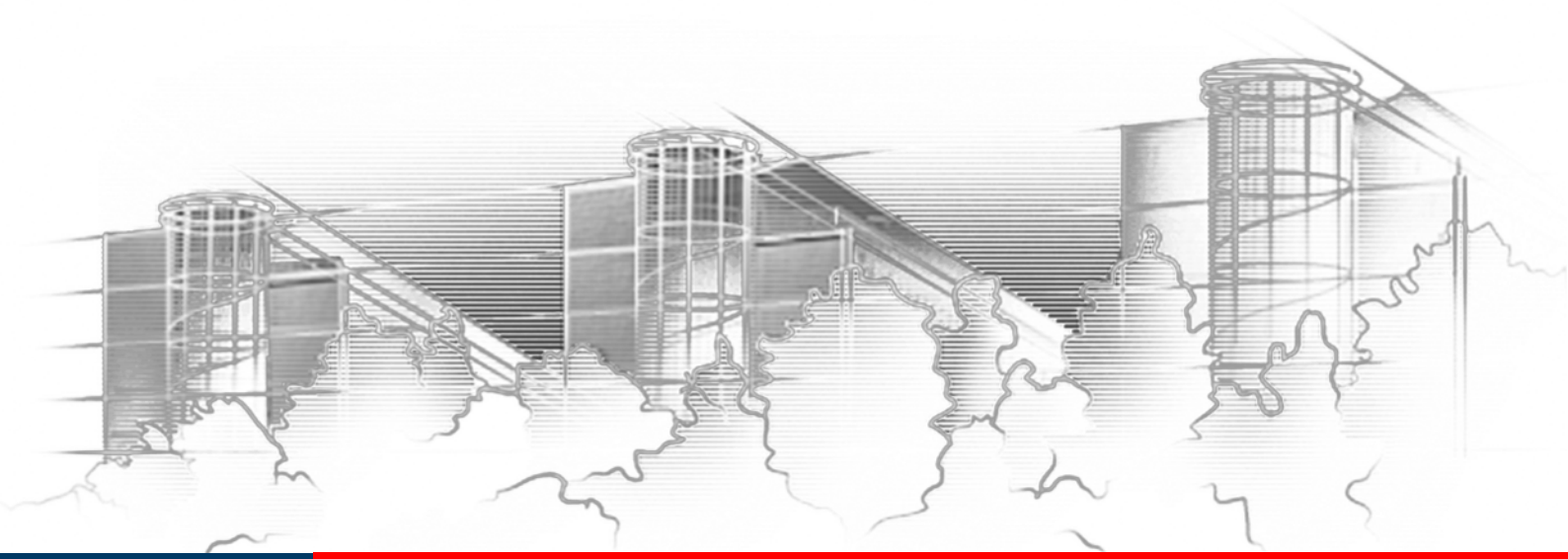
Freie
Hansestadt
Bremen

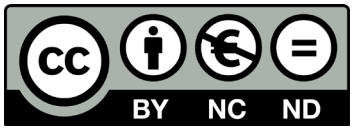


MAKROS – FORTFÜHRUNG

EXCEL 2016

AFZ
wir bilden zukunft
Aus- und Fortbildungszentrum
für den bremischen öffentlichen Dienst

Diese Lizenz ermöglicht nicht die Nutzung folgender eventuell enthaltener Inhalte:
- Hoheits- und Wahrzeichen der Freien Hansestadt Bremen
- Titelbild und Logo
- Bildschirmfotos aus dem Internet
- Personenbezogene Daten
- Unrechtmäßig veröffentlichtes Material



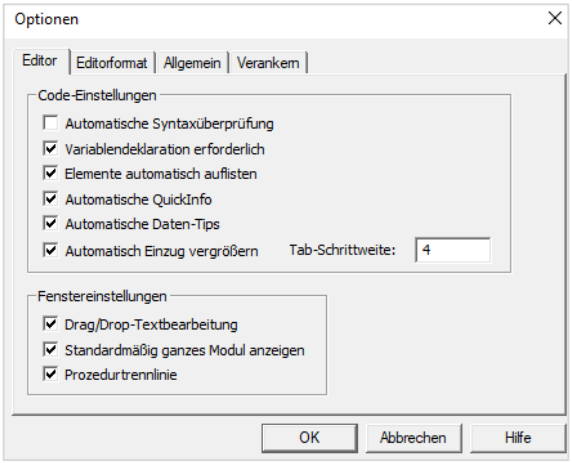
Namensnennung - Nicht-kommerziell - Keine Bearbeitung

1. Einleitung

Diese Kursmappe baut auf den Inhalten der Mappe **Makros – Einführung** auf und setzt deren Inhalte zum Verständnis voraus. Grundlegende Bedienung und Einstellungen zum Visual Basic-Editor (VBE) entnehmen Sie bitte auch der Einführungsmappe.

1.1 Ergänzende Einstellungen im VB-Editor vornehmen

Die Einstellungen zum VB-Editor (VBE) rufen Sie über den Menüpunkt **Extras** und dort über den Eintrag **Optionen** auf. Auf der ersten Registerkarte **Editor** werden zwei Optionen näher erläutert.

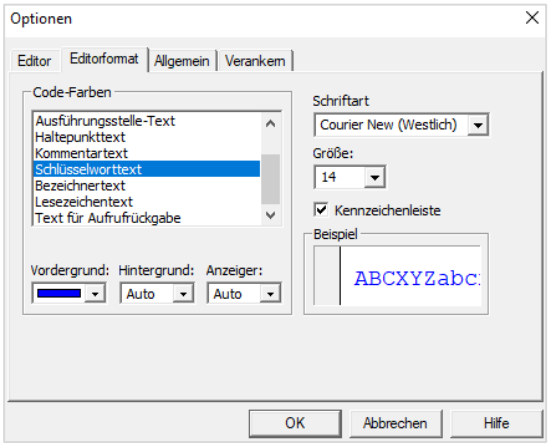


Die zweite Option **Variablendeklaration erforderlich** sollte immer gesetzt sein. Zu Beginn eines Moduls erscheint damit automatisch beim Einfügen des Moduls die Anweisung **Option Explicit**. Diese Anweisung muss vor den Prozeduren stehen und zwingt dazu, dass innerhalb aller Prozeduren genutzte Variablen deklariert, also bestimmt werden müssen.

Was sind die Vorteile dabei?

- Variablen haben den gewollten Datentypen, Typenkonflikte lassen sich somit vermeiden
- Variablen reservieren nur so viel Speicherplatz, wie sie auch wirklich benötigen
- Der Programmcode läuft schneller ab, als mit undeklarierten Variablen.

Wenn Sie das Häkchen bei **Automatische Syntaxprüfung** entfernen, schalten Sie damit das Meldungsfenster aus, das bei einem Syntaxfehler eingeblendet wird. Die Syntax wird trotzdem geprüft und bei Fehlern rot unterstrichen. Der Arbeitsfluss wird durch das Meldungsfenster jedoch nicht unterbrochen.



Auf der Registerkarte Editorformat können Sie beispielsweise die Schriftgröße verändern, um den Programmtext besser lesbar zu machen. Die Schriftart Courier New sollten Sie nicht verändern, da sich bei der Wahl einer Proportionalschrift die Lesbarkeit des Programmcodes verschlechtern kann.

2. Variablen und deren Deklaration

Variablen sind Platzhalter für die Daten, die verarbeitet werden sollen. Mit dem VBA-Schlüsselwort **Dim** und einem selbstgewählten Bezeichner, z. B. **int_Zahl,** stellen Sie sozusagen einen Container bereit, der Daten aufnehmen kann. Durch die Anweisung **As Integer** bestimmen Sie, von welcher Art die Daten sind und wieviel Speicherplatz sie einnehmen.

2.1 Namenskonventionen bei der Deklaration von Variablen

Bei den Variablennamen müssen folgende Regeln eingehalten werden:

- Sie müssen mit einem Buchstaben beginnen,
- sie dürfen maximal 255 Zeichen haben,
- sie dürfen nicht wie VBA-Schlüsselwörter heißen,
- sie dürfen keine Sonderzeichen enthalten, z. B. Leerzeichen, +, -, -, / usw.

2.2 Datentypen

Der Datentyp Variant wird von VBA automatisch gewählt, wenn Sie keinen Datentypen vorgeben. Der Datentyp Variant kann sowohl Text als auch verschiedene Zahlen-Datentypen enthalten. Wenn Sie wissen, die Daten sind nur Zahlen einer bestimmten Größenordnung, dann ist es immer sinnvoller einen passenden Datentyp zu wählen. So sparen Sie Speicherplatz und der Programmcode läuft schneller ab.

Der Datentyp Integer hat z.B. einen Speicherbedarf von 2 Bytes und der Datentyp Double einen Speicherbedarf von 8 Bytes. Der Datentyp Variant reserviert grundsätzlich einen Speicher von mindestens 16 Bytes.

Einige andere Datentypen und ihre Wertebereiche:

Name	Typ	Wertebereich	Präfix
Byte	Ganze Zahl	0 bis 255	byt
Integer	Ganze Zahl	-32.768 bis 32.767	int
Long	Ganze Zahl	-2.147.483.648 bis 2.147.483.647	lng
Single	Dezimalzahl mit 8 Stellen Genauigkeit	$\pm 3{,}4*10^{38}$	sng
Double	Dezimalzahl mit 16 Stellen Genauigkeit	$\pm 1{,}8*10^{308}$	dbl
Currency	Festkommazahl mit vier Nachkommastellen	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807	cur
String	Text	Beliebige Zeichenfolge	str
Variant	Text oder Zahlen		var
Boolean	Wahrheitswert	True oder False	bln
Object	Beliebiger Verweis auf ein Objekt		obj

Wenn Sie Variablen mit Datentypen deklarieren ist es sinnvoll dem Variablennamen ein Präfix voranzustellen, um das Lesen und nachvollziehen einfacher zu machen.

Zwar können Sie Variablen an beliebigen Stellen in der Prozedur deklarieren. Das sollte man allerdings nicht tun, da die Übersichtlichkeit des Programmcodes darunter leidet. Üblich ist es, am Anfang einer Prozedur die nötigen Variablen zu deklarieren.

2.3 Der Gültigkeitsbereich von Variablen

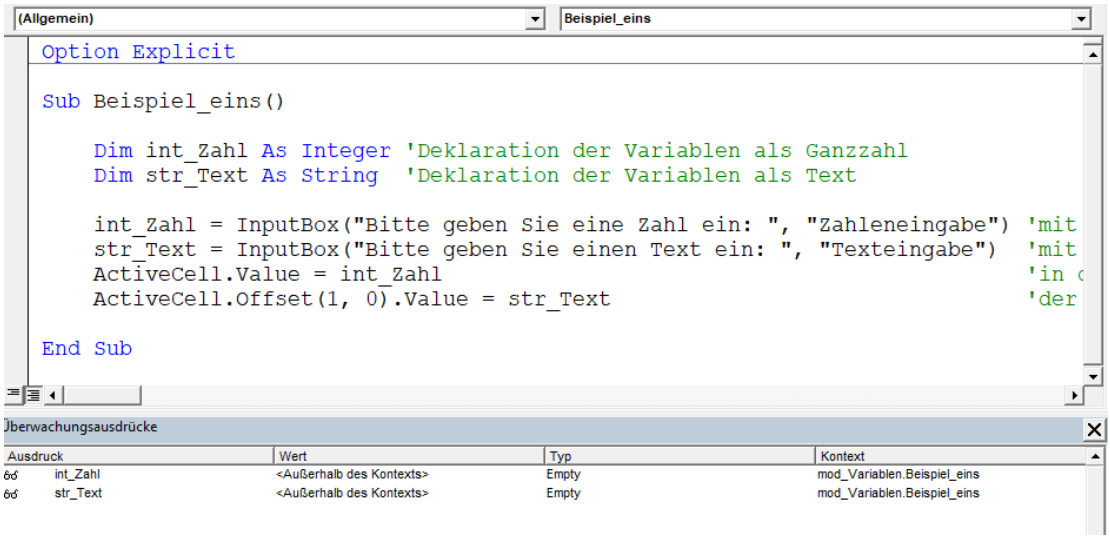
Variablen sind nur in dem Bereich gültig, in dem Sie deklariert sind. Wenn Sie einer Variablen einen Wert zuweisen, die am Anfang einer Prozedur deklariert wurde, gilt sie nur innerhalb der Prozedur.

Variablen werden beim Start einer Prozedur übrigens automatisch initialisiert. Wenn eine Variable als Zahl deklariert wird, wird sie auf den Anfangswert 0 gesetzt. Bei einer String-Variablen wird eine leere Zeichenkette ("") gesetzt.

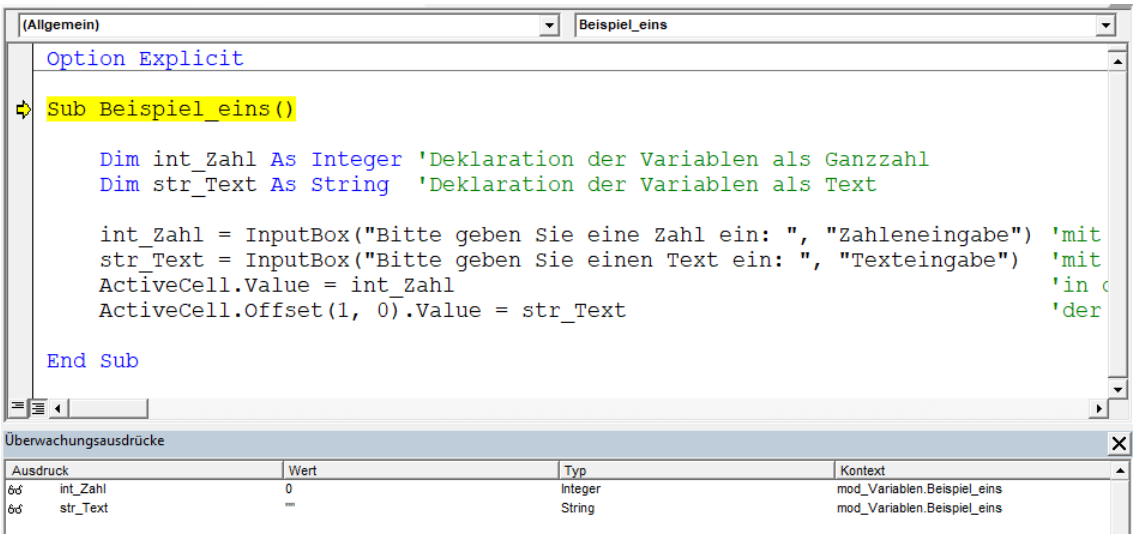
Geben Sie das folgende Beispiel ein, um den Gültigkeitsbereich von Variablen nachvollziehen zu können:

```
Option Explicit

Sub Beispiel_eins()

    Dim int_Zahl As Integer 'Deklaration der Variablen als Ganzzahl
    Dim str_Text As String  'Deklaration der Variablen als Text

    int_Zahl = InputBox("Bitte geben Sie eine Zahl ein: ", "Zahleneingabe") 'mit einer Inputbox wird eine Zahl eingelesen
    str_Text = InputBox("Bitte geben Sie einen Text ein: ", "Texteingabe")  'mit einer Inputbox wird ein Text eingelesen
    ActiveCell.Value = int_Zahl                                             'in die aktive Zelle wird der Wert der Zahlenvariable eingelesen
    ActiveCell.Offset(1, 0).Value = str_Text                                'der Wert der Textvariablen wird in die Zelle darunter eingelesen

End Sub
```

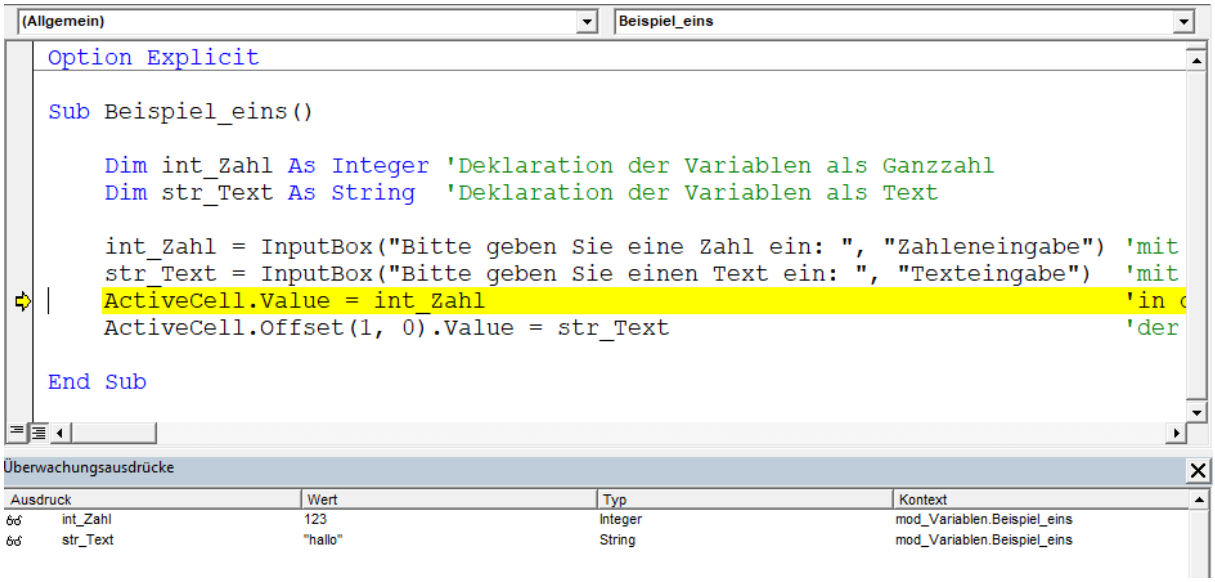
Um den Inhalt der Variablen zu prüfen, klicken Sie auf den Menüpunkt **Ansicht**. Wählen Sie dort den Eintrag **Überwachungsfenster**. Markieren Sie dann nacheinander die Variablennamen und ziehen Sie diese ins Überwachungsfenster.



Da die Prozedur noch nicht läuft, steht in der Spalte Wert: **Außerhalb des Kontextes**. Es gibt auch noch keine Variablen. Wenn Sie jedoch mit der Taste F8 die schrittweise Ausführung des Makros starten, wird die Zuweisung vorgenommen.



Sobald durch die Inputbox der Wert in die Variable eingelesen wurde, können Sie im Überwachungsfenster die eingegebenen Werte sehen.



Mit der Beendigung der Prozedur sind auch die Werte in den Variablen verschwunden, da der Gültigkeitsbereich verlassen wurde.

Variablen können auch auf der **Modul-Ebene** unterhalb der Anweisung Option Explicit deklariert werden. Die Variablen können dann von allen im Modul vorhandenen Prozeduren genutzt werden.

2.4 Variablen Werte zuweisen

Unter 2.3 Der Gültigkeitsbereich von Variablen haben Sie bereits gesehen, wie den Variablen mittels einer **Inputbox** Werte zugewiesen werden. Sie können den Variablen aber auch direkt Werte im Code aus Tabellenblättern zuweisen.

Zwei Beispiele dazu:

```
Sub Beispiel_zwei()

    Dim int_Zahl As Integer 'Deklaration der Variablen als Ganzzahl
    Dim str_Text As String  'Deklaration der Variablen als Text

    int_Zahl = 144
    str_Text = "Hallo"


End Sub
```

```
Sub Beispiel_drei()

    Dim int_Zahl As Integer 'Deklaration der Variablen als Ganzzahl
    Dim str_Text As String  'Deklaration der Variablen als Text

    int_Zahl = ActiveCell.Value
    str_Text = ActiveCell.Offset(1, 0).Value


End Sub
```

Wenn Sie mit Objektvariablen (Tabellenblätter, Zellen und Zellbereiche usw.) arbeiten wollen, müssen Sie die **Set-Anweisung** zum Zuweisen eines Objekts in eine Variable benutzen.

Dazu drei Beispiele:

```
Sub Beispiel_vier()

    Dim obj_Bereich As Range                    'Deklaration der Variablen als Range-Objekt

    Set obj_Bereich = ActiveSheet.Range("a1:c2") 'Zuweisung des Bereichs a1 bis c2 zu der Variablen
    obj_Bereich.Select                          'Markieren des Bereichs

End Sub
```

```
Sub Beispiel_fuenf()


    Dim obj_Bereich As Range                    'Deklaration der Variablen als Range-Objekt

    Set obj_Bereich = ActiveSheet.UsedRange     'Zuweisung des Bereichs benutzten Bereichs zu der Variablen

    obj_Bereich.Select                          'Markieren des Bereichs

End Sub
```

```
Sub Beispiel_sechs()
     Dim obj_Bereich As Range                   'Deklaration der Variablen als Range-Objekt

     Set obj_Bereich = Application.InputBox("Wählen Sie einen Zellbereich aus ", "Zellen auswählen", Type:=8)
     'Zuweisung des Bereichs durch Markieren mit der Inputbox

     obj_Bereich.Select                         'Markieren des Bereichs
End Sub
```

3. Kontrollstrukturen: Wenn-Dann-Sonst-Anweisungen und Schleifen

Zu den wesentlichen Kontrollstrukturen in der Programmierung gehören die Wenn-Dann-Sonst-Anweisungen und Schleifen, die wir Ihnen in den folgenden Abschnitten vorstellen.

Wollen Sie eine Aktion mehrfach ausführen, können Sie die Anweisungen mehrfach eingeben. Um das zu vermeiden, können Schleifen genutzt werden. Dabei muss zwischen zählergesteuerten Schleifen und bedingungsgesteuerten Schleifen unterschieden werden.

Lassen Sie uns aber mit der Wenn-Dann-Sonst-Anweisung beginnen.

3.1 Die Wenn-Dann-Sonst-Anweisung

Die Wenn-Dann-Sonst-Anweisung ist die einfachste Form einer Fallunterscheidung. Wie aus der Bezeichnung schon ableitbar, wird bei einer zutreffenden, wahren Bedingung (True) etwas gemacht. Wenn die Bedingung nicht zutrifft und falsch (False) ist, dann wird etwas Anderes gemacht.

Für die Bedingung selbst muss ein Vergleich mit einem gültigen Vergleichsoperator durchgeführt werden. Als Beispiel nutzen wir die Zahlen 5 und 3:

Operator	Bedeutung	Beispiel	Ergebnis
=	gleich	5 = 3	False
>	größer als	5 > 3	True
>=	größer oder gleich als	5 >= 3	True
<	kleiner als	5 < 3	False
<=	kleiner oder gleich als	5 <= 3	False
<>	ungleich (nicht)	5 <> 3	True

Ein Codebeispiel dazu:

```
Sub WennAnweisung()
'Deklaration zweier Integer-Variablen
Dim int_Zahl1 As Integer
Dim int_Zahl2 As Integer

  'Einlesen von den beiden Zahlen über InputBoxen
  int_Zahl1 = InputBox("Bitte die erste Zahl eingeben:", "Erste Zahl")
  int_Zahl2 = InputBox("Bitte die zweite Zahl eingeben:", "Erste Zahl")

  'Durchführen der Wenn-Dann-Sonst-Anweisung mit dem Vergleich,
  'ob die erste Zahl größer als die zweite Zahl ist und Ausgabe
  'entsprechender Meldungen.
  If int_Zahl1 > int_Zahl2 Then
    MsgBox "Die erste Zahl ist größer als die zweite Zahl.", , "Größer"
  Else
    MsgBox "Die erste Zahl ist NICHT größer als die zweite Zahl.", , "NICHT Größer"
  End If
End Sub
```

3.2 Die zählergesteuerte For … Next-Schleife

Bei einer zählergesteuerten Schleife geben Sie vor, wie oft die Anweisungen in der Schleife wiederholt werden sollen.

```
Sub Schleife01()
Dim int_Zaehler As Integer 'Laufvariable der zu wiederholenden Anweisungen
Dim int_Endwert As Integer 'Setzt den Endwert der Schleife

  int_Endwert = InputBox("Anzahl neuer Tabellenblätter", "Blätter einfügen")

  'Zuweisung in die Zählervariable
  'Setzt den Anfangswert 1 und geht die Schleife
  'solange durch bis der Endwert erreicht ist.
  For int_Zaehler = 1 To int_Endwert
    'Hinzufügen eines neuen Tabellenblatts an letzter Stelle
    Worksheets.Add after:=Worksheets(Worksheets.Count)
  Next int_Zaehler

End Sub
```

3.3 Die bedingungsgesteuerte Do Until / While … Loop-Schleife

Bedingungsgesteuerte Schleifen werden solange durchgeführt bis eine Ende-Bedingung (until = solange bis) erfüllt ist oder eine Anfangsbedingung nicht mehr (while = solange wie) erfüllt ist. Diese Bedingung wird ebenfalls über einen Vergleich formuliert, siehe 3.1 Die Wenn-Dann-Sonst-Anweisung.

Das folgende Beispiel zeigt eine Do Until-Schleife mit einer Ende-Bedingung.

```
Sub Schleife02()
Dim int_Antwort As Integer 'Variable für den Rückgabewert der Messagebox

  Do Until int_Antwort = vbNo 'vbYes = 6 / vbNo = 7
    'Einfügen eines neuen Tabellenblatts an letzter Stelle
    Worksheets.Add after:=Worksheets(Worksheets.Count)
    'Messagebox mit der Abfrage, ob ein weiteres Tabellenblatt
    'eingefügt werden soll.
    int_Antwort = MsgBox("Soll ein weiteres Tabellenblatt eingefügt werden?", _
                         vbYesNo, "Weiteres Tabellenblatt?")
  Loop

End Sub
```

Als Anfangsbedingung könnte der Code wie folgt geändert werden:

```
Sub Schleife03()
Dim int_Antwort As Integer 'Variable für den Rückgabewert der Messagebox

  Do While int_Antwort <> vbNo 'int_Antwort ist bei der Initialisierung 0.
    'Einfügen eines neuen Tabellenblatts an letzter Stelle
    Worksheets.Add after:=Worksheets(Worksheets.Count)
    'Messagebox mit der Abfrage, ob ein weiteres Tabellenblatt
    'eingefügt werden soll.
    int_Antwort = MsgBox("Soll ein weiteres Tabellenblatt eingefügt werden?", _
                         vbYesNo, "Weiteres Tabellenblatt?")
  Loop

End Sub
```

Neben der Prüfung am Anfang der Schleife **Do Until/While** (kopfgesteuert) kann die Prüfung der Bedingung auch am Ende der Schleife **Loop Until/While** (fußgesteuert) erfolgen.

3.4 Die For Each-Schleife für Auflistungsobjekte

Diese Schleife können Sie mit Auflistungsobjekten verwenden. Der Vorteil darin besteht, dass alle Objekte im Auflistungsobjekt abgearbeitet werden, ohne diese durch getrennte Anweisungen einzeln ansprechen zu müssen.

Im folgenden Beispiel werden alle Tabellenblätter der aktiven Arbeitsmappe durchlaufen und umbenannt in Monat gefolgt von einem Zähler.

```
Sub Schleife04()
'Deklaration der Variablen für das Auflistungsobjekt aller
'Arbeitsblätter
Dim obj_ArbeitsblaetterAlle As Sheets
'Deklaration der Objektvariablen für ein Arbeitsblatt
Dim obj_ArbeitsblattEins As Worksheet
'Deklaration einer Zähler-Variablen
Dim int_Zaehler As Integer

  int_Zaehler = 1

  Set obj_ArbeitsblaetterAlle = ActiveWorkbook.Worksheets

  For Each obj_ArbeitsblattEins In obj_ArbeitsblaetterAlle
    obj_ArbeitsblattEins.Name = "Monat " & int_Zaehler
    int_Zaehler = int_Zaehler + 1
  Next

End Sub
```

Sie merken, dass die Auflistungsobjekte in der Regel immer ein Mehrzahl-S am Ende des Objektbezeichners haben, z.B. Sheet**s**. Doch auch davon gibt es Ausnahmen.

Die wichtigste Ausnahme ist das Range-(Auflistungs)Objekt. Dieses stellt einen markierten Bereich in einem Arbeitsblatt dar. Aufgelistet werden in diesem Objekt alle Zellen, die markiert sind. Um die Ausnahme an dieser Stelle auf die Spitze zu treiben, ist eine einzelne Zelle wiederum ein Range-Objekt. Dazu ein Beispiel, welches für den markierten Bereich nacheinander alle Adressen der markierten Zellen in einer MessageBox ausgibt:

```
Sub Schleife05()
Dim obj_Bereich As Range
Dim obj_Zelle As Range

  'Der aktive markierte Bereich steckt im Range-Objekt Selection
  Set obj_Bereich = Selection

  For Each obj_Zelle In obj_Bereich
    MsgBox "Adresse der Zelle: " & obj_Zelle.Address
  Next

End Sub
```

4. Dialoge programmieren (Formulare)

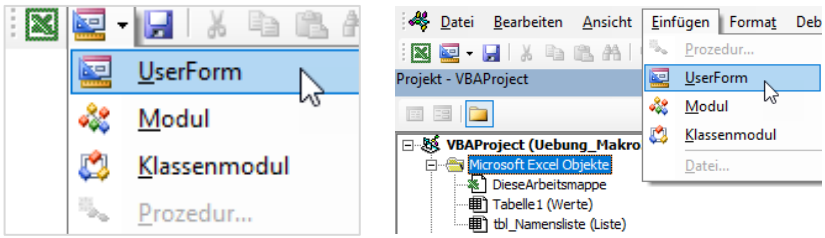
Nehmen wir an, Sie wollen ein Makro erstellen, welches eine bestimmte Anzahl an neuen Tabellenblättern mit einem beliebigen Präfix und einer laufenden Nummer als Namen erstellt, z.B. 5 Tabellenblätter mit den Namen Projekt 1 bis Projekt 5.

Mit dem bisherigen Wissen können Sie für die Eingabe der Anzahl und des Präfixes zwei InputBoxen programmieren. Anstelle der InputBoxen können Sie aber auch ein UserForm nutzen, auf der Sie beide Angaben in einem Arbeitsschritt machen.

4.1 Formular (UserForm) erzeugen

Um eine UserForm zu erstellen, gehen Sie wie folgt vor:

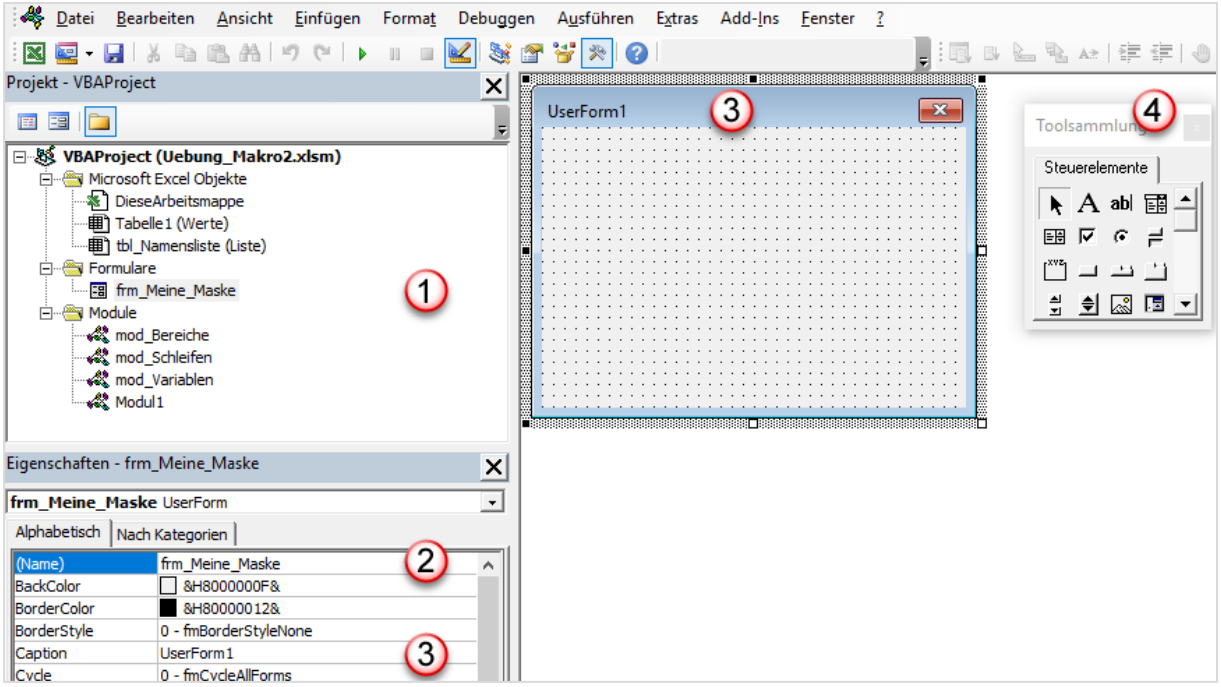
- Klicken Sie im **VBE** auf den Listenpfeil das **zweite Symbol** von links.
- Wählen Sie den Eintrag **UserForm**. Alternativ dazu können Sie in der Menüleiste auf den Befehl **Einfügen** klicken und dort ebenfalls den Eintrag **UserForm** auswählen.



Auch hier müssen Sie, wie bei der Modul-Erzeugung, darauf achten, dass im Projektexplorer die richtige Arbeitsmappe markiert ist, in die das UserForm eingefügt werden soll.

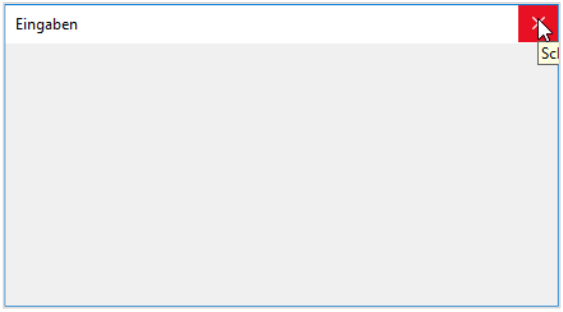
4.2 Elemente der UserForm

Im Projektexplorer des VBEs sehen Sie nun in der Struktur den Ordner Formulare, der die eben erzeugte UserForm enthält.



1. Die neue UserForm heißt eigentlich **Userform1**. Fügen Sie weitere Userformen ein, heißen diese erstmal Userform2, Userform3 usw.
2. Sie sollten der UserForm einen Namen geben. Dazu klicken Sie in das Eigenschaften Fenster und tragen dort den Namen unter **(Name)** ein. Dabei sollten Sie auch hier ein Präfix, z.B. frm für Formular, benutzen.
3. Im Eigenschaftenfenster taucht im Feld **Caption** nochmal die Bezeichnung **Userform1** auf. Hierbei handelt es sich um den Anzeigenamen, den Sie im Formularfenster sehen. Diesen Namen können Sie überschreiben, z.B. in Eingaben.
4. Die UserForm hat eine bestimmte Größe und ist am Rand mit den Markierungsklötzchen markiert. Wenn Sie auf ein Markierungsklötzchen zeigen und ziehen, können Sie die UserForm vergrößern oder verkleinern. Sobald die UserForm markiert ist, erscheint auch das Fenster **Toolsammlung**. Die Toolsammlung enthält diverse **Steuerelemente**, die Sie in dem Formular benutzen können.

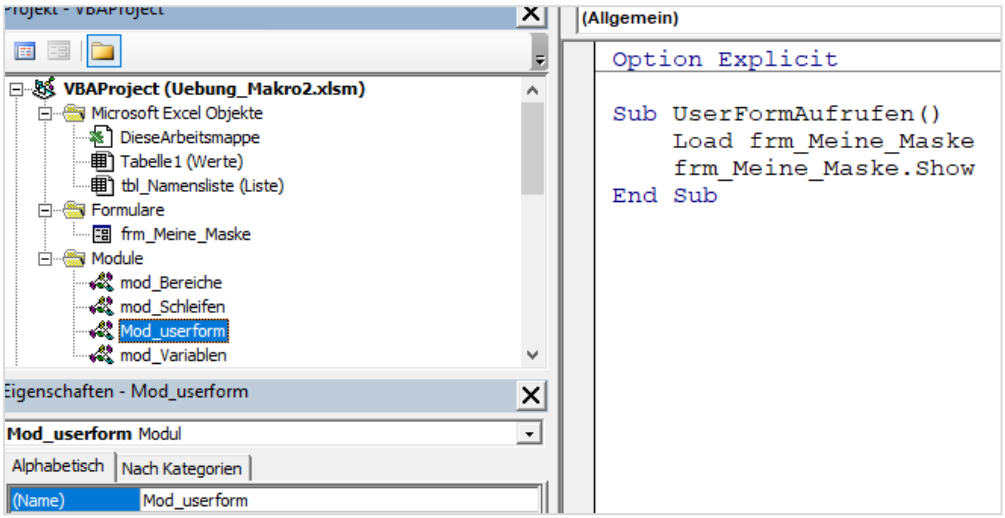
Wenn Sie nun die markierte UserForm z. B. mit der Taste F5 ablaufen lassen, sieht das in Excel wie folgt aus:



Die UserForm wird als leeres Formular angezeigt, noch sind keine Steuerelemente hinzugefügt. Es lässt sich über die Schließen-Schaltfläche, die im Standard immer enthalten ist, wieder schließen.

4.3 UserForm über eine Schaltfläche aufrufen

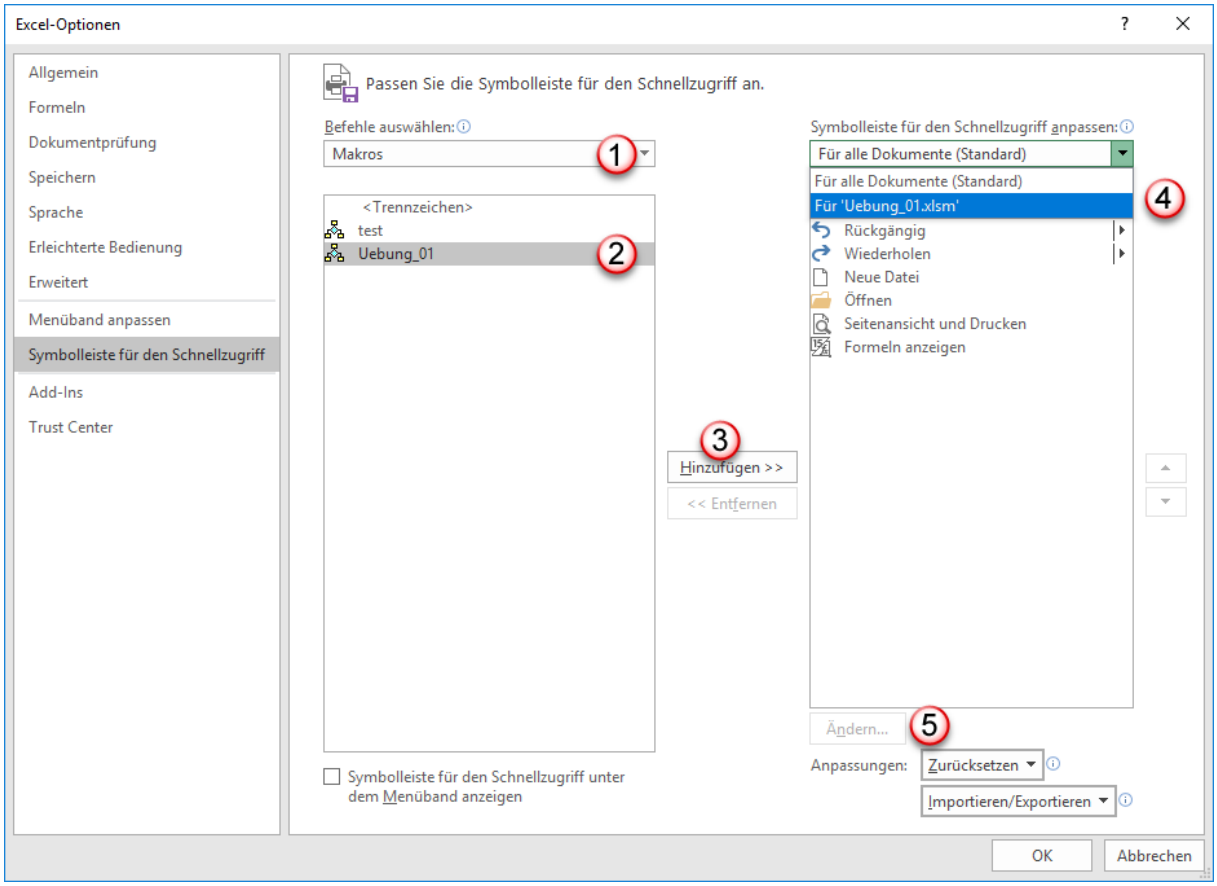
Damit Sie die UserForm aufrufen können, müssen Sie den Aufruf in einer Prozedur eines Moduls erzeugen. Es ist am besten, wenn Sie ein neues Modul erzeugen, es gleich umbenennen und den Programmtext eingeben.



Die Anzeige eines Formulars erfolgt immer in zwei Schritten. Ein Formular muss erst geladen (**Load**)werden. Anschließend kann es mit der Methode **Show** eingeblendet werden.

Wenn Sie diese Prozedur erzeugt haben, können Sie sie mit einer Schaltfläche oder einem Symbol in der **Symbolleiste für den Schnellzugriff** starten. Das funktioniert wie folgt:

Klicken Sie in der **Symbolleiste für den Schnellzugriff** ganz rechts auf den Listenpfeil und wählen dort den Eintrag **Weitere Befehle** aus. Sie gelangen in das Dialogfeld **Excel-Optionen**.



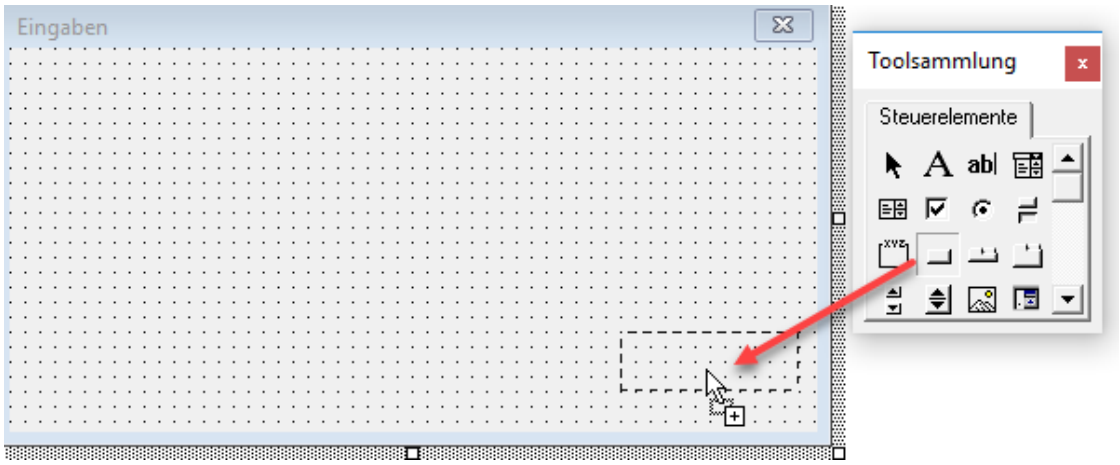
1. Wählen Sie aus dem Listenfeld den Befehl **Makros** aus.
2. Aus der Liste der vorhandenen Makros wählen Sie das gewünschte Makro aus
3. Klicken Sie auf die Schaltfläche **Hinzufügen**, so dass das Makro in der rechten Fensterhälfte erscheint.
4. Im Listenfeld darüber können Sie auswählen, ob das Makro immer in der Symbolleiste angezeigt werden soll oder nur dann, wenn die Datei, die das Makro enthält, geöffnet ist.
5. Sobald das Makro in der Symbolleiste eingefügt ist, wird die Schaltfläche **Ändern** aktiv. Sie können nun dem Makro ein anderes Symbol und ein Quickinfo zuweisen.

4.4 UserForm mit Steuerelementen ausstatten

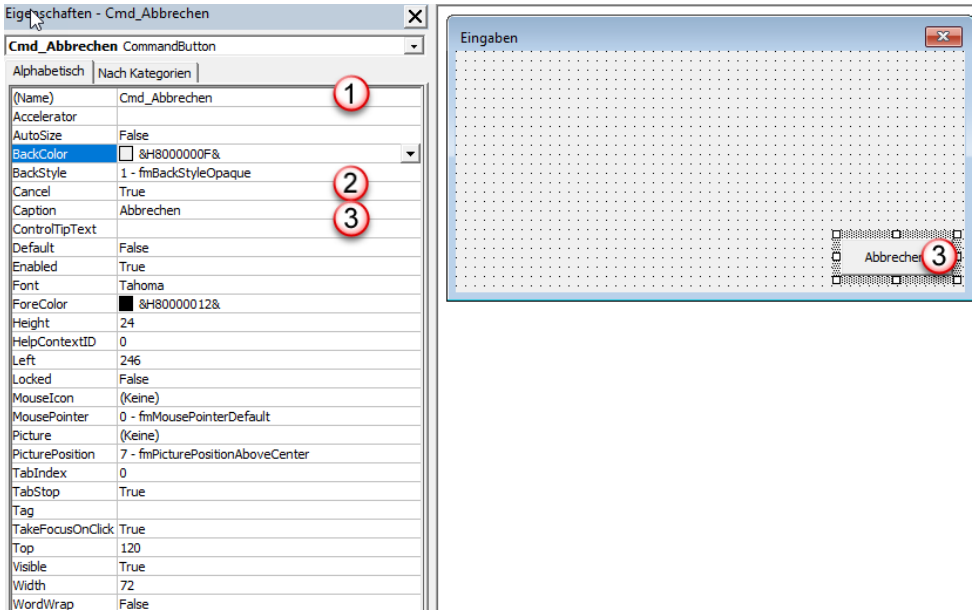
Die UserForm muss mit Steuerelementen ausgestattet werden, damit das Formular vernünftig genutzt werden kann. Zunächst sollen die Schaltflächen **OK** und **Abbrechen** erzeugt werden.

4.4.1 CommandButton (Befehlsschaltflächen) einfügen und bearbeiten

Ziehen Sie das Steuerelement **CommandButton** aus der Toolsammlung auf die Formularfläche.



Sobald Sie die Maustaste loslassen, erscheint die Schaltfläche auf dem Formular

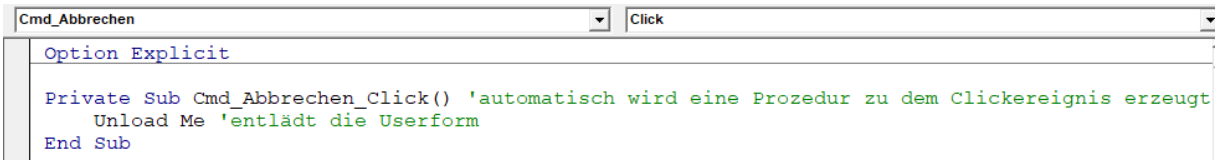


Damit Sie die Übersicht über die Schaltflächen und deren Programmierung behalten, sollten Sie die Schaltflächen entsprechend ihrer Aufgabe umbenennen.

1. Diese Schaltfläche ist ein CommandButton und sollte deshalb mit Präfix **cmd** und dem, was dieser Button macht, in der Eigenschaft **(Name)** beschriftet werden.
2. Damit man auch die **ESC**-Taste zum Abbrechen benutzen kann, sollten Sie auf die Eigenschaft **Cancel** doppelklicken, um sie auf True zu schalten.
3. Im Feld **Caption** tragen Sie die Beschriftung, die auf der Schaltfläche stehen soll, ein. In unserem Beispiel den Text Abbrechen.

Zu guter Letzt müssen Sie die Schaltfläche noch programmieren, damit sie die UserForm schließt, wenn man draufklickt.

Doppelklicken Sie auf die Schaltfläche Abbrechen und Sie gelangen in die Code-Ansicht für diese Schaltfläche.



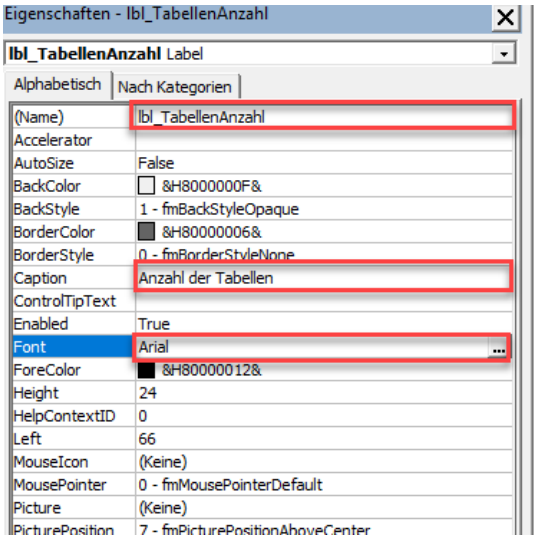
Die Schlüsselworte (Private Sub und End Sub), der Schaltflächenname und das Standardereignis Click sind bereits eingestellt. Sie brauchen nur noch die Zeilen zu schreiben, was passieren soll, wenn die Schaltfläche geklickt wird.

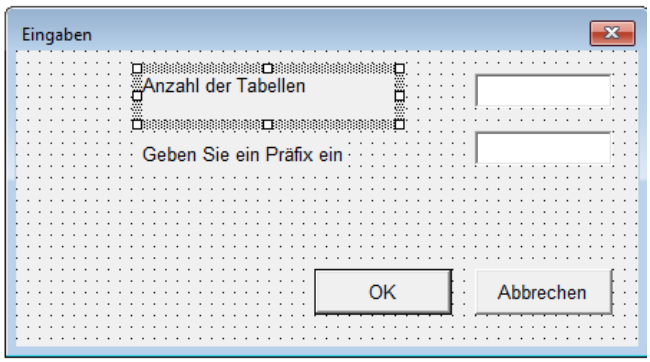
So wie für die Anzeige ein Formular geladen werden muss, muss es für ein vollständiges Schließen wieder entladen werden. Der Befehl lautet **Unload Me**. Me als Schlüsselwort ist dabei als indirekt deklarierte Variable zu betrachten, die sich auf das Formular bezieht.

Auf die gleiche Art produzieren Sie eine **OK**-Schaltfläche, die aber noch nicht programmiert werden kann. Zunächst müssen Eingabefelder erzeugt werden.

4.4.2 TextBoxen (Textfelder) und Label (Beschriftungsfelder)

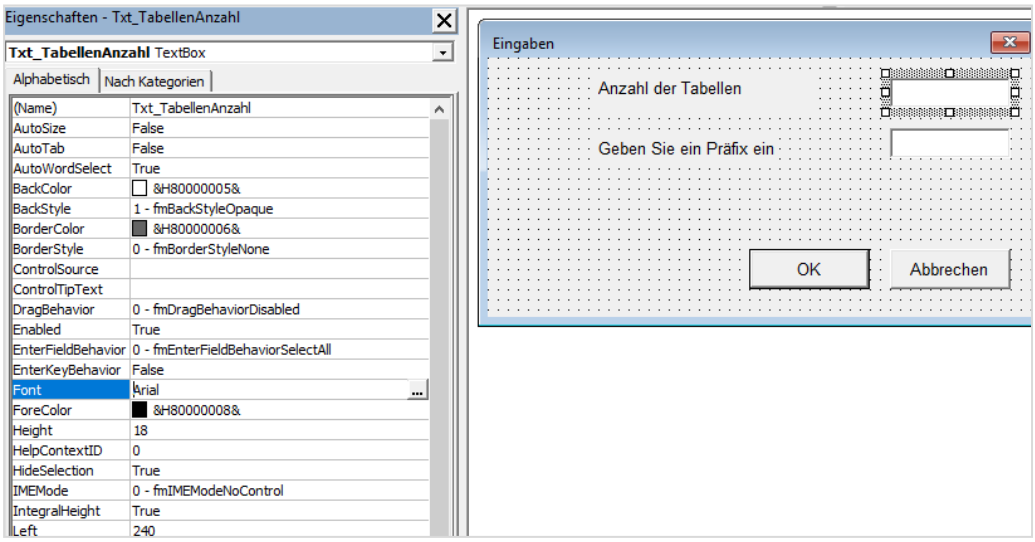
Um Eingaben in das Formular machen zu können, müssen Sie Textfelder benutzen und Beschriftungen, die den Text anzeigen, was eingegeben werden soll.





Wenn Sie mehrere sogenannte Labels, also Beschriftungsfelder, nutzen wollen, ist es sinnvoll, ein Feld zu gestalten in Schriftart, Schriftgrad und sonstigen Eigenschaften und die weiteren Labels zu kopieren.

Zu dem Beschriftungsfeld sollte eine TextBox hinzukommen, die die Eingaben aufnehmen kann. Auch hier gestalten Sie zunächst und kopieren dann, wenn eine weitere TextBox notwendig ist. Die TextBox kann nicht nur Text, sondern beliebige Zeichenfolgen - auch Zahlen oder ein Datum - aufnehmen. Allerdings müssen Sie ggf. prüfen, ob der Datentyp korrekt ist oder diesen ggf. umwandeln.



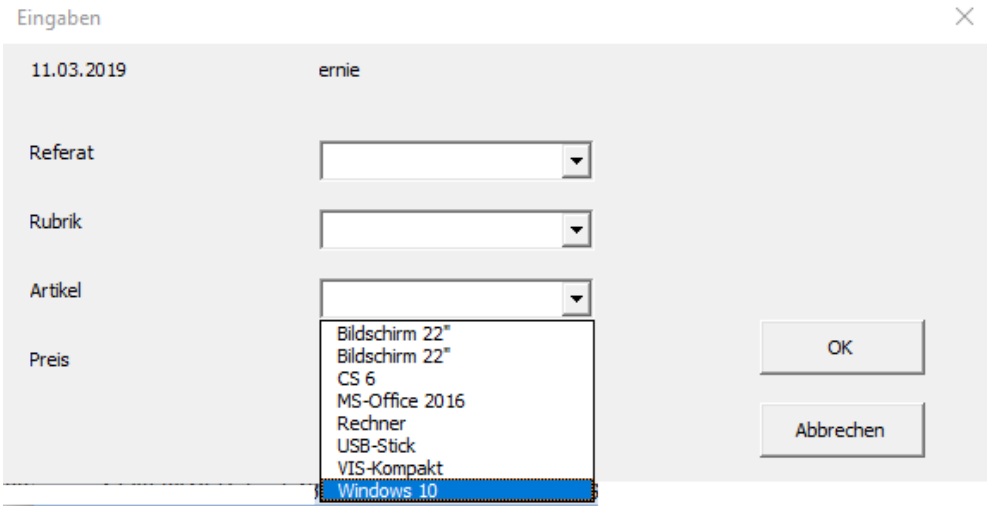
Wenn die Felder befüllt wurden, muss hinter der Schaltfläche **OK** hinterlegt werden, was mit den Eingaben passieren soll. Doppelklicken Sie auf die Schaltfläche Ok, um das Grundgerüst des Codes für das Standardereignis Click zu erhalten. Sie gelangen in das Codefenster und können hier den Programmcode eingeben.

```
Private Sub cdm_OK_Click()
Dim int_Zaehler As Integer 'Laufvariable der zu wiederholenden Anweisungen
Dim int_Endwert As Integer 'Setzt den Endwert der Schleife
Dim str_Praefix As String 'Variable für den Präfis der Blattnamen
Dim obj_ArbeitsblattNeu As Worksheet 'Objektvariable für neue Arbeitsblätter

On Error GoTo Fehler

  int_Endwert = Me.txt_TabellenAnzahl.Value
  str_Praefix = Me.txt_TabellenPraefix.Value

  'In einer For-Schleife den Code abarbeiten
  For int_Zaehler = 1 To int_Endwert
    'Hinzufügen eines neuen Tabellenblatts an letzter Stelle und
    'Umbenennen mit dem Präfix und der lfd. Nummer als neuer Name
    Set obj_ArbeitsblattNeu = Worksheets.Add(, _
                              Worksheets(ActiveWorkbook.Worksheets.Count))
    obj_ArbeitsblattNeu.Name = str_Praefix & " " & int_Zaehler
  Next int_Zaehler

  Unload Me
  Exit Sub

Fehler:

  MsgBox "Ein Fehler ist aufgetreten. Entweder ist bereits eine " & _
         "Tabelle mit dem Namen vorhanden oder die Eingaben zur " & _
         "Anzahl oder dem Präfix der Tabellen sind ungültig.", _
         vbOKOnly, "Fehler bei der Ausführung"

End Sub
```

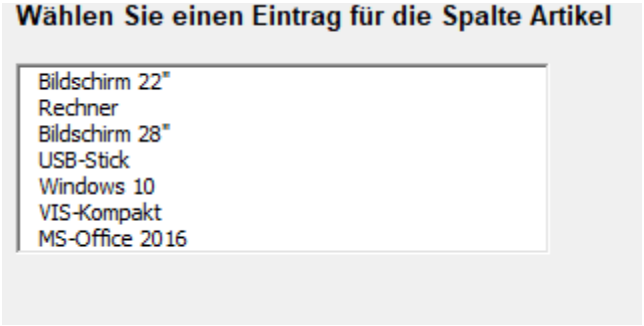
4.4.3 Eingabeauswahl durch Listboxen oder Comboboxen

Wenn Sie keine freien Eingaben machen wollen, können Sie aus einer ListBox (Listenfeld) oder einer ComboBox (Kombinationsfeld) auswählen.

In der folgenden Abbildung sehen Sie ein Kombinationsfeld für die Eingabe.



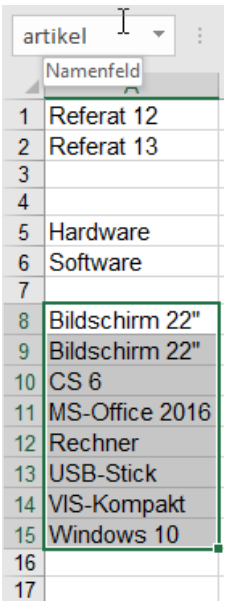
In der folgenden Abbildung sehen Sie die gleiche Liste in einem Listenfeld. Sie können sich alle Einträge zugleich anschauen. Wenn nicht alle Einträge in die Liste passen, erscheint am Rand eine Bildlaufleiste.

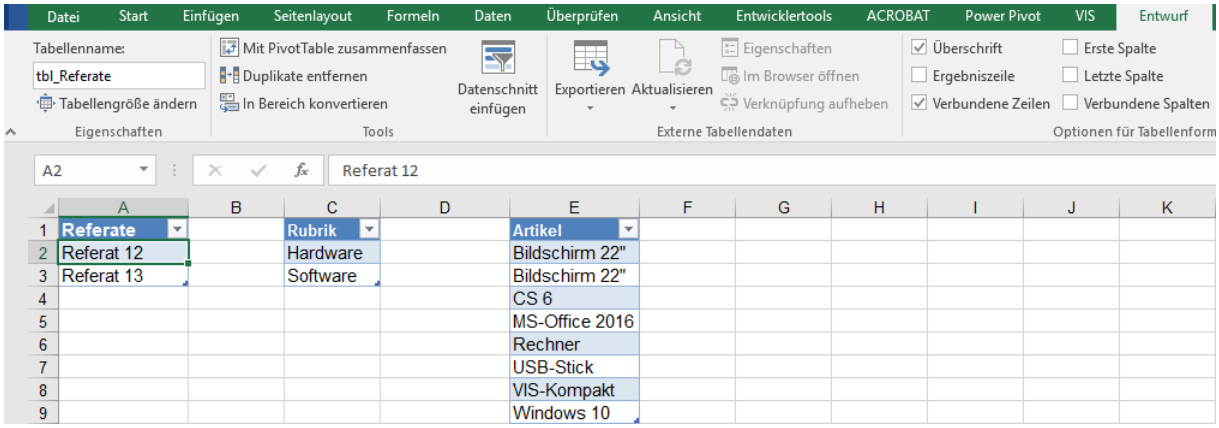


Für beide Objekte gilt, dass Sie sie mit den Elementen befüllen müssen. Dazu bietet es sich an, in einem separaten Tabellenblatt die notwendigen Einträge zu erzeugen und die Bereiche mit Namen zu versehen oder diese als Tabelle zu definieren.

In der Abbildung sehen Sie, dass z. B. dem Bereich A8 bis A15 der Name **Artikel** zugewiesen wurde. Wenn Sie Namen benutzen können Sie Zeilen dazwischen fügen und die neuen Zeilen werden in der List- bzw. Combobox angezeigt.

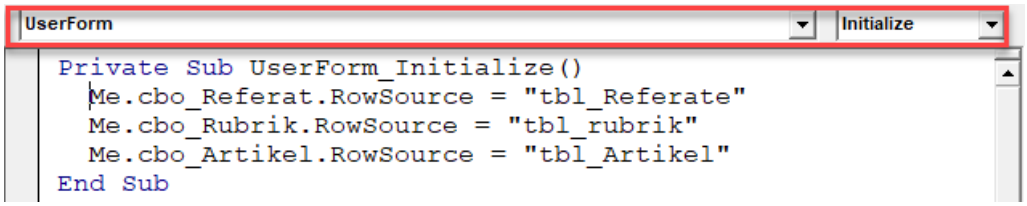
Wenn Sie die Bereiche als Tabellen definieren, können Sie die neuen Einträge unten anfügen.





Um die Einträge der List- bzw. der Combobox zuzuweisen, können Sie das über das Eigenschaftenfeld **RowSource** tun oder beim Initialisieren der UserForm.

- Klicken Sie doppelt auf die **UserForm** oder wechseln in das Codefenster und wählen dort im Objekt-Kombinationsfelds **UserForm** aus.
- wählen Sie im Kombinationsfeld für die Prozedur das Ereignis **Initialize** aus.

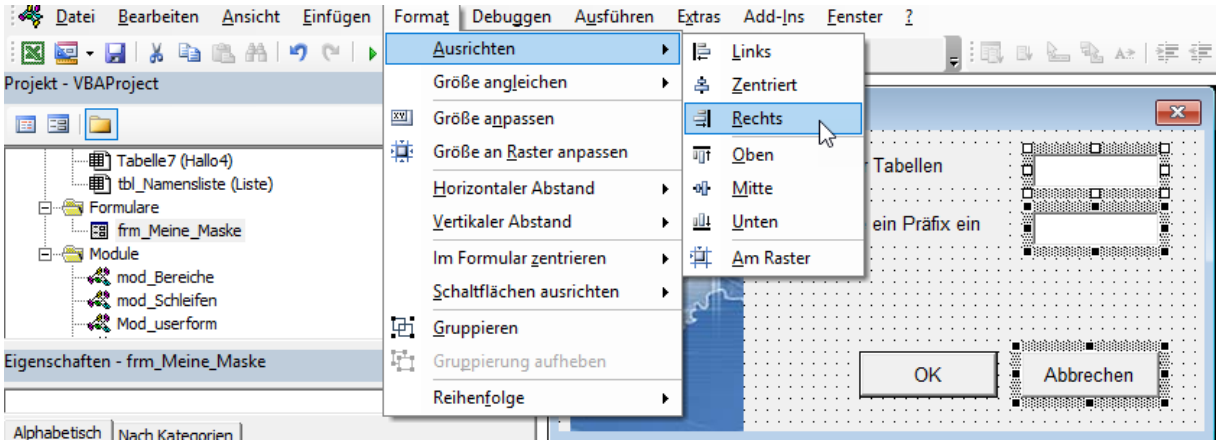


- der aktuellen UserForm (Me) und dort dem Objekt cbo_Referat werden die Einträge aus der Tabelle tbl_Referate zugeordnet, ebenso den Kombinationsfeldern für die Rubrik und die Artikel.

4.5 Optische Gestaltung des Formulars durch Ausrichten der Elemente

Aus Gründen der Arbeitsergonomie und der Barrierefreiheit sollen alle Steuerelemente bündig und Fluchten ausgerichtet sein. Gehen Sie dazu wie folgt vor:

- Klicken Sie die erste Schaltfläche an, halten die Shift-Taste gedrückt und klicken die weiteren Schaltflächen an, die Sie ausrichten wollen.
- Klicken Sie in der Menüleiste auf **Format** und dort auf den Eintrag **Ausrichten**.
- Sie können nun die markierten Objekte in einer Fluchtlinie anordnen.



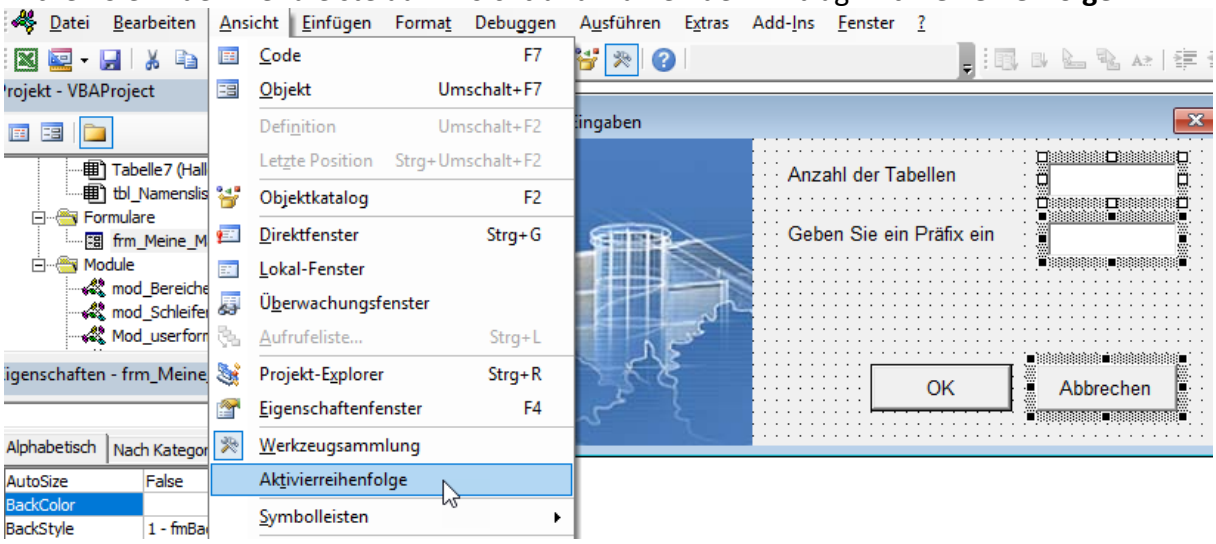
4.6 Die Aktivierungsreihenfolge der Steuerelemente

Zur Barrierefreiheit gehört auch, dass Formulare mit der Tastatur bedienbar sein müssen. Die einfachste Form der Bedienung über die Tastatur ist dabei das Durchwandern aller Steuerelemente mit der Tabulator-Taste. Entscheidend dabei ist, dass die Steuerelemente in einer sachlich logischen Reihenfolge aktiviert werden. Das ist oft die Reihenfolge der Elemente von oben nach unten, muss es aber nicht unbedingt.

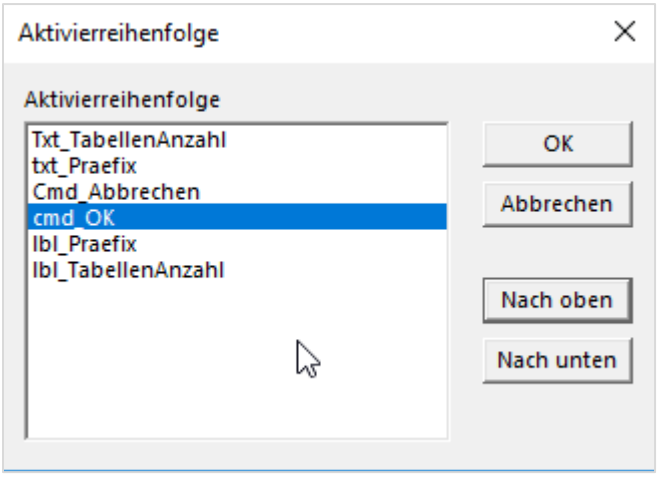
Die Aktivierungsreihenfolge der Steuerelemente in einem Formular kann unabhängig der Platzierung auf dem Formular erfolgen. Jedes Steuerelement hat dazu die Eigenschaft **TabIndex**. Der TabIndex ist eine aufsteigende Zahlenreihe beginnend mit 0 (Null), die die Reihenfolge der Aktivierung der Elemente bestimmt.

Der VB-Editor bietet aber auch einen Dialog zum Definieren der Aktivierungsreihenfolge:

- Klicken Sie in der Menüleiste auf **Ansicht** und wählen den Eintrag **Aktivierreihenfolge**



- Das Dialogfenster **Aktivierreihenfolge** öffnet sich



- Sie können nun die einzelnen Steuerelemente markieren und
- mit den Schaltflächen **Nach oben** oder **Nach unten** in die entsprechende Position schieben.

Da hier die Steuerelemente mit ihren Namen angezeigt werden, sehen Sie auch, wie wichtig es ist, die Schaltflächen aussagekräftig benannt zu haben.

Platz für Ihre Notizen

Platz für Ihre Notizen

Lernmaterial, Beratung und Kontakt

Auf der Internetseite

http://www.afz.bremen.de/lernen

stellt das AFZ Ihnen Kursunterlagen zu den IT-Kursen in elektronischer Form zur Verfügung. Diese werden regelmäßig aktualisiert und an neue Programmversionen angepasst. Das bietet Ihnen die Möglichkeit, jederzeit Kursthemen zu wiederholen und Ihre Kenntnisse zu aktualisieren.

Bei unseren Kursunterlagen handelt es sich um PDF-Dokumente, die Sie am Bildschirm lesen können. Die Dateien sind barrierefrei und können nach Stichworten durchsucht (strg + F) werden. Das Inhaltsverzeichnis und Links sind dynamisch verwendbar. Sie können die Dateien auf Ihrem Rechner speichern und bei Bedarf ausdrucken.

Auskünfte und Beratung

Sollten Sie als Beschäftigte der Freien Hansestadt Bremen bei Ihrer Arbeit auf Probleme stoßen, die beim Einsatz Ihrer Softwareausstattung auftreten (Probleme mit Word-Dokumenten, Excel-Tabellen etc.), können Sie sich mit Ihren Fragen, Problemstellungen oder Fehlermeldungen telefonisch oder per E-Mail an uns wenden:

it-fortbildung@afz.bremen.de **Tel. 361-16 999**

Beschreiben Sie Ihre Frage bzw. die Fehlersituation und Ihre bisherige Vorgehensweise und fügen Sie die Dateien im Original-Dateiformat als Anlage bei. Wir beantworten Ihre Fragen so schnell wie möglich, in jedem Fall melden wir uns innerhalb weniger Tage bei Ihnen.

Kontakt

Wir sind sehr an Ihren Anregungen und Verbesserungsvorschlägen zu unseren Kursangeboten, zu den Lernmaterialien und Ihrer Meinung zu unseren E-Learning-Kursen interessiert. Bitte nutzen Sie das

Kontaktformular

auf unserer Internetseite oder senden Sie eine Nachricht an **it-fortbildung@afz.bremen.de**.

Version vom November 2021

Impressum

Redaktion und Koordination

Referat 20 – Informationstechnologie – Qualifizierung und Beratung
Aus- und Fortbildungszentrum
Doventorscontrescarpe 172C

28195 Bremen

Telefon: +49 421 361-16999
E-Mail: it-fortbildung@afz.bremen.de

Herausgeber

Aus- und Fortbildungszentrum
für den bremischen öffentlichen Dienst
Doventorscontrescarpe 172C

28195 Bremen