

Eingabehilfe: Eingabeformular

Inhalt

1. Anlage einer Tabelle ("Sammeltabelle")	2
2. Entwurf des Formulars	3
3. Verbesserungen am Eingabeformular	8
4. Übernahme der Summenwerte einzelner Ausgabearten aus der Sammliste in die Gesamtübersicht	10
5. Modul zum Formularaufruf	12
6. Verbesserter Aufruf	13

Aufgabe

Anzulegen ist eine Übersicht persönlicher Ausgaben, die etwa wie folgt aussehen könnte:

Monat	Ausgabeart 1		Ausgabeart n
1			
...			
12			
Jahressumme			

Erwünscht ist, eine solche Ausgabenübersicht bequem bedienbar zu machen. Das verlangt den Aufbau einer Eingabehilfe (eines Eingabe-Formulars). Es sollen die Einzelausgaben monatsweise summiert und in eine Ausgabenübersicht eingetragen werden, welche die jeweiligen Monatssummen sammelt.

Anmerkung:

Ja, ich weiß, dass man sich so eine Excel-Anwendung bequem beschaffen kann. Alles was hier folgt, ist für Leute gedacht, die bereit sind, auch unterhalb des Nobelpreis-Niveaus herumzuwerkeln und zu Lernzwecken selbst solche Probleme anzupacken, die die Menschheit schon mal gelöst hat.

1. Anlage einer Tabelle ("Sammeltabelle")

Diese soll die Einzelausgaben eines Monats sammeln. Sie kann außerhalb des sichtbaren Bereiches einer Tabelle angelegt werden. Die Zahl möglicher Einträge im Monat bestimmt die Zahl zu reservierender Zeilen.

Ausgabeart 1	Ausgabeart 2		Ausgabeart n
...
SUMME(Ausgabeart 1)		SUMME(Ausgabeart n)

Die Summenberechnungen in der letzten Zeile sind über alle Zahlenfelder der jeweiligen Spalte zu erstrecken. Wie man das macht? Müsste inzwischen klar sein:

The image shows the 'SUMME' (SUM) dialog box in Microsoft Excel. The formula bar at the top displays '=SUMME(B13:B22)'. The dialog box has a 'Zahl1' field containing 'B13:B22' and a 'Zahl2' field which is empty. Below these fields, it shows '= 0'. The text inside the dialog reads: 'Summiert die Argumente. Zahl1: Zahl1; Zahl2; ... sind 1 bis 30 Argumente, deren Summe Sie berechnen möchten.' At the bottom of the dialog, there is a 'Formelergbnis = 0' and buttons for 'Ende' and 'Abbrechen'. Below the dialog, a portion of an Excel spreadsheet is visible. The columns are labeled 'Miete', 'Ernährung', 'Auto', 'Bücher', and 'Katze'. Row 23 is highlighted, with 'Summe' in cell A23 and '(B13:B22)' in cell B23. A dashed box highlights the range B13:B22 in the spreadsheet.

Die Syntax der Summenberechnung ist erkennbar. Man kann, muss aber nicht diesen Text eintragen. Es genügt, im Eingabefeld nach dem "=" die "SUMME" zu klicken und alle beteiligten Zellen mit der Maus auszuwählen, wie im Bild erkennbar.

2. Entwurf des Formulars

Der Entwurf wird innerhalb der IDE (Integrated Development Environment = integrierte Entwicklungsumgebung) von vba (= Visual Basics for Applications) gestaltet. Diese wird über die Menüleiste <Extras>, <Makro>, <Visual Basic-Editor> erreicht. Erforderlich sind ein Modul und ein Formular.

Modul:

Container für eventuell erforderliche Variablendeklarationen und für den Aufruf des Formulars. Wird erst später gebraucht.

Formular:

das Bedien- und Eingabefenster selbst, sowie der Programmcode, der "Ereignisse" der auf dem Formular angebrachten "Objekte" auswertet.

2.1 Einfügen des Formulars:

Über die Menüleiste wird aufgerufen <Einfügen>, <UserForm>. Es erscheinen ein Leerformular und die Toolbox (= Werkzeugkasten).

2.2 Gestaltung des Formulars, einfachste Fassung

Auf dem Formular werden benötigt (nur ein Vorschlag!):



Ein Kombinationsfeld (ComboBox) für die Anzeige und Auswahl der Ausgabeart,



ein Textfeld (TextBox) zum Eintippen der Ausgabesumme,

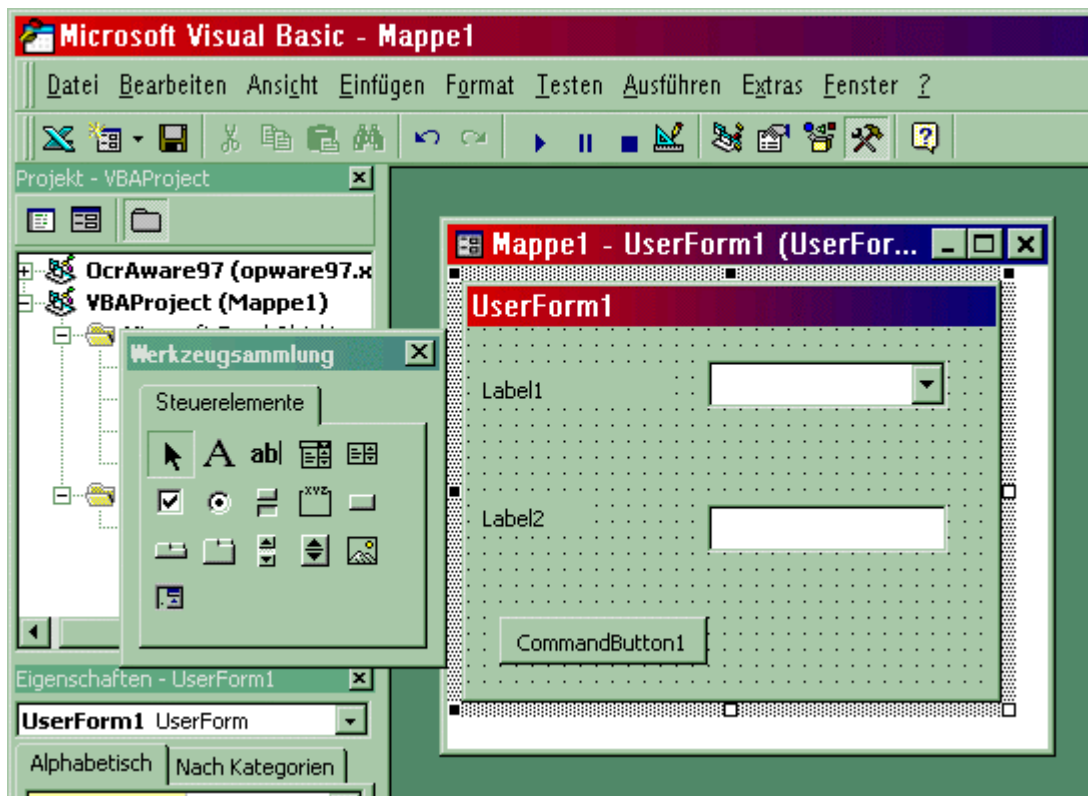


Bezeichnungsfelder (Labels) zur Beschriftung der beiden Boxen, nach dem Einfügen können sie sogleich einsatzfertig gemacht werden; dazu im Eigenschaftfenster (im Bild unten in der linken unteren Ecke zu erkennen) in der Zeile "Caption" den anzuzeigenden Text eintragen. Unter "Font" lässt er sich auch gleich "fein machen".



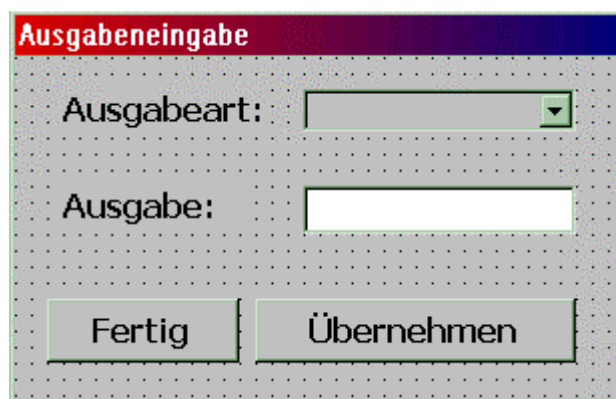
Befehlsschaltflächen (CommandButtons), die die Eingabewerte übernehmen bzw. die Eingabe beenden. Damit sie ihre Funktion verraten, erhalten sie, wie die Labels, auch gleich eine "Caption". (Achtung, nicht "(Name)" verändern, der bleibe ruhig, wie er ist!).

Es gibt viele andere Möglichkeiten, die Eingaben zu realisieren, bei der Gestaltung des Formulars je nach Geschmack sollte beachtet werden, dass die Bedienung (Nutzung) dadurch erleichtert und nicht erschwert werden sollte.



So etwa könnte das Formular beim Einfügen der Steuerelemente aussehen.

Es fehlt noch: Ein zweiter Button. Einer für die Eingabe, einer für Beendigung der Eingabe. Außerdem sieht alles noch etwas unfertig aus. Im Eigenschaftfenster kann man auch später für das Fenster, die Labels und die Buttons (jeweils nach Markierung des gewünschten Elements) das Verhalten, das Aussehen (etwa der Ränder, ("Border..."), die Farbe ("BackColor") und die Auf- bzw. Überschriften ("Caption") verändern. Das Formular hatte ja noch keine sinnvolle Überschrift. Alles kann ruhig etwas traurig aussehen, denn das Geld, das hier eingetragen wird, ist ja weg!



2.3 Programmcode zu den Bedienelementen:

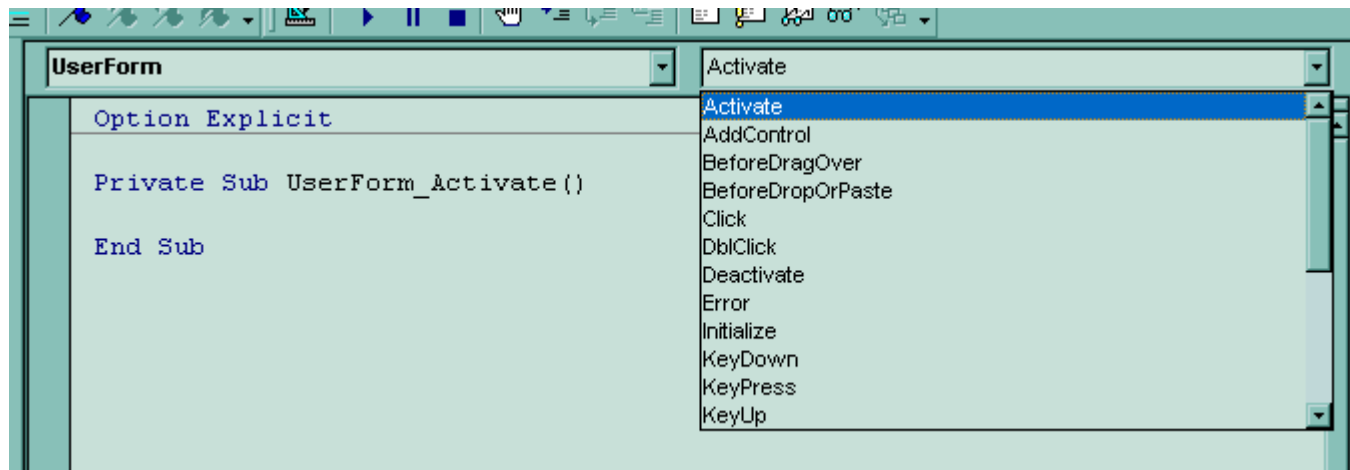
Zunächst muss die ComboBox mit den Namen der Ausgabearten gefüllt werden, unter denen man später auswählen können will (gehabt haben ..).

Diese Vorbereitung des Formulars wird auf dem zugehörigen Programmblatt programmiert. Nach Doppelklick auf eine freie Stelle des Formulars erscheint ein Codefenster mit einem vorgefertigten Text, einem Prozedurrahmen, leider nicht mit dem erforderlichen.

```
Private Sub UserForm_Click()

End Sub
```

Also einfach löschen, im oberen rechten Auswahlfeld des Codefensters auf "Activate" klicken und schon sieht es so aus:



Diese Prozedur "UserForm_Activate" wird bei Öffnung des Formulars mit "Show" automatisch ausgeführt und muss nicht extra angesprungen werden.

Wir verweilen, dies ist der Moment der größten Ratlosigkeit, wie soll es denn weiter gehen?

Nun, die "ComboBox1" soll ihre Einträge erhalten, ihr sollen die "Items" "addiert" werden. Deshalb heißt es ComboBox1.AddItem "irgendwas".

```
Option Explicit      'erzwingt Variablendeklaration

Private Sub UserForm_Activate()
    UserForm1.ComboBox1.AddItem "Miete" 'Ausgabearten
    UserForm1.ComboBox1.AddItem "Ernährung"
    UserForm1.ComboBox1.AddItem "Auto"
    .....
End Sub
```

Das kann man gleich testen, in der Menüleiste das kleine schwarze Dreieck tippen, schon ist das Formular zu sehen, in der Combobox steht nicht etwa "Miete", sondern nichts. Erst wenn man es aufblättert, sieht man, es hat sich etwas getan.

Nun ja, das geht - in mehrfacher Hinsicht - auch eleganter und lässt sich noch etwas ergänzen:

```
Private Sub UserForm_Activate()
    Dim reihe As Integer, spalte As Integer
    reihe = 1
    For spalte = 2 To 6
        UserForm1.ComboBox1.AddItem Cells(reihe, spalte)
    Next
    'ersten Eintrag vorwählen
    UserForm1.ComboBox1.ListIndex = 0
    'Zugabe: Tooltip = Erklärung, wenn die Maus darüber steht
    UserForm1.ComboBox1.ControlTipText = "Ausgabeart"
End Sub
```

Diesmal werden die Spaltenüberschriften nacheinander ausgelesen. Man sieht, wie auf der Inhalt der Excel-Zellen zu lesen ist. Einfach mit dem Schlüsselwort "Cells(i, j)". Die beiden Indizes adressieren die Zelle. Der jetzt folgende Test bringt (wahrscheinlich) eine Enttäuschung, die ComboBox zeigt nicht, was sie soll. Grund: Wir haben "reihe" und "spalte" mit Zahlen gestartet, die nicht unbedingt stimmen. Hier sind die Werte für Ihre konkrete Tabelle anzupassen. Diese Anpassung bietet Stoff zum Nachdenken. Ihre Ausgabearten mögen etwa ab Spalte "E" stehen, wieso ist dann "For spalte = 5 To ..." einzutragen? Fragen über Fragen!

CommandButton1 heie der "Fertig"-Knopf. Fr den knnte eingetragen werden:

```
Private Sub CommandButton1_Click()
    UserForm1.Hide      'beendet Anzeige des Formulars
    Range("a1").Select  'zeigt linke obere Ecke des Excel-Formulars
End Sub
```

Nun funktioniert schon alles Unwichtige, blo der Eintrag in die Tabelle nicht.

Jetzt nutzen wir den zweiten Button, der die bernahme des eingetragenen Wertes veranlassen soll. Zuerst mal eine einfache Variante, welche die eigentliche Arbeit macht. Spter soll sie noch verbessert werden. Also: Es muss eine Zelle gefunden werden, die leer ist. Dazu wird die gewnschte Spalte (deren berschrift in der ComboBox gewhlt ist) von oben (Nummer der ersten Zeile nach dem Kopf) nach unten (Zeile vor der Summenzelle) durchsucht. So etwa geht es:

```
Private Sub CommandButton2_Click()
    Dim reihe As Integer, spalte As Integer
    spalte = UserForm1.ComboBox1.ListIndex + 1 + 1 'Spalte whlen
    reihe = 14
    While ActiveSheet.Cells(reihe, spalte).Value <> "" 'erste leere Zeile suchen
        reihe = reihe + 1
    Wend
    Cells(reihe, spalte) = TextBox1.Text 'Zahl eintragen
End Sub
```

Interessant die Zeile mit dem "... +1+1"! Die Zhlung der ComboBox beginnt mit 0, deshalb meldet sie beim ersten Eintrag 0 statt 1. Das wird mit dem ersten +1 korrigiert. Das zweite +1 ist der Offset (der Abstand der ersten Spalte zum Rand), hier bei mir war das 1, knnte bei Ihnen auch 123 oder so sein. Die beiden Zeilennummern sind ebenfalls jeweils anzupassen. Das liee sich zwar eleganter lsen, wie sagt der Volksmund: "Warum einfach, wenn es auch kompliziert geht?"

Ein zweites Problem bietet die Abfragezeile. Gesucht wird nach der ersten leeren Zelle. Eigentlich wre es aber besser, alle Zellen vorab auf "0" zu setzen und sie zu formatieren. Danach steht in jeder Zelle "0,00 ". Wie man leicht herausfindet, kann dann in unserer Programmzeile getrost nach "0" gefragt werden, Excel findet die richtige Zelle.

Notwendige Ergnzung ist, wenn es denn schon mehrere Tabellen in der Mappe gibt, die richtige Tabelle fr den Eintrag auszuwhlen. Das geschieht mit dem ".Activate"-Befehl. In der Testphase ist es auerdem bequem, den zu fllenden Tabellenbereich ins aktive Fenster zu holen. Das erledigt "Range.Select ()". Beide Zeilen werden am Anfang der letzten Prozedur eingefgt:

```
Sheets("Tabelle1").Activate  'Excel-Tabelle aktivieren
Range("b13").Select         'Tabelle in obere linke Ecke
...
```

Bei "Range" ist die Angabe der oberen linken Ecke des Bereiches ntig, den man sehen will. Dann kann man eine fehlerhafte Adressierung schnell erkennen.

Die Excel-Rechenroutine, wenn sie denn ordnungsgem eingesetzt wurde, erzeugt innerhalb dieser Sammeltablelle

automatisch zu jedem neuen Eintrag eine neue Spaltensumme.
Wenn alles funktioniert, stellen sich erste Schwächen heraus:

- Ist mein letzter Eintrag übernommen worden?
- ich habe versehentlich zweimal geklickt? Was nun?
- Was geschieht, wenn ich statt einer Zahl etwa aus Wut "zu viel" eingebe?

Man muss etwas tun. Dazu ein neues Kapitel.

3. Verbesserungen am Eingabeformular

Die Vorhaben ergeben sich aus den genannten Mängeln.

Werteübernahme wird angezeigt

Da man auf die TextBox leicht zugreifen kann, bietet es sich an, diese entweder bei Übernahme des Wertes zu löschen, oder sie zur Bestätigung zu nutzen. Dazu muss innerhalb der "Sub CommandButton2_Click()" eine entsprechende Zeile eingefügt werden. Das könnte etwa so aussehen:

```
Private Sub CommandButton2_Click()  
    ...  
    TextBox1.Text = "gebucht: " & Cells(reihe, spalte).Value  
End Sub
```

Interessant ist, dass, auch wenn in der Tabelle 34,22 € steht, die Meldung nach Übernahme nur "gebucht : 34,22" lautet. Aus Gründen der Schönheit sollte man hinter das Textfeld selbst die Währungsangabe, also "€" (Strg+Alt+E) in ein Label setzen.

Alternativ kann man aber auch den wirklichen Tabelleninhalt auslesen, dann muss statt ".Value" (was auch entfallen kann), nur ".Text" geschrieben werden.

Zweifache Eingabe verhindern

Ist eigentlich schon fast erledigt. Denn wenn man die gerade beschriebene Kontrollanzeige im Fenster stehen hat und erneut klickt, gibt es einen fehlerhaften Eintrag in der Tabelle, der Text wird mit eingetragen, das Ganze nicht als Zahl erkannt. Das erinnert daran, dass eine reine Textübernahme aus dem Textfeld eh' leichtsinnig ist. Man ersetze in der fraglichen Prozedur (in welcher wohl?) die Zeile "Cells(row, column) = TextBox1.Text" durch "Cells(row, column) = CCur(TextBox1.Text)", was die Eingabe in das richtige Währungsformat umwandelt. Neuer Test, jetzt gibt es beim zweiten Klick wenigstens eine Fehlermeldung. Was nun?

Nägel mit Köpfen: Wenn der Eintrag erzeugt wird, sollte der Button deaktiviert werden. Steuerelemente werden mit ".Enabled = False" deaktiviert. Deshalb tragen wir noch ein:

```
Private Sub CommandButton2_Click()  
    ...  
    CommandButton2.Enabled = False  
End Sub
```

Und wie bekommen wir ihn wieder aktiv? Natürlich, indem die gleiche Eigenschaft ".Enabled" wieder auf "True" gesetzt wird. Bloß wann und wo?

Mein Vorschlag: Sobald die Maus in die Textbox klickt, um einen neuen Eintrag zu beginnen, soll netterweise die Box geleert werden und dann könnte man ja gleichzeitig den Button wieder enablen (doch, das gibt es offenbar, ist bloß Neudeutsch!).

```
Private Sub TextBox1_MouseDown(ByVal Button As Integer, ByVal Shift As Integer,  
    ByVal X As Single, ByVal Y As Single)  
    TextBox1.Text = ""  
    CommandButton2.Enabled = True  
End Sub
```

Diesen speziellen Prozedurkopf (Sub-Kopf klingt zu doof!), holt man sich aus den beiden Auswahlfeldern im Kopf des Codefensters (links die TextBox1 wählen, rechts dazu MouseDown klicken.)

Drittes Problem lösen: Fehleingaben verhindern

Die Übernahme von Text haben wir durch die Umwandlungsfunktion "CCur()" schon verhindert, allerdings haben wir uns mögliche Fehlermeldungen eingehandelt. Da gibt es ja auch noch das Problem des versehentlich statt eines Kommas eingegebenen Punktes.

Nun, man könnte doch einfach nur gültige Einträge zulassen. Wenn man wüsste wie es geht. Wird sofort verraten. Wir überwachen die Tasteneingaben in die Textbox und lassen nur Zulässiges zu. Dazu gibt es das Ereignis "KeyPress" der Textbox, dessen Prozedurkopf man sich holt, wie oben für "MouseDown" beschrieben. Der Rest ist etwas komplexer, eine Funktion, die unsere eingegebene Zahl bei jedem Tastendruck prüft. Probieren Sie's!

```
Private Sub TextBox1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    KeyAscii = PrüfeZeichen(KeyAscii, TextBox1.Text)
End Sub

Function PrüfeZeichen(Prüfling As MSForms.ReturnInteger, vorhTxt As String) As Integer
    Dim erlaubt As String
    If Not (Prüfling = 8) Then '8 ist Backspace zum Korrigieren
        erlaubt = "0123456789"
        If Len(vorhTxt) > 0 Then erlaubt = ",0123456789"
        If InStr(vorhTxt, ",") Then erlaubt = "0123456789"
        If InStr(erlaubt, Chr(Prüfling)) = 0 Then
            PrüfeZeichen = 0
            Exit Function 'Prüfling war unzulässig!
        End If
    End If
    PrüfeZeichen = Prüfling 'Prüfling ist zulässig!
End Function
```

4. Übernahme der Summenwerte einzelner Ausgabearten aus der Sammliste in die Gesamtübersicht

Am Monatsende haben wir ein Problem. Was soll nun geschehen?

Möglichkeit 1:

Man müsste die jeweilige Summe in die Jahresübersicht übernehmen. Das könnte eine Programmroutine werden, die die Summenzeile in die zum Monat gehörige Zeile der Jahrestabelle übernimmt.

Bloß, wann genau soll denn die Trennung in die jeweiligen Monateinträge erfolgen? Muss man etwa immer genau am letzten Tag des Monats den Rechner anschmeißen, damit die Übernahme pünktlich erfolgt? Darüber ist noch nachzudenken.

Gleichzeitig muss die Hilfs-Liste gelöscht werden, weil sonst Werte des Vormonats im laufenden Monat erneut eingerechnet werden würden. Das ginge etwa so:

```
Sub Listelöschen()  
Dim reihe As Integer, spalte As Integer  
Sheets("Tabelle1").Activate  
Range("B13").Select  
For reihe = 13 To 22  
    For spalte = 2 To 8  
        Cells(reihe, spalte) = "0" 'löschen  
    Next  
Next  
Range("A1").Select 'zur Gesamtübersicht zurückgehen  
End Sub
```

Allerdings sind nun alle Einzeleinträge weg. Hatten wir nun den Autokauf eingetragen oder nicht?
Denke, denke, denke.

2. Möglichkeit

An sich haben unsere Rechner ja genug Platz, um die Hilfstabelle einfach aufzuheben. Das bedeutet, eine Hilfstabelle für **jeden** Monat.

Das hat noch mehr Vorteile.

Die Übernahme der Werte lässt sich nun ohne Programmhilfe dadurch erreichen, dass jede Zeile der Jahrestabelle ihren Inhalt aus der zugehörigen Sammeltablette holt.

Nun muss nur noch bei der Eingabe von Werten in Abhängigkeit vom Datum der Eintragung die passende Sammeltablette ausgewählt werden. Wenn man die Sammeltabellen etwa um eine konstante Zeilenzahl nach unten kopiert, dann lassen sich die Lagen dieser Monatstabellen berechnen. Das bedeutet in der oben angegebenen Routine `Sub CommandButton2_Click()` eine geringe Veränderung, Berechnung des zu aktivierenden Bereiches. Hier wurden die Sammeltabellen um jeweils 20 Zeilen verschoben.

```
Private Sub CommandButton2_Click()  
...  
Dim bereich As String  
...  
    reihe = 17 + (Month(Now) - 1) * 20  
    bereich = "b" & CStr(reihe)  
    Range(bereich).Select  
...  
End Sub
```

Die obige "17" ist natürlich wieder anzupassen, dort beginnt nun die Reihe der Januar-Einträge. `Month(Now)` bedeutet, was man auch vermuten würde, die Monatsnummer. Schließlich wird für die Anzeige der zutreffenden Monatstabelle der `"bereich"` berechnet. Das `"b"` steht für die zweite Spalte, wenn man eine andere an den Rand schieben will, muss natürlich deren Buchstabe hier hin.

5. Modul zum Formularaufruf

Ein Modul wird innerhalb der IDE wie der/die oder das UserForm eingefügt: **Menüleiste** <Einfügen>, <Modul>.

Globale Variable gibt es nicht. Es genügt, hier den Formularaufruf einzutragen:

```
Option Explicit

Sub Daten_Eingeben()      'frei vergebbarer Name
    UserForm1.Show       'Aufruf des Formulars
End Sub
```

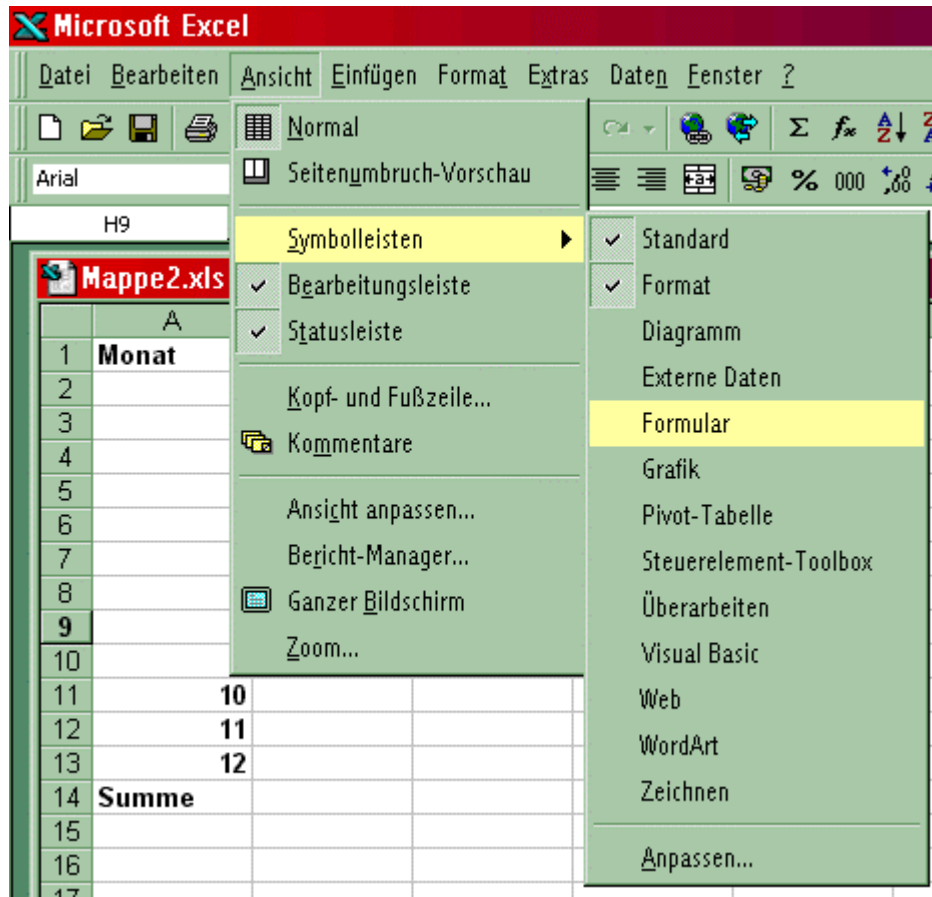
Nach diesen Einträgen ist der Name dieser Startprozedur, hier "Daten_Eingeben" als Makroname in der Makroliste des Projektes eingetragen und kann wie alle Makros aus der Excel-Menüleiste über <Extras>, <Makro>, <Makros...> ausgewählt und gestartet werden. Dies wäre aber unpraktisch, deshalb soll später noch eine freundlichere Aufrufmöglichkeit realisiert werden.

6. Verbesserter Aufruf

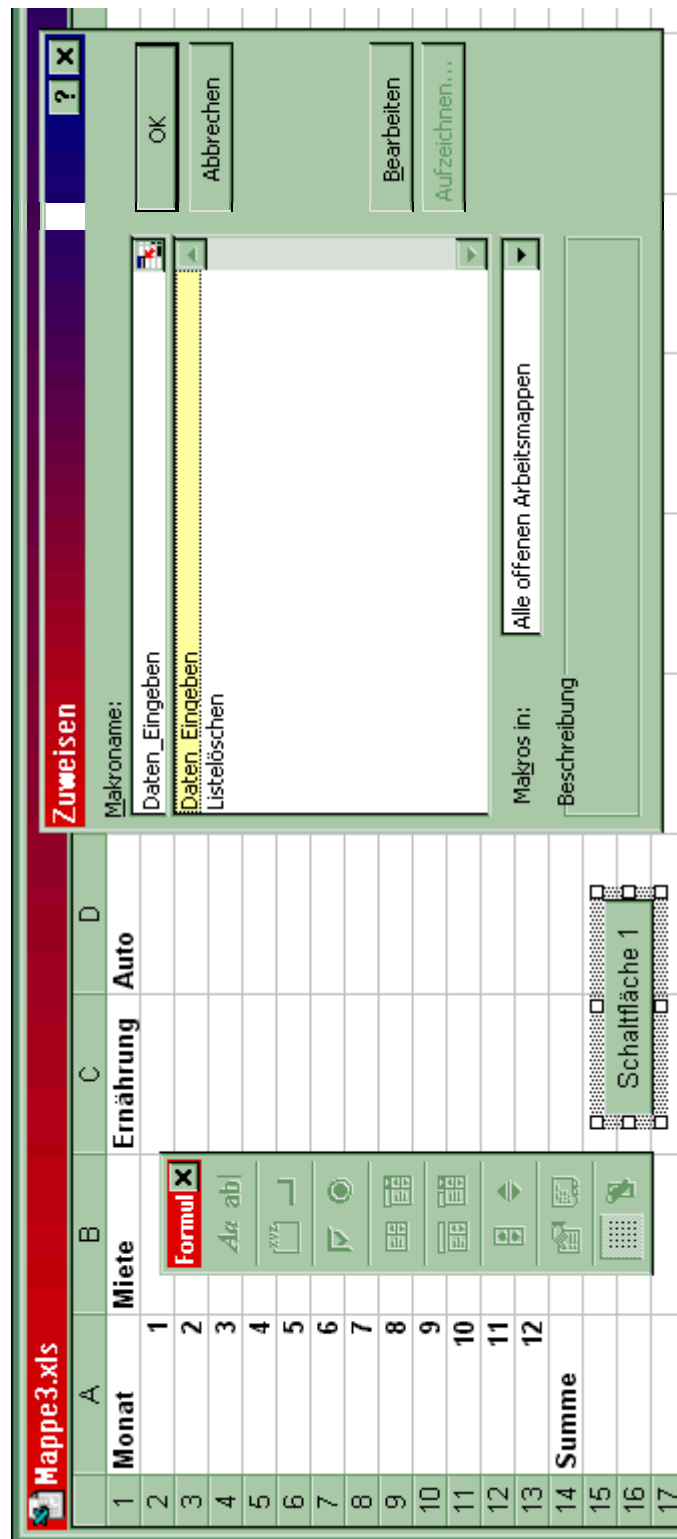
Zum Aufruf von Makros bietet Excel mehrere Möglichkeiten. Besonders bequem ist es, wenn im Bereich der Übersichtstabelle spezielle Befehlsschaltflächen dafür vorhanden sind.

Deren Einbau wird nun beschrieben:

Im Menü von Excel wird aufgerufen `<Ansicht>`, `<Symbolleiste>`, `<Formular>`.



Aus der Formulsymbolleiste zieht man einen Button neben die Tabelle auf das Tabellenblatt. Es erscheint ein Fenster "Zuweisen", das die vorhandenen Makros zur Zuweisung anbietet. Es wird der Name (hier "Daten_Eingeben") gewählt.



Zum Schluss ist noch der Bedienknopf zu beschriften (Anwahl ohne Makrostart mit "Shift"). Per Doppelklick kann man noch die Formatierung verändern:

Schaltfläche 1 =

Mappe3.xls

	A	B	C	D
1	Monat	Miete	Ernährung	Auto
2		1		
3		2		
4		3		
5		4		
6		5		
7		6		
8		7		
9		8		
10		9		
11		10		
12		11		
13		12		
14	Summe			
15				
16				
17				
18				
19				

Formularfenster: Steuer-element formatieren

Schrift: Ausrichtung Größe Schutz Eigenschaften Abstände

Schriftart: Antiqua Oliv (w1) | Antiqua Oliv (w1) | Animals 1 | Animals 2 | Antiqua Oliv (w1) | Arial

Schriftschnitt: Standard | Standard | Kursiv | Fett | Fett Kursiv

Schriftgrad: 10 | 8 | 9 | 10 | 11

Unterstreichung: Ohne | Automatisch | Standardschrift

Farbe: Automatisch

Darstellung: Durchgestrichen Hochgestellt Tiefgestellt

Vorschau: AaBbCcYyZz

Maßstäbliche Druckerschriftart: Ausdruck kann von der Bildschirmansicht abweichen.

OK Abbrechen

Werte eingeben

Nun ist das Projekt arbeitsfähig. Anschließend sollte über sinnvollen Ausbau nachgedacht werden. Wäre ja nicht verkehrt, auch über die doch hoffentlich vorhandenen Einkünfte Buch zu führen. Im Eingabeformular lässt sich das jeweils (noch) vorhandene Geld anzeigen. Die erforderlichen Techniken, um das zu realisieren, sind hier alle schon vorgekommen: Tabelle für die Einkünfte, Verknüpfung mit den Ausgaben, Formular erweitern. Viel Erfolg!