Excel VBA

Teil 11.8

Zusammenfassung!

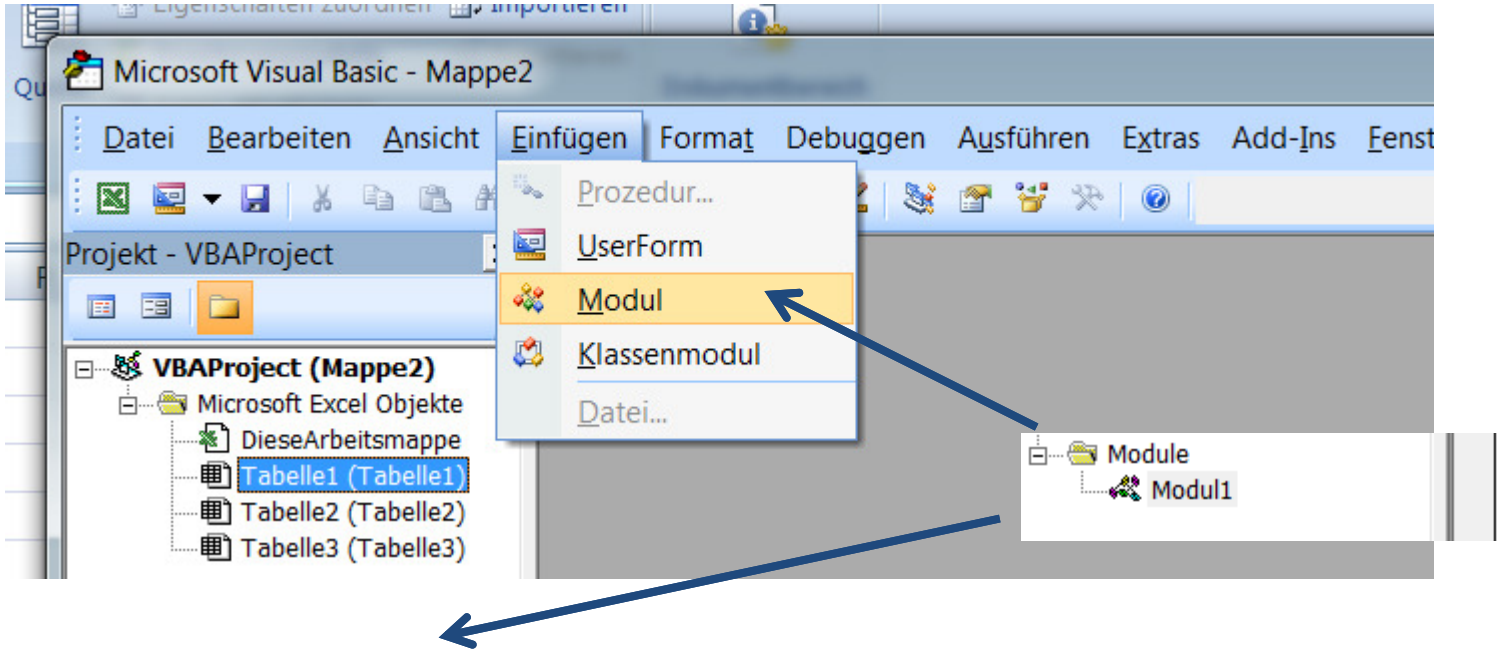
Was wir können sollten!

V0.5 5.4.2013

Inhaltsverzeichnis

Modul einfügen

Um ein Makro zu erstellen, muss man ein Modul eingefügen.



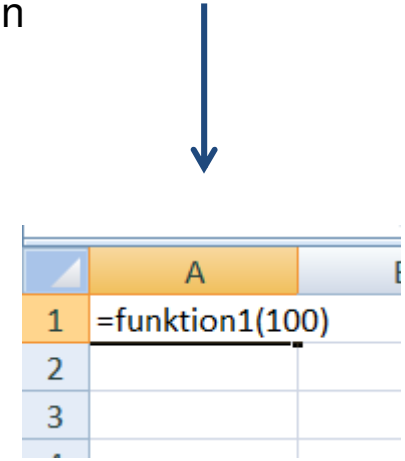
Prozeduren oder Funktionen

Ausführbare Makros sind in der Regel Programme welche Excel fernsteuern.

```
Sub makro1()

 MsgBox ("Das ist ein Makro")

End Sub
```

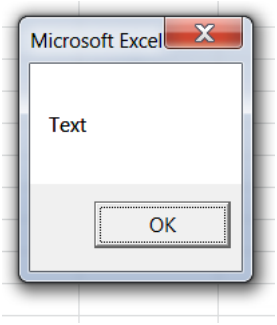
Funktionen kann man in Excel Formeln direkt verwenden.

```
Function funktion1(Zahl As Double) As Double

    funktion1 = Zahl * Zahl

End Function
```

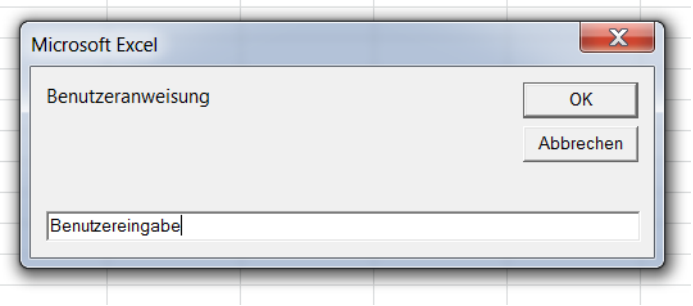


Ein/Ausgaben

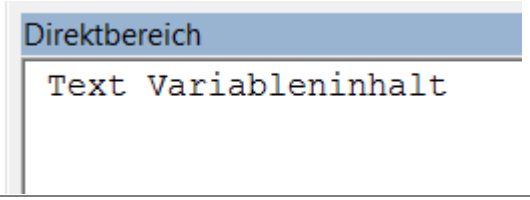
- MsgBox (“Text“)



- variable = Inputbox(“Benutzeranweisung“)



- Debug.Print “Text“ & Variable



Variablen

Variablen sind
Zwischenspeicher
für Programm-Daten

So wird die Variable varname mit den Type Integer deklariert (= Speicher reserviert).

```
dim variblen_name as integer
```

Nummerische Datentypen

Excel verfügt über 6 nummerische Datentypen:

Ganzahlige Typen (ohne Komma)

- Byte 0…255 (1Byte)
- Integer -32768 … 32767 (2 Byte)
- Long -2147483648 ..2147483647 (4 Byte)

Gleitkommazahlen

- Single $\pm 3{,}402823 * 10^{36}$ …. $\pm 1{,}401298 * 10^{-45}$ (4 Byte)
- Double $\pm 1{,}7976931348623 1 * 10^{308}$ …
  $\pm 4{,}940656458412 47 * 10^{-324}$ (8 Byte)

Festkomma

- Currency -922337203685477,5808 … 922337203685477,5807.

(8 Byte im Ganzzahligen Format aber durch 10000 dividiert)

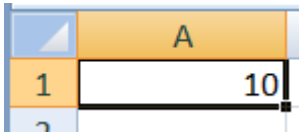
Sonstige Datentypen

- Date zum Speichern von Datum und Zeitangaben (8Byte)
- String zum Speichern von Texten
- Boolean zum Speichern von logischen Werten,
  also „True“ oder „False“
- Variant hier entscheidet Excel VBA welcher Datentyp
  notwendig ist. In Variant können alle Datentypen
  gespeichert werden.

Wert einer Zelle zuweisen

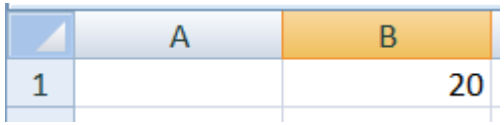
```
ActiveSheet.Range("A1").Value = 10
```

Ergebnis:




```
ActiveSheet.Cells(1, 2).Value = 20
```

Ergebnis:

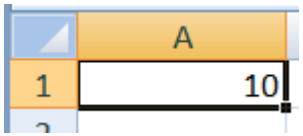


Wert aus Zelle lesen

Zellenwert einer Variable (a,b) zuweisen

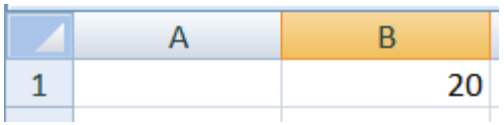
```
a =  ActiveSheet.Range("A1").Value
```

Ergebnis: a =10



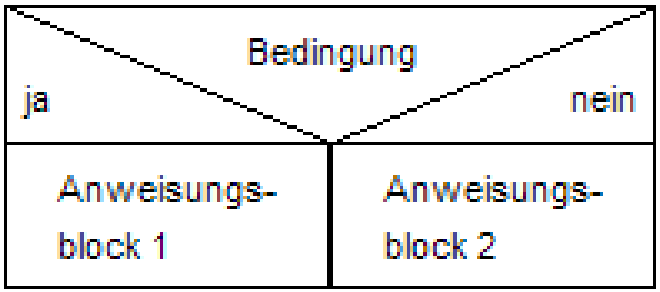

```
b = ActiveSheet.Cells(1, 2).Value
```

Ergebnis: b =20



If Anweisung

Die If Anweisung ermöglicht uns Anwiesungsblöcke nur dann auszuführen, wenn eine Bedingung erfüllt ist.




```
If (x = 1) Then
    MsgBox (“x = 1")
Else
    MsgBox (“x <>1")
End If
```

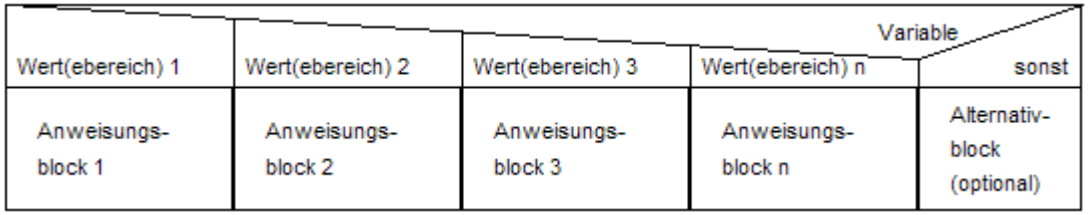
Vergleichsoperatoren

für Bedingungen

>	größer
<	kleiner
=	gleich
<>	ungleich
>=	größer gleich
<=	kleiner gleich

Select Case

Mit Select Case können Sie ganz einfach mehrere Fälle unterscheiden.

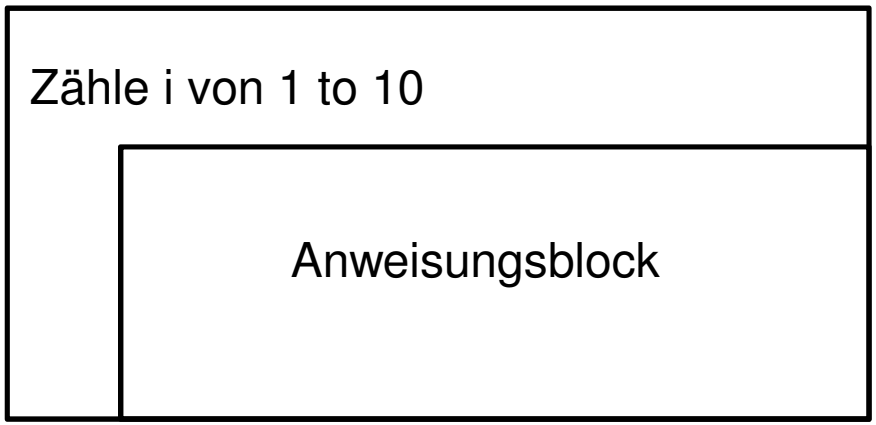


Select Case

```
Select Case  variable
Case 1
  Debug.Print „Fall 1"
Case 2
   Debug.Print „Fall 2"
Case 3
   Debug.Print „Fall 3"
Case 4
   Debug.Print „Fall 4"
Case 5 To 7
   Debug.Print „Fall 5 bis 7„
Case Else
   Debug.Print „sonstiger Fall"
End Select
```

For … Next Schleife

Die For-Next Schleife ermöglicht es einen Anweisungsblock, in einer vorgegebenen Anzahl von Schleifendurchläufen, auszuführen.

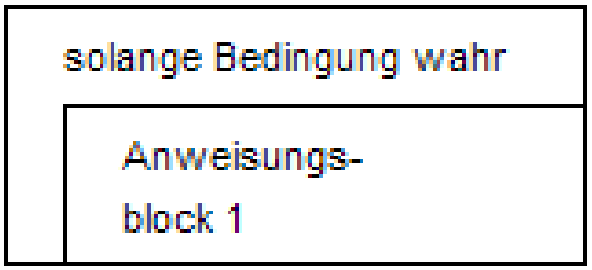



```
For i = 1 to 10
    z = z +1
    Debug.print z
Next i
```

Anweisungsblock

Do while loop - Schleife

Die do while loop Schleife wird nur so lange durchgeführt wie die Bedingung erfüllt ist.

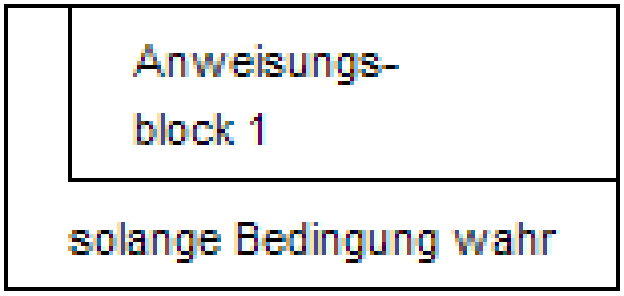



```
Sub dountilschleife()
Dim i As Integer

i = 1

Do While i < 5
    Debug.Print i
    i = i + 1
Loop

End Sub
```

Befehlssequenz wir keinmal durchlaufen, wenn der logische Ausdruck schon zu Beginn falsch ist.

Do loop while- Schleife

Die do while loop Schleife wird nur so lange durchgeführt wie die Bedingung erfüllt ist.




```
Sub dountilschleife()
Dim i As Integer

i = 1

Do
   Debug.Print i
    i = i + 1
Loop While  i < 5

End Sub
```

Befehlssequenz einmal durchlaufen, wenn der logische Ausdruck schon zu Beginn falsch ist.