VBA-Einführung

mit Excel 201x 

nur das Wichtigste

Agenda

A. Die ersten Schritte
- Was ist VBA ?
- Makro-Rekorder
- VBA-Editor
- „Hallo Welt“

B. Generelles
- Subs & Functions
- Module, Formulare
- Programmaufruf
- Variablen & Datentypen
- Operatoren
- Excelobjekte
- Datenaustausch mit Register

C. Techniken
- Schleifen
- Verzweigungen
- Debuggen
- Arrays
- Strings
- Fehlerbehandlung

D. Nützliches
- Ereignisse
- Eigene Datentypen
- Datenaustausch mit txt-file
- Pfad auslesen
- Split

Tag 1

Tag 2

Visual Basic for Applications

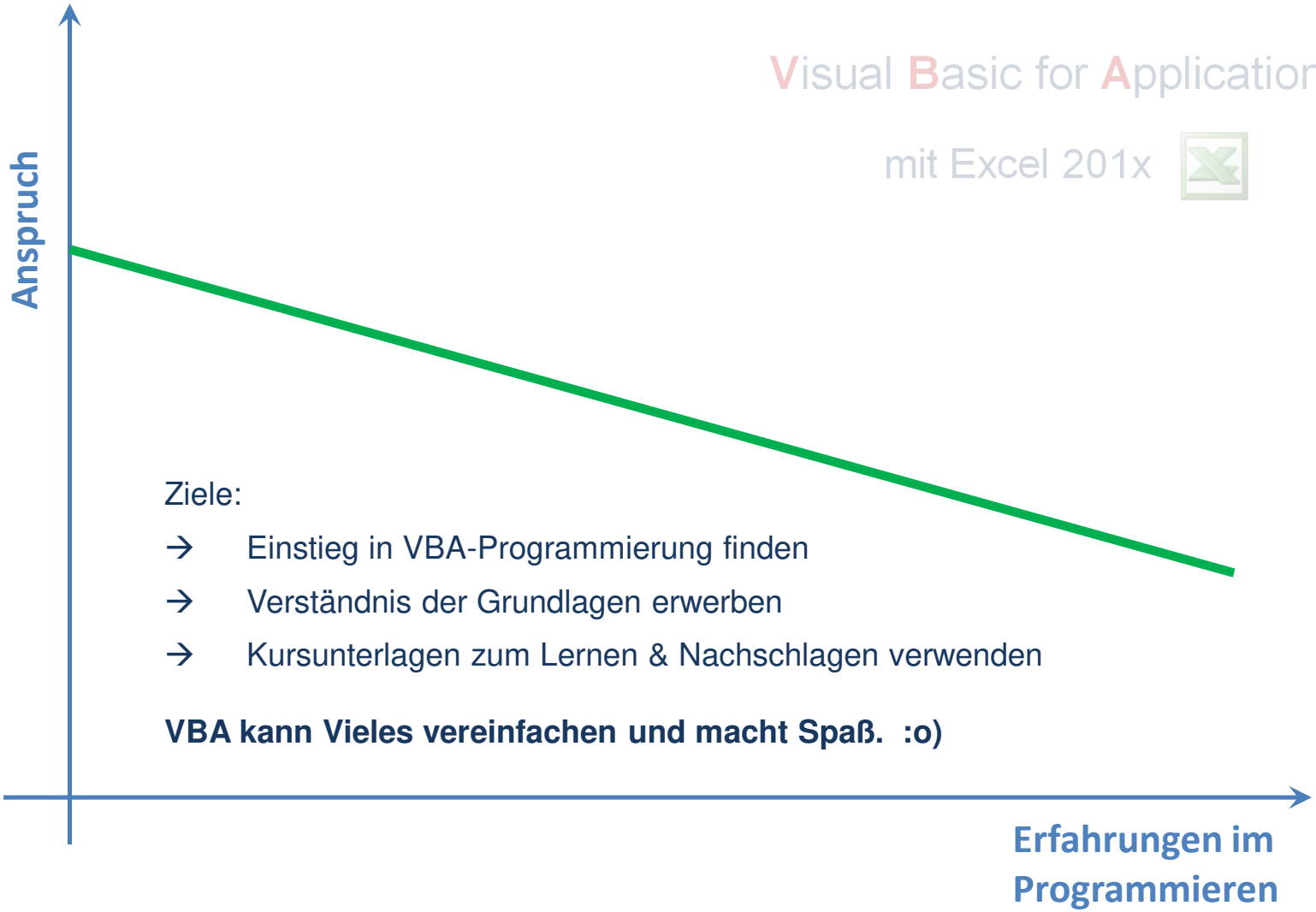
mit Excel 201x 

= Übung,
bitte direkt ausprobieren

Anspruch/Ziel des Trainings

Visual Basic for Applications

mit Excel 201x



Ziele:

→ Einstieg in VBA-Programmierung finden

→ Verständnis der Grundlagen erwerben

→ Kursunterlagen zum Lernen & Nachschlagen verwenden

VBA kann Vieles vereinfachen und macht Spaß. :o)

VBA

…ist eine **Programmiersprache** (interpretiert)
…ist in sämtlichen **Office**-Programmen enthalten
…wurde von Visual Basic abgeleitet
...ist in ein Wirtsprogramm **(z.B. Excel)** integriert
…verfügt über viele **Objekte** des Wirtsprogramms
…dient der
- **Automation** von Abläufen des Wirts-Programms
- „freien“ **Programmierung**

…kann **externe** Programme (Famos, Inca…) steuern, z.B. über COM
…wird häufig zur automatisierten **Datenverarbeitung** verwendet **:**

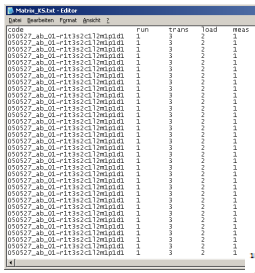
Visual Basic for Applications

Bsp

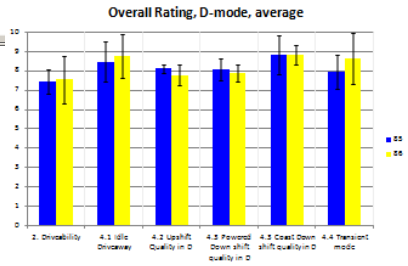
Daten
Daten
Daten
Daten
Daten
Daten
Daten
…

VBA-Programm

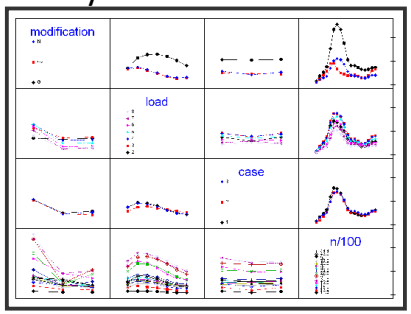
Extraktion



Visualisierung



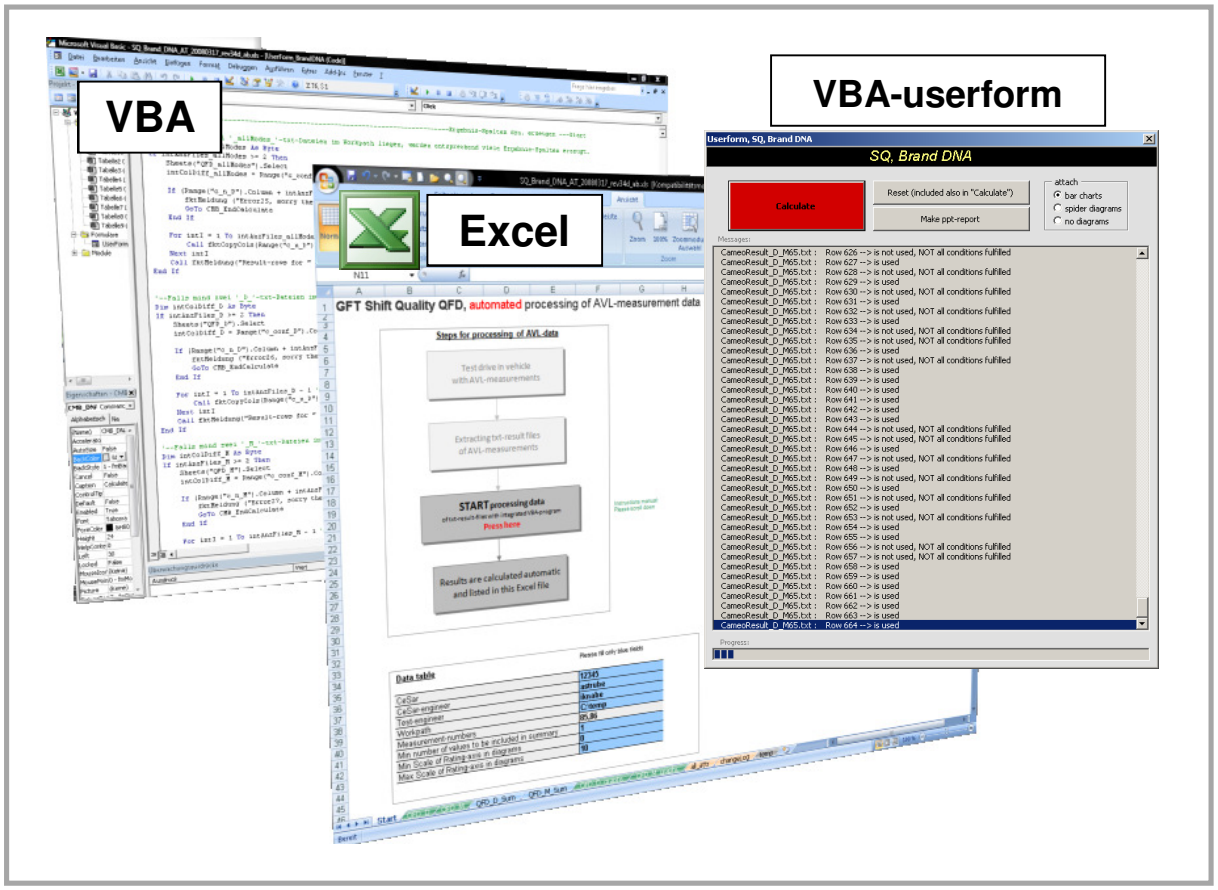
Analyse



z.B. mit Minitab

VBA

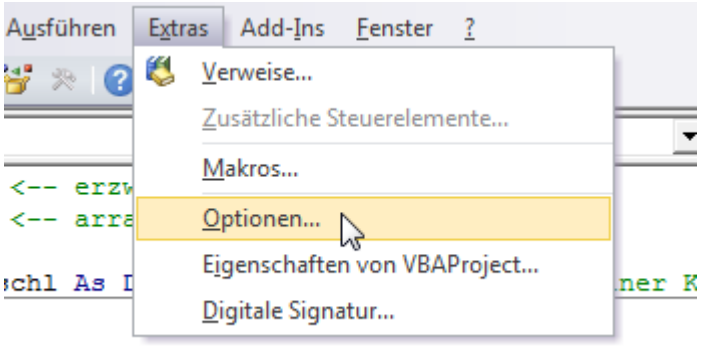
Jede **Excel-Mappe** kann um VBA-Programme erweitert werden.



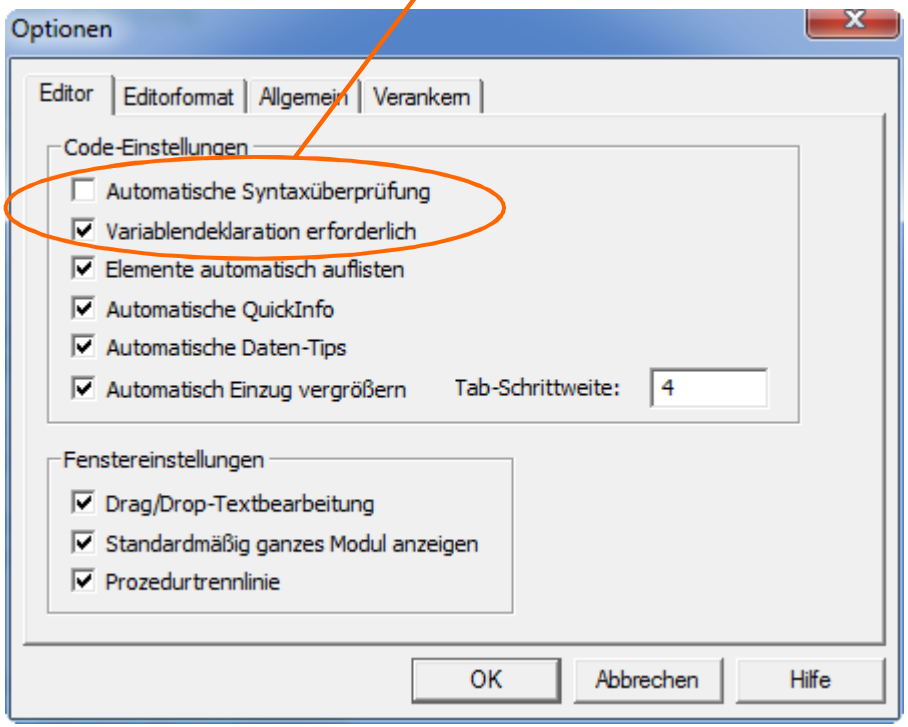
VBA

Empfehlungen für Voreinstellungen im VBA-Editor

Alt + F11 startet den VBA-Editor



Empfehlung



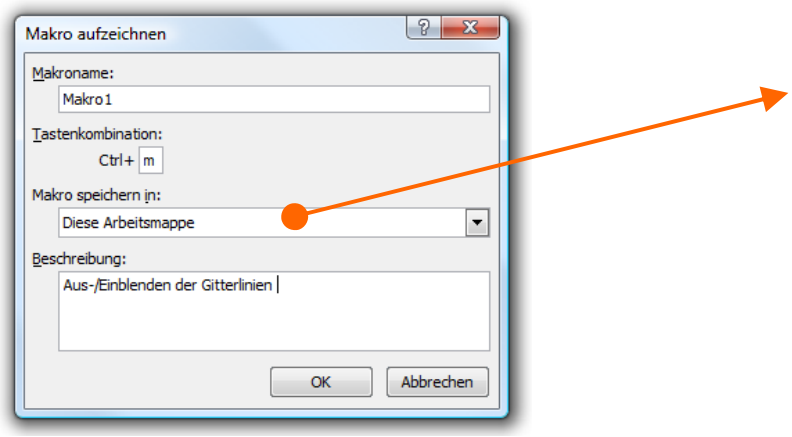
*„**Automatische Syntaxprüfung**“*: **Haken nicht gesetzt** → fehlerhafte Syntax wird rot markiert, aber es erscheint nicht andauernd ein Popup-Fenster (**empfohlen**)

*„**Variablendeklaration erforderlich**“*: **Haken gesetzt** → jedes neue Modul wird automatisch mit „Option explicit“ versehen. (**empfohlen**)

Makro-Rekorder

1. Menüband: Ansicht, Makros, **“Makro aufzeichnen“**

2.



3. Aufzuzeichnende Aktionen durchführen

4. Menüband: Ansicht, Makros, **“Aufzeichnung beenden“**

Speicherort „**Persönliche Arbeitsmappe**“ :

→ Makro wird ausserhalb der Excelmappe gespeichert

→ und steht in jeder Excelmappe auf dem eigenen Rechner zur Verfügung

Speicherort „**Diese Arbeitssmappe**“ :

→ Makro wird in der Excelmappe gespeichert

→ und steht mit dieser Excelmappe auch auf anderen Rechnern zur Verfügung

Ergebnis

Makro, das

- Excel-Aktionen in VBA-Code übersetzt hat
- jederzeit z.B. mit „Ctrl+m“ ausgeführt werden kann
- jederzeit angepasst werden kann

```
Sub Makro1()
'
' Makro1 Makro
' Aus-/Einblenden der Gitterlinien
'
' Tastenkombination: Strg+m
'
    ActiveWindow.DisplayGridlines = False
    Range("A9").Select

End Sub
```

Ü

Übung, Makro-Rekorder

1. Leere Excel-Mappe öffnen
2. Unter Ansicht, Makros, "**Makro aufzeichnen**" anklicken
3. Einträge vornehmen (s. unten links)
4. Excel-Register hinzufügen und eine Mini-Tabelle + Chart erstellen (s. unten rechts)
5. Unter Ansicht, Makros, "**Aufzeichnung beenden**" anklicken
6. Excelmappe schliessen
7. Neue Excelmappe öffnen und Makro starten (Ctrl+m)
8. Code kürzen/anpassen und ausprobieren

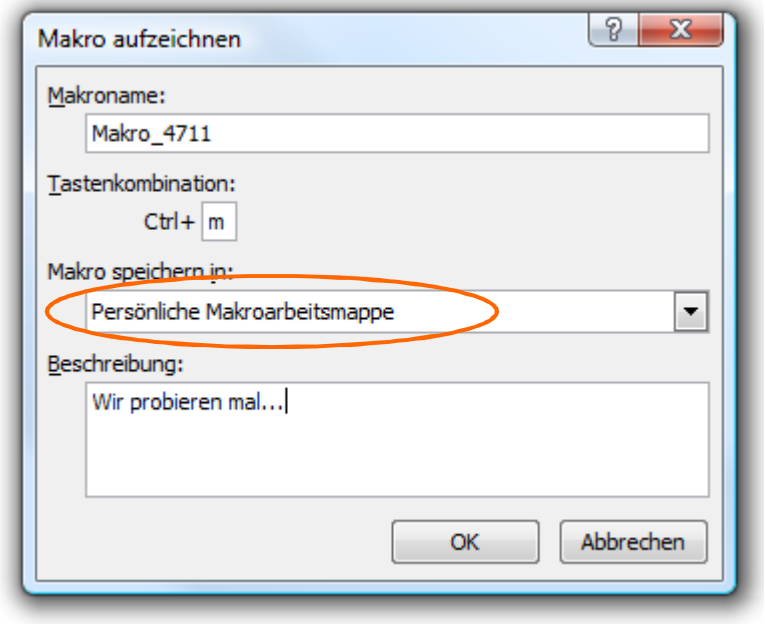
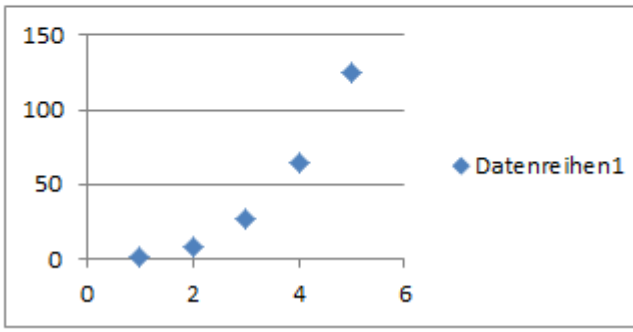
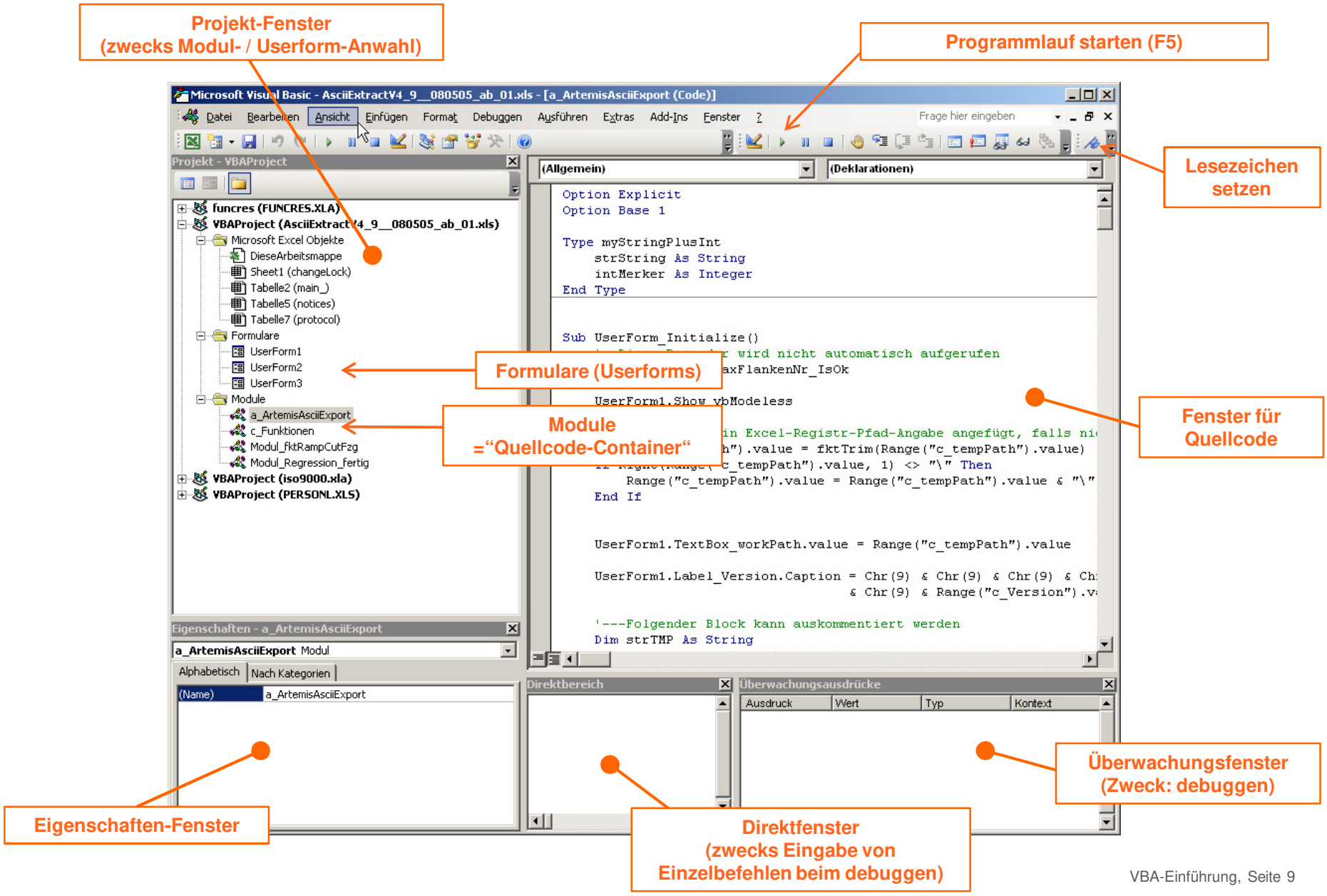


Tabelle4

	A	B	C	D
1	1	1		
2	2	8		
3	3	27		
4	4	64		
5	5	125		
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				



VBA-Editor (Alt+F11)

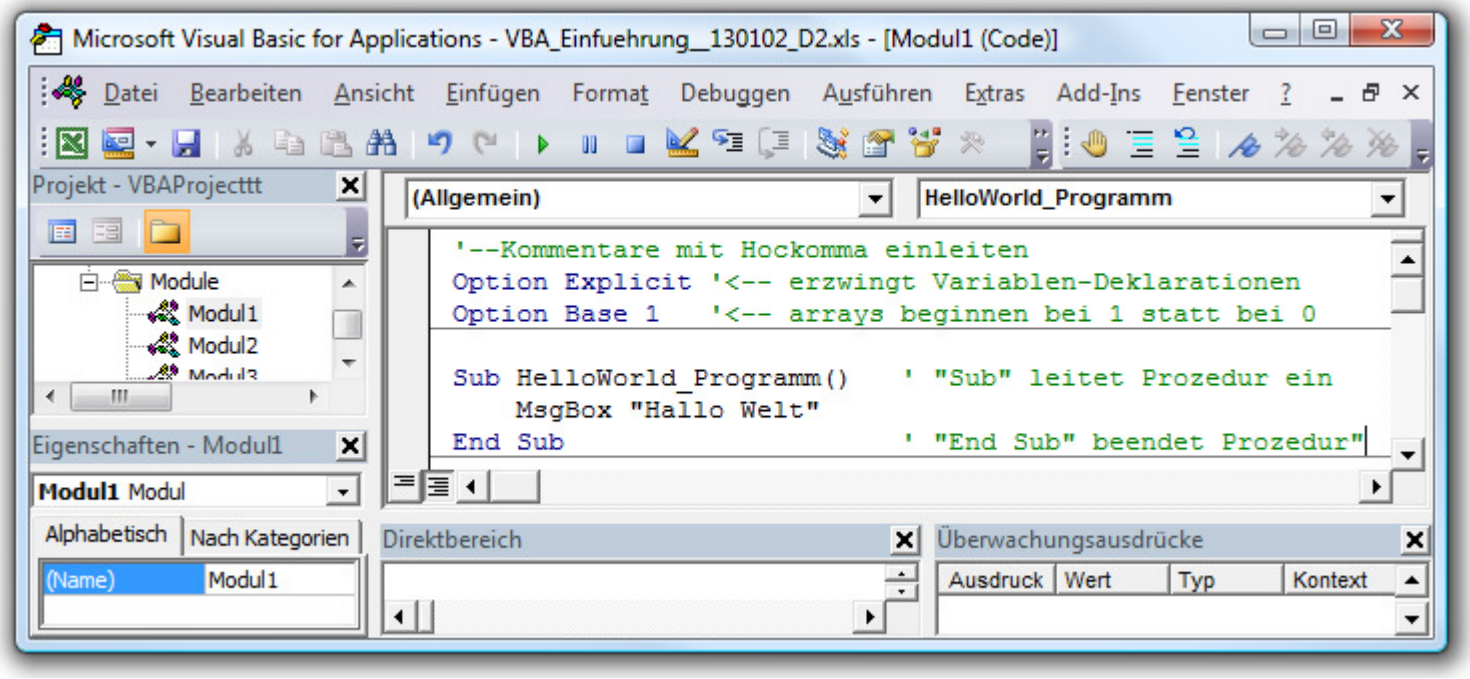


Hallo Welt

Ein “**Hallo Welt**“-Prográmmchen mit Excel-VBA

1. Excel starten
2. VBA-Editor starten (Alt+F11)
3. RM in Projekt-Fenster, “Einfügen, Modul“
4. Quellcode eintippen
5. Cursor innerhalb der Sub platzieren
6. Programm starten (F5)

s. Modul1



Subs & Functions

Was ist eine Sub ?

```
Sub HelloWorld_Programm()    ' "Sub" leitet Prozedur ein
    MsgBox "Hallo Welt"
End Sub                      ' "End Sub" beendet Prozedur"
```

Eine Sub…

…ist eine eigenständige Prozedur (= Gruppe von Programmbefehlen)

…liefert keinen Rückgabewert

…kann eine andere Sub aufrufen

Ein VBA-Programm kann mit jeder Sub gestartet werden

Was ist eine Function ?

```
Function fktMal2(dblX As Double) As Double
    fktMal2 = dblX * 2   'Rückgabewert der Funktion
End Function
```

Eine Function…

…ist eine Prozedur die funktional abgegrenzten Code auslagert

…wird i. d. Regel mehrfach aufgerufen

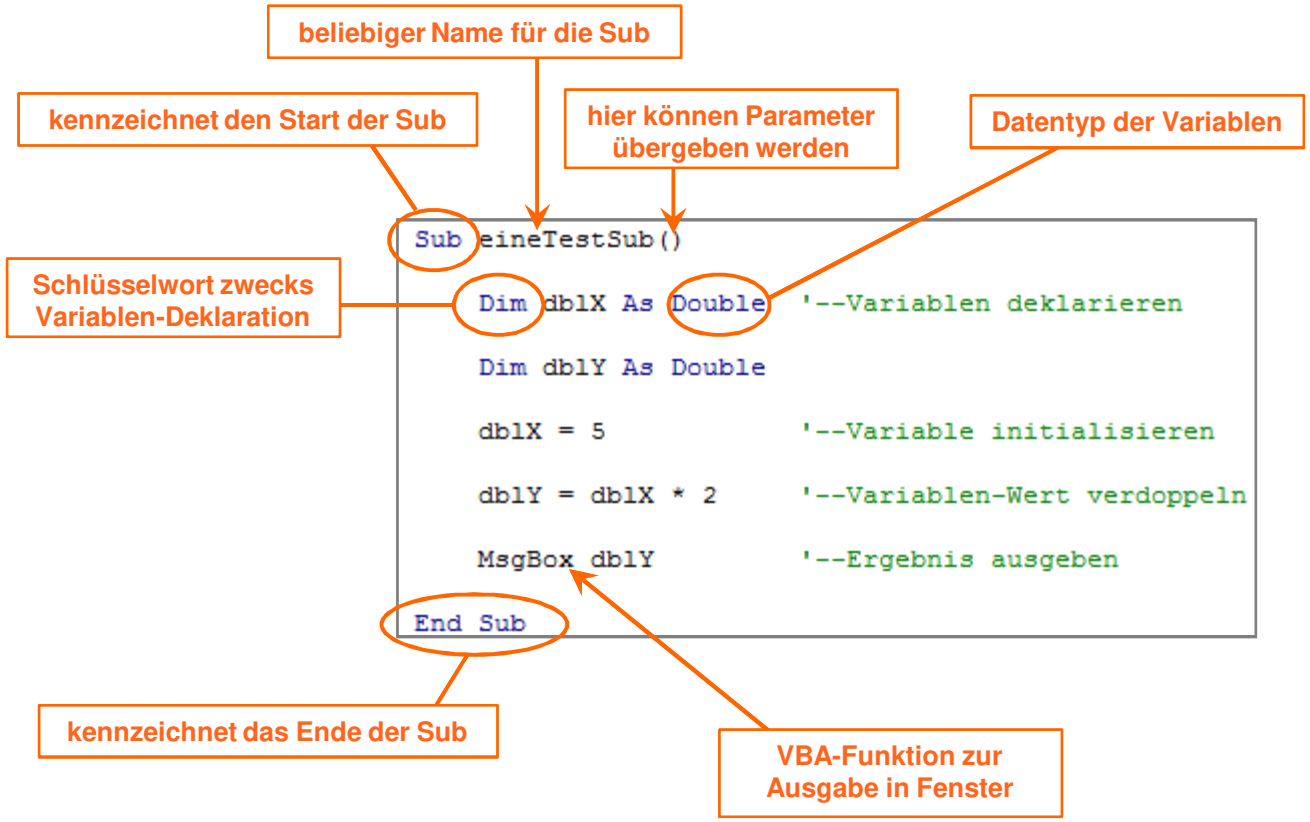
…dient der besseren Lesbarkeit und Fehlervermeidung

…liefert einen Rückgabewert (in der Regel)

VBA ist für die prozedurale Programmierung konzipiert

Subs & Functions

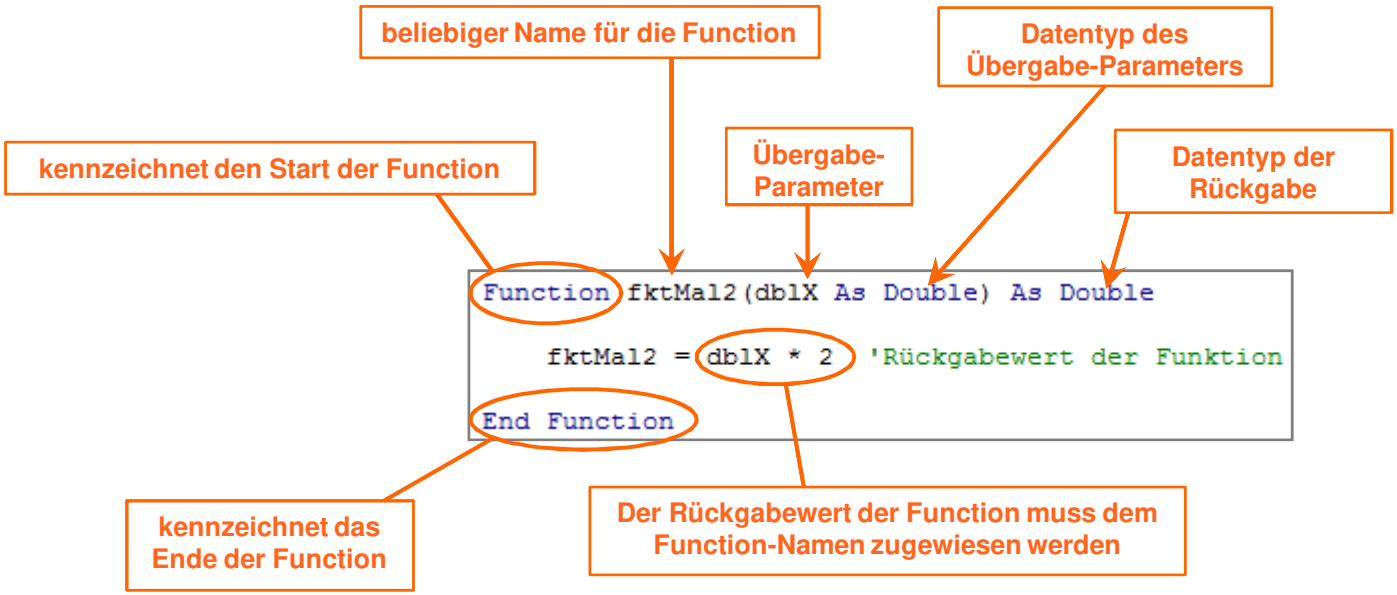
Wie ist eine Sub aufgebaut ?



Modul1

Subs & Functions

Wie ist eine Function aufgebaut ?



Modul1

Subs & Functions

Wie wird eine Function aufgerufen?

Function-Aufruf mit Parameterübergabe

Ergebnis der Function wird dblY zugewiesen

```
Sub eineTestSub_mitFktAufruf() '--Test zu Function-Aufruf

    Dim dblX As Double          '--Variable deklarieren

    Dim dblY As Double

    dblX = 5                    '--Variable initialisieren

    dblY = fktMal2(dblX)        '--Variablen-Wert mittels Funktion verdoppeln

    MsgBox dblY                 '--Ergebnis ausgeben

End Sub
```

```
Function fktMal2(dblX As Double) As Double

    fktMal2 = dblX * 2   'Rückgabewert der Funktion

End Function
```

Modul1

Übung

Erstelle eine Funktion die zwei Strings mit einem Bindestrich verkettet

Was sind Strings?

Strings sind Zeichenfolgen und werden mit Anführungszeichen angegeben.

Bsp: “Schnee“ ***&*** “mann“ → “Schneemann“

*Der **&-Operator** verkettet Strings*

oder: sText1 = “Schnee“

sText2 = “mann“

sErgebnis = sText1 ***&*** sText2

1. Öffne den VBA-Editor (Alt+F11)
2. Schreibe eine Function **fktVerkette** mit den Übergabe-Parametern **sText1** und **sText2**
3. Schreibe eine Sub, die die Function aufruft

   In der Sub sollen die Variablen s1, s2, sErgebnis verwendet werden.

   sErgebnis soll am Ende der sub mittels msgbox ausgegeben werden.
4. Führe die Sub im Einzelsatz (F8) aus und beobachte.

Modul2

Subs & Functions

Wie wird eine Sub aufgerufen ?

```
Sub test_1()
'--Eine Sub ruft andere Sub auf
    MsgBox " bin nun in der sub 'test_1'"
    Call test_2 '<-- Aufruf der 2ten Sub
    MsgBox " bin nun WIEDER in der sub 'test_1'"
End Sub

Sub test_2()
'--das ist die andere Sub
    MsgBox " bin nun in der sub 'test_2'"
End Sub
```

Mit „Call“ werden
- **andere Subs**
- **oder Functions aufgerufen**

Modul2

(auch in T-Version)

Module, Formulare

Was ist eine Formular (="Userform") ?

Ein Formular ist …

… ein seperates Fenster

… eine „Bediener-Schnittstelle“

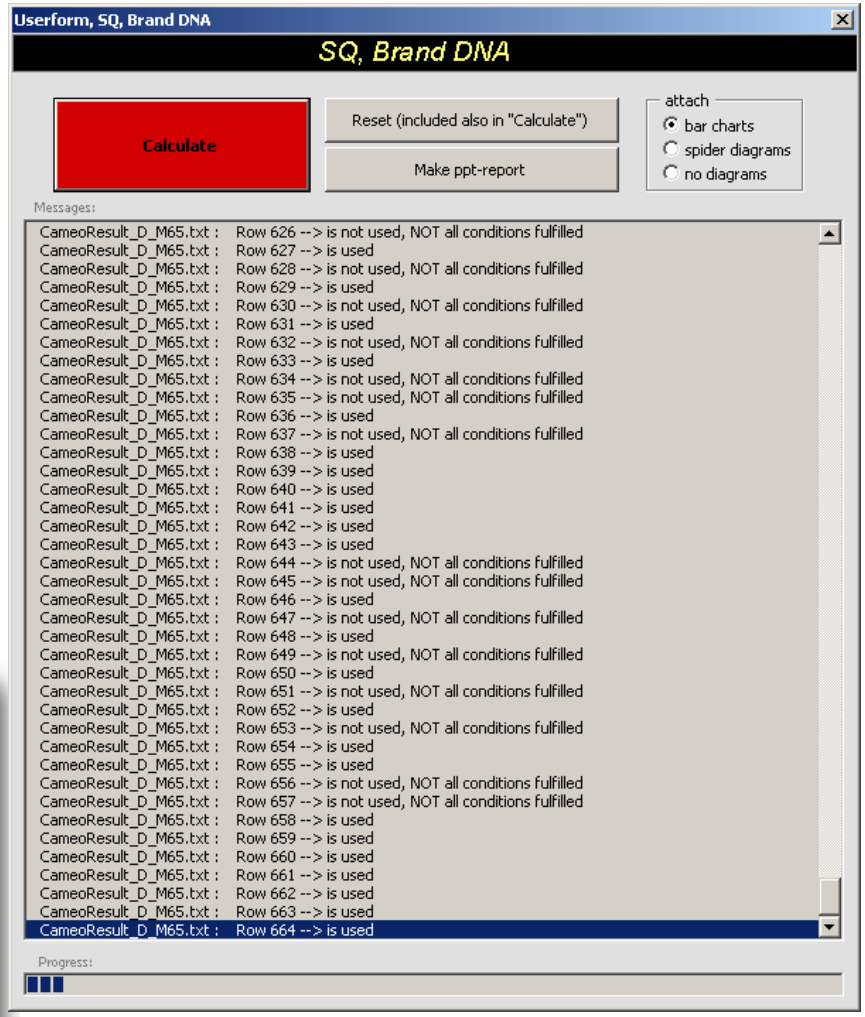
Es kann verwendet werden zur …

…Darstellung

…Eingabe/Ausgabe von Infos

…Programmsteuerung

…und kann VBA-Code enthalten



Beispiele

Module, Formulare

Was ist ein Modul ?

Ein Modul …

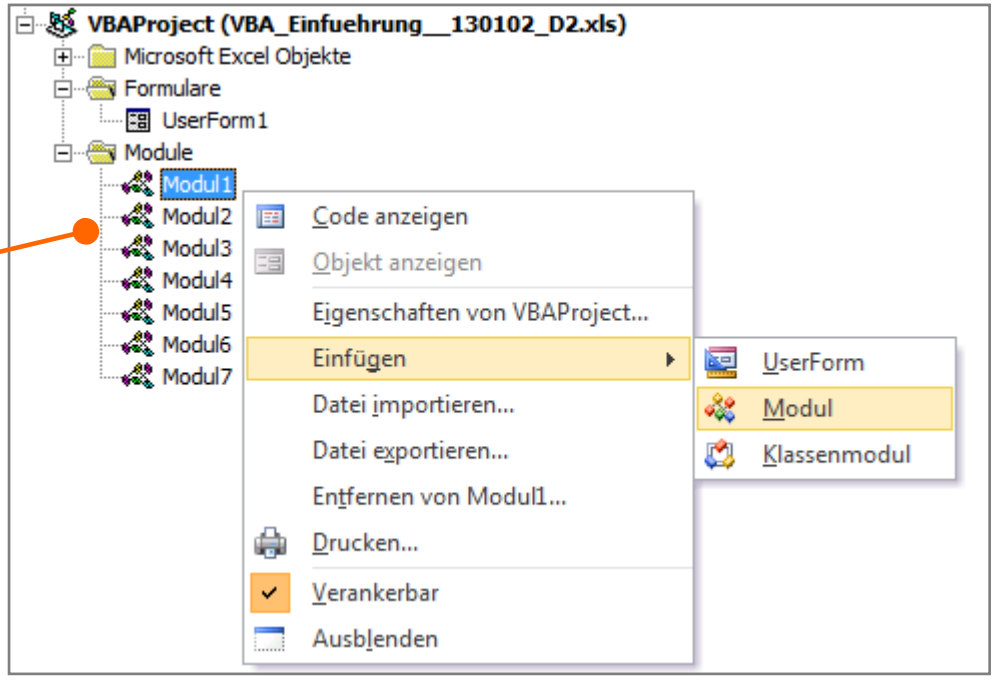
… ist ein “Quellcode-Container“

… besteht aus Deklarationen und Prozeduren

… kann importiert / exportiert werden (txt-Format)

… dient der Strukturierung

im Projekt-Fenster:



Mittels RM* im Projekt-Fenster

können Module erstellt/gelöscht werden

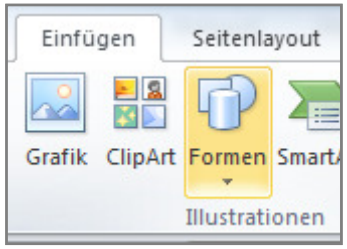
* RM = rechte Maustaste

Programmaufruf

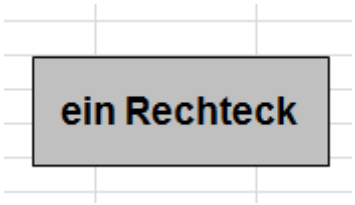
Variante1

VBA-Prozeduren/Makros können mit **Zeichenobjekten** (z.B. Rechtecke) verknüpft werden.

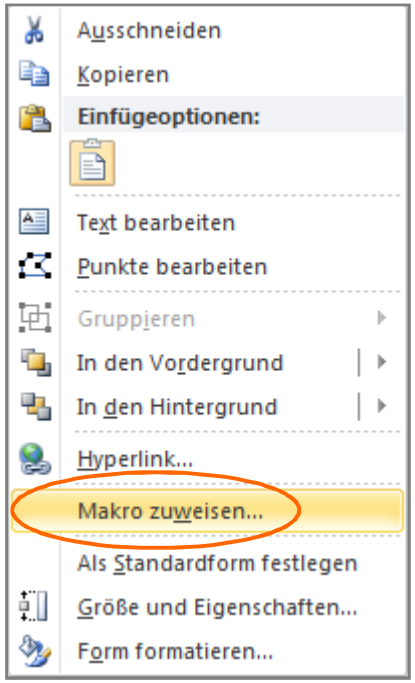
Ü



1. Zeichenobjekt erstellen, beschriften



2. RM, „Makro zuweisen“ anklicken

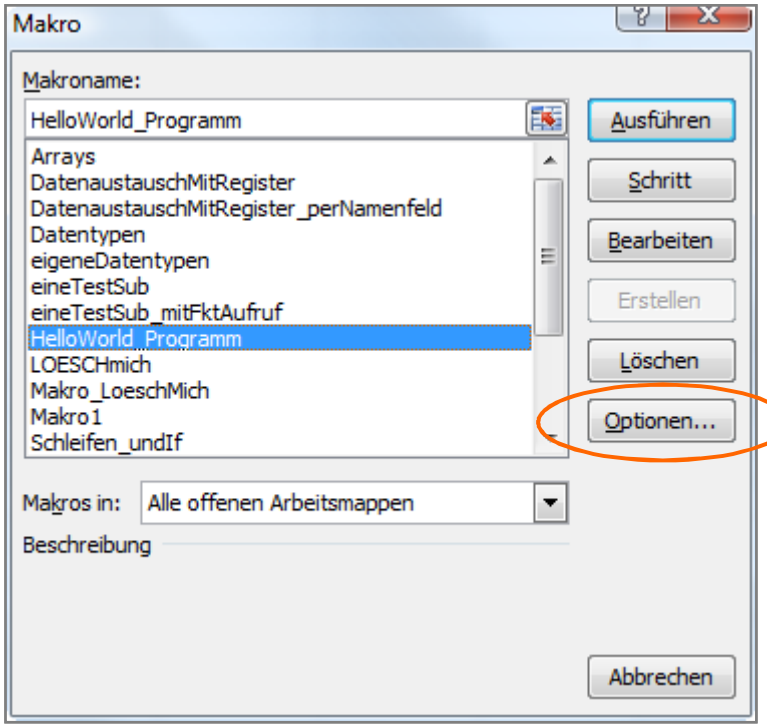


3. VBA-Prozedur auswählen

Programmaufruf

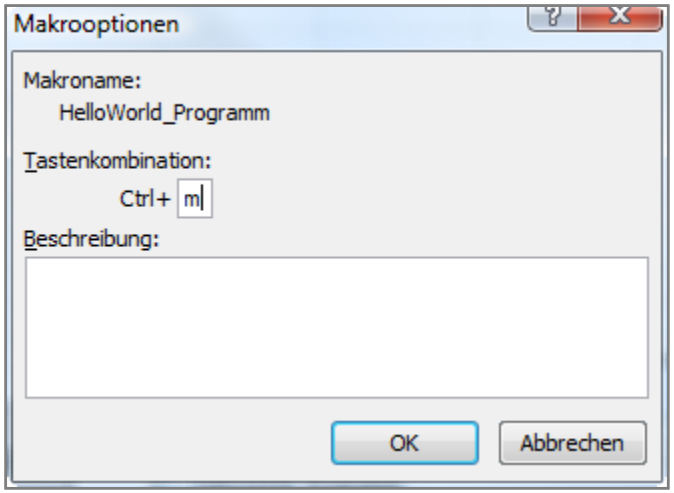
Variante 2

VBA-Prozeduren/Makros können mittels **Tastenkombination** aufgerufen werden.



1. Alt + F8 drücken

2. Optionen anklicken

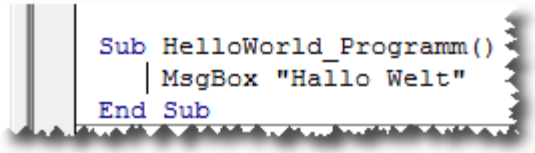


3. Tastenkombination vergeben, z.B: Ctrl + m

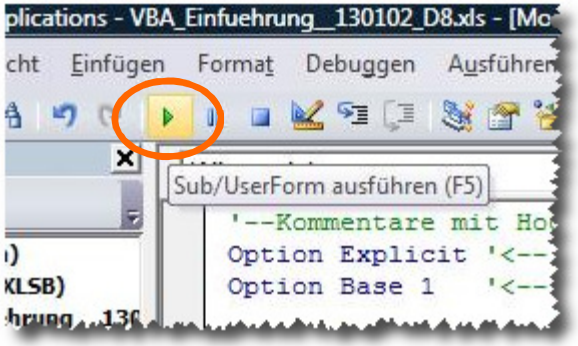
Programmaufruf

Variante 3

VBA-Prozeduren/Makros können **im VBA-Editor** direkt gestartet werden.



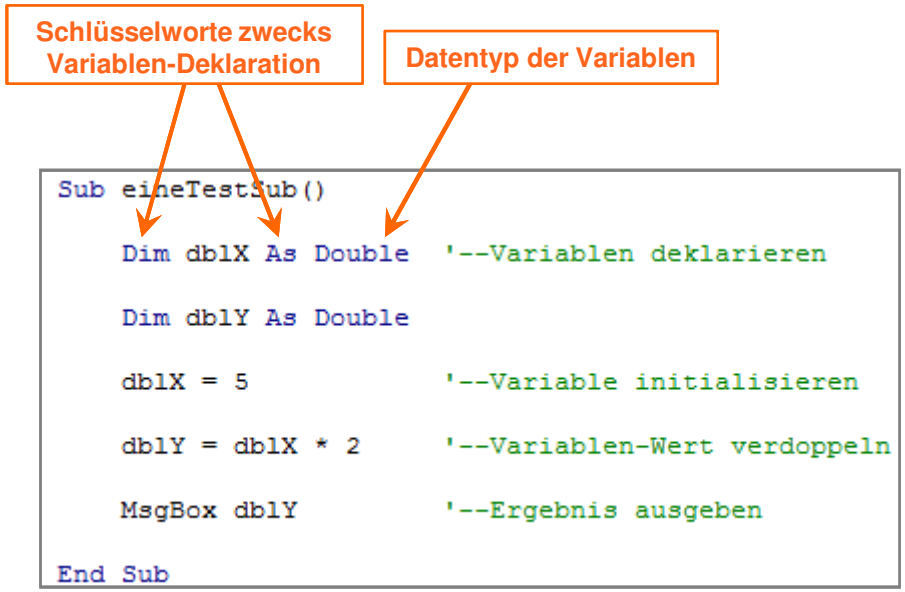
1. **Cursor** in auszuführende Sub setzen



2. **Grünen Pfeil** anklicken
   oder **F5**
   oder **F8 (Einzelschritt)**

Variablen & Datentypen

Deklaration von **Variablen:** **Dim** *Variablenname* **As** *Datentyp*



Modul1

Achtung:
Jede Variable in eigener Zeile deklarieren !!

Die Variable ist **gültig**

- in der Sub/Function in der sie deklariert ist oder
- in allen Subs des akt Moduls wenn sie vor der ersten Sub deklariert ist

Variablen & Datentypen

```
Option Explicit  'erzwingt Variablendeklaration, empfohlen
Option Base 1    'arrays beginnen bei 1

Sub Datentypen()
'*************************************************************
'*********** Bsp-Prozedur für wichtige Datentypen *************
'*************************************************************

    Dim intI As Integer          'Ganzzahl, -32768 bis 32767
    Dim lZahl As Long            'Ganzzahl, -2.147.483.648 bis 2.147.483.647
    Dim dblX As Double           'Dezimalzahl mit doppelter Genauigkeit
    Dim sglY As Single           'Dezimalzahl mit einfacher Genauigkeit
    Dim boolJaNein As Boolean    'Boolsche Variable, true oder false
    Dim strText As String        'String, Zeichenketten
    Dim objWs As Excel.Worksheet 'Objekt
    Dim varWert As Variant       'Variant, Datentyp kann wechseln

    Set objWs = ThisWorkbook.Worksheets("Tabelle1")

    intI = 17
    dblX = 3.14159265
    sglY = 17.8
    boolJaNein = True
    strText = "Zeichenfolge"
    objWs.Range("A1").Value = "einText"
    varWert = "erst ein string"
    varWert = 555 'und dann eine Zahl

    Set objWs = Nothing
End Sub
```

dbl: ca 310 Stellen
sgl: ca 40 Stellen

Modul2_c
(auch in T-Version)

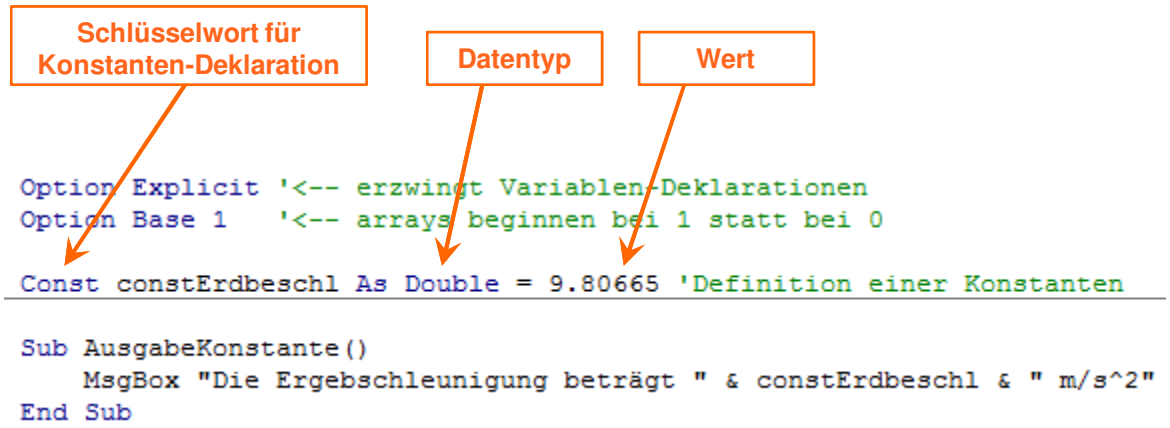
Empfehlung für Variablennamen:
kleingeschrieb. Datentyp-Kürzel +
aussagekräftiger Name
(beginnt mit Grossbuchstaben)

Bsp.:
iNummer
dblBeschleunigung
sVorname
boolGroesser
objRegister

→ **bessere Lesbarkeit , wichtig !!!**

Variablen & Datentypen

Deklaration von **Konstanten:** **Const** *Variablenname* **As** *Datentyp = Wert*



Ü

Modul1

Die Konstante ist **gültig**

- in der Sub/Function in der sie deklariert ist
- oder in allen Subs des akt Moduls, wenn sie vor der ersten Sub deklariert ist

Variablen & Datentypen

Gültigkeit von Variablen

```
Option Explicit
Option Base 1

'Globale Variable deklarieren
Dim dblG As Double

Sub GlobalLokalTest()

    'Lokale Variable deklarieren
    Dim dblY As Double

    'Wert von globaler V. ändern
    dblG = 1
    'Wert von lokaler V. ändern
    dblY = 2

    'Funktions-Aufruf
    Call fktTest

End Sub

Function fktTest()
    'Wert von globaler Variablen ändern
    dblG = 8.88
    'Wert von lokaler Variablen ändern
    dblY = 8.88 '<--Fehler, Variable hier unbekannt
End Function
```

Variablen, die

- ausserhalb einer Sub / Function deklariert werden, sind **global** verwendbar.

- innerhalb einer Sub / Function deklariert werden, sind **lokal** verwendbar.

Modul2_b

(auch in T-Version)

Operatoren

Bei Kombination von Operationen gilt

→Punkt vor Strichrechnung

→Klammerausdrücke haben Vorrang

abnehmende Priorität

Arithmetisch	Vergleich	Logisch
Potenzierung (^)	Gleich (=)	**Not**
Negation (-)	Ungleich (<>)	**And**
Multiplikation und Division (*, /)	Kleiner als (<)	**Or**
Ganzzahldivision (\)	Größer als (>)	**Xor**
Restwert (**Mod**)	Kleiner oder gleich (<=)	**Eqv**
Addition und Subtraktion (+, –)	Größer oder gleich (>=)	**Imp**
Zeichenverkettung (&)	**Like** **Is**	

abnehmende Priorität

```
Sub OperatorTest()
    Dim dblY As Double
    Dim intWert As Integer
    Dim strText As String

    dblY = 2 + 2 * (2 + 3) ^ 2   'ergibt 52
    intWert = 7 Mod 3            'ergibt 1
    strText = "Fuß" & "ball"     'ergibt "Fußball"
End Sub
```

Modul2_d

Operatoren

Bedeutung der Operatoren

Operatoren (arithmetisch)

+, -, *, /	plus, minus, mal, geteilt
\	Ganzzahldivision
^	Potenz
Mod	Modulo (Rest bei Ganzzahldivision)

Operatoren (Vergleich)

<	kleiner als
<=	kleiner gleich
>	größer als
>=	größer gleich
=	gleich
<>	ungleich
is	gleiches Objekt
like	gleiche Zeichenfolge

Operatoren (logisch)

Not	nicht
And	sowohl als auch
Or	entweder ... oder
Xor	genau einer von beiden
Eqv	beide wahr oder beide unwahr
Imp	außer wenn Ausdruck1 = true und Ausdruck2 = false
	immer wenn a dann b

Diverses

&	verkettet Zeichenfolgen (empfohlen)
+	verkettet Zeichenfolgen
.	Zugriff auf Methoden und Eigenschaften (Objekte)
:	trennt Anweisungen in einer Zeile
_	Fortführung der akt Anweisung in nächster Zeile

Übung

1. Tippe das gezeigte Programm „OperatorTest_2“ ab
2. Überlege erst wie das Ergebnis lauten müßte
3. Wie lautet das vom Programm ermittelte Ergebnis ?
4. Was fällt auf ?


```
Sub OperatorTest_2()
    Dim a As Double
    Dim b As String
    Dim c As Integer
    Dim d As Boolean

    a = 100 / 4
    c = 2 + 1.5 * a ^ 0.5

    '--Wie lautet das Ergebnis? Korrekt?
    MsgBox "Das Ergebnis ist: " & CStr(c)
End Sub
```

Modul2_d

Excelobjekte

Viele Elemente von Excel sind als „**Objekt**“ einer Klasse ansprechbar

Objekte/Klassen verfügen i.d.R. über **Eigenschaften und Methoden**,
die mittels **Punkt-Operator** nutzbar sind

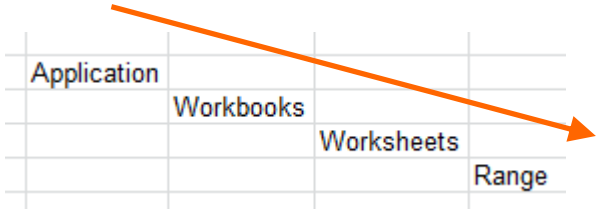
Bsp zur Begriffserklärung:

Klasse	→	Ford Fiesta
Objekt	→	Tante Gerda‘s Fiesta
Eigenschaft	→	Farbe, Anzahl Türen
Methode	→	Hupen, Beschleunigen

Aufruf des Objektkatalogs : F2 (zurück mit F7)

Excelobjekte

Objekte verwenden



Man unterscheidet Objekt-Auflistungen (z.B. “Workbooks“)
und Objekte selbst (z.B. Workbook(“Mappe1.xls“))

Objekte können

→ über ihre **Hierarchie** angesprochen werden:

```
Application.Workbooks("Mappe1.xls").Worksheets("Tabelle1").Range("A12").Value = 77
```

→ in Variablen gespeichert und verwendet werden:

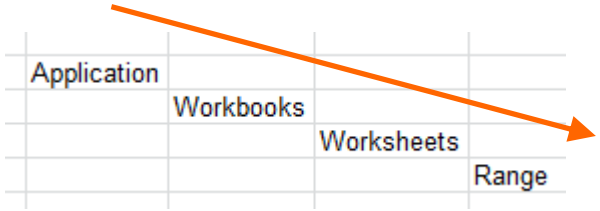
```
Sub ObjektVariablen()
'Objekt-Variablen deklarieren
Dim objWb As Workbook
Dim objWs As Worksheet
Dim objR As Range

'Objekt-Variablen initialsieren
Set objWb = Application.Workbooks("VBA_Einfuehrung_130102_D4.xls")
Set objWs = objWb.Worksheets("Tabelle1")
Set objR = objWs.Range("A1")

'Objekt-Variable verwenden
objR.Value = 17.2

End Sub
```

Modul2_e

(auch in T-Version)

Excelobjekte

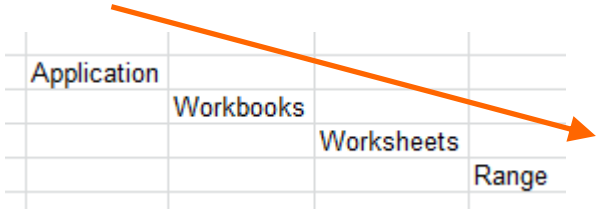
eine Auswahl…



Wichtige Objekte	Beschreibung
Application	Hauptobjekt, repräsentiert die Excel-Anwendung
Workbooks	Auflistung der Arbeitsmappen
Worksheets	Auflistung der Register (von der akt Arbeitsmappe)
ActiveWorkbook	Die aktive Arbeitsmappe
ActiveSheet	Das aktive Tabellenblatt
ActiveCell	Die aktive Zelle
ActiveChart	Das aktive Diagramm
ThisWorkbook	Die Arbeitsmappe des ausgeführten VBA-Code
Cells(Row,Column)	Eine bestimmte Zelle
Range	Zelle oder Zellverbund
Selection	Selektiertes Objekt
ActiveSheet.UsedRange	Aktuell verwendeter Bereich im Register

Excelobjekte

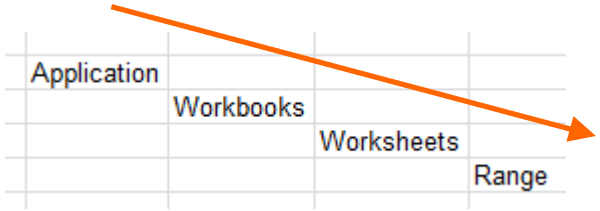
eine Auswahl bzgl **Arbeitsmappe** und **Register**



Wichtiges zu (Application / Arbeitsmappe / Register)	Beschreibung
Application.Screenupdating	Bildschirmaktualisierung aktivieren/deaktivieren
Application.DisplayAlerts	Warnungen und Meldungen aktivieren/deaktivieren
Workbooks.Open "Mappe1.xls"	Öffnet eine Arbeitsmappe
Workbooks ("Mappe1.xls").Close	Schliesst eine Arbeitsmappe
ActiveWorkbook.Name	Name der aktiven Arbeitsmappe
ActiveWorkbook.Save	Sichert die aktive Arbeitsmappe
ThisWorkbook.Path	Vollständiger Pfad der vorliegenden Arbeitsmappe
ThisWorkbook.SaveAs	Sichert die vorliegende Arbeitsmappe unter…
Worksheets("Tabelle1").Activate	Aktiviert ein Register
Worksheets("Tabelle1").Select	Aktiviert ein Register und holt es in den Vordergrund
Worksheets.Add	Neues Register hinzufügen
Worksheets.Count	Anzahl der Register
Worksheets(2).Name	2tes Register umbenennen

Excelobjekte

eine Auswahl bzgl Zellen



Wichtige Methoden (Zelle)	Beschreibung
Range("A1").Value	Inhalt der Zelle
Range("Namenfeld").Value	Inhalt der Zelle
Range("A1").Offset(Row, Col).Value	Inhalt der Offset-Zelle
Range("A1").Interior.ColorIndex	Hintergrundfarbe der Zelle
Range("A1").Value = ""	Inhalt der Zelle löschen
Range("A1").Adress	Zell-Adresse lesen
Range("A1").AddComment	Zell-Kommentar hinzufügen
Range("A1").Row	ZeilenNr der Zelle
Range("A1").Column	SpaltenNr der Zelle
Cells(Row,Col).Value	Inhalt der Zelle
Cells(Row,Col).Offset(Row, Col).Value	Inhalt der Offset-Zelle
Cells.ClearContents	Inhalt aller Zellen des akt Registers löschen

Datenaustausch mit Register

Excel-Zellen des aktiven Registers lesen / beschreiben mittels

Range() → Bsp. : dblX = Range("B3").Value ‘lesen
→ Bsp. : Range("B3").Value = 38 ‘schreiben

Cells() → Bsp. : dblX = Cells(intZeile, inSpalte).Value ‘lesen
→ Bsp. : Cells(intZeile, inSpalte).Value = 38 ‘schreiben

Namenfeld → Bsp. : dblX = Range(„c_Name").Value ‘lesen
→ Bsp. : Range(„c_Name").Value = 38 ‘schreiben

Register aktivieren mittels:
“Select“

Modul4

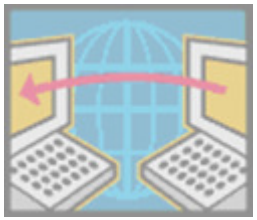
```
Sub Worksheet_Bsp()
Dim iAnzRegister As Integer

iAnzRegister = Worksheets.Count   'Anzahl der Register ermitteln
Worksheets("Tabelle2").Select     'Register über Namen aktivieren
Worksheets(2).Select              'Register über Index aktivieren
Worksheets(iAnzRegister).Select   'Letztes Register aktivieren
Worksheets.Add                    'Register hinzufügen

End Sub
```

Ü

Datenaustausch mit Register

```
Sub DatenaustauschMitRegister()
'--Daten mit Register austauschen

Dim dblX As Double
Dim dblY As Double
Dim intI As Integer

'--Zellen leeren
Worksheets("Tabelle1").Range("B5").Value = ""
Worksheets("Tabelle1").Range("B6").Value = ""

'--Wert aus Zelle LESEN (mehrere Möglichkeiten)
dblX = Worksheets("Tabelle1").Range("B3").Value      'mittels Range
dblX = Worksheets("Tabelle1").Range("c_NAME").Value 'mittels Namenfeld
dblY = Worksheets("Tabelle1").Cells(4, 2).Value      'mittels Cells
dblY = Cells(4, 2).Value 'so wird das aktive Register genommen

'--Wert in Zelle SCHREIBEN
Worksheets("Tabelle1").Range("B5").Value = dblX + dblY

'--Text in Zelle SCHREIBEN
Worksheets("Tabelle1").Range("B6").Value = "Hallo"

'--Quadratzahlen ins Register SCHREIBEN
For intI = 1 To 10
    Worksheets("Tabelle1").Cells(intI, 1).Value = intI ^ 2
Next intI
End Sub
```

zusammen



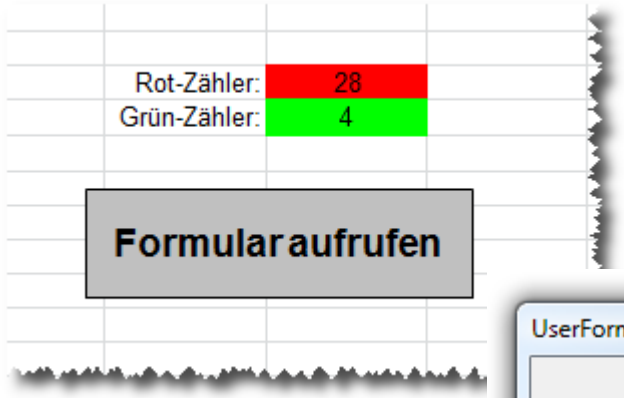
Modul4

(auch in T-Version)

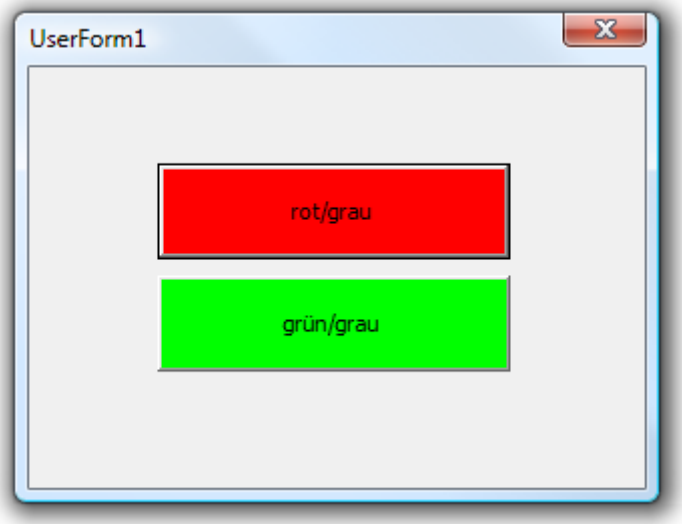
Datenaustausch mit Register

1. Erstelle ein Formular (=“UserForm1“) mit zwei Knöpfen deren Farbe beim Klicken wechselt
2. Das Formular soll vom Register mittels eines Rechtecks gestartet werden können.
3. Die Farbwechsel sollen im Register gezählt werden
4. Die „Zähl-Zellen“ sollen über Namenfelder angesprochen werden (z.B. „c_rot“)
   Die Farbe der Knöpfe mit “Knopfname.BackColor = RGB(255,0,0) “ setzen

Fehler finden



Userform1



Farbwerte

RGB-Wert	Farbe
RGB(0,0,255)	Blau
RGB(70,30,20)	Braun
RGB(255,255,0)	Gelb
RGB(80,80,80)	Grau
RGB(0,255,0)	Grün
RGB(150,0,255)	Lila
RGB(255,128,0)	Orange
RGB(255,0,255)	Rosa
RGB(255,0,0)	Rot
RGB(0,0,0)	Schwarz

Schleifen

…werden verwendet um Code-Zeilen zu wiederholen

!

for-Schleife

- Bei vorgegebener Anzahl an Wdh
- Step-Wert darf negativ sein

```
Sub dieForSchleife()
    Dim i As Integer

    For i = 1 To 10 Step 1 'Step=Schrittweite, darf negativ sein
        Cells(i, 1).Value = i
    Next i
End Sub
```

Modul3

(auch in T-Version)

for each-Schleife

- Jedes Element eines Datenfeldes wird durchlaufen
- Praktische Verwendung bei Zellbereichen

```
Sub dieForEachSchleife()
    Dim objCell As Range

    For Each objCell In Range("B3:B5")
        MsgBox objCell.Value
    Next objCell
End Sub
```

Schleifen

…werden verwendet um Code-Zeilen zu wiederholen

!

While-Schleife

- Solange eine Bedingung wahr ist
- Bedingung muss irgendwann falsch sein, sonst → Endlosschleife

```
Sub dieWhileSchleife()
    Dim i As Integer

    i = 1
    While i <= 10
        Cells(i, 3).Value = i
        i = i + 1 'Zähler inkrementieren
    Wend
End Sub
```

Do Loop-Schleife

- Solange (while) eine Bedingung wahr ist

oder

- Bis (until) eine Bedingung wahr ist
- wird mind. einmal durchlaufen

```
Sub dieDoLoopSchleife()
    Dim i As Integer

    i = 1
    Do
        Cells(i, 3).Value = i
        i = i + 1 'Zähler inkrementieren
    Loop While (i <= 10)
End Sub
```

oder so:

```
Sub dieDoLoopSchleife()
    Dim i As Integer

    i = 1
    Do
        Cells(i, 3).Value = i
        i = i + 1 'Zähler inkrementieren
    Loop Until (i > 10)
```

Modul3

(auch in T-Version)

Verzweigungen

Die if-Abfrage ist eine **Programmverzweigung**

If …Then … Else

- Wenn eine Bedingung wahr ist,
  wird der "**Then**"-Block abgearbeitet
  ansonsten der "**Else**"-Block abgearbeitet

```
Sub dieIfAbfrage()
    Dim i As Integer

    i = InputBox("Bitte Zahl eingeben")

    If i < 0 Then
        MsgBox "Zahl ist < 0"
    Else   'else-Block ist optional
        MsgBox "Zahl ist >= 0"
    End If
End Sub
```

- Mit "**ElseIf**" können weitere Bedingungen abgefragt werden

```
Sub dieIfAbfrage_2()
    Dim i As Integer

    i = InputBox("Bitte Zahl eingeben")

    If i < 0 Then '1te Bedingung
        MsgBox "Zahl ist < 0"
    ElseIf i = 0 Then '2te Bedingung
        MsgBox "Zahl ist = 0"
    Else
        MsgBox "Zahl ist > 0"
    End If
End Sub
```

Modul3

(auch in T-Version)

Debuggen

Möglichkeiten nach Fehlern zu suchen :

- # F8 → Sub **im Einzelschritt** ausführen und in Funktion springen
- # Shift+F8 → Sub **im Einzelschritt** ausführen und nicht in Funktion springen
- # Strg+F8 → Sub bis zum Cursor ausführen
- # F9 → Haltepunkte setzen

```
'--Alle Felder des Arrays füllen
For i = 1 To UBound(arrStrings) 'ubound=letzter Index
    '--In Feld 7 hat anschliessend z.B. den Wert: "Inhalt: 7"
    arrStrings(i) = "Inhalt: " & CStr(i) 'cstr wandelt Zahl in Text
Next i                                   i = 1
```

Einzelschritt:

Aktive Zeile ist gelb hinterlegt.

Cursor auf Variable plazieren → zeigt Wert

Haltepunkt,

Programm

stoppt hier

```
For i = 1 To UBound(arrStrings) 'ubound=letzter Index
    '--In Feld 7 hat anschliessend z.B. den Wert: "Inhalt: 7"
    arrStrings(i) = "Inhalt: " & CStr(i) 'cstr wandelt Zahl in Text
Next i
```

Überwachungsausdrücke

Ausdruck	Wert	Typ	Kontext
i	3	Integer	Modul8.Bsp2FuerArrays

Im Einzelschritt:

Cursor auf Variable setzen,

mittels RM „**Überwachung hinzufügen**“

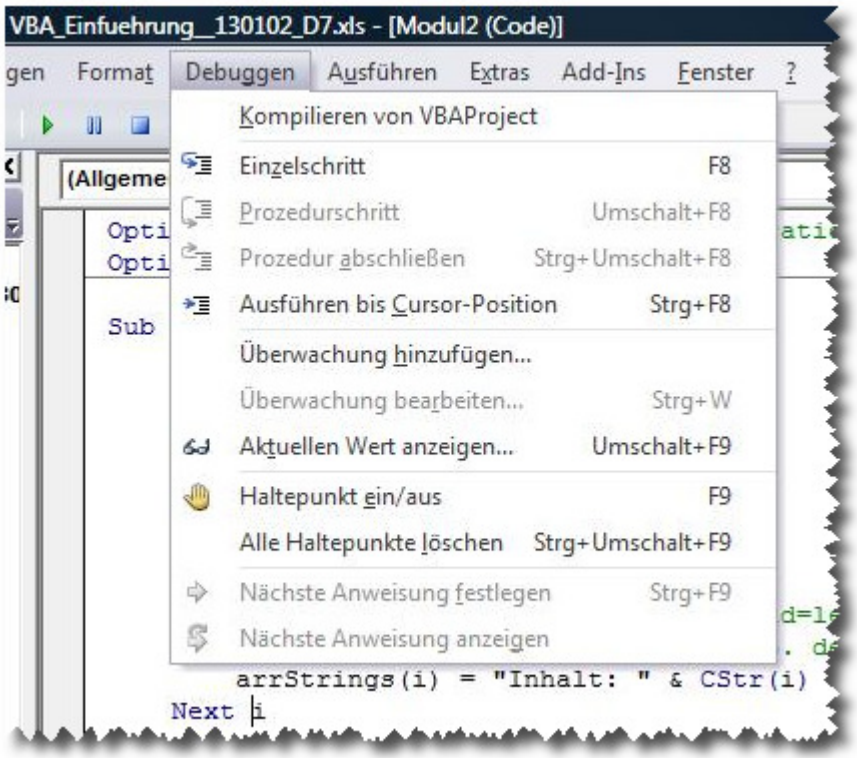
Debuggen

Möglichkeiten das Programm auf Fehler zu untersuchen

- # F8 → Sub **im Einzelschritt** ausführen und in Functions springen
- # Shift+F8 → Sub **im Einzelschritt** ausführen und nicht in Functions springen
- # Strg+F8 → Sub bis zum Cursor ausführen
- # F9 → Haltepunkte setzen

Im Menu-Punkt “Debuggen“
sind viele Überwachungsmöglichkeiten gelistet.

Hier können auch z.B. alle Haltepunkte gelöscht werden



Arrays

Arrays sind Feldvariablen

→ Speichern / lesen von vielen Daten **gleichen Typs**

→ Zugriff auf Daten über **Index**

→ Anzahl der Elemente kann dynamisch geändert werden

Arrays werden verwendet zur Verarbeitung von **umfangreichen gleichartigen** Daten

Arrays

Deklaration: **Dim** *Arrayname()* **As** *Datentyp*

```
Dim arrStrings() As String
```

Festlegen der Grösse:

Redim *Arrayname(Untergrenze* **to** *Obergrenze)*

→bestehende Werte werden **gelöscht**

```
ReDim arrStrings(1 To 10)
```

oder

Redim Preserve *Arrayname(Untergrenze* **to** *Obergrenze)*

→bestehende Werte werden **beibehalten**

```
ReDim Preserve arrStrings(1 To 15)
```

Arrays

```
Option Explicit  'erzwingt Variablendeklaration, empfohlen
Option Base 1    'Arrays beginnen bei 1

Sub BspFuerArrays()
    '--Array-Deklaration
    Dim arrStrings() As String
    Dim i As Integer

    '--Datenfeldgrösse festlegen
    ReDim arrStrings(10)

    '--Alle Felder des Arrays füllen
    For i = 1 To UBound(arrStrings) 'ubound=letzter Index
        '--In Feld 7 hat anschliessend z.B. den Wert: "Inhalt: 7"
        arrStrings(i) = "Inhalt: " & CStr(i) 'cstr wandelt Zahl in Text
    Next i

    '--Array erweitern, bestehende Elemente werden beibehalten.
    ReDim Preserve arrStrings(15) 'Nachteil: Befehl kostet Zeit !!

    '--Array löschen und mit 20 leeren Feldern anlegen
    ReDim arrStrings(20)

    '==> "redim" löscht und legt neu an,
    '==> "redim preserve" behält Bestehendes bei und fügt neue Felder hinzu
End Sub
```

Modul3

(auch in T-Version)

Strings

Strings (Datentyp)

→ sind Zeichenketten

→ werden durch **Anführungsstriche** kenntlich gemacht

→ können mittels…

…**&**-Operator verknüpft werden

…string-Funktionen verarbeitet werden

```
Dim sName As String
sName = "Max " & "Mustermann"
```

```
'-----------------------------------Stringbefehle-------------------------------------------------------------
'strx           = Right(strZeile, 5)                    '-- liefert die letzten fünf Zeichen
'strx           = Left(strZeile, 5)                     '-- liefert die ersten fünf Zeichen
'intPos1        = InStr(Startpos, strZeile, strSuch, 1) '-- liefert Pos von strSuch in strZeile
'strNeuerText   = Trim(strZeile)                        '-- entfernt führende und angehängte Leerzeichen
'strErgebnistext= Mid(strZeile, StartPos, [Länge])      '-- liefert den Text ab StartPos
'boolPruef      = IsNumeric("23.14")                    '-- 'true' wenn Text als Zahl auswertbar ist
'dblZ           = Val(" 1.1e-001 ")                     '-- liefert die Zahl 0.11
'intVgl         = StrComp(string1, string2, 1)          '-- Stringvergleich, liefert für string1="aaa" und
                                                        '-- string2="AAA" eine 0,
                                                        '-- was bedeutet daß die strings ohne Beachtung der
                                                        '-- Groß und Kleinschreibung gleich sind
'intAnzSlashes  = UBound(Split(strTemp, "\"))           '-- Anzahl des Vorkommens eines Zeichens
'-------------------------------------------------------------------------------------------------------------
```

Modul3

(auch in T-Version)

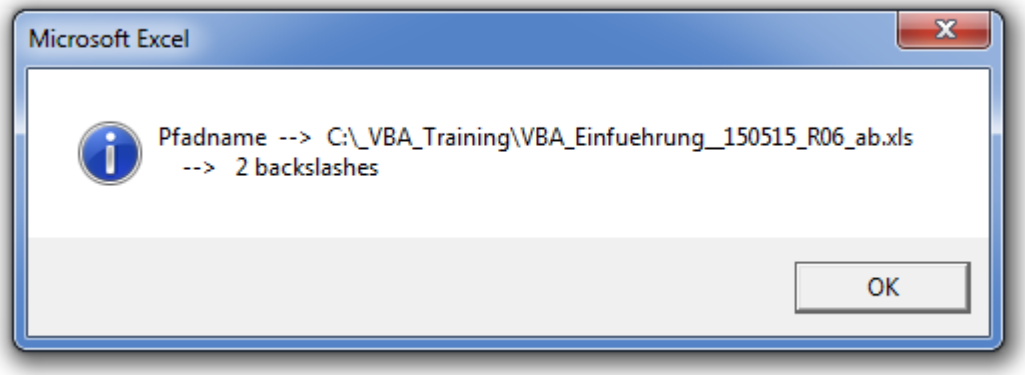
Übung

Erstelle eine Sub „AnzahlZeichenInString“ die folgendes macht

- Pfad + Mappenname ermitteln und in einem string merken
  → (Tipp: Siehe Folie ‚Excelobjekte‘)
- Anzahl des Zeichens „\“ ermitteln und ausgeben
  → (Tipp: Siehe Folie ‚Strings‘)

Modul3

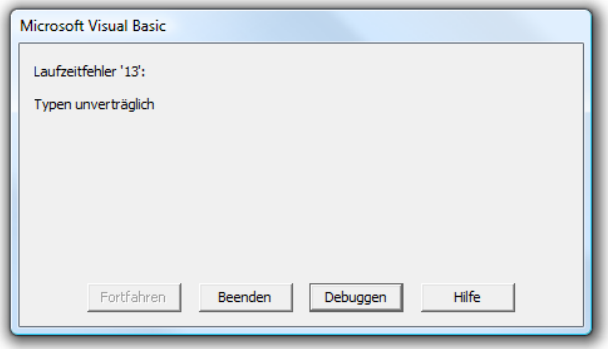
Ergebnis:



Fehlerbehandlung

Laufzeitfehler

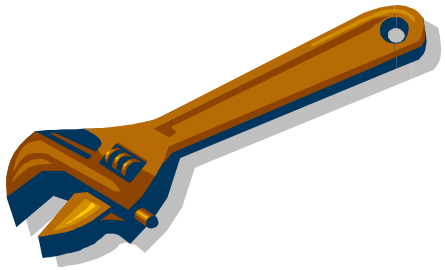
- verursachen eine Unterbrechung des Programms
- können mit “On Error“ abgefangen werden



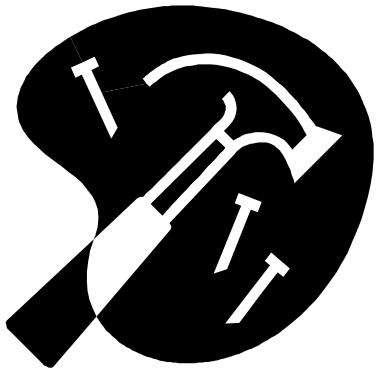

```
Sub BspFuerFehlerbehandlung()
    Dim i As Integer

    '--Fehlerbehandlungsroutine aktivieren
    On Error GoTo errorHandler

    '--FEHLER, Datentypen unterschiedlich
    i = "nix"

    '--Fehlerbehandlungsroutine DEaktivieren
    On Error GoTo 0

    MsgBox "Ok, durchgelaufen"

    '--Sub beenden, wenn kein Fehler auftrat
    Exit Sub

'--Sprungmarke
errorHandler: '<--Hier wird bei einem Fehler hingesprungen

    '--Fehlerbeschreibung ausgeben
    MsgBox "FehlerNr: " & Err.Number & ", " & Err.Description

End Sub
```

Modul3_b

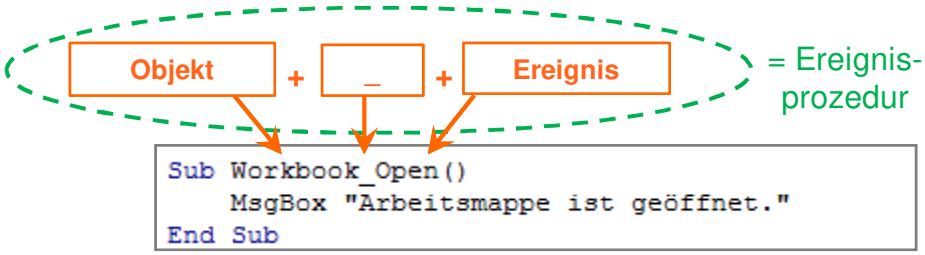
(auch in T-Version)



Nützliches…



Ereignisse



…können in Excel VBA-Code aktivieren

Beispiele für Ereignisse

Öffnen / Schliessen / Speichern einer Arbeitsmappe

Aktivieren / Deaktivieren eines Tabellenblatts (Register)

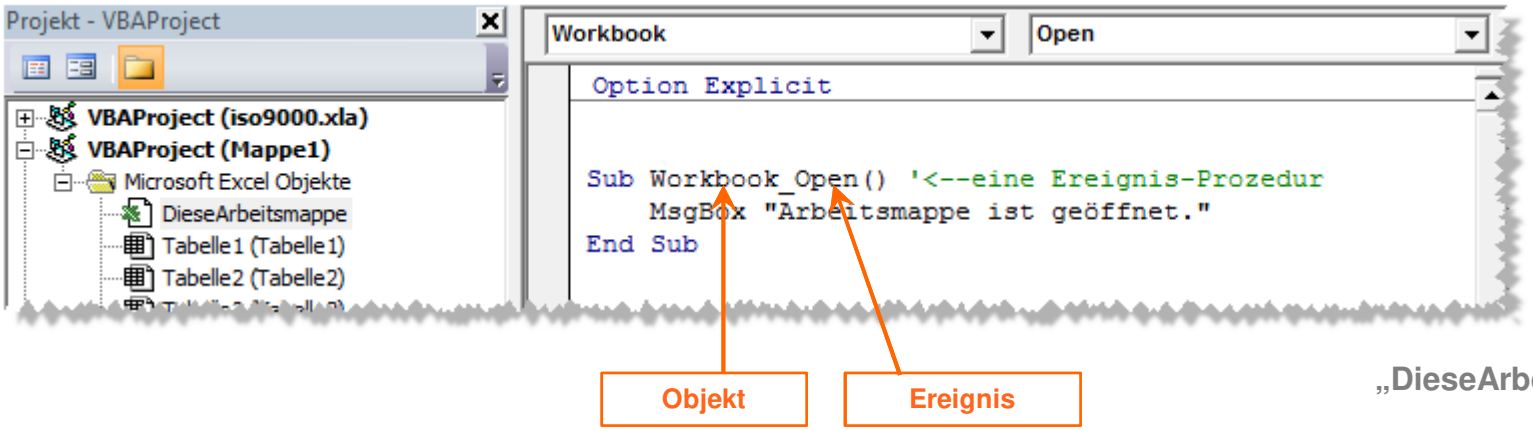
Selektieren einer Zelle

Für die Objekte „Workbook“ und „Worksheet“ können Ereignisprozeduren erstellt werden

→ Dafür im Projektfenster „**DieseArbeitsmappe**“ bzw „**TabelleX**“ auswählen

→ und per Listenfelder das Objekt und das Ereignis auswählen

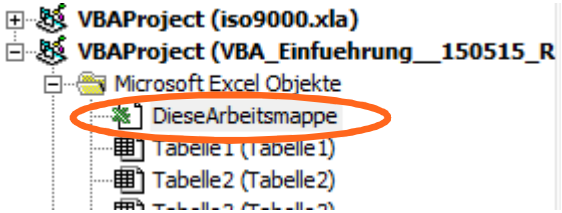
Eine Codeschablone wird erstellt, hier den gewünschten Code eingeben



„DieseArbeitsmappe“

Ereignisse

Bsp für Ereignisprozeduren des Workbooks




```
'--Wenn die Arbeitsmappe geschlossen wird
Sub Workbook_BeforeClose(Cancel As Boolean)
    MsgBox "Das Ereignis 'BeforeClose' wird ausgelöst. Aktuelle Uhrzeit: " & Time
End Sub
```

```
'--Wenn die Arbeitsmappe gespeichert wird
Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
    Worksheets(1).Select
    Range("I2") = Format(Date, "Long Date")
    Range("I3") = Format(Time, "Long Time")
    Range("I4") = Environ("USERNAME")
    MsgBox "Das Ereignis 'BeforeSave' wurde ausgelöst."
End Sub
```

Ü

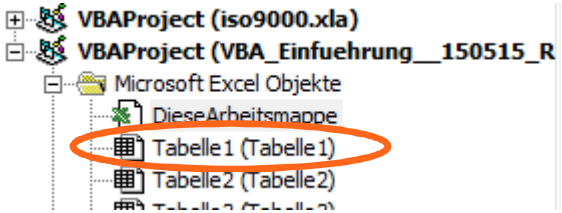
```
'--Wenn die Arbeitsmappe geöffnet wird
Sub Workbook_Open()
    MsgBox "Hallo " & Environ("USERNAME") & ", das Ereignis 'Workbook_Open' wurde ausgelöst."
End Sub
```

```
'--Wenn in dieser Arbeitsmappe das Register gewechselt wird.
Sub Workbook_SheetActivate(ByVal Sh As Object)
    MsgBox "Name des aktivierten Registers: " & Sh.Name
    MsgBox "Index des aktivierten Registers: " & Sh.Index
End Sub
```

„DieseArbeitsmappe“

Ereignisse

Bsp für Ereignisprozeduren des Worksheets



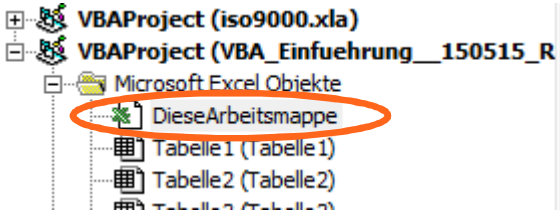

```
'--Wenn das Tabellenblatt gewechselt wird
Sub Worksheet_Activate()
    MsgBox "Ereignis Activate wurde für Register " & ActiveSheet.Name & " ausgelöst."
End Sub
```

```
'--Wenn in diesem Tabellenblatt eine Zelle angeklickt wird
Sub Worksheet_SelectionChange(ByVal Target As Range)
    Target.Interior.ColorIndex = 4 'die angeklickte Zelle wird grün
End Sub
```

```
'--Wenn mit der Rechten Maustaste geklickt wird
Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As Boolean)
    MsgBox "Klick mit RM"
End Sub
```

„Tabelle1“

Übung



Programmiere eine Ereignisprozedur für das Objekt „Workbook“

welche ausgelöst wird sobald das Tabellenblatt gewechselt wird.

In Zelle A18 des jeweiligen Tabellenblatts soll gezählt werden wie oft dieses Tabellenblatt bisher aufgerufen wurden.

Anschliessend soll stets Zelle A1 automatisch selektiert werden.

```
'--Wenn in dieser Arbeitsmappe das Register gewechselt wird.
Sub Workbook_SheetActivate(ByVal Sh As Object)
    Dim i As Integer '<--Zähler
    If Range("A18") = "" Then 'Zähler bisher wenn Zelle leer ist
        i = 0
    Else                      'Zähler bisher wenn Zelle nicht leer ist
        i = Range("A18").Value
    End If

    Range("A18").Value = i + 1  'Zähler inkrementieren und schreiben

    Range("A1").Select          'Zelle A1 selektieren
End Sub
```

„DieseArbeitsmappe“

Eigene Datentypen

```
Option Explicit '<-- erzwingt Variablen-Deklarationen
Option Base 1   '<-- arrays beginnen bei 1 statt bei 0

'--Eigenen Datentypen definieren
Type myDatentyp
    dblBreite As Double
    dblHoehe As Double
End Type

Sub eigeneDatentypen()
'*********************************************************
'*********** Bsp-Prozedur für eigene Datentypen ********
'*********************************************************

    '--Variable mit eigenem Datentypen deklarieren
    Dim myX As myDatentyp
    Dim dblFlaeche As Double

    '--Variable mit eigenem Datentypen initialisieren
    myX.dblBreite = 13.6
    myX. = 12.87
      dblBreite
      dblHoehe
    '--Flächenberechnung
    dblFlaeche = myX.dblBreite * myX.dblHoehe

End Sub
```

Eigene Datentypen
können zu Beginn eines Moduls mittels “type“ definiert werden.

Auf die Komponenten
kann mittels Punkt zugegriffen werden

Modul2_c
(auch in T-Version)

Datenaustausch mit txt-Datei

Text-Datei lesen

```
'--------------------------------txt lesen----------start
'-- txt-file öffnen
Open "c:\temp\test.txt" For Input As #50

'--Solange das Dateiende nicht erreicht ist...
Do While Not EOF(50)
    Line Input #50, sRow '...wird die akt Zeile gelesen
    arrSplittedRow = Split(sRow, Chr(9)) 'und gesplittet
Loop
Close 50
'--------------------------------txt lesen----------end
```

Modul4

(auch in T-Version)

Typische txt-Datei (Tab-getrennt)

Matrix.txt - Editor

Datei Bearbeiten Format Ansicht ?

lfNr	MNr	SipNr	Clutch	Cond	ChannelName	Tsump_s	Mue1	Mue2	Mue3	Mue5
1	175	1	1	1	chStepNr03_SIP01_K1_T40_l2s_loop00	40	0.2685	0.2658	0.2459	0.233
2	175	1	1	2	chStepNr03_SIP01_K1_T40_s2l_loop00	40	0.2607	0.2429	0.2107	0.203
3	175	2	1	1	chStepNr03_SIP02_K1_T40_l2s_loop00	40	0.1993	0.1995	0.1891	0.1845
4	175	2	1	2	chStepNr03_SIP02_K1_T40_s2l_loop00	40	0.1937	0.1846	0.1704	0.1664
5	175	3	1	1	chStepNr03_SIP03_K1_T40_l2s_loop00	40	0.1671	0.1675	0.168	0.1676
6	175	3	1	2	chStepNr03_SIP03_K1_T40_s2l_loop00	40	0.1641	0.1602	0.1539	0.1518
7	175	4	1	1	chStepNr03_SIP04_K1_T40_l2s_loop00	40	0.1583	0.1614	0.1624	0.1607
8	175	4	1	2	chStepNr03_SIP04_K1_T40_s2l_loop00	40	0.1557	0.1522	0.147	0.1446
9	175	5	1	1	chStepNr03_SIP05_K1_T40_l2s_loop00	40	0.1539	0.1568	0.1575	0.156
10	175	5	1	2	chStepNr03_SIP05_K1_T40_s2l_loop00	40	0.1513	0.1481	0.1431	0.1405
11	175	6	1	1	chStepNr03_SIP06_K1_T40_l2s_loop00	40	0.1471	0.1489	0.1478	0.144
12	175	6	1	2	chStepNr03_SIP06_K1_T40_s2l_loop00	40	0.145	0.1432	0.1377	0.1347
13	175	7	1	1	chStepNr03_SIP07_K1_T40_l2s_loop00	40	0.1404	0.1442	0.1443	0.1429
14	175	7	1	2	chStepNr03_SIP07_K1_T40_s2l_loop00	40	0.1374	0.1361	0.1315	0.1285
15	175	8	1	1	chStepNr03_SIP08_K1_T40_l2s_loop00	40	0.1458	0.1506	0.1557	0.1556
16	175	8	1	2	chStepNr03_SIP08_K1_T40_s2l_loop00	40	0.1426	0.141	0.1384	0.1363

Datenaustausch mit txt-Datei

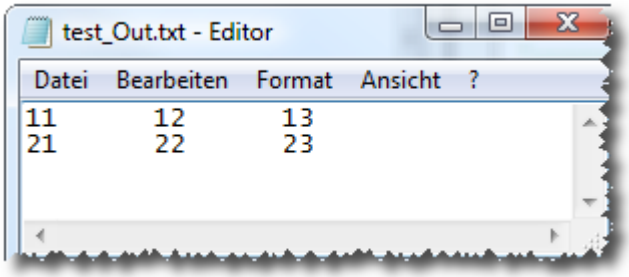
Text-Datei schreiben

```
'--------------------------------txt schreiben------start
sString1 = "11" & Chr(9) & "12" & Chr(9) & "13" 'chr(9) = Tabzeichen
sString2 = "21" & Chr(9) & "22" & Chr(9) & "23"

Open "c:\temp\test_Out.txt" For Output As #51
Print #51, sString1
Print #51, sString2
Close 51
'--------------------------------txt schreiben------end
```

chr(9) = Tab-Zeichen

Modul4
(auch in T-Version)

Ergebnis



Dateien in Pfad

Alle Dateinamen in Pfad ermitteln

```
Sub AlleDateinamenInPfad_Bsp()
'--Im angegebenen Pfad werden alle Dateinamen ermittelt
'--die der Dir-Abfrage entsprechen und im array gemerkt
Dim sPfad As String
Dim i As Long

sPfad = "C:\temp"

ReDim arrFileNames(1 To 1000)

'---Alle Dateinamen im Arbeitspfad ermitteln
i = 0
sDatei = Dir(sPfad & "*.txt")
Do While sDatei <> ""
    i = i + 1
    arrFileNames(i) = sDatei
    sDatei = Dir() 'Dir()=Nächster Dateiname
Loop
ReDim Preserve arrFileNames(1 To i)

End Sub
```

Modul5

(auch in T-Version)

Split

Die Split-Funktion

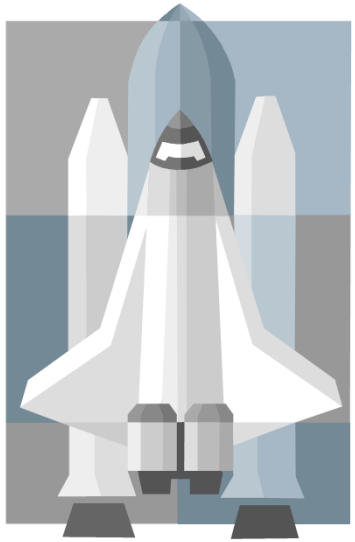
```
Sub Split_Bsp()
'--String der durch Tabs unterteilt ist in array aufsplitten
'--Hilfreich beim Lesen aus txt-file
Dim arrText() As String
Dim arrSplittedString() As String
Dim sEinString As String

'--String erzeugen der durch Tabs unterteilt ist
sEinString = "77" & Chr(9) & "88" & Chr(9) & "99" 'chr(9) = Tabzeichen
'--nun lautet der string: "77   88   99"

'--string wird an den Tab-Stellen "zerschnitten"
arrSplittedString = Split(sEinString, Chr(9))
'--Ergebnis:
'arrSplittedStrings(0) = "77"
'arrSplittedStrings(1) = "88"
'arrSplittedStrings(2) = "99"

End Sub
```

Modul5

(auch in T-Version)

Achtung, “Split“ gibt IMMER ein 0-basiertes array zurück

Vielen Dank für‘s Mitmachen und viel Spaß mit VBA !!!



Buch-Empfehlung