Visual Basic for Applications als Einstieg in das Programmieren [ARGE AINF HTL NÖ]

LV-Nummer 351F4WWJ01

HTBLuVA Wiener Neustadt

28. November 2014

Prof. Mag. Martin Schilk

INHALTSVERZEICHNIS

EINLEITUNG

Wieso Visual Basic for Applications?

Visual Basic for Applications (kurz VBA) ist eine Skriptsprache, die es erlaubt kleinere, auf Microsoft-Office-Anwendungen (Excel, Word, Access …) basierende Programme zu erstellen.

In der betrieblichen Praxis wird VBA häufig verwendet, um beispielsweise in Excel kleinere IT-Lösungen rasch zu realisieren. Im Unterrichtsbereich erlaubt VBA einen unkomplizierten Einstieg in die Grundlagen des Programmierens.

Vorteile von VBA:

- Mit Microsoft Office bereits mitgeliefert.
- Vergleichsweise einfach zu erlernen. Wenig „Überbau“ erleichtert die Fehlersuche.
- Programmierung von Funktionen und Makros erhöht den Nutzwert der Basisanwendung.
- Vielzahl von Literatur, Foren und Programmsammlungen.

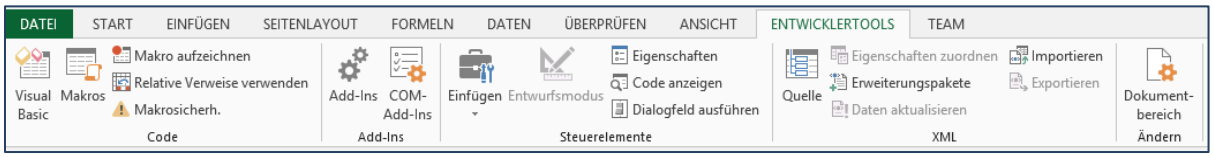
Erste Schritte

In diesem Skriptum wird sofern nicht anders erwähnt Microsoft Excel 2013 als Basisanwendung herangezogen.

Eine umfassende englischsprachige Dokumentation der Objekte, Funktionen und Befehle von VBA findet man unter ***Concepts (Excel 2013 developer reference)*** auf den Webseiten von Microsoft msdn.microsoft.com/en-us/library/office/jj733879(v=office.15).aspx.

Einstellungen in der Basisanwendung:

DATEI > Optionen > Menüband anpassen > Kontrollkästchen ***Entwicklertools*** anhaken, um dieses Menü anzuzeigen:

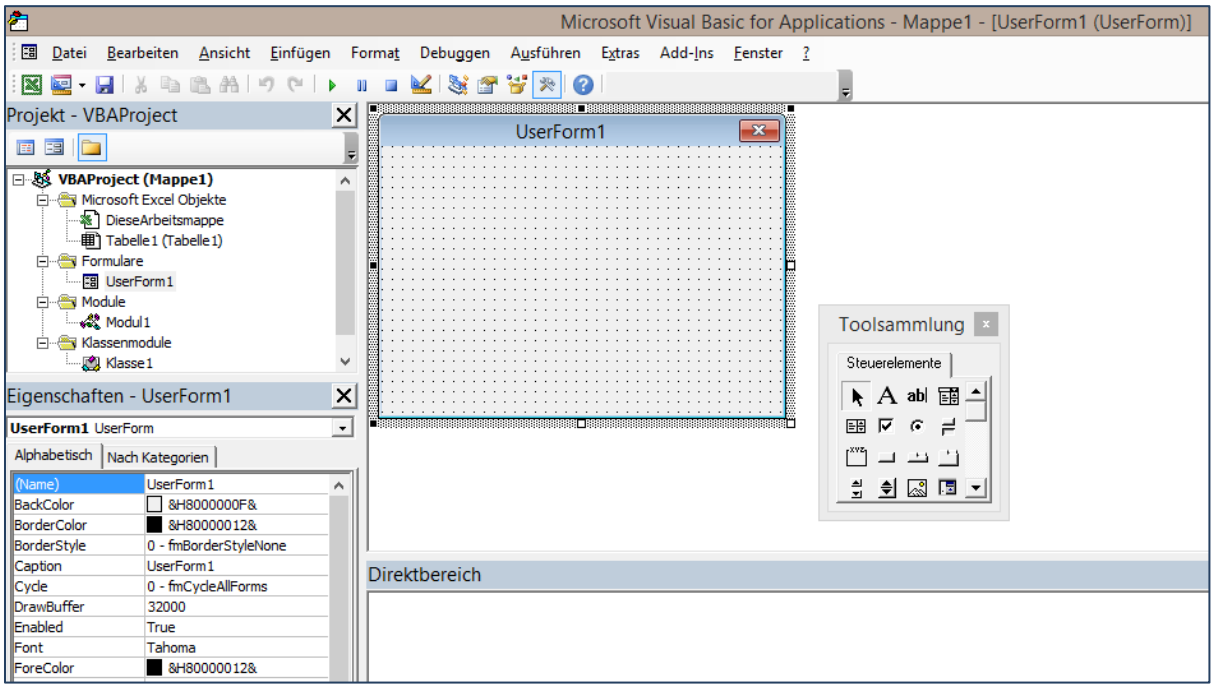


Menü ***ENTWICKLERTOOLS > Makrosicherheit***: Option ***Alle Makros aktivieren*** auswählen.

WICHTIG:	Eine Excel-Arbeitsmappe, die einen VBA-Code enthält, muss unbedingt mit ***DATEI > Speichern unter…*** Dateityp: ***Excel-Arbeitsmappe mit Makros (*.xlsm)*** gespeichert werden!

Entwicklungsumgebung

Einblenden über das Menü ***ENTWICKLERTOOLS > Visual Basic*** oder mit ***Alt + F11***.



Ansicht > Projekt-Explorer, ***Ansicht > Eigenschaftenfenster (F4)***
Ansicht > Direktfenster (erlaubt direkte Kontrollausgaben mit dem Befehl ***Debug.Print***)

In einem **Benutzerdefinierten *Formular*** („eigenes Fenster“, Erstellung mit ***Einfügen > UserForm***) werden ***Formularsteuerelemente*** aus dem Werkzeugkasten („Toolsammlung“) platziert. Um den VBA-Code eines Steuerelementes zu sehen, reicht ein Doppelklick auf das jeweilige Element oder man wählt ***Ansicht > Code (F7)***. Mit ***Ansicht > Objekt (⇧+ F7)*** gelangt man wieder zurück zur Formularansicht.

Formularsteuerelemente (***ENTWICKLERTOOLS > Einfügen > Formularsteuerelemente***) dienen auch dazu, um auf einem Excel-Arbeitsblatt direkt auf Zellendaten zu verweisen und mit diesen ohne Verwendung eines VBA-Codes zu interagieren. Mit ***Rechtsklick > Steuerelement formatieren…*** kann z.B. die gewünschte ***Zellverknüpfung*** festgelegt werden.

ActiveX-Steuerelemente (***Entwicklertools > Einfügen > ActiveX-Steuerelemente***) sind Objekte auf einem Excel-Arbeitsblatt, die zum Steuern von komplexeren Ereignissen dienen. Die Eigenschaften von ActiveX-Steuerelementen lassen sich auch während der Programmausführung verändern. Der VBA-Programmcode ist in der Entwicklungsumgebung unter ***Tabelle_*** ersichtlich.

Module sind Container für VBA-Routinen. In einem Modul (Erstellung mit ***Einfügen > Modul***) wird der Code von ***Funktionen*** („Function“) und ***Prozeduren*** („Makro“ - „Sub“) abgelegt bzw. bearbeitet. Bei der ***Aufzeichnung eines Makros*** wird automatisch ein Modul erzeugt, in dem der entsprechende VBA-Code betrachtet und weiterbearbeitet werden kann.

GRUNDLAGEN

Variablendeklaration & Variablentypen

Eine Variable ist ein „Container“ für einen Wert. Dieser Wert kann z.B. eine ganze Zahl, eine Kommazahl oder ein Text sein. Eine wie in anderen Programmiersprachen obligate Variablendeklaration ist in VBA zwar nicht zwingend notwendig, sollte aber trotzdem vorgenommen werden (Stichworte: Fehlersuche, guter Programmierstil, Übersichtlichkeit).

Mit der Anweisung ***Option Explicit*** (erscheint automatisch bei der Voreinstellung ***Extras > Optionen > Variablendeklaration erforderlich***) erzwingt man in VBA die Deklaration jeder benutzten Variablen.

Typ	kurz	Dez.	
Boolean			True oder False
Byte			0 ... +255
Integer	%		-32.768 ... +32.767
Long	&		-2.147.483.648 ... +2.147.483.647
Currency	@	32	-922.337.203.685.477,5808 ... +922.337.203.685.477,5807
Single	!	8	±3,402823E38 ... ±1,401298E-45
Double	#	16	-1.79769313486231E308 … -4,94065645841247E-324 (neg.) 4,94065645841247E-324 … 1,79769313486232E308 (pos.)
Date			Datum und Zeit
String	**$**		Zeichenkette, Text
Object			Objekte
Variant			alle Typen (Voreinstellung)

Beispiel: ***Dim X As Double*** oder kurz ***Dim X#***

Konstante werden mit ***Const*** deklariert. Beispiel: ***Const Pi = 3.1415926535897932***

Ein ***Array*** oder „Datenfeld“ ist eine ein Container für mehrere Werte. Einem Arraynamen folgt ein Klammerpaar mit einem ***Index*** (die Zählung beginnt standardmäßig bei 0). Bei mehrdimensionalen Arrays werden die Indizes durch Beistriche getrennt.

Beispiel: ***Dim X(3) As Integer*** … Ganzzahlige Variablen X(0), X(1), X(2), X(3)

Beispiel: ***Dim Y(1 To 4) As Integer*** … Ganzzahlige Variablen Y(1), Y(2), Y(3), Y(4)

Beispiel: ***Dim Z(1,1) As Integer*** … Ganzzahlige Variablen Z(0,0), Z(0,1), Z(1,0), Z(1,1)

Lebensdauer von Variablen: Soll eine Variable auch in allen anderen Modulen zur Verfügung stehen, so muss sie als ***Public*** definiert werden. Wird eine Variable mit ***Dim*** deklariert, so steht sie nur innerhalb des aktuellen Moduls zur Verfügung (entspricht ***Private***). Wird eine Variable innerhalb einer Prozedur (**Sub**) oder einer Funktion (***Function***) deklariert, so steht sie nur innerhalb dieser Prozedur bzw. Funktion zur Verfügung.

Operatoren

Mathematische Operatoren: ***+***, ***-***, *******, ***/***, ***^*** (hoch), ***Mod*** (modulo, ganzzahlige Rest)

Verknüpfung von Zeichenketten: **&**

Vergleichsoperatoren: ***=***, ***<***, ***<=***, ***>***, ***>=***, ***<>*** (ungleich)

Logische Operatoren: **Not**, **And**, **Or**, **Xor** (exklusives oder, “oder - aber nicht beide”)

Mathematische Funktionen in Visual Basic for Applications

msdn.microsoft.com/en-us/library/office/jj692811(v=office.15).aspx

Abs(x) .....(Absolut)Betrag einer Zahl x

Atn(x).....Arkustangens von x (Ergebnis im Bogenmaß)

Cos(x).....Kosinus eines Winkels x (im Bogenmaß)

Exp(x).....Natürliche Exponentialfunktion e^x (Basis Eulersche Zahl e = 2,71828...)

Int(x).....Ganzzahliger Anteil einer Zahl Int(-3.4) = -4

Fix(x).....Ganzzahliger Anteil einer Zahl Fix(-3.4) = -3

Log(x).....Natürlicher Logarithmus einer Zahl … ln(x), Basis e

Rnd .....Zufallszahl zwischen 0 und 1, zuvor: **Randomize**

Round(x,n).....Zahl x auf n Stellen runden

Sin(x).....Sinus eines Winkels x (im Bogenmaß)

Sqr(x) .....Quadratwurzel einer Zahl x

Tan(x) .....Tangens eines Winkels x (im Bogenmaß)

Umrechnungsformeln Grad (Degree) – Bogenmaß (Radian): $\alpha° = \frac{\alpha_{rad} \cdot 180°}{\pi}$, $\alpha_{rad} = \frac{\alpha° \cdot \pi}{180°}$

Arkussinus: $asn(x) = Atn\left(\frac{x}{\sqrt{1-x^2}}\right)$ $für$ $-1 < x < 1$, $asn(1) = \frac{\pi}{2}$, $asn(-1) = -\frac{\pi}{2}$

Arkuskosinus: $acs(x) = Atn\left(\frac{-x}{\sqrt{1-x^2}}\right) + 2 \cdot Atn(1), -1 < x < 1$, $acs(1) = 0$, $acs(-1) = \pi$

Mathematische “Worksheet Functions”

msdn.microsoft.com/en-us/library/office/ff836235(v=office.15).aspx

Application.WorksheetFunction.Acos(x)..... Arkuskosinus von x (Ergebnis im Bogenmaß)

Application.WorksheetFunction.Asin(x)..... Arkussinus von x (Ergebnis im Bogenmaß)

Application.WorksheetFunction.Fact(n)..... Fakultät n!

Application.WorksheetFunction.Gcd(m,n)... größter gemeinsamer Teiler von m, n, …

Application.WorksheetFunction.Lcm(m,n)... kleinstes gemeinsames Vielfaches von m, n, …

Application.WorksheetFunction.Log10(x).... Dekadischer Log. einer Zahl … lg(x), Basis 10

Application.WorksheetFunction.Pi..... π = 3,14159265…

Verzweigungen

```
If Bedingung1 Then
        'Anweisungen
ElseIf Bedingung2 Then
        'Anweisungen
Else
        'Anweisungen
End If
```

```
Select Case Variable
Case Variablenwert1
        'Anweisungen
Case Variablenwert2, Variablenwert3, Variablenwert4
        'Anweisungen
Case Else
        'Anweisungen
End Select
```

Schleifen

Zählschleife:

```
For i = Anfang To Ende Step Schrittweite
        'Anweisungen
Next i
```

Ausstiegsmöglichkeit: `If Bedingung Then Exit For`

Schleife mit Bedingung am Anfang:

```
Do While [oder Do Until] Bedingung
        'Anweisungen
Loop
```

Ausstiegsmöglichkeit: `If Bedingung Then Exit Do`

Schleife mit Bedingung am Ende:

```
Do
        'Anweisungen
Loop While [oder Loop Until] Bedingung
```

Ausstiegsmöglichkeit: `If Bedingung Then Exit Do`

PROGRAMMIERUNG VON FUNKTIONEN

Die Programmierung eigener Funktionen dient zur Erweiterung der in Excel integrierten Standardfunktionen.

Funktionen werden in ein Modul geschrieben (VBA-Entwicklungsumgebung Menü: ***Einfügen > Modul***), sind von einem oder mehreren Variablen abhängig und liefern genau einen Rückgabewert (Zahl oder Text).

Beispiel 1: Volumen eines Drehkegels

gegeben: *Radius r, Höhe h*
gesucht: *Volumen V* $V = \frac{\pi}{3} \cdot r^2 \cdot h$
Bedingungen: *r > 0, h > 0*
Fkt. DREHKEGELVOLUMEN(r,h): *wenn r <= 0 dann Rückgabe Text1*
sonst wenn h <= 0 dann Rückgabe Text2
sonst (wenn alle Bed. erfüllt): Berechnung & Rückgabe Zahl

```
Option Explicit

Function DREHKEGELVOLUMEN(R As Double, H As Double) As Variant
    Dim Pi, V As Double
    If R <= 0 Then
        DREHKEGELVOLUMEN = "#Radius unzulässig!"   'Rückgabe Text
    ElseIf H <= 0 Then
        DREHKEGELVOLUMEN = "#Höhe unzulässig!"     'Rückgabe Text
    Else
        Pi = 4 * Atn(1)
        V = Pi / 3 * R ^ 2 * H                        'Berechnung
        DREHKEGELVOLUMEN = V                        'Rückgabe Zahl
    End If
End Function
```

Wurde eine Funktion mit VBA in einem Modul erstellt, steht sie in der gesamten Excel-Arbeitsmappe unmittelbar zur Verfügung. Die Funktion kann (beginnend mit **=**) direkt in eine Zelle eingetippt werden oder mittels Funktionsassistenten (***f_x*** in der Bearbeitungsleiste, ***Kategorie: Benutzerdefiniert***) aufgerufen werden:

In der VBA-Entwicklungsumgebung können alle in einem Modul erstellten Funktionen mit dem Menü ***Datei > Datei exportieren…*** in eine ***Basicdatei *.bas*** übertragen werden. Die Funktionen lassen sich bei Bedarf ebenso bequem wieder in einer anderen Excel-Arbeitsmappe einfügen und anwenden (***Datei > Datei importieren…***).

Speichert man die Datei unter dem Dateityp: ***Excel-Add-In (*.xlam)*** und aktiviert im Menü ***DATEI > Optionen > Add-Ins > Gehe zu…*** diese Add-In-Datei, so stehen die selbst erstellten Funktionen in Zukunft in jeder Excel-Arbeitsmappe auf diesem PC zur Verfügung.

PROGRAMMIERUNG VON PROZEDUREN

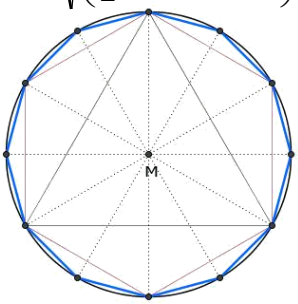
Prozeduren können einerseits erzeugt werden, indem man sich wiederholende Befehlsabläufe als Makros aufzeichnet (***ENTWICKLERTOOLS > Makro aufzeichnen***).

Andererseits lässt sich eine Prozedur („Sub“) in ein Modul (VBA-Entwicklungsumgebung Menü: ***Einfügen > Modul***) auch zur Gänze selbst programmieren.

Beispiel 2: Näherungsberechnung von π

Ausgehend von einem n = 6-Eck mit der Seitenlänge s_n = 0,5 und dem Umfang (n*s_n) = 3 nähert sich der Umfang durch Verdoppelung der Ecken rasch an die Kreiszahl π an.

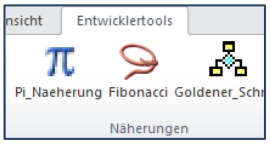
$$s_{2n} = \sqrt{\left(\frac{1}{2} \cdot \left(1 - \sqrt{1 - s_n^2}\right)\right)}$$




```
Sub Pi_Naeherung()
    Dim N As Long
    Dim SN, Pi_N As Double
    Dim I As Byte

    Cells(1, 1).Value = "n-Eck"
    Cells(1, 2).Value = "Seitenlänge s"
    Cells(1, 3).Value = "Umfang"
    Cells(1, 4).Value = "Differenz zu Pi"
    N = 6
    SN = 0.5

    For I = 1 To 14
        Pi_N = N * SN
        'Debug.Print I, N, Pi_N
        Cells(I + 1, 1).Value = N
        Cells(I + 1, 2).Value = SN
        Cells(I + 1, 3).Value = Pi_N
        Cells(I + 1, 4).Value = 4 * Atn(1) - Pi_N
        N = 2 * N
        SN = Sqr(1 / 2 * (1 - Sqr(1 - SN ^ 2)))
    Next I
End Sub
```

Eine erstellte Prozedur („Makro“ - „Sub“) lässt sich in Excel über ***ENTWICKLERTOOLS > Makros > Ausführen*** starten.

Über ***DATEI > Optionen > Menüband anpassen >*** Befehle auswählen: ***Makros*** > ***Hinzufügen*** (ev. ***Umbenennen…***) lassen sich Prozeduren über Symbole rasch starten. Es empfiehlt sich, davor eine neue Gruppe im Hauptregister „Entwicklertools“ anzulegen:



Über ***DATEI > Optionen > Symbolleiste für den Schnellzugriff >*** Befehle auswählen: ***Makros*** > ***Hinzufügen*** (ev. ***Ändern…***) lässt sich eine Prozedur auch bequem über ein Symbol in der Schnellzugriffsleiste starten.

Wird mit ***ENTWICKLERTOOLS > Einfügen*** (Toolbox) ***> Schaltfläche*** (Formularsteuerelement) eine Befehlsschaltfläche auf einem Arbeitsblatt gezeichnet, so kann über diese eine Prozedur nach ***Rechtsklick > Makro zuweisen…*** ebenfalls gestartet werden.

Beispiel 3: Zellbereiche durchlaufen: **Range(" : ") – *Cells(,)* – *Selection***:

```
Option Explicit                'Zellbereich mit Zufallszahlen befüllen

Sub Bereich_1()                'Bereich "A2:J11" am Blatt "Tabelle1"
    Dim Zelle As Object
    Tabelle1.[A1].Value = "100 Zufallszahlen:"
    For Each Zelle In Worksheets("Tabelle1").Range("A2:J11").Cells
      Zelle.Value = Rnd()
    Next Zelle
End Sub

Sub Bereich_2()                                 'aktives Blatt
    Dim I, J As Byte                            'Zellenindizes I und J
    Cells(1, 1).Value = "100 Zufallszahlen:"
    For I = 2 To 11                             'Zeilenindex
      For J = 1 To 10                           'Spaltenindex
        Cells(I, J).Value = Rnd()
      Next J
    Next I
End Sub

Sub Bereich_3()                                 'markierter Bereich
    Dim Zelle As Object
    For Each Zelle In Selection
      Zelle.Value = Rnd()
    Next Zelle
End Sub
```

zu Beispiel 3: Berechnungen von Bereichen mit Formeln oder Worksheet-Functions

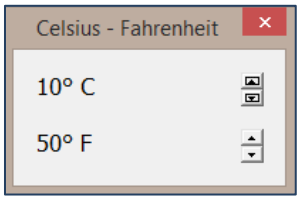
```
Sub Mittelwert_in_D1()  'Formel in D1 am Blatt "Tabelle1" schreiben
    Dim Formel As String
    Formel = "=MITTELWERT(A2:J11)"
    Worksheets("Tabelle1").Range("D1").FormulaLocal = Formel
    'oder: .Formula mit englischem Funktionsnamen "=AVERAGE(A2:J11)"
End Sub

Sub Mittelwert_der_Auswahl()
    Dim M As Double
    M = Application.WorksheetFunction.Average(Selection)
    MsgBox M, vbOKOnly, "Der Mittelwert ist:"      'Meldungsfenster
End Sub
```

So wie Funktionen können auch Prozeduren in eine **Basicdatei *.bas** exportiert oder als **Excel-Add-In** gespeichert werden (siehe Seite 7 unten).

Programmierung von Benutzerdefinierten Formularen

VBA-Entwicklungsumgebung Menü: ***Einfügen > UserForm***

Beispiel 4: Gradumwandlung Celsius – Fahrenheit



$$F = \frac{9}{5} \cdot C + 32 \quad C = \frac{5}{9} \cdot (F - 32)$$

1) Objekte (Steuerelemente) mit Hilfe der Werkzeugsammlung (engl. Toolbox) zeichnen und Eigenschaften festlegen – vgl. dazu auch Seite 11:

Objekt	Eigenschaft	Wert
UserForm (Formular / Fenster)	Name	frm_Temperatur
	Caption	Celsius – Fahrenheit
Label (Beschriftungsfeld)	Name	lbl_Celsius
	Caption	10° C
SpinButton (Drehfeld)	Name	spb_Celsius
	Max	1000
	Min	-273
	Value	10
Label (Beschriftungsfeld)	Name	lbl_Fahrenheit
	Caption	50° F
SpinButton (Drehfeld)	Name	spb_Fahrenheit
	Max	1832
	Min	-460
	Value	50

2) Ereignisse programmieren (***F7***, ***Ansicht > Code*** oder ***Doppelklick auf Steuerelement***):

```
Option Explicit

Private Sub spb_Celsius_Change()                        'EREIGNIS
    Dim C, F As Double
    lbl_Celsius.Caption = spb_Celsius.Value & " °C"
    C = spb_Celsius.Value                               'Einlesen
    F = 9 / 5 * C + 32                                  'Berechnen
    lbl_Fahrenheit.Caption = Round(F, 2) & " °F"        'Ausgeben
End Sub

Private Sub spb_Fahrenheit_Change()                     'EREIGNIS
    Dim C, F As Double
    lbl_Fahrenheit.Caption = spb_Fahrenheit.Value & " °F"
    F = spb_Fahrenheit.Value                            'Einlesen
    C = 5 / 9 * (F - 32)                                'Berechnen
    lbl_Celsius.Caption = Round(C, 2) & " °C"           'Ausgeben
End Sub
```

3) Prozedur zum Starten des Formulars (siehe auch Seite 8):

```
Sub Gradumrechnung_starten()
    frm_Temperatur.Show
End Sub
```

Analog zu Funktionen & Prozeduren können benutzerdefinierte Formulare in eine **Formular-datei *.frm** exportiert oder als **Excel-Add-In** gespeichert werden (siehe Seite 7 unten).

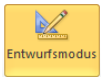
PROGRAMMIERUNG VON ACTIVEX-STEUERELEMENTEN

Formular- und ActiveX-Steuerelemente erfüllen sehr ähnlich Zwecke, wobei letztere eine wesentlich flexiblere Programmierung erlauben.

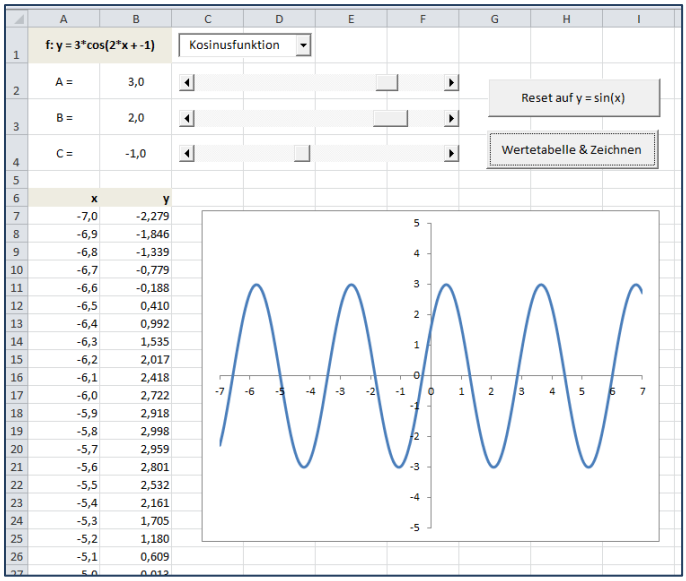
ActiveX-Steuerelemente werden nicht in benutzerdefinierte Formulare (UserForms), sondern **direkt auf Excel-Arbeitsblättern** eingefügt. Nach dem Einfügen legt man ihre Eigenschaften (wichtig: Name) fest. Nach einem Doppelklick werden für sie Ereignisse mit den gewünschten Befehlen in VBA programmiert.

Eine englischsprachige Beschreibung aller ActiveX-Steuerelemente findet sich unter msdn.microsoft.com/en-us/library/office/ff837582(v=office.15).aspx.

Objekt	Eigenschaften	Standardereignis	Methoden
Checkbox Kontrollkästchen	Caption **Value**: True / False	Click()	
ComboBox Kombinationsfeld	Text, Value	Change()	**.Additem** "Punkt"
Commandbutton Befehlsschaltfläche	Caption	Click()	
Image Bild	Picture PictureSizeMode	Click()	
Label Bezeichnung	Caption	Click()	
ListBox Listenfeld	Text, Value Multiselect	Click()	**.Additem** "Punkt"
OptionButton Optionsfeld	Caption Groupname **Value**: True / False	Click()	
ScrollBar Bildlaufleiste	Max, Min LargeChange **Value**	Change()	
SpinButton Drehfeld	Max, Min **Value**	Change()	
TextBox Textfeld	Text	Change()	
ToogleButton Umschaltfläche	Caption **Value**: True / False	Click()	

Zum Wechseln zwischen Anwendung bzw. Erstellung und Programmierung der ActiveX-Steuerelemente muss die Schaltfläche **ENTWICKLERTOOLS > Entwurfsmodus** deaktiviert bzw. aktiviert werden: 

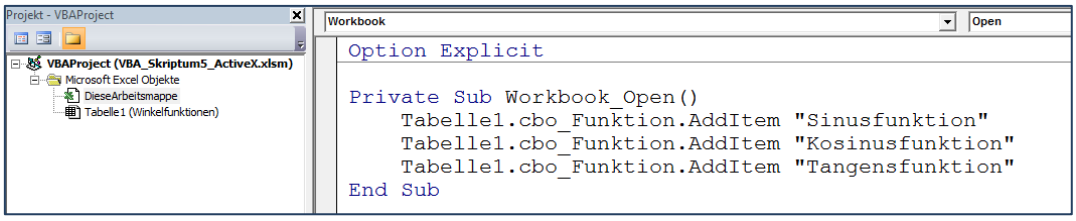
Beispiel 5: Winkelfunktionen



1) Objekte (ActiveX-Steuerelemente) mit Hilfe der Werkzeugsammlung (engl. Toolbox) auf dem Tabellenblatt „Tabelle1“ zeichnen und deren Eigenschaften festlegen:

Objekt	**Eigenschaft**	**Wert**
ComboBox (Kombinationsfeld)	Name	cbo_Funktion
	Text	Sinusfunktion
ScrollBar (Bildlaufleiste)	Name	scb_Amplitude
	Max	50
	Min	-50
	LargeChange	10
ScrollBar (Bildlaufleiste)	Name	scb_Frequenz
	Max	30
	Min	-30
	LargeChange	10
ScrollBar (Bildlaufleiste)	Name	scb_Phase
	Max	70
	Min	-70
	LargeChange	10
Commandbutton (Schaltfläche)	Name	cmb_Reset
	Caption	Reset auf y = sin(x)
Commandbutton (Schaltfläche)	Name	cmb_Zeichnen
	Caption	Wertetabelle & Zeichnen

2) Beim Öffnen der Excel-Arbeitsmappe das Kombinationsfeld auf Tabelle1 befüllen.
Unter „Diese Arbeitsmappe“: oben Objekt „Workbook“ und Ereignis „Open“ auswählen:



3) Unter „Tabelle1“ alle Ereignisse und eine eigene Prozedur programmieren:

```
Option Explicit

Dim Auswahl As Byte     'Variablendeklaration auf Tabelle1
Dim A, B, C As Double
Dim Formel As String
```

```
Private Sub cbo_Funktion_Change()
    Auswahl = cbo_Funktion.ListIndex        'beginnend bei 0
    Call Funktionserstellung                'Prozeduraufruf
End Sub
```

```
Private Sub scb_Amplitude_Change()
    [B2].Value = scb_Amplitude.Value / 10   '1 Dezimalstelle
    Call Funktionserstellung                'Prozeduraufruf
End Sub
```

```
Private Sub scb_Frequenz_Change()
    [B3].Value = scb_Frequenz.Value / 10
    Call Funktionserstellung                'Prozeduraufruf
End Sub
```

```
Private Sub scb_Phase_Change()
    [B4].Value = scb_Phase.Value / 10
    Call Funktionserstellung                'Prozeduraufruf
End Sub
```

```
Private Sub cmb_Reset_Click()
    cbo_Funktion.Value = "Sinusfunktion"
    scb_Amplitude.Value = 10
    scb_Frequenz.Value = 10
    scb_Phase.Value = 0
    Call Funktionserstellung                'Prozeduraufruf
End Sub
```

```
Private Sub cmb_Reset_Click()
    cbo_Funktion.Value = "Sinusfunktion"
    scb_Amplitude.Value = 10
    scb_Frequenz.Value = 10
    scb_Phase.Value = 0
    Call Funktionserstellung                'Prozeduraufruf
End Sub
```

```
Private Sub Funktionserstellung()           'eigene Prozedur (Unterprogramm)
    Range("A7:B147").Clear                      'löschen der Wertetabelle in A7:B147
    A = scb_Amplitude.Value / 10                'Einlesen
    B = scb_Frequenz.Value / 10
    C = scb_Phase.Value / 10
    Dim Vorzeichen_C As String
    If C >= 0 Then Vorzeichen_C = "+ " Else Vorzeichen_C = "- "
    Select Case Auswahl
    Case 0:
        [A1].Value = "f: y = " & A & "*sin(" & B & "*x " & Vorzeichen_C & Abs(C) & ")"
                        'Ausgabe der Winkelfunktion in der Zelle A1
        Formel = "=R2C2*SIN(R3C2*RC[-1]+R4C2)"
                        'Formel am Tabellenblatt mit absoluten und relative Bezügen
    Case 1:
        [A1].Value = "f: y = " & A & "*cos(" & B & "*x " & Vorzeichen_C & Abs(C) & ")"
        Formel = "=R2C2*COS(R3C2*RC[-1]+R4C2)"
    Case 2:
        [A1].Value = "f: y = " & A & "*tan(" & B & "*x " & Vorzeichen_C & Abs(C) & ")"
        Formel = "=R2C2*TAN(R3C2*RC[-1]+R4C2)"
    End Select
End Sub
```

```
Private Sub cmb_Zeichnen_Click()
    Dim I As Integer                    'Zeilenindex I
    I = 7                                   'Startwert für Wertetabellenzeile
    Dim X As Double                     'Variable X
    X = -7                                  'Startwert für die Wertetabelle
    Do Until X > 7
        Cells(I, 1).Value = X
        Cells(I, 1).NumberFormat = "0.0"
        Cells(I, 2).FormulaR1C1 = Formel
        Cells(I, 2).NumberFormat = "0.000"
                'um bei Tangens Bereiche nahe nicht definierter Stellen zu entfernen:
        If Auswahl = 2 And Abs((A * Tan(B * X + C))) > 5 Then
            Cells(I, 2).Value = ""
        End If
        I = I + 1                       'nächste Zeile in der Wertetabelle
        X = X + 0.1                     'Schrittweite für X: 0.1
    Loop
End Sub
```

DATEI > Optionen > Menüband anpassen: Entwicklertools

ENTWICKLERTOOLS > Makrosicherheit > Alle Makros aktivieren

ENTWICKLERTOOLS > Visual Basic (Alt + F11)

In der VBA-Entwicklungsumgebung: ***Einfügen > Modul***

Beispiel 6: Datumsberechnung ab aktuellem Datum & Ergebnis an Cursorposition einfügen

```
Option Explicit

Sub DatumPlus()
    Dim T As Integer
    On Error GoTo Fehler                        'Fehlerbehandlung
    T = InputBox("Tage ab heute:", "Eingabe", 14)
    Selection.InsertBefore Format((Date + T), "dd. MMMM yyyy")
    Exit Sub                                    'Ausstieg aus Prozedur
Fehler:
    MsgBox "Berechnung nicht ausgeführt!", vbCritical, "Abbruch"
End Sub
```

Anwenden: ***ENTWICKLERTOOLS*** **> Makros >** „DatumPlus“ **Ausführen**

DATEI > Speichern unter… Dateityp: **Word Dokument mit Makros (*.docm)**