Excel – VBA-Grundlagen

Allgemein	
Globale Variable:	PUBLIC variable[nliste]
Dimensionierung:	PUBLIC feld1(100), x(5) *oder* DIM feld1(100), x(5)
Makro:	SUB name( ) / END SUB
Aufruf Makro:	CALL name
Function:	FUNCTION name(argumente) / END FUNCTION
Aufruf Function:	variable = name(argumente)

Ein- / Ausgabe		
Makro ⇒ Tabelle:		[B12] = variable
	oder	CELLS(zeile%, spalte%) = variable
Tabelle ⇒ Makro:		variable = [B12]
	oder	variable = CELLS(zeile%, spalte%)

Operatoren

- Arithmetisch: **+ - * / ^ \ mod**
- Addition von Zeichenketten: **+ &**
- Vergleichsoperatoren: **= < > <= >= <>**
- Prüfung, ob zwei Objekte auf dasselbe Objekt verweisen: **is**
- Vergleich von Zeichenmustern: **like**
- Logische Operatoren: **and or not eqv**

Operatoren:

- Arithmetisch ^ * / + - \ (abgerundete Ganzzahldivision) mod (Rest einer Ganzzahldivision)
- Zeichenketten + & ("15" & "3" ergibt "153")
- Vergleichsoperatoren = < > <= >= <>
- **is** *(Wenn zwei Objekte auf dasselbe Objekt verweisen, ist das Ergebnis True; andernfalls ist Ergebnis False.)*
- **like** zum Vergleich von Zeichenmuster: "abc" like "a*"
- Logische Operatoren **and or not** (Implikation) **eqv** (Äquivalenz)

Numerische Datentypen	Sonstige Datentypen
• Byte • Integer } ganzzahlig • Long • Single • Double • Currency	• Boolean • String • Date • Object • Variant

Datentypen:

- **Byte** 0 ... 255
- **Integer** - 32.768 ... 32.767
- **Long** - 2.147.483.648 ... 2.147.483.647 +/- 2 Mio
- **Single** -3,4E38 .. 3,5E38 8 Stellen
- **Double** -1,8E308 ... 1,8E308 16 Stellen
- **Currency** Festkomma: 15 Stellen vor + 4 Stellen nach Komma
- **Boolean** True - False
- **String** < 65.400 Zeichen
- **Date** Datum und Uhrzeit mit versch. Formatierungen
- **Object** Adressen, die auf Objekte in der Anwendung verweisen können

Gültigkeitsbereiche von Variablen

Deklaration im Modulkopf	Bekanntheit der Variable	Lebensdauer
`Public Var As Byte`	in allen Modulen	bis zum
`Dim Var As Byte`	nur im aktuellen	Neustart des
`Private Var As Byte`	Modul	Moduls
Deklaration in Prozedur/Funktion		
`Dim Var As Byte`	nur in der aktuellen	bis zum Ende der Prozedur/Funktion
`Static Var As Byte`	Prozedur/Funktion	bis zum Neustart des Moduls

Variablen, die nur in einer Prozedur oder Funktion bekannt sein sollen, deklariert man innerhalb derselben mit dem Schlüsselwort DIM.

Globale Variablen, die in allen Prozeduren/Funktionen eines Moduls gültig sein sollen, **deklariert man im Deklarationsteil eines Moduls** (Abschnitt vor der 1. Prozedur/Funktion).

Modulübergreifende Variablen, die darüber hinaus in allen Modulen Verwendung finden, muss man mit dem Schlüsselwort **Public** deklarieren.

Nicht deklarierte Variablen sind standardmäßig **nur lokal gültig**.

Auf Modulebene deklarierte Variablen erhalten ihren Wert, bis das Modul zurückgesetzt bzw. neu gestartet wird.

Option: Ausführen/Zurücksetzen

In der Funktion/Prozedur deklarierte Variablen entsprechen nicht den auf Modulebene deklarierten globalen Variablen (auch bei Namensgleichheit).

Sie behalten ihren Wert nur bis zur Beendigung der Prozedur bzw. Funktion, wenn sie nicht STATIC sind - dann werden sie erst wieder beim Zurücksetzen des Moduls neu initialisiert.

Auswahlbefehl - vollständige Alternative

(siehe Beispiel 1)

```
If Bedingung1_erfuellt Then
      Anweisung1
  ElseIf Bedingung2_erfuellt Then
      Anweisung2
  Else
      Anweisung3
End If
```

If...Then...Else-Anweisung

Syntax

If *Bedingung* **Then** [*Anweisungen*] [**Else** *elseAnweisungen*]

Alternativ können Sie die Block-Syntax verwenden:

If *Bedingung* **Then**
  [*Anweisungen*]

[**ElseIf** *Bedingung-n* **Then**
  [*elseifAnweisungen*] . . .

[**Else**
  [*elseAnweisungen*]]

End If

Indizierte Schleife

```
For Index = 1 To 10 Step 1
   Anweisungen
Next Index
```

Anweisungen werden hier genau 10x durchlaufen

Die Angabe der Schrittweite `Step` ist optional
Standard-Schrittweite = 1

Pre-Check Schleifen

= Überprüfung VOR Durchführung der Anweisungen

```
Do While Zahl > 10
Zahl = Zahl - 1
Wert = Wert + 1
Loop
```

```
Do Until Zahl = 10
Zahl = Zahl + 1
Wert = Wert + 1
Loop
```

Wiederholung der Anweisungen,

solange die Bedingung WAHR ist **bis die Bedingung WAHR ist**

Do...Loop-Anweisungen dienen dazu, einen Block von Anweisungen eine unbestimmte Anzahl von Wiederholungen ausführen zu lassen.

Die Anweisungen werden wiederholt, solange eine Bedingung dem Wert **True** entspricht oder bis eine Bedingung dem Wert **True** entspricht.

Sie können eine **Do...Loop**-Anweisung unter Verwendung der **Exit Do**- Anweisung verlassen. Verwenden Sie z.B. die **Exit Do**-Anweisung im **Do**- Anweisungsblock einer **If...Then...Else**-Anweisung oder einer **Select Case**-Anweisung, wenn Sie eine Endlosschleife verlassen möchten.
Entspricht die Bedingung dem Wert **False**, so wird die Schleife wie üblich ausgeführt.

Anmerkung

Drücken Sie **ESC** oder **STRG+PAUSE**, um eine Endlosschleife zu beenden.

Post-Check Schleifen

= Überprüfung NACH Durchführung der Anweisungen

```
Do
Zahl = Zahl - 1
Wert = Wert + 1
Loop While Zahl > 10
```

```
Do
Zahl = Zahl + 1
Wert = Wert + 1
Loop Until Zahl = 10
```

Wiederholung der Anweisungen,

solange die Bedingung WAHR ist **bis die Bedingung WAHR ist**

Prüfung der Bedingung, nachdem die Schleife mindestens einmal durchlaufen wurde

Quelle:
Einführung in die Excel-Makroprogrammierung von J. Abulawi,
Hochschule für Angewandte Wissenschaften Hamburg, SS 2005
*www.haw-hamburg.de/pers/**Abulawi**/**Excel**makrosSS05.pdf*

Ablaufsteuerung

Schleife:		DO UNTIL bedingung / LOOP
	oder	DO / LOOP UNTIL bedingung
Ausstieg - unbedingt:		EXIT DO *oder* EXIT FOR
- bedingt:		IF bedingung THEN EXIT FOR (*oder* EXIT DO)
leereZelle		ISEMPTY( )

Tabelle

Tabellenblatt wählen:	SHEETS("Tabelle1").SELECT
Zelle wählen:	[B12].SELECT
Bereich wählen + löschen:	RANGE("A3:B15").CLEARCONTENTS
oder	RANGE (CELLS(3,1), CELLS(15,2)).CLEARCONTENTS
Zahlenformat festlegen:	RANGE("A2:C10").NumberFormat = "0.000"
	RANGE("A2:C10").NumberFormat = "0.00E+00"

Datenfiles

Öffnen + Lesen:	OPEN pfad$ FOR INPUT AS #nummer
Öffnen + Schreiben:	OPEN pfad$ FOR OUTPUT AS #nummer
Öffnen + Anhängen:	OPEN pfad$ FOR APPEND AS #nummer
Schließen:	CLOSE #nummer
Lesen:	INPUT #nummer, variable[nliste]
Schreiben:	WRITE #nummer, variable[nliste]

Einlesen bis File-Ende

(siehe Beispiel 2)

Öffnen + Lesen:	Open Pfad$ For Input As #1
Pre-Check Schleife (Not "End of File"):	Do While Not EOF(1)
Zeilenweise Lesen:	Line Input #1, textzeile
Schleifenbefehl:	Loop
Schließen:	Close #1