Visual Basic Basisbefehle

Hinweis: Der Text in eckigen Klammern [ ] ist variabel, z.B. [var] => 5.3.
Eckige Klammern sind stets wegzulassen!

Grundstrukturen:

```
Sub [name]([Übergabe])
End Sub
```

[Übergabe] ist optional

```
Function [funktion]([Übergabe])
End function
```

[Übergabe] ist Pflicht

Zur Übergabe:
Ruft man eine Funktion (oder Sub) aus einer anderen Stelle des Programms auf, so muss man diesem Ruf noch ein paar Daten anhängen, mit denen die Funktion (oder Sub) starten soll. Diese werden in die runden Klammern hinter dem Befehl (s.u.) geschrieben. In der Function (oder Sub) müssen in den Klammern dann die Variablen noch deklariert werden.

Aufrufen eines Programm(abschnitts):

```
Call [name]([Übergabe])
```

```
[funktion]([Übergabe])
```

Beispiel:

```
Call Programm1(a,b)
```
→ Es wird „Programm1“ aufgerufen mit den Daten a und b

```
Sub Programm1(x As Single, y As Integer)
```
→ Das Sub übernimmt die erste Variable als Single und die zweite als Integer.
a wird zu x und b wird zu y. Es können auch gleiche Namen gewählt werden.

```
GoTo [Textmarke]
```
→ Springt innerhalb einer Prozedur zur Stelle [Textmarke]

```
[Textmarke]: [CODE]
```
→ Die Textmarke steht vor der Programm-Zeile mit einem Doppelpunkt

Variablen: Deklaration und Typen:

```
Dim [var] As [typ], [var2] As [typ]
```
oder
```
Dim [var]([Bedingungen]) As [typ]
```

Zu Bedingungen:
Bedingungen sind z.B. `abc(1 To 3,1 To 3).` Damit deklariert man gleich 3 Variablen, nämlich `abc(1), abc(2), abc(3).`

Einige mögliche Typen:

`Single` (Gleitkommazahl)
`Double` (Gleitkommazahl, genauer)
`String` (Text)
`String * 10` (Text mit max. 10 Zeichen)
`Date` (Zeitformat: z.B. `#TT/MM/JJ#`)

`Integer` (-32.000 bis +32.000)
`Long` (-2 Mrd. bis +2 Mrd.)
`Range` (Zellinformation)
`Boolean` (1 oder 0, True oder False)

Konstanten:

```
Const [Con] = [Wert]
```

Dateneingabe:

```
[var]()=InputBox("[Anzeigetext]","[Titel]",[Default])
```

→ ( ) sind optional, wenn die Variable mit ( ) deklariert wurde, muss hierin die Nummer stehen, sonst werden die Klammern weggelassen.

Datenausgabe:

```
MsgBox("[Text]" & [Var])
```

→ Eine Box wird angezeigt mit den Inhalten der Klammern. Text muss in "" geschrieben werden. Verschiedene Variablen oder Text und Variablen müssen mit einem & voneinander getrennt sein

```
Debug.Print("[Text]" & [var])
```

→ Der Inhalt der Klammern wird im Direktbereich ausgegeben. Details s.o.)

Auskommentierung:

```
'[Kommentar]
```

→ Kommentierung über mehrere Zeilen ist nicht möglich.

Schleifen und Abfragen:

```
If [var] >/</= [Bed] Then
[CODE]
Else
[CODE]
End If
```

→ End If muss bei mehr als einem Befehl im [CODE] und bei Else gesetzt werden. **>/</=**.ist ein Beispiel, welche Operatoren möglich sind.
(Es gibt auch <= (kleinergleich) und >= (größergleich), sowie <> (ungleich)

```
For [var]=1 To [Zahl]
[CODE]
Next [var]
```

→ Keine Abbruchbedingung nötig, da diese Schleife von selbst hochzählt.

```
Do While [var] </>/= [Bed]
Loop
```

```
Do Until [var] </>/= [Bed]
Loop
```

→ Schleifen mit der Abfrage vorher, sie werden nur durchlaufen, wenn die Bedingung erfüllt ist.

Achtung: Innerhalb der Do Loop Schleife muss die Abbruchbedingung erfüllt werden!

```
Do
Loop While [var] </>/= [Bed]
```

```
Do
Loop Until [var] </>/= [Bed]
```

→ Schleifen mit Abfrage hinterher, Schleife läuft immer mindestens einmal

Achtung: Innerhalb der Do Loop Schleife muss die Abbruchbedingung erfüllt werden!

```
Select Case [Var]
Case "[Wert]"
[CODE]
Case [Zahl]
[CODE]
Case Else
[CODE]
End Select
```

→ Funktioniert ähnlich wie die `If-Then-Else` Abfrage, doch kann man hiermit wesentlich einfacher mehrere Möglichkeiten einbauen ohne komplizierte Verschachtelungen.

→Text wird in **""** dargestellt, Zahlen können auch ohne geschrieben werden.

→ `Case Else` schließt dann alle anderen Fälle mit ein.

Besondere Operationen:

```
[var1]=Left([Textvar], InStr([Textvar], " ")-1)
[var2]=Right([Textvar], Len([Textvar]) - InStr([Textvar], " ")
```

→ Ein in `[Textvar]` eingegebener Text wird bei seinem ersten Leerzeichen (`" "`) getrennt, alles links davon kommt in die Variable `[var1]`, alles rechts davon in die Variable `[var2]`

```
[var]=FormatNumber([var], [Nachkommastellen])
```

→ Die Zahl in der Variablen wird auf die angegebenen Nachkommastellen gerundet und wieder auf sich selbst übertragen (`[var]=`)

EXCEL:

Zellwerte verändern:

`ActiveCell.Value =`	→ Ändert den Wert der aktiven Zelle
`ActiveCell.Offset([X],[y]).Value =`	→ Ändert den Wert der von der aktiven Zelle um x nach unten und um y nach rechts verschobenen Zelle
`Cells([x],[y]).Value =`	→ Ändert den Wert der Zelle x,y
`ActiveSheet.Cells([x],[y]).Value =`	→ Ändert den Wert der Zelle x,y im aktiven Arbeitsblatt
`Range("A1").Value =`	→ Ändert den Wert der Zelle A1

Zellwerte lesen:

Befehle wie oben nur umgedreht.

Beispiele:

```
[var] = ActiveCell.Value
[var] = Range("A1").Value
```

Zellen/Blätter auswählen:

`ActiveCell.Offset([X],[y]).Select`	→ Wählt die Zelle um x nach unten und um y nach rechts als neue aktive Zelle
`Range("A1").Select`	→ Wählt die Zelle A1 als aktive Zelle
`Range("A1:B3").Select`	→ Wählt die zellen von A1 bis B3 aus
`Sheets("[Sheetname]").Activate` `Sheets("[Sheetname]").Select`	→ Wählt das Blatt mit dem Namen [Sheetname] aus (alternativ mit Select)

Zellen und Text formatieren:

`ActiveCell.Font.Bold = True/False`	→ Neben Bold auch möglich: `Italic, Underline`
`ActiveCell.Font.Colorindex = 1,2,3,4...`	→ Ändert die Farbe der Schriftart der aktiven Zelle, z.B.: `3` = Rot
`ActiveCell.Interior.Colorindex = 1,2...`	→ Ändert die Farbe des Hintergrunds der aktiven Zelle
`Cells([x],[y]).Font.Bold = True/False`	→ Ändert die Formatierung (s.o.)
`Cells([x],[y]).Font.ColorIndex = 1,2`	→ Ändert die Farbe der Schriftart der Zelle x,y
`Cells([x],[y]).Interior.ColorIndex = 1,2`	→ Ändert die Farbe des Hintergrunds der Zelle x,y

Zellen und Spalten ausblenden:

`ActiveCell.EntireRow.Hidden = True`	→ Versteckt die Zeile der aktiven Zelle (bei True) Statt Row auch Column möglich
`Cells([x],[y]).EntireRow.Hidden = True`	→ Versteckt die Zeile der Zelle x,y (bei True) Statt Row auch Column möglich
`Rows([x]).Hidden = True/False` `Columns([y]).Hidden = True/False`	→ Wie oben, doch simpler

Speziellere Befehle:

`IsNumeric(ActiveCell.Value) =`	→ True, wenn der Ausdruck in der Klammer eine Zahl ist, False, wenn der Ausdruck keine Zahl ist.
`IsEmpty(ActiveCell.Value) =`	→ True, wenn der Ausdruck in den Klammern leer ist, False, wenn der Ausdruck Zeichen hat..
`ActiveSheet.UsedRange.Rows.Count` `ActiveSheet.UsedRange.Columns.Count`	→ Die Anzahl der genutzten Zeilen, bzw. Anzahl der genutzten Spalten.
`ActiveSheet.UsedRange.ClearContents`	→ Löscht alle Inhalte im aktiven Blatt (nicht die Formatierung)

Kopieren, Ausschneiden und Einfügen:

`ActiveCell.Copy` `ActiveCell.Cut`	→ Kopiert den Inhalt der aktiven Zelle bzw. schneidet ihn aus
`Cells([x],[y]).Copy` `Cells([x],[y]).Cut`	→ Kopiert den Inhalt der Zelle x,y bzw. schneidet ihn aus.

`Selection.EntireRow.Copy`
`Selection.EntireRow.Cut`
`Selection.EntireColumn.Copy`
`Selection.EntireColumn.Cut`
	→ Markiert die ganze Zeile/Spalte und kopiert sie, bzw. schneidet sie aus.

`ActiveSheet.Paste`	→ Fügt die ausgeschnittenen Zellen ein
`Application.CutCopyMode = False`	→ Beendet die Kopiermarkierung
`Selection.Copy`	→ Kopiert die Auswahl

Sonstige Befehle

Aufbau einer Case-Abfrage für Zellen:

```
Select Case (ActiveCell.Value)
Case Is = [Wert]
End Select
```

For-Each-Schleife mit Range:

```
Dim [var] As Range

For Each [var] In Selection
```
oder: ***(For Each [var] In UsedRange)***
```
[var].Value = ...
Next
```

For-Schleife mit Count:

```
For ... to UsedRange.Count
```