R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Leere erste Zeile in einer Zeile füllen

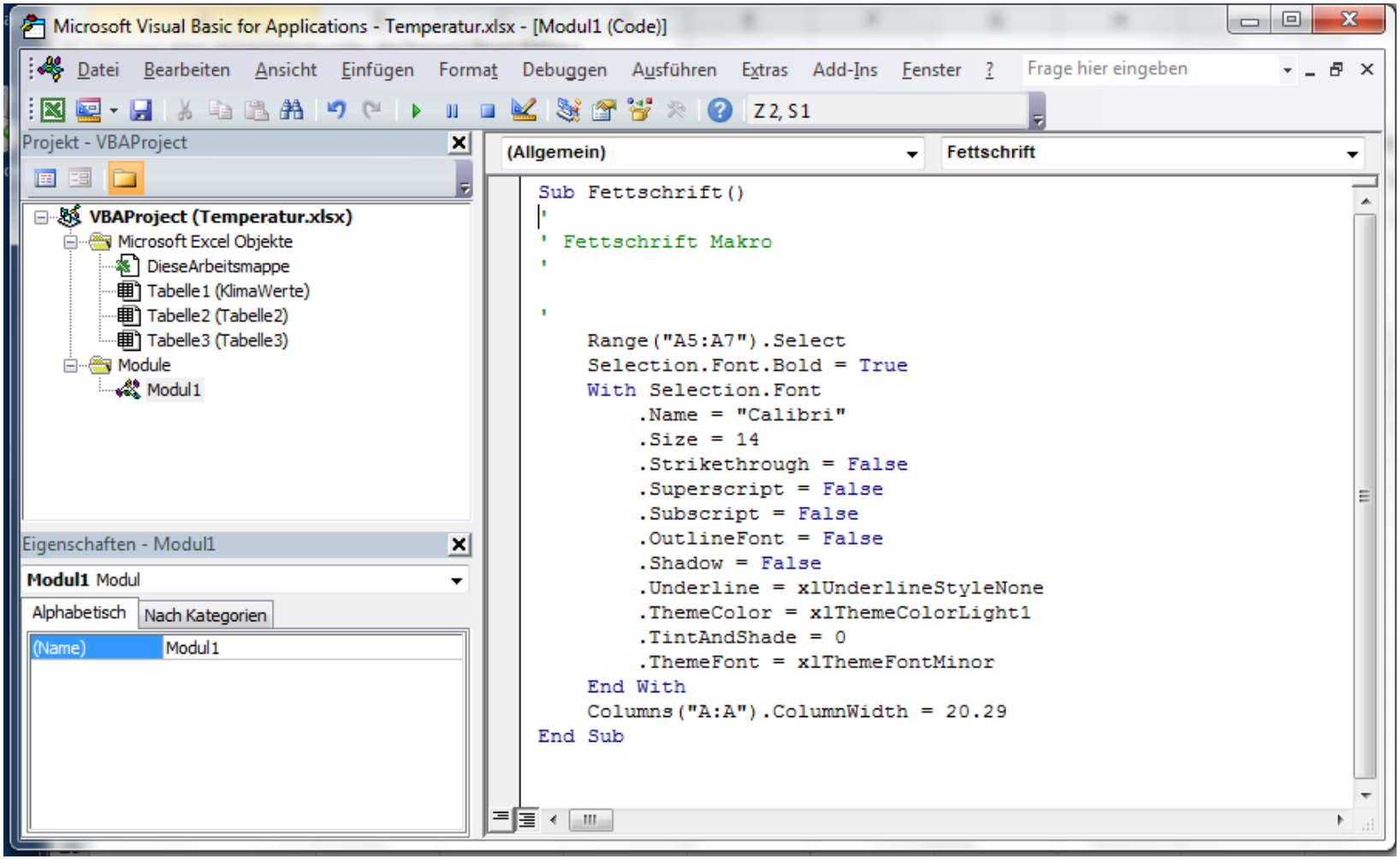
Aufgabe

- In einem Excel-Arbeitsblatt werden zeilenweise verschiedene Reihen unterschiedlichsten Inhaltes angezeigt.
- In Abhängigkeit der Differenz der zwei vorherigen Zellen wird die leere Zelle gefüllt.
- Hinweis: Diese Aufgabe ist nicht durch die Aufzeichnung eines Makros lösbar.

VBA-Editor öffnen

- Das Menüband Entwicklertools ist eingeblendet.
- Mit einem Klick auf *Visual Basic* in der Gruppe Code wird der VBA-Editor geöffnet.

Beispiel


Microsoft Visual Basic for Applications - Temperatur.xlsx - [Modul1 (Code)]
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Frage hier eingeben
Z 2, S 1
Projekt - VBAProject
VBAProject (Temperatur.xlsx)
Microsoft Excel Objekte
DieseArbeitsmappe
Tabelle1 (KlimaWerte)
Tabelle2 (Tabelle2)
Tabelle3 (Tabelle3)
Module
Modul1
Eigenschaften - Modul1
Modul1 Modul
Alphabetisch
Nach Kategorien
(Name)
Modul1
(Allgemein)
Fettschrift
Sub Fettschrift()
'
' Fettschrift Makro
'
'
Range("A5:A7").Select
Selection.Font.Bold = True
With Selection.Font
.Name = "Calibri"
.Size = 14
.Strikethrough = False
.Superscript = False
.Subscript = False
.OutlineFont = False
.Shadow = False
.Underline = xlUnderlineStyleNone
.ThemeColor = xlThemeColorLight1
.TintAndShade = 0
.ThemeFont = xlThemeFontMinor
End With
Columns("A:A").ColumnWidth = 20.29
End Sub

VBA-Editor ...

- ist eine integrierte Entwicklungsumgebung (IDE) für die Programmiersprache V(isual)B(asic for)A(pplication).
- ist in jeder Office-Anwendung vorhanden.
- ist eine eigenständige Anwendung, die in der Taskleiste als Symbol eingeblendet wird.
- bietet die Möglichkeit VBA-Code zu lesen und zu bearbeiten.

Aufbau

- Titelleiste zur Anzeige von Informationen.
- Menüleiste. Sammlung aller Befehle.
- Symbolleiste. Anzeige der wichtigsten Befehle als Icon.
- Projekt-Explorer. Schaltzentrale einer Excel-Datei im Bereich VBA.
- Eigenschaftenfenster. Attribute des gewählten Objekts.
- Codefenster. Anzeige von Code zu dem gewählten Objekt.
- Mit Hilfe des Rahmens um den Editor herum, wird das Fenster vergrößert oder verkleinert.

Titelleiste ...

- befindet sich am oberen Rand der Anwendung.
- hat in der linken Ecke ein Icon, welches die Anwendung symbolisiert. Mit einem Klick auf das Icon wird das dazugehörige Systemmenü geöffnet.
- zeigt den Namen der Excel-Datei sowie des aktiven Moduls an.
- hat am rechten Rand Schaltflächen zum Minimieren, Verkleinern / Maximieren oder Schließen der Anwendung.

Menüleiste ...

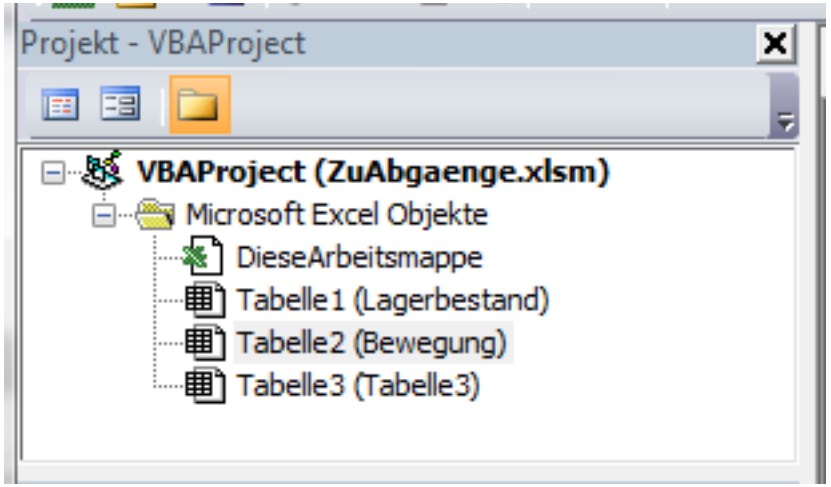
- befindet sich unterhalb der Titelleiste.
- sammelt alle Befehle der Anwendung.
- bietet aufklappbare Menüs zu verschiedenen Themen an. In diesen Menüs werden Befehle und Funktionen der Anwendung gesammelt.
- zeigt auf der obersten Ebene Kategorien an. In Abhängigkeit dieser Kategorien werden die Befehle zusammengefasst. Der Name der Kategorie gibt einen ersten Hinweis auf die Benutzung der darin enthaltenen Befehle.

Symbolleisten ...

- sammeln häufig genutzte Aktionen zu einem Thema. Die Aktionen werden durch Icons dargestellt.
- beginnen mit der senkrechten gestrichelten Linie am linken Rand. Sobald die Maustaste über diese Linie liegt, kann die Symbolleiste mit Hilfe von Drag (Maustaste gedrückt) & Drop (Maustaste loslassen) verschoben werden.
- werden über das Menü *Ansicht* – *Symbolleiste* ein- oder ausgeblendet.
- haben einen Pfeil nach unten am rechten Rand. Mit einem Klick auf den Pfeil wird ein Menü zum Ein- und Ausblenden von Icons in der Symbolleiste angezeigt.

Projekt-Explorer ...

- ist die Schaltzentrale für die Programmierung einer Excel-Anwendung.
- verwaltet die, zu dem Projekt gehörende Arbeitsmappe sowie die darin enthaltenen Arbeitsblätter.
- zeigt aufgezeichnete Makros in Modulen an.
- ist frei platzierbar.
- kann über das Menü *Ansicht* ein- oder ausgeblendet werden.



Aufbau des Projekt-Explorers

- In der Titelleiste wird die Schließen-Schaltfläche angezeigt.
- Darunter befindet sich die Symbolleiste mit den Icons
  - „Zeige zum gewählten Element den Code an“.
  - „Zeige das passende Objekt zum Code an“.
  - „Sortierung mit Hilfe von Ordnern“.
- Unterhalb der Symbolleiste werden die Module alphabetisch oder in Abhängigkeit ihres Typs sortiert angezeigt.

Module ...

- kapseln Code zu einem Thema.
- sind Container für Code. In dem Container wird eine bestimmte Aufgabe gelöst.
- fassen Programmiercode und Deklarationen zu einem Thema zusammen.
- werden automatisch durch die Aufzeichnung eines Makros angelegt.
- können vom Entwickler oder der Anwendung angelegt werden.

Modul-Typen ...

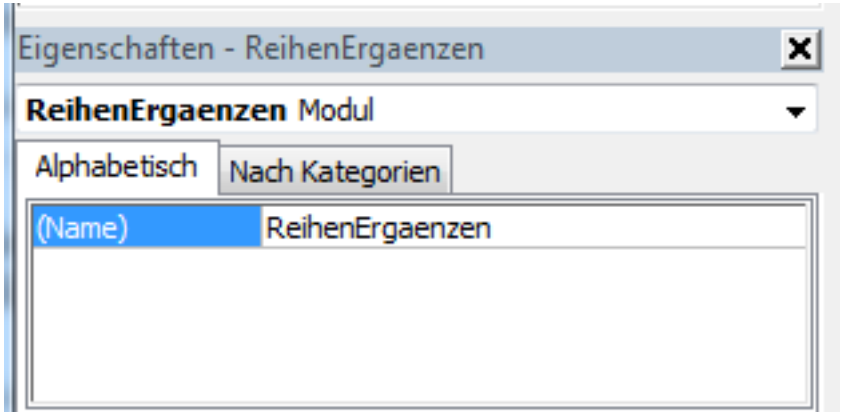
- werden mit Hilfe von Ordnern im Projekt-Explorer dargestellt.
- *(Microsoft Excel Objekte)* enthalten Module, die Code für die Arbeitsmappe oder die darin enthaltenen Arbeitsblätter enthalten.
- *(Module)* enthalten aufgezeichnete Makros oder vom Entwickler selbst geschriebene Prozeduren.

Standardmodul einfügen

- *Einfügen* – *Modul* im VBA-Editor.
- Im Code-Fenster wird das leere Modul angezeigt.

Namen eingeben

- Das Eigenschaftenfenster ist geöffnet.
- Rechts von der Eigenschaft *Name* wird eine Bezeichnung für das Modul eingegeben.
- Das Modul wird unter diesen Namen gespeichert.

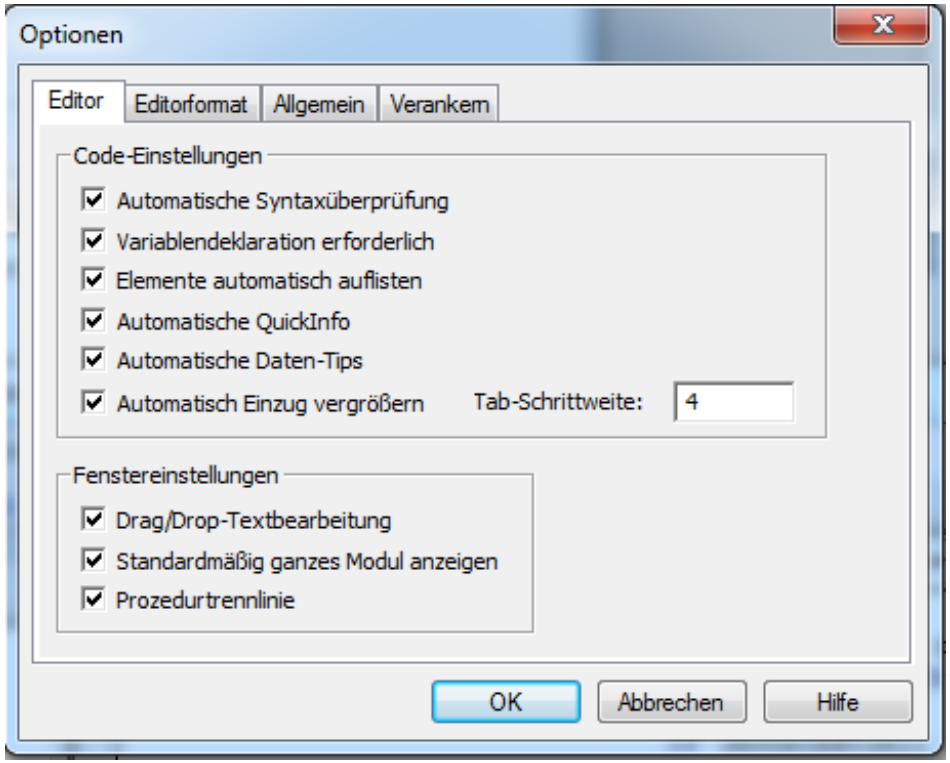


Anweisung „Option Explicit”

- Diese Anweisung steht in der ersten Zeile eines Moduls
- Nicht deklarierte Variablen werden als Fehler gemeldet.
- Doppelt vergebene Bezeichner werden als Fehler gemeldet.

… automatisiert bei einem neuen Modul einfügen

- *Extras – Optionen*.
- Registerkarte Editor.
- Klick auf das leere Kästchen *Variablendeklaration erforderlich*.



Der Rahmen eines „Makros” in dem Codefenster

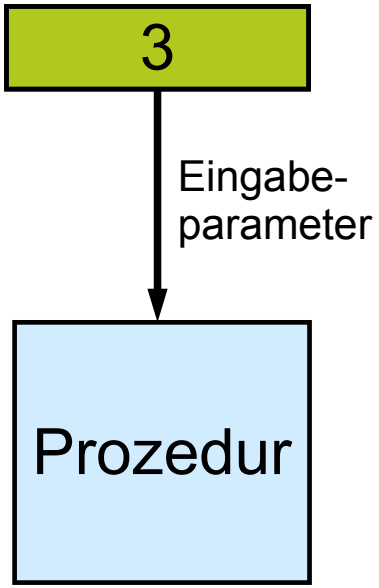
```
Sub LeereZelle()


End Sub
```

- Durch das Schlüsselwort Sub wird der Beginn einer Prozedur / eines Makros gekennzeichnet.
- Anschließend folgt ein selbsterklärender, benutzerdefinierter Name.
- Dem Namen folgen leere runde Klammern. Der Prozedur werden in diesem Beispiel keine Werte übergeben.
- Die Prozedur / das Makro endet mit den Schlüsselwörtern End Sub. Diese Schlüsselwörter werden automatisch von VBA gesetzt.

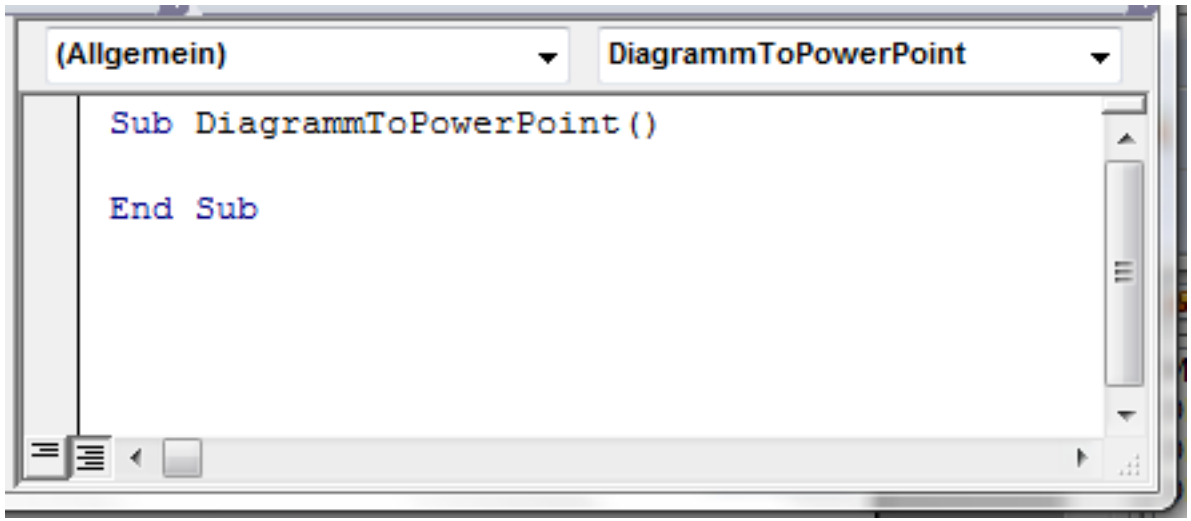
Arbeitsweise einer Prozedur

- Der Prozedur können Eingabeparameter (Argumente) übergeben werden.
- Diese Eingabeparameter werden in der Prozedur verarbeitet.
- Die Verarbeitung selber ist dem Aufrufer aber nicht bekannt. Der Nutzer kennt nur die Schnittstelle (den Prozedurkopf) nach außen.



Anzeige im Codefenster

- Die Schlüsselwörter aus VBA werden standardmäßig mit blauer Schrift angezeigt.
- Im Codefenster wird das Modul mit Hilfe der Prozeduren in kleine Code-Abschnitte unterteilt.
- Ein Modul kann aus beliebig vielen Modulen bestehen.



Regeln für den Bezeichner

- Beginn mit einem Buchstaben.
- Maximale Länge 255 Zeichen.
- Keine Beachtung der Groß- und Kleinschreibung.
- Der Bezeichner besteht nur aus den Buchstaben a..z, A..Z, dem Unterstrich und den Zahlen 0..9.
- Jeder Bezeichner wird in einem Modul nur einmal genutzt und in dem Dialog „Makro” angezeigt.
- VBA-Schlüsselwörter oder von VBA, definierte Funktionsnamen können nicht als benutzerdefinierte Namen genutzt werden.

Geeignete Prozedur-Namen ...

- geben Auskunft über die Nutzung der Prozedur.
- beschreiben die in der Prozedur, gekapselten Aktionen.
- entsprechen dem Sprachraum des Entwicklers.
- sind an die Sprache der realen Welt angelehnt.

Zusammengesetzte Prozedur-Namen

- Sub BionomischeFormel(). Jedes Wort in dem benutzerdefinierten Namen beginnt mit einem Großbuchstaben.
- Sub Umrechnung_Meter_Centimeter(). Die Wörter werden mit Hilfe des Unterstrichs getrennt. Der Unterstrich ersetzt Leerzeichen in einem Wort.

Code eingeben

```
Sub LeereZelle()


End Sub
```

- Mit einem Mausklick wird die Einfügemarke zwischen den Schlüsselwörtern Sub und End Sub gesetzt.
- Mit Hilfe der Tastatur werden dort die benötigten Arbeitsschritte zeilenweise eingegeben.

Aufbau einer Prozedur

```
Sub LeereZelle()
        ' Deklaration der Variablen


        ' Anweisungen
End Sub
```

- Am Anfang der Prozedur sollten die Variablen deklariert werden.
- Anschließend werden die Anweisungen geschrieben.

Variablen ...

- sind Platzhalter für bestimmte Werte. Als Werte können Zahlen, Datums- und Zeitwerte sowie Texte gespeichert werden.
- sind von einem bestimmten Standard-Datentyp. Die Datentypen in VBA werden im Internet auf der Seite http://msdn.microsoft.com/en-us/library/gg278937.aspx aufgelistet.
- haben einen eindeutigen Namen. Der Name sollte selbsterklärend sein.

… deklarieren

```
Sub LeereZelle()
    Dim zeile As Long
    Dim spalte As Long
    Dim zeileLastInArbeitsblatt As Long
    Dim abstand As Integer
    Dim asciicode As Byte
    Dim monat As Byte
    Dim monatAktuell As Byte
    Dim monatDatum As String
End Sub
```

Erläuterung

- Jede Variable hat einen eindeutigen Namen (zeile, spalte, etc.). Der Name sollte über den zu speichernden Wert Auskunft geben. Die Bezeichnung ist frei wählbar.
- Jede Variable verweist auf einen bestimmten Datentyp. Mit Hilfe des Schlüsselwortes As wird der Variablen ein Typ zugewiesen.
- Auf die Variablen kann nur innerhalb der Prozedur LeereZelle zugegriffen werden (Dim). Diese Prozedur ist Besitzer dieser Variablen.

Datentypen

- beschreiben den zu speichernden Wert.
- legen Regeln für die Verwendung der Variablen fest.
- beschreiben den Datenbereich des Platzhalters.
- legen den Speicherbedarf fest.
- sind für Ganzzahlen, Dezimalzahlen, Datums- und Zeitwerte sowie Zeichenfolgen vorhanden.

… für Ganzzahlen

	Speicherbedarf in Bytes	Datenbereich
As Byte	1	0 - 255
As Integer	2	-32.768 - +32.767
As Long	4	-2.147.483.648 - +2.147.483.647
As Boolean	2	0 (falsch, false) <> 0 (wahr, true)

Datentypen für Zeichenfolgen (Text)

	Länge
As String	Variable Länge.
As String * 5	Die Länge des Strings wird auf eine bestimmte Anzahl eingeschränkt. In diesem Beispiel besteht die Zeichenfolge aus fünf Zeichen.

Konvertierung von Datentypen

- Unterschiedliche Datentypen in Ausdrücken werden häufig automatisch (implizit) umgewandelt.
- Mit Hilfe von Funktionen kann eine Konvertierung des Ausdruckes vom Programmierer explizit erzwungen werden.

Funktionen zur Konvertierung

	Konvertierung zu ...
CBool(variable)	Boolean
CByte(variable)	Byte
CInt(variable)	Integer
CLng(variable)	Long
CSng(variable)	Single
CDbl(variable)	Double
CCur(variable)	Currency
CDate(variable)	Date
CStr(variable)	String

Werte zuweisen

```
Sub VariablenZuweisen()
    Dim lngSpalte As Long
    Dim strDatum As String
    Dim dateDatum As Date

    lngSpalte = 1
    strDatum = "01.12.2013"
    dateDatum = #01.12.2013#


End Sub
```

Zuweisungsoperator

- Als Zuweisungsoperator wird das Gleichheitszeichen genutzt.
- Rechts vom Gleichheitszeichen steht ein Ausdruck. Der Ausdruck besteht aus festen Werten (Konstanten), Variablen und Operatoren.
- Der, durch den Ausdruck berechnete Wert, wird der Variablen links vom Gleichheitszeichen zugewiesen.

Integrierte Funktionen zur Berechnung von Werten

- Mathematische Funktionen.
- Berechnung von Datums- und Zeitwerten.
- Bearbeitung von Strings.
- Konvertierung von Datentypen
- Daten aus Dateien oder vom Bildschirm ein- oder auslesen.
- Ordner oder Datei nutzen.
- Die Funktionen werden alphabetisch in der VBA-Hilfe (*Visual Basic-Sprachverzeichnis - Funktionen*) alphabetisch aufgelistet.

… aufrufen

```
variable = Funktion(argument01, argument02, ...)
```

- Die Funktion wird mit Hilfe ihres Namens aufgerufen.
- Dem Namen folgen die runden Klammern. Die Klammern enthalten eine Liste von Argumenten. Die Anzahl der zu übergebenden Argumente ist abhängig von der Aufgabe der Funktion. Leere Klammern bedeuten, dass der Funktion keine Werte übergeben werden.
- Die Elemente der Liste werden durch ein Komma getrennt.
- Eine Funktion gibt immer einen Wert von einem bestimmten Datentyp zurück. Der Rückgabewert kann in einer Variablen gespeichert werden.

Länge einer Zeichenfolge

```
Len(zellbereich.Value)
```

- Der Funktion Len() wird in runden Klammern die zu untersuchende Zeichenfolge übergeben.
- Die Funktion liefert die Anzahl der Zeichen zurück.

Zeichencode eines Zeichens

```
Asc(zellbereich.Value)
```

- Der Funktion Asc() liefert den ASCII-Zeichencode eines Textzeichens zurück.

ASCII-Zeichentabelle

0 =	18 = ↕	36 = $	54 = 6	72 = H	90 = Z	108 = l
1 = ☺	19 = ‼	37 = %	55 = 7	73 = I	91 = [	109 = m
2 = ☻	20 = ¶	38 = &	56 = 8	74 = J	92 = \	110 = n
3 = ♥	21 = §	39 = ’	57 = 9	75 = K	93 =]	111 = o
4 = ♦	22 = ▬	40 = (	58 = :	76 = L	94 = ^	112 = p
5 = ♣	23 = ↨	41 =)	59 = ;	77 = M	95 = _	113 = q
6 = ♠	24 = ↑	42 = *	60 = <	78 = N	96 = ‘	114 = r
7 = •	25 = ↓	43 = +	61 = =	79 = O	97 = a	115 = s
8 = ◘	26 = →	44 = ,	62 = >	80 = P	98 = b	116 = t
9 = ◇	27 = ←	45 = -	63 = ?	81 = Q	99 = c	117 = u
10 = ◙	28 = ∟	46 = .	64 = @	82 = R	100 = d	118 = v
11 = ♂	29 = ↔	47 = /	65 = A	83 = S	101 = e	119 = w
12 = ♀	30 = ▲	48 = 0	66 = B	84 = T	102 = f	120 = x
13 = ♪	31 = ▼	49 = 1	67 = C	85 = U	103 = g	121 = y
14 = ♫	32 =	50 = 2	68 = D	86 = V	104 = h	122 = z
15 = ☼	33 = !	51 = 3	69 = E	87 = W	105 = i	123 = {
16 = ►	34 = "	52 = 4	70 = F	88 = X	106 = j	124 = ¦
17 = ◄	35 = #	53 = 5	71 = G	89 = Y	107 = k	125 = }

… im Web

- http://www.bettersolutions.com/vba/VIX243/VD622016333.htm
- http://www.vba-proggen.de/attachments/ASCII-Zeichensatz.pdf

Strings formatieren

```
Format(Variable, Formatstring)
```

- Die Funktion Format() liefert einen formatierten String zurück.
- Der Funktion wird als erstes Argument der zu formatierende String übergeben.
- Als zweites Argument wird der Formatstring übergeben. Der Formatstring beginnt und endet mit den Anführungszeichen. Mit Hilfe von Formatierungszeichen wird der String gebildet.
- Hilfe zum Formatstring finden Sie unter http://msdn.microsoft.com/en-us/library/office/gg251755%28v=office.14%29.aspx

Beispiele

```
' Der Monat wird zweistellig formatiert
Format("1." & zelleAktuell.Value & "." & Year(Date), "mm")

' Der Name des Monats
Format(DateSerial(Year(Date), monat + 1, 1), "mmmm")

' Formatierung als Währung
Format(einnahme, "Currency")
```

Datumsfunktionen

```
heute = Date                                    ' Aktuelles Systemdatum
aktuellZeit = Time                              ' Aktuelle Systemzeit

gestern = DateDiff("d", 1, heute)               ' heute – 1 (in Tagen)
morgen = DateAdd("d", 1, heute)                 ' heute + 1 (in Tagen)

jahr = Year(Date)                               ' Rückgabe des Jahres
monat = Month(Date)                             ' Rückgabe des Monats
tag = Day(Date)                                 ' Rückgabe des Tages

datum = DateSerial(jahr, monat + 1, tag)        ' Datumswert
```

Objektvariablen ...

- verweisen auf ein bestimmtes Objekt in einer Office-Anwendung.
- enthalten eine Referenz auf die Office-Anwendung selbst oder auf Elemente in einem PowerPoint-Dokument, Excel-Arbeitsmappe etc.
- werden für Objekte definiert, die häufig in einem Programm genutzt werden.

… deklarieren

```
Sub LeereZelle()
    Dim zelleVorher As Range
    Dim zelleAktuell As Range

End Sub
```

Erläuterung

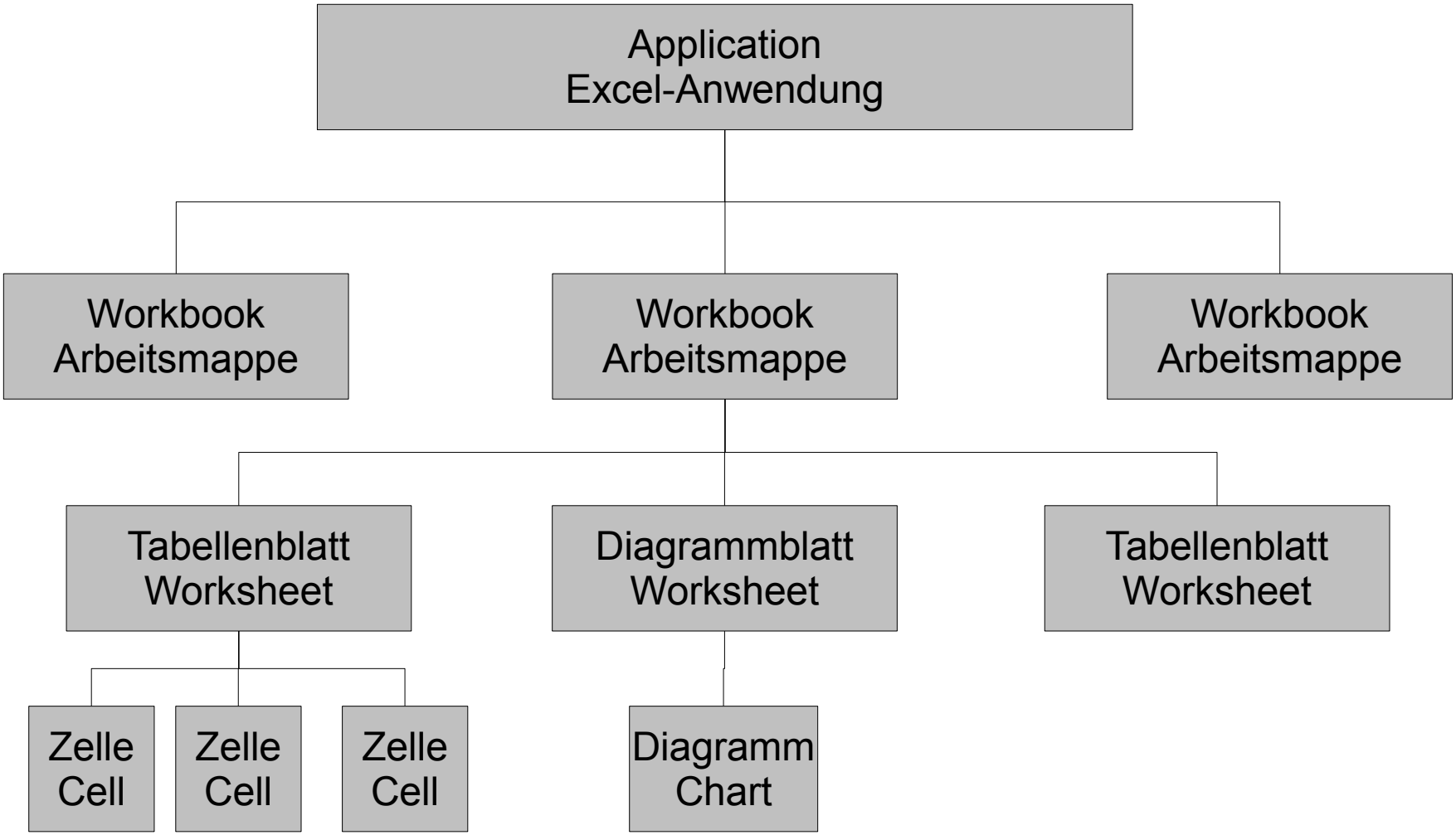
- Jede Objektvariable hat einen eindeutigen Namen (zelleVorher). Der Name sollte über die Art des Verweises Auskunft geben. Die Bezeichnung ist frei wählbar.
- Jede Objektvariable verweist auf einen bestimmten Typ von Objekt. Mit Hilfe des Schlüsselwortes As wird der Variablen ein Typ zugewiesen.
- Mit Hilfe des Schlüsselwortes Dim wird der Zugriff auf Variablen geregelt. Auf die Variablen kann nur innerhalb der Prozedur LeereZelle zugegriffen werden (Dim). Diese Prozedur ist Besitzer dieser Variablen.

Zuweisung an eine Objektvariable

```
Sub LeereZelle()
    Set zelleVorher = ActiveSheet.Cells(zeile, spalte - 1)
    Set zelleVorher = Nothing
End Sub
```

- Das Schlüsselwort Set leitet eine Zuweisung an eine Objektvariable ein.
- Mit Hilfe des Gleichheitszeichens wird der Objektvariablen ein Verweis zugewiesen.
- Der Wert Nothing symbolisiert die leere Objektvariable. Die Variable verweist auf kein Objekt.

Excel-Objektmodell



Worksheet (Arbeitsblatt)

- Die Daten in einer Excel-Datei.
- Tabellen in Excel. Tabellen bestehen aus Zeilen und Spalten.
- Die Registerkarte am unteren Rand der Arbeitsmappe zeigt die verschiedenen Arbeitsblätter an.

ActiveSheet = Aktives Arbeitsblatt

```
Sub Objektvariable()
    Dim arbeitsblatt As Worksheet

    Set arbeitsblatt = ActiveSheet
End Sub
```

- Anzeige in Fettschrift auf den Registerkarten am unteren Rand.
- Das angezeigte Arbeitsblatt in Excel.

Cells ...

Cells(1)				
		Cells(4,3)		

- enthält alle Zellen eines Arbeitsblattes.
- beschreibt mit Hilfe des Zeilen- und Spaltenindex eindeutig eine Zelle.

Eigenschaften von Zellen

- .Value speichert den Inhalt einer Zelle. Der Wert kann mit Hilfe von VBA gelesen (wert = zelle.Value) oder verändert (zelle.Value = wert) werden.
- .Address gibt die Position der Zelle in dem Arbeitsblatt zurück. Der Wert kann nur Hilfe von VBA gelesen (pos = zelle.Address). Zum Beispiel die Positionsangabe "A1" bezeichnet eine Zelle in der ersten Zeile und ersten Spalte.

Range ...

Range("A1")				
	Range("B3:C4")			

- beschreibt eine Zelle, Zeile, Spalte oder Zellbereich.
- nimmt auf einen Zellbereich Bezug.
- kann durch die Angabe von Zellen und Spalten für eine Zelle beschrieben werden.

Zellbereich definieren

- Als Index wird die Zeilen- und Spaltenangabe einer Zelle in Form von Text genutzt. Konstanter Text wird in Anführungsstriche gesetzt.
- "A1". Die angegebene Zelle.
- "A1:B3". Ein zusammenhängender Zellbereich. Die linke obere Zelle steht links vom Doppelpunkt. Die rechte untere Zelle steht rechts vom Doppelpunkt.
- "A1, B2, C3". Nicht zusammenhängende Zellen.
- "A1:A3, B2:B5, C3:C6". Nicht zusammenhängende Zellbereiche.

ActiveSheet.UsedRange

- Der genutzte Zellbereich auf dem aktiven Arbeitsblatt.

Auflistungen von Objekten

- Zum Beispiel: Die Auflistung
  - Worksheets enthält alle Arbeitsblätter einer Arbeitsmappe.
  - Rows ist eine Auflistung aller Zeilen eines Arbeitsblattes.
- Auflistungen werden in VBA als Collections bezeichnet.
- Collections sind Sammlungen von bestimmten Objektarten.
- Der Name einer Collection endet immer mit einem „s“.

Elemente in einer Collection identifizieren

- Zum Beispiel: `Worksheets("Tabelle1"), Rows(1).`
- Die Elemente in einer Auflistung werden eindeutig durch die Angabe eines Indizes identifiziert.
- Als Index kann eine Ganzzahl oder der Name des Objekts genutzt werden.

Ganzzahl als Index nutzen

- Zum Beispiel: Rows(1).
- Das erste Element hat den Index eins, das zweite Element den Index zwei und so weiter.
- In diesem Beispiel wird die erste Zelle auf einem Tabellenblatt angesprochen.
- Der ganzzahlige Index ist von der Anzahl der Elemente in der Auflistung abhängig. Sobald ein Objekt nicht mehr zur Verfügung steht, wird das Objekt aus der Auflistung gelöscht und die Auflistung neu durchnummeriert. Alle Objekte nach dem gelöschten Objekt bekommen einen neuen Index.

Namen als Index nutzen

- Zum Beispiel: `Worksheets("Tabelle1")`
- Als Index wird der eindeutige Name des Elements genutzt.
- Der Name wird durch Anführungsstriche begrenzt. Der Name ist ein beliebiger Text in VBA.
- Diese Art von Index ist unabhängig von der Position des Elements in der Liste.

Anzahl der Elemente in einer Auflistung

```
zeileLastInArbeitsblatt = ActiveSheet.UsedRange.Rows.Count
```

- Die Methode Count gibt die Anzahl der Elemente in der Auflistung zurück.
- In diesem Beispiel wird die Anzahl der Zeilen im benutzten Bereich zurückgegeben.

Anweisungen ...

- werden aus Ausdrücken gebildet.
- entsprechen einer Zeile im Code-Fenster
- symbolisieren eine aufgezeichnete Makro-Aktion.

Beispiele

```
Sub Anweisung()

    zeileLastInArbeitsblatt = ActiveSheet.UsedRange.Rows.Count
     spalte = 1
     Set zelleAktuell = ActiveSheet.Cells(zeile, spalte)

End Sub
```

Ausdrücke ...

- bestehen aus Operanden und Operatoren, die nach bestimmten Regeln zusammengesetzt werden.
- berechnen mit Hilfe von Operatoren und Operanden einen neuen Wert.
- sind Verarbeitungsvorschriften, die ein Ergebnis zurückgeben.
- verändern den Wert von Variablen entsprechend des angegebenen Datentyps.

Beispiele

- Arithmetische Berechnung: preis * 0.16
- Vergleichsopratoren nutzen: messpunkt > 0
- Ausdrücke miteinander verknüpfen: (a >= b) AND (a >= c)
- Prozeduren aufrufen: Len(zeichenkette)

Variable Operanden

```
Sub Variablen()
    Dim lngSpalte As Long
    Dim strDatum As String
    Dim dateDatum As Date

    lngSpalte = 1
    strDatum = "01.12.2013"
    dateDatum = #01.12.2013#

End Sub
```

Konstante Operanden (Kursivschrift)

```
Sub Konstante()
    Dim lngSpalte As Long
    Dim einzahlung as Currency
    Dim strDatum As String
    Dim dateDatum As Date

    lngSpalte = 1
    strDatum = "01.12.2013"
    dateDatum = #01.12.2013#
    einzahlung = 1.4

End Sub
```

Operatoren

- Arithmetische Operatoren berechnen einen Wert.
  Beispiel: a + b.
- Vergleichsoperatoren vergleichen zwei Werte.
  Beispiel: a > b. Als Ergebnis liefert der Ausdruck einen boolschen Wert zurück.
- Logische Operatoren verknüpfen boolsche Werte. Boolsche Werte sind wahr oder falsch.
  Beispiel: (a >= 10) And (a <= 20) .
- Der Zuweisungsoperator weist einer Variablen einen Wert zu.

Arithmetische Operatoren

Operator	Rechenart	Beispiel
+	Positives Vorzeichen	ergebnis = +3
-	Negatives Vorzeichen	ergebnis = -3
+	Addition	ergebnis = 3 + 4
-	Subtraktion	ergebnis = 3 - 4
^	Potenzrechnung	ergebnis = 3^4

Arithmetische Operatoren

Operator	Rechenart	Beispiel
*	Multiplikation	ergebnis = 3 * 4
/	Division	ergebnis = 3 / 4 ergebnis = 0.75 Fehler = 3 / 0
\	Ganzzahlige Division	ergebnis = 3 \ 4 ergebnis = 0
Mod	Modula (Division mit Rest). Nur für Ganzzahlen.	ergebnis = 3 % 4 ergebnis = 3 ergebnis = 4 % 3 ergebnis = 1

Bedingte Anweisungen

```
If (Bedingung) Then
   Anweisung
End If
```

```
If spalte > 1 Then
      ' Anweisung
End If
```

Erläuterung

- Die Bedingung beginnt mit dem Kopf If (Bedingung) Then und endet mit dem Schlüsselwort End If. Der Ende-Befehl wird nicht automatisch ergänzt.
- Wenn If die Bedingung wahr ist, dann Then führe die Anweisungen aus.
- Wenn die Bedingung nicht wahr ist, werden die Anweisungen nicht ausgeführt.
- if-Anweisungen können verschachtelt werden.

Bedingungen ...

- sind Ausdrücke, die wahr oder falsch sind.
- vergleichen Werte.
- nutzen Vergleichsoperatoren.
- überprüfen das boolsche Ergebnis von Funktionen, die mit „Is“ beginnen.
- können verknüpft werden.
- werden häufig in runde Klammern gesetzt. Die runden Klammern dienen der besseren Lesbarkeit.

Vergleichsoperatoren

Operator	Erläuterung	Beispiel
=	ist gleich	3 = 4 ≈ *falsch*
<>	ungleich	3 <> 4 ≈ *richtig*
<	kleiner als	3 < 4 ≈ *richtig*
<=	kleiner gleich	3 <= 4 ≈ *richtig*
>	größer	3 > 4 ≈ *falsch*
<=	größer gleich	3 >= 4 ≈ *falsch*

Hinweise

- Alle Variablen in einer Bedingung sind von dem gleichen Datentyp. Der zu vergleichende Wert und der Vergleichswert sollten den gleichen Datentyp besitzen.
- Gleitkommazahlen (As Single, As Double) sollten nicht für Vergleiche „ist gleich“ genutzt werden. Gleitkommazahlen speichern immer nur einen Nährungswert.

Die Bedingung trifft nicht zu ...

```
If (Bedingung) Then
   Anweisung
Else
  Anweisung
End If
```

```
If IsNumeric(zellbereich.Value) Then
        ' Anweisung
Else
        ' Anweisung
End If
```

Erläuterung

- Wenn If die Bedingung wahr ist, dann Then führe die nachfolgenden Anweisungen aus. Andernfalls Else ...
- Der else-Zweig beschreibt den Standardfall. Was passiert, wenn alle vorhergehenden Bedingungen nicht zutreffen?
- Der else-Zweig ist optional.

Fallunterscheidung

```
If (Bedingung) Then
   Anweisung
ElseIf (Bedingung) Then
   Anweisung
Else
  Anweisung
End If
```

```
If IsNumeric(zellbereich.Value) Then
        ' Anweisung
 ElseIf IsDate(zellbereich.Value) Then
        ' Anweisung
 Else
        ' Anweisung
 End If
```

Erläuterung

- Wenn « If » die Bedingung wahr ist, dann « Then » führe die nachfolgenden Anweisungen aus. Andernfalls wenn « ElseIf » wahr ist, dann « Then » führe die dazugehörigen Anweisungen aus. Andernfalls « Else » ...
- « ElseIf » folgt immer einer if-Anweisung. Mit Hilfe dieses Schlüsselwortes können weitere bekannte Fälle definiert und behandelt werden.
- « ElseIf » ist optional.

Überprüfung von Variablen

- IsNumeric(zellbereich.Value). Ist der Wert als Zahl interpretierbar?
- IsDate(zellbereich.Value). Ist der Wert als Datum- oder Zeitwert interpretierbar?
- IsNull(zellbereich.Value). Ist der Wert der Variablen undefiniert? In VBA hat jede Variable einen definierten Standardwert.
- In den runden Klammern wird der zu überprüfenden Wert definiert. Der Funktion wird der zu überprüfende Wert als Variable oder Konstante übergeben.
- Wenn die Aussage zutrifft, wird als Rückgabe wahr (True) zurück gegeben. Andernfalls falsch (False).

Negation des Rückgabewertes

```
If Not (IsNull(ActiveSheet.Cells(zeile, spalte - 1))) Then
```

- Der Rückgabewert wird umgekehrt. True (Wahr) wird zu False (falsch) und umgekehrt.
- Falls die Zelle nicht leer ist, ...

Verknüpfung von Bedingungen

```
If (monat >= 1) And (monat <= 12) Then
```

- Beide Bedingungen müssen zutreffen.

Anweisungen wiederholen

- In Abhängigkeit einer bestimmten Bedingung.
- In Abhängigkeit eines vorgegebenen Zahlenbereichs.
- Verschachtelungen sind möglich.

Zählschleifen

```
For zählvariable = anfangswert To endwert Step schrittweite
    Anweisungen
 Next
```

```
For zeile = 1 To zeileLastInArbeitsblatt
        Anweisung
Next
```

- läuft von … bis.
- läuft eine genau festgelegte Anzahl von Durchläufen.

For zeile = 1 To zeileLastInArbeitsblatt.

- Schleifenkopf.
- Der Zählvariable wird ein Startwert zugewiesen (zeile = 1).
- Die Zählvariable läuft bis zu dem angegebenen Endwert (To zeileLastInArbeitsblatt.).

Next

- Die Zählvariable wird automatisch auf den nächsten Wert gesetzt. Standardmäßig wird die Zählvariable um eins erhöht.
- Falls der Wert kleiner gleich dem Endwert ist, wird die Schleife nochmals durchlaufen. Andernfalls wird die Schleife abgebrochen.

Schrittweite

```
For zeile = 1 To zeileLastInArbeitsblatt Step 1 …
For zaehler = 1 To 10 Step 2 …
For messwerte = 1 To maxWert Step 0.5 ...
```

- Ohne Angabe der Schrittweite wird die Zählvariable um eins erhöht.
- Dem Schlüsselwort Step kann die Angabe einer Schrittweite folgen. Die Schrittweite wird in Abhängigkeit des Datentyps der Zählvariablen angegeben. Eine ganzzahlige Zählvariable kann nicht mit einer Dezimalzahl als Schrittweite arbeiten.

Do-While-Schleifen

```
Do While Abbruchbedingung
        Anweisung
Loop
```

```
Do While (ActiveSheet.Cells(zeile, spalte).Value <> "")
        Anweisung
Loop
```

Do-Loop-Schleifen ...

- beginnen mit « Do ».
- beginnen mit « Loop » automatisiert einen neuen Schleifendurchlauf.
- können eine Abbruchbedingung haben, müssen aber nicht. Wenn keine Abbruchbedingung vorhanden ist oder diese nie erfüllt wird, läuft die Schleife endlos.

Abbruchbedingung « While » ...

```
Do While Abbruchbedingung
        Anweisung
Loop
```

```
Do
        Anweisung
Loop While Abbruchbedingung
```

- läuft solange die Bedingung erfüllt ist. Wenn die Bedingung nicht erfüllt ist, bricht die Schleife ab.
- kann im Kopf oder Fuß der Schleife stehen. Falls die Abbruchbedingung im Fuß steht, wird die Schleife mindestens einmal durchlaufen.