R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Suche nach Werten in Zellen

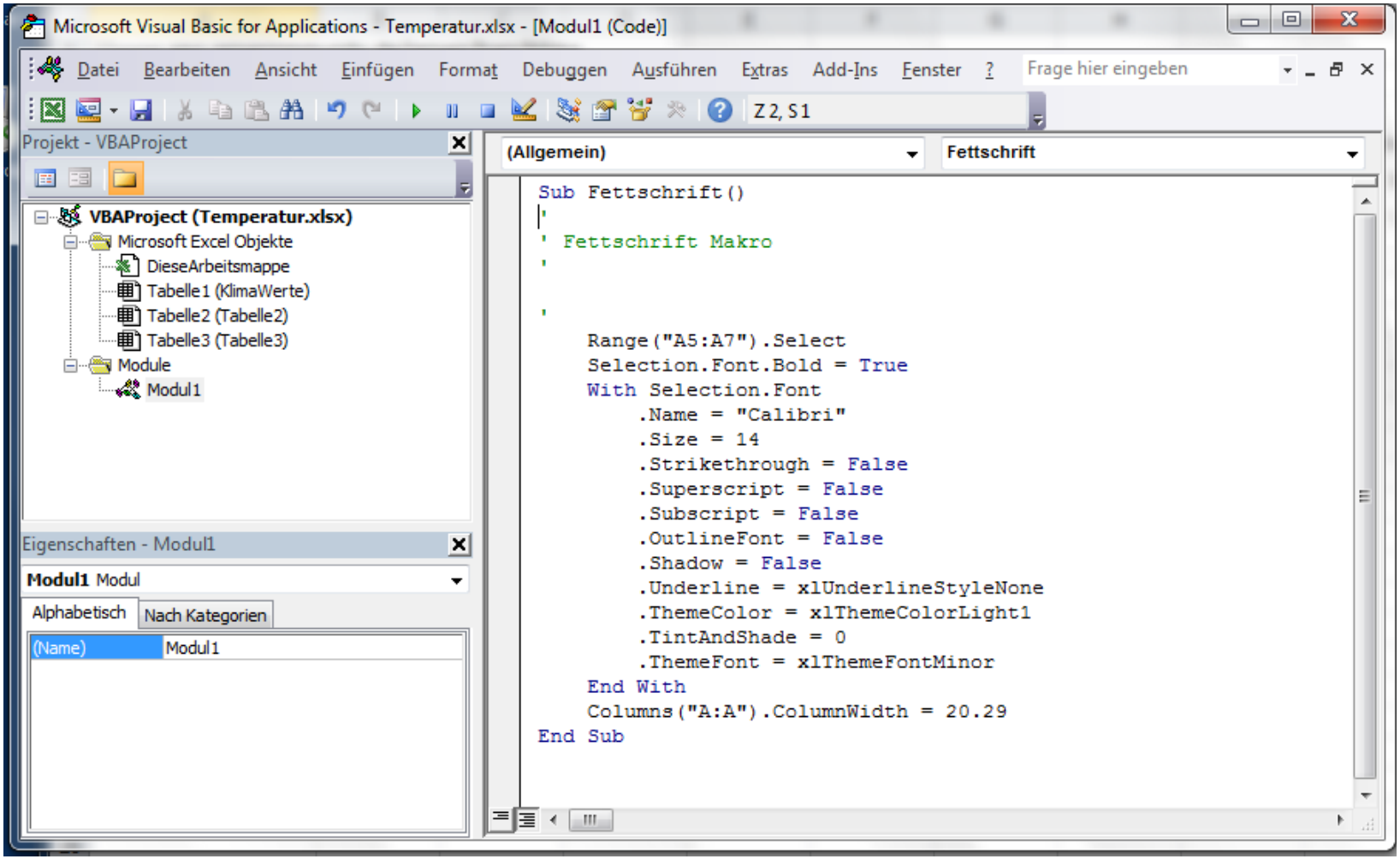
Aufgabe

- Suche eines Kunden mit Hilfe des Nachnamens und Vornamens.
- In Abhängigkeit des Suchergebnisses wird die Einzahlung in einer Monatsspalte vermerkt.
- Die Monatsspalte wird in Abhängigkeit des Einzahlungsdatums ermittelt.
- 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 ...

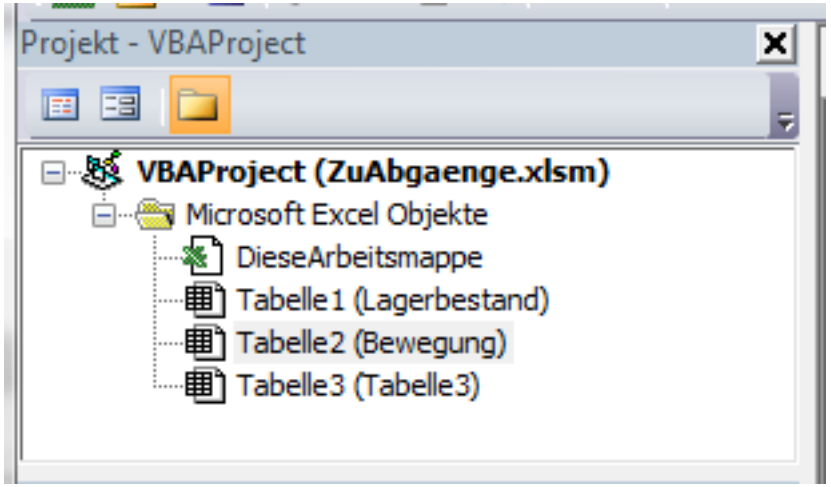
- befindet sich unterhalb der Titelleiste.
- sammelt alle Befehle der Anwendung.
- bietet aufklappbare Menüs zu verschiedenen Themen an. In diesen Menüs werden Befehle und Funktionen der Anwendung gesammelt.
- zeigt auf der obersten Ebene Kategorien an. In Abhängigkeit dieser Kategorien werden die Befehle 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 ...

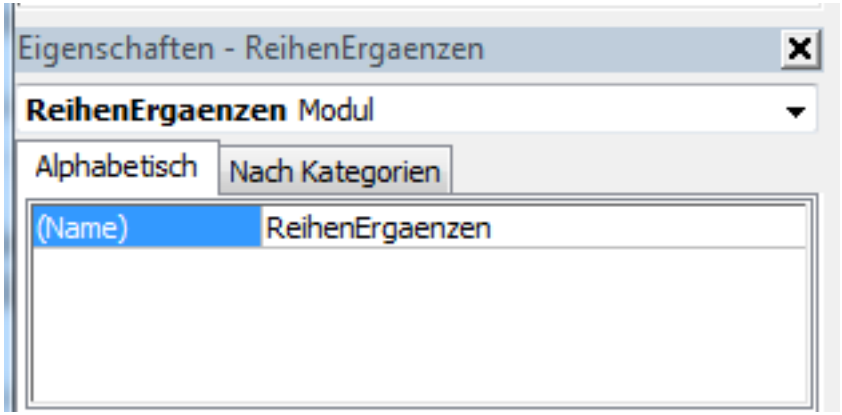
- werden mit Hilfe von Ordnern im Projekt-Explorer dargestellt.
- *(Microsoft Excel Objekte)* enthalten Module, die Code für die Arbeitsmappe oder die darin enthaltenen Arbeitsblätter enthalten.
- *(Module)* enthalten aufgezeichnete Makros oder vom Entwickler selbst geschriebene Prozeduren.

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.

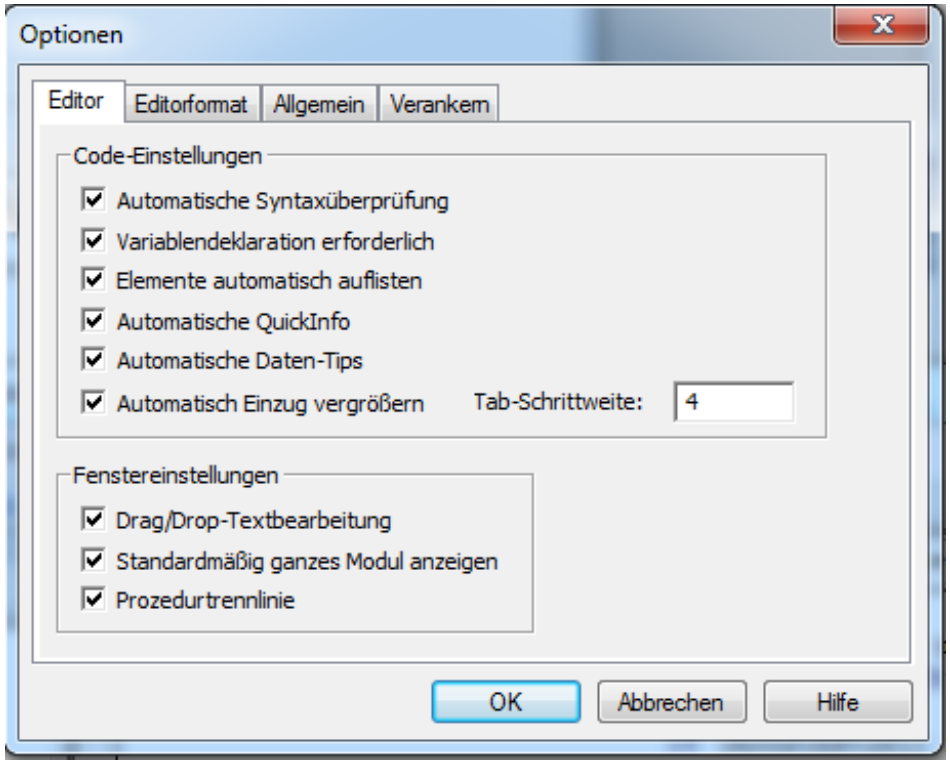


Anweisung „Option Explicit”

- 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*.



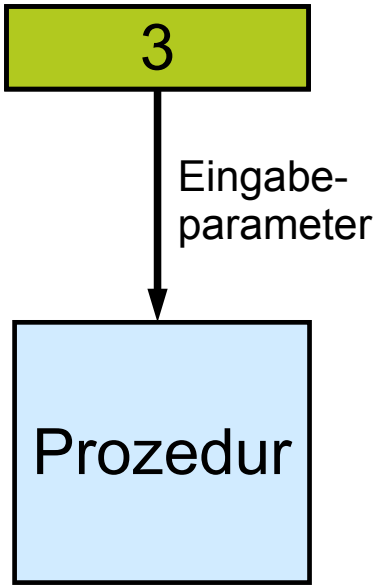
Der Rahmen eines „Makros” in dem Codefenster

```
Sub SucheName()


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.

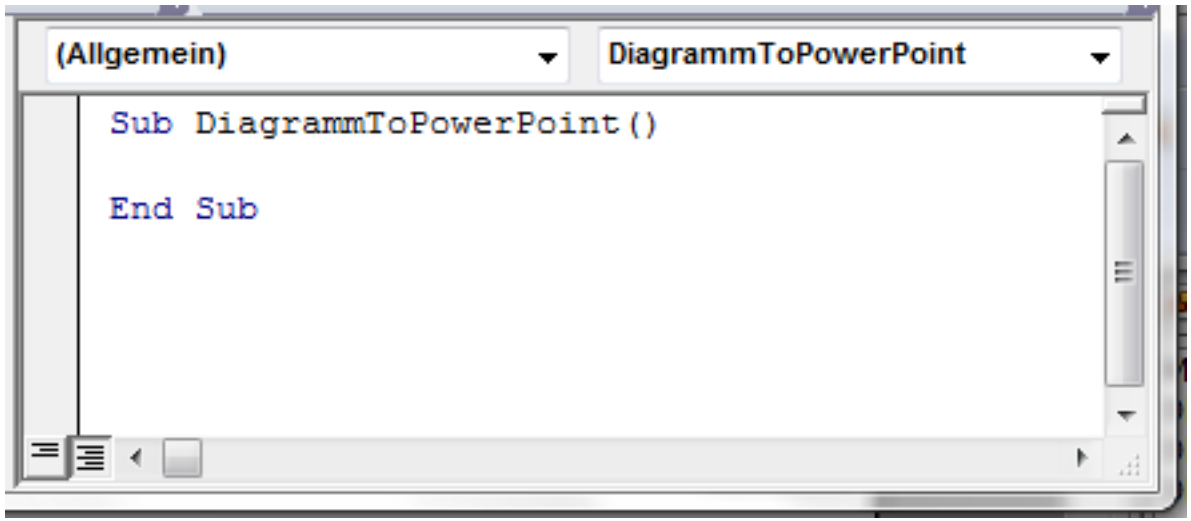
Arbeitsweise einer Prozedur

- 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 SucheName()


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 SucheName()
        ' 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 SucheName()
    Dim spalteVorname As Long
    Dim zeileEinzahlung As Long
    Dim spalteEinzahlung As Long
    Dim zeile As Long

    Dim strNachname As String
    Dim strVorname As String

    Dim aktuellDatum As Date
    Dim aktuellMonat As Byte
End Sub
```

Erläuterung

- Jede Variable hat einen eindeutigen Namen (zeile, aktuellDatum, 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 SucheName 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.

… für Datums und Zeitwerte

	Speicherbedarf	Datenbereich
As Date	8 Bytes	1.Januar 100 – 31. Dezember 9999 00:00:00 - 23:59:59

Hinweise

- Zeitwerte werden in dem Format Stunde : Minute : Sekunde eingegeben. Zeitangaben werden in einem 12- oder 24-Stunden-Format in Abhängigkeit des Computers dargestellt.
- Datumswerte werden intern als Integer-Zahl interpretiert. Zum Beispiel wird das Datum #7/9/2008# intern als 39638 gespeichert. Die Zählung der Tage beginnt am 30.12.1899. Datumswerte werden in Abhängigkeit der Ländereinstellungen und Formatierungen in Excel gespeichert.

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.

Konstanten ...

- haben einen festen Wert, der nicht verändert werden kann.
- sind Platzhalter für einen bestimmten Wert.
- werden auch als Literale bezeichnet.

… deklarieren

```
Sub VariablenZuweisen()
    Const anfangZeile As Byte = 5
End Sub
```

- Die Deklaration beginnt mit dem Schlüsselwort Const.
- Dem Schlüsselwort folgt der Name der Konstanten.
- In der Deklaration muss Konstanten mit Hilfe eines Gleichheitszeichens ein Wert zugewiesen werden.
- Der Bezeichnung kann die Angabe eines Datentyps folgen. Falls keine Angabe gemacht wird, legt der Wert rechts vom Gleichheitszeichen den Datentyp fest.

Konstante Werte

```
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
```

Systemdefinierte Konstanten ...

- werden von der Programmiersprache oder der Office-Anwendung bereitgestellt.
- Das Präfix vb kennzeichnet Konstanten der Programmiersprache VBA.
- Das Präfix mso kennzeichnet Konstanten der Office-Anwendung.
- Das Präfix xl kennzeichnet Konstanten der Office-Anwendung Excel.

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, ...)
```

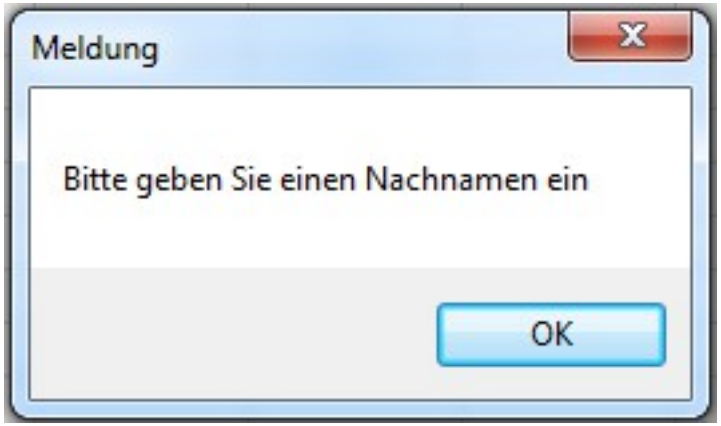
- Die Funktion wird mit Hilfe ihres Namens aufgerufen.
- Dem Namen folgen die runden Klammern. Die Klammern enthalten eine Liste von Argumenten. Die Anzahl der zu übergebenden Argumente ist abhängig von der Aufgabe der Funktion. Leere Klammern bedeuten, dass der Funktion keine Werte übergeben werden.
- 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.

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
```

Nachrichtenfenster

```
MsgBox ("Bitte geben Sie einen Nachnamen ein")
MsgBox Prompt:="Nachnamen fehlt", Title:="Meldung"
```



Erläuterung der Funktion

- Argumente:
  - prompt. Der im, Fenster angezeigte Text.
  - title. Eine Beschreibung in der Titelleiste.
- Rückgabewert vom Datentyp vbMsgBoxResult. Welche Schaltfläche wurde gedrückt?

Nutzung von benannten Argumenten

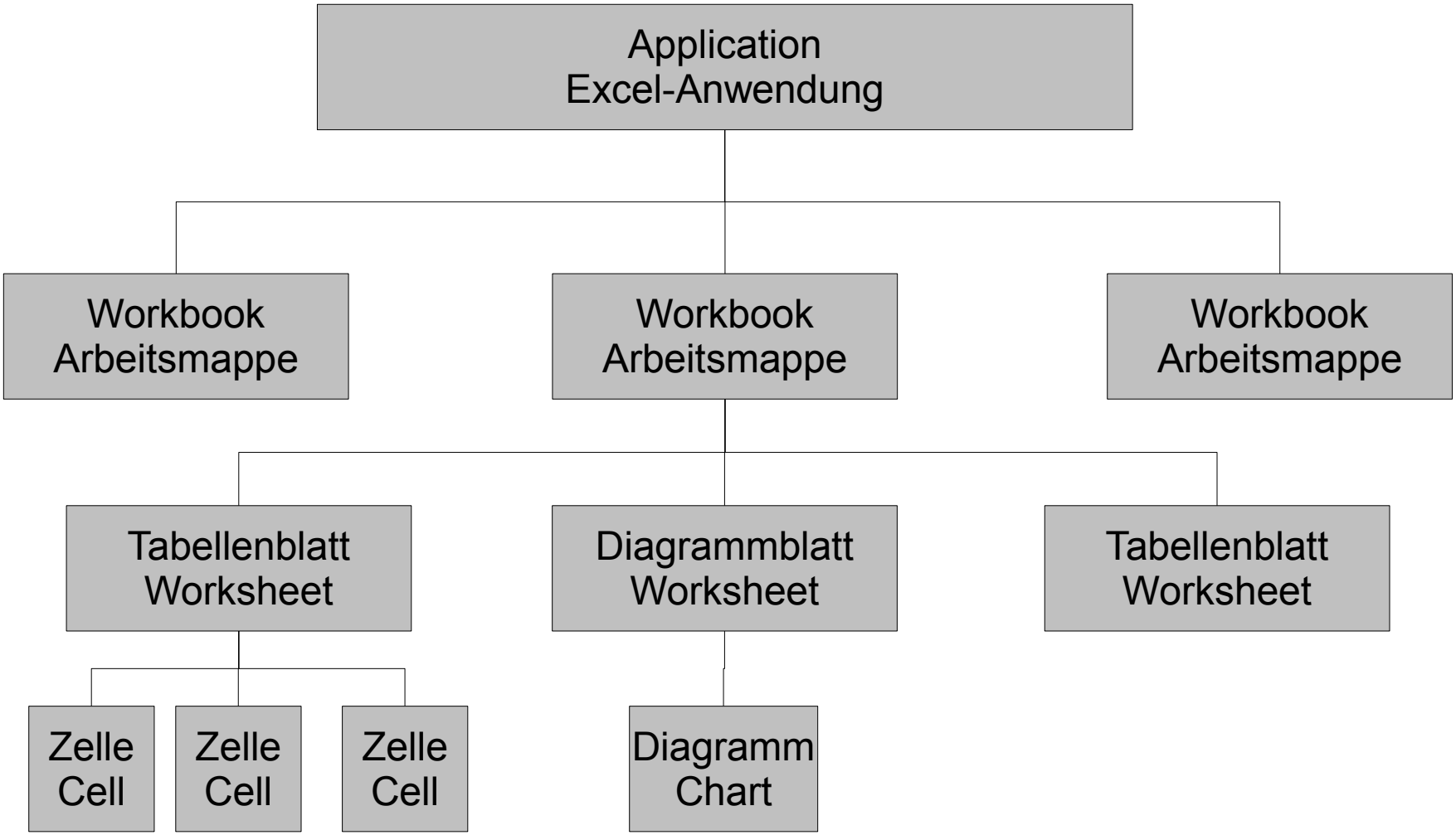
```
MsgBox ("Bitte geben Sie einen Nachnamen ein")
MsgBox Prompt:="Nachnamen fehlt", Title:="Meldung"
```

- Dem Namen des Arguments wird mit Hilfe des Operators := ein Wert zugewiesen.
- Die Zuweisung zu einem Argument ist unabhängig der Position in der Liste.
- Falls der Rückgabewert nicht genutzt wird, dürfen die Klammern der Argumentliste nicht gesetzt werden. Andernfalls wird der Fehler „erwartet =” angezeigt.

Objekte ...

- sind Bestandteile der Anwendung Excel oder anderer Office-Anwendungen.
- sind hierarchisch geordnet.
- haben Eigenschaften (Attribute) und Methoden (Funktionen).
- reagieren auf Aktionen des Benutzers mit Hilfe von Ereignissen.

Excel-Objektmodell



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 Objektvariable()
    Dim zellNachname As Range
    Dim suchbereichNachname As Range
    Dim suchbereichVorname As Range


End Sub
```

Erläuterung

- Jede Objektvariable hat einen eindeutigen Namen (zellNachname). 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. In diesem Beispiel kann auf die Variable kann nur innerhalb der Prozedur zugegriffen werden. Diese Prozedur ist Besitzer dieser Variablen.

Zuweisung an eine Objektvariable

```
Sub Zuweisung()
    Set zellNachname = ActiveSheet.Cells(zeile, spalte - 1)
    Set zellNachname = suchbereichNachname.Find(What:=strNachname,
                                                LookIn:=xlValues)
    Set zellNachname = 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.

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.

Cells ...

Cells(1)				
		Cells(4,3)		

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

Definition eines Zellbereichs

```
Set suchbereichNachname = ActiveSheet.Range(Cells(1, anfangZeile), Cells(zeile, 1))
```

- Mit Hilfe von .Cells wird die obere, linke und die untere, rechte Zelle definiert.
- Die Angaben werden durch ein Kommata getrennt.

Eigenschaften eines Objekts

- beschreiben ein Objekt.
- beeinflussen das Aussehen eines Objektes und deren Inhalt.
- sind Merkmale eines Objekts.
- werden durch einen Punkt vom Objekt getrennt.
- können mit dem Gleichheitszeichen einen Wert zugewiesen bekommen.
- haben in VBA englische Bezeichnungen.

Zelladresse

- .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.

Inhalt einer Zelle

- .Value speichert den Inhalt einer Zelle.
- Mit Hilfe des Zuweisungsoperators kann der Zelle ein neuer Wert zugewiesen werden.
- Die Formatierungen der Zelle werden nicht beachtet.

Formatierung einer Zelle

- .NumberFormat formatiert den Inhalt einer Zelle für die Anzeige.
- entspricht der Einstellungen des Registers *Zahlen* im Dialog *Zellen formatieren* (*Start* – *Zahlenformat* in Gruppe *Zahl*).
- kann ein benutzerdefiniertes Format zugewiesen bekommen (siehe http://office.microsoft.com/en-us/excel-help/create-or-delete-a-custom-number-format-HP010342372.aspx).

Einstellungsmöglichkeiten

- "General". Es wird die Standardformatierung genutzt.
- "@". In den Zellen wird Text angezeigt.
- "#,##0.000 $" beschreibt ein Währungsformat.
  - Das Format hat drei Dezimalstellen.
  - Das Dezimaltrennzeichen wird durch den Punkt beschrieben.
  - Das Tausender-Zeichen durch das Komma.
  - Vor dem Dezimaltrennzeichen muss mindestens eine Zahl stehen.

Anweisungen für ein Objekt zusammenfassen

```
ActiveSheet.Cells(zeileEinzahlung, spalteEinzahlung).NumberFormat =     "#,##0.00 $"
ActiveSheet.Cells(zeileEinzahlung, spalteEinzahlung).Value =
                                                    CCur(ActiveSheet.Range("G2").Value)
```

```
With ActiveSheet.Cells(zeileEinzahlung, spalteEinzahlung)
        .NumberFormat = "#,##0.00 $"
        .Value = CCur(ActiveSheet.Range("G2").Value)
End With
```

Zusammenfassungen für ein Objekt ...

- beginnen mit Hilfe der Schlüsselwort With und enden End With. Zwischen den beiden Schlüsselwörtern stehen Anweisungen, die sich auf ein bestimmtes Objekt beziehen.
- Dem Schlüsselwort With folgt der Objektname. In diesem Beispiel ActiveSheet.Cells(zeileEinzahlung, spalteEinzahlung).
- Alle Anweisungen zwischen With und End With, die mit einem Punkt beginnen, beziehen sich auf das angegebene Objekt.

Methoden

- beschreiben das Verhalten eines Objekts.
- verändern oder lesen Attribute des dazugehörigen Objekts.
- sind Prozeduren, die an ein Objekt gebunden sind.
- werden mit Hilfe des Punktes an ein Objekt gebunden.

Beispiel

```
Set zellNachname = suchbereichNachname.Find(What:=strNachname,
                                            LookIn:=xlValues)
Set zellNachname = suchbereichNachname.FindNext(zellNachname)


Set suchbereichVorname = zellNachname.Offset(0, 1)
```

- Methoden werden mit ihren Namen aufgerufen.
- Dem Namen folgt in runden Klammern eine Argumentliste. Die Liste kann leer sein.
- Die Listenelemente werden durch ein Kommata getrennt.

Suche nach Werten

```
Set zellNachname = suchbereichNachname.Find(What:=strNachname,
                                            LookIn:=xlValues)

Set zellNachname = suchbereichNachname.FindNext(zellNachname)
```

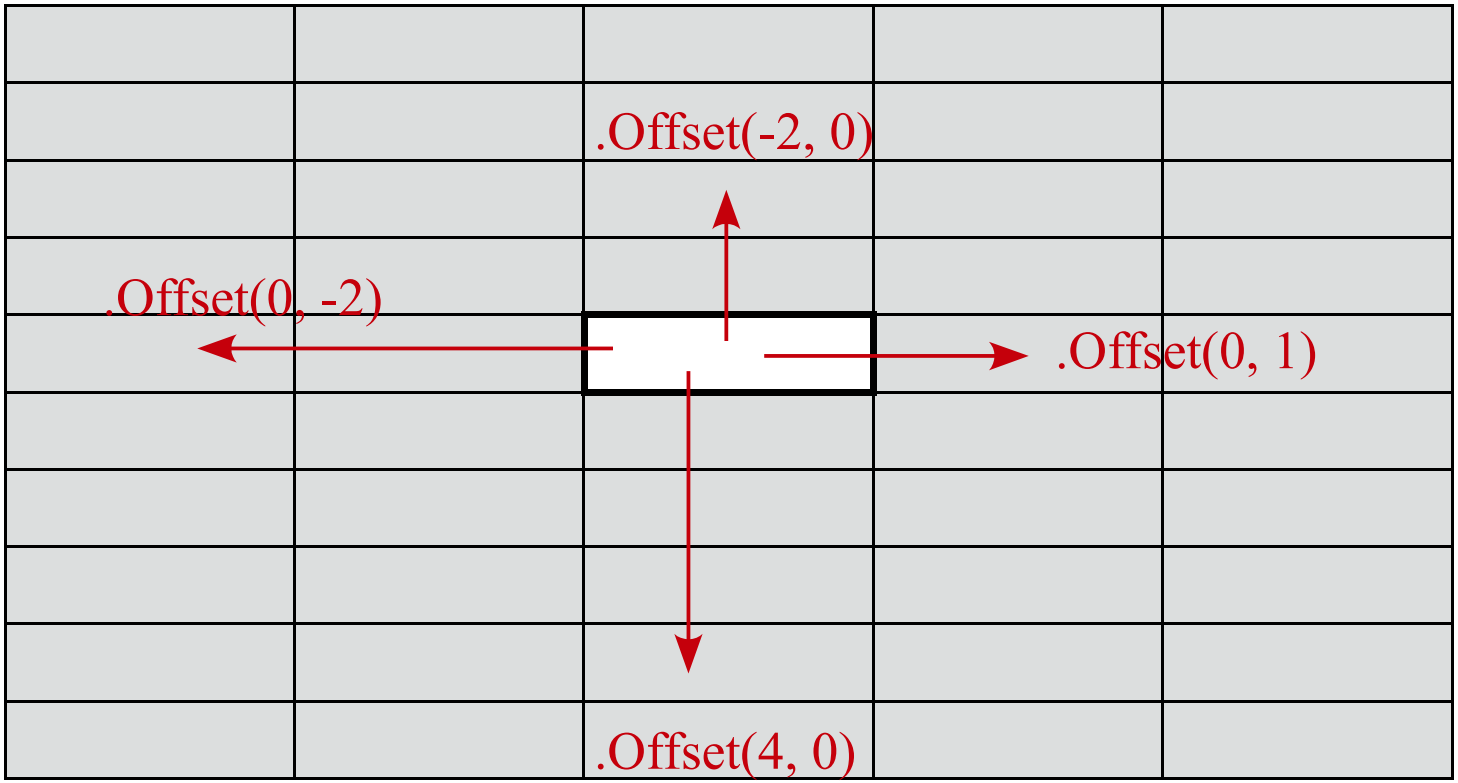
- Find sucht nach einem bestimmten Begriff in einem Zellbereich.
  - What enthält den Suchbegriff.
  - LookIn gibt über den Typ, nach dem gesucht wird, Auskunft.
- FindNext setzt an der übergebenen Position die Suche fort.

In Abhängigkeit einer Postion ...

```
Set suchbereichVorname = zellNachname.Offset(0, 1)
```

- In Abhängigkeit einer bestimmten Zelle (zellNachname) wird zu einer anderen Zelle gegangen.
- Der Zellbereich wird in Abhängigkeit eines anderen Zellbereichs gewählt.

Beispiel



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.

Beispiel „Arbeitsblatt”

```
Sub Objektvariable()
    Dim arbeitsblatt As Worksheet

    Set arbeitsblatt = ActiveSheet                  ' Aktives Arbeitsblatt
    Set arbeitsblatt = Worksheets("Tabelle1")       ' Aufruf von Tabelle1
    Set arbeitsblatt = Worksheets(2)                ' Das zweite Arbeitsblatt
End Sub
```

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. Mit Fettschrift wird das aktive Arbeitsblatt gekennzeichnet.

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).

is nothing

```
If zellNachname Is Nothing Then ...
```

- Die Objektvariable verweist auf kein Objekt.
- Die Variable enthält keinen Verweis.

Negation des Rückgabewertes

```
If (strVorname <> "") Then …
If Not(strVorname = "") Then ...
```

- Der Rückgabewert wird umgekehrt. True (Wahr) wird zu False (falsch) und umgekehrt.
- Falls die Variable nicht leer ist, … Die Variable muss einen String enthalten.

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.

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.

Exit

```
Exit Sub        ' Prozedur vorzeitig verlassen
Exit Do         ' Do-While-Schleife vorzeitig verlassen
```

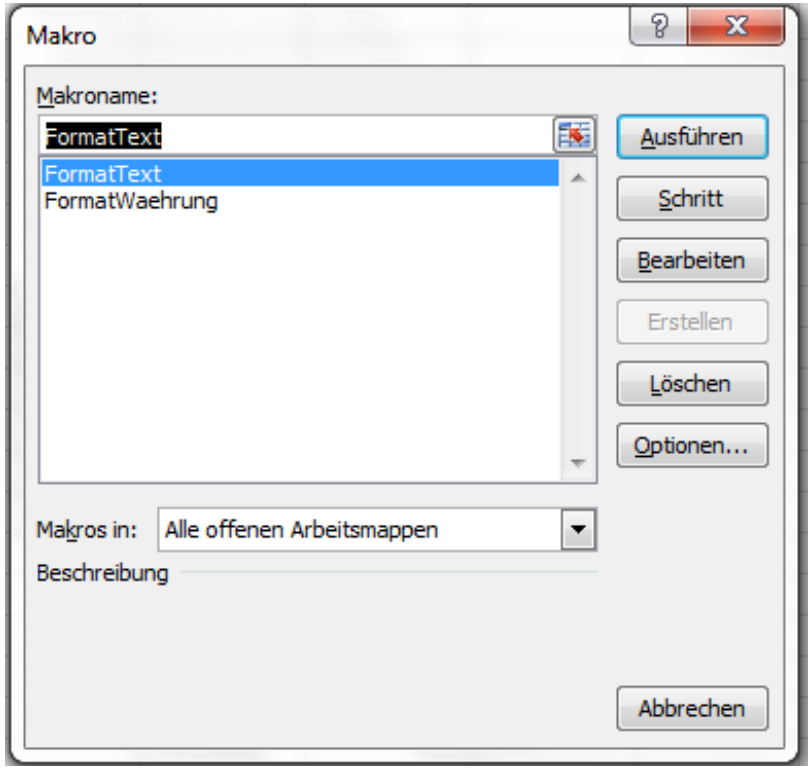
- Mit Hilfe des Schlüsselwortes Exit für eine Schleife oder eine Prozedur abgebrochen und verlassen.

„Makros” starten

- über das Menüband *Entwicklertools* in der Anwendung.
- über eine eigenes Menüband in der Anwendung.
- über eine Schaltfläche in der Arbeitsmappe.
- über das Menü *Ausführen* des VBA-Editors.

… über die Entwicklertools starten

- Das Menüband *Entwicklertools* ist aktiv.
- Mausklick auf das Icon *Makros* in der Gruppe Code.
- In dem Dialogfenster Makro wird aus der Liste ein Makro ausgewählt. Die Schaltfläche *Ausführen* startet das markierte Makro.



… über ein eigenes Menüband

- Erstellung eines neuen Menübandes.
- Eventuell: Unterteilung des Menübandes in Gruppen für die verschiedenen Makros.
- Makro hinzufügen

Neues Menüband anlegen

- *Datei – Optionen.*
- Die Kategorie *Menüband anpassen* wird ausgewählt.
- In der Liste Menüband anpassen werden alle Hauptregisterkarten angezeigt.
- Mit Hilfe der Schaltfläche *Neue Registerkarte* wird ein neues Menüband plus eine neue Gruppe innerhalb des Menübandes angelegt. Das neue Menüband wird hinter der, in der Liste markierten Registerkarte eingefügt.
- Die neue Registerkarte wird in der Liste ausgewählt. Mit Hilfe der Schaltfläche *Umbenennen* wird dem neuen Menüband ein „sprechender“ Name gegeben.

Neue Gruppe umbenennen

- Jedes Menüband hat mindestens eine Gruppe.
- Die neue Gruppe wird in der Liste ausgewählt. Mit Hilfe der Schaltfläche *Umbenennen* wird der Gruppe ein „sprechender“ Name gegeben. Zusätzlich kann die Gruppe durch ein Log gekennzeichnet werden.

Weitere Gruppen hinzufügen

- Mit Hilfe der Schaltfläche *Neue Registerkarte* wird ein neues eine neue Gruppe innerhalb des markierten Menübandes angelegt.
- Die neue Gruppe wird hinter der, in der Liste markierten Gruppe eingefügt.

Makros einer Gruppe hinzufügen

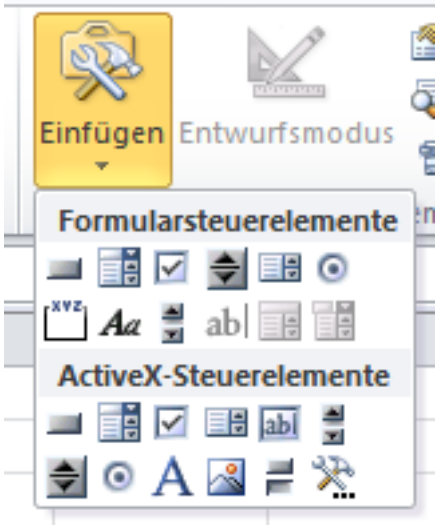
- In dem Kombinationsfeld Menüband anpassen ist eine Hauptregisterkarte markiert. In dem gewählten Menüband wird die passende Gruppe für das neu hinzufügende Makro ausgewählt.
- In dem Kombinationsfeld Befehle auswählen ist das Element Makros ausgewählt. In der Liste darunter werden alle Makros angezeigt.
- Mit einem Mausklick wird das gewünschte Makro ausgewählt.
- Mit Hilfe der Schaltfläche *Hinzufügen* wird ein Makro der gewählten Gruppe rechts hinzugefügt.

… über eine Schaltfläche im Arbeitsblatt

- Das Makro wird aufgezeichnet oder im VBA-Editor geschrieben.
- Mit Hilfe der *Entwicklertools* wird eine Schaltfläche in der Arbeitsfläche abgelegt und mit einem Makro verbunden.
- Mit einem Mausklick auf die Schaltfläche wird das Makro gestartet.

Steuerelemente

- Das Menüband *Entwicklertools* ist aktiv.
- Öffnen des Menüs *Einfügen* in der Gruppe Steuerelemente.



Formularsteuerelemente (Controls) ...

- sind Platzhalter für die Eingabe von Informationen.
- bieten verschiedene Eingabe- und Auswahlmöglichkeiten zu einem Fragenkomplex an.
- können Aktionen starten.
- zeigen Bilder an.
- werden in den *Entwicklertools* als Icon dargestellt.

Schaltfläche einfügen

- Das Menüband *Entwicklertools* ist aktiv.
- Öffnen des Menüs *Einfügen* in der Gruppe Steuerelemente.
- Mausklick auf die *Schaltfläche* in der Gruppe Formularsteuerelemente (1. Steuerelement in der ersten Reihe).
- An einer bestimmten Position in der Arbeitsmappe wird mit Hilfe der gedrückt gehaltenen linken Maustaste ein Rahmen aufgezogen.
- Sobald die Maustaste losgelassen wird, wird das Dialogfenster Makro zuweisen geöffnet. Das zu startende Makro wird aus der ausgewählt.

Makro zuweisen

- Im Dialogfenster Makro zuweisen wird in der Liste das gewünschte Makro markiert.
- *OK* schließt das Dialogfenster.
- Anschließend wird die Schaltfläche an der gewünschten Position eingefügt.

Beschriftung der Schaltfläche ändern

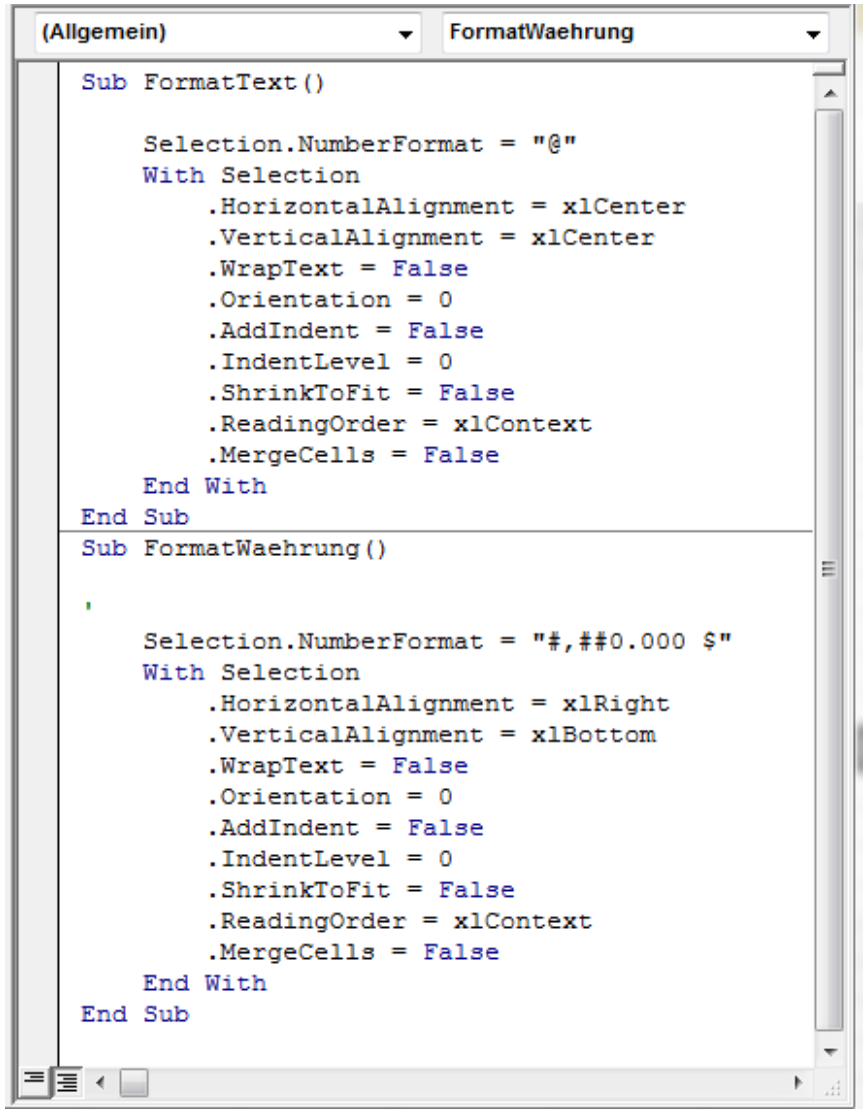
- Mit Hilfe eines Rechtsklicks auf die Schaltfläche kann der Bearbeitungsmodus aktiviert werden. Die Schaltfläche wird in einem Auswahlrahmen angezeigt. In den Ecken sowie mittig auf den Kanten werden „Anfasser” eingeblendet.
- Mit Hilfe der Maus wird der Text auf der Schaltfläche markiert.
- Der markierte Text kann mit Hilfe der Tastatur überschrieben werden.
- Sobald mit der Maus außerhalb der Schaltfläche auf die Arbeitsmappe geklickt wird, wird der Bearbeitungsmodus für die Schaltfläche deaktiviert.

im VBA-Editor

- Das Makro wird im Code-Fenster gestartet.
- Die Funktionalität des Makros wird getestet.

Codefenster ...

- zeigt Schlüsselwörter aus VBA in blauer Schrift an.
- zeigt die Arbeitsschritte eines Makros zeilenweise in der Programmiersprache VBA an.
- sammelt Arbeitsschritte zu einem Thema in einer Prozedur.



Objekt-Kombinationsfeld ...

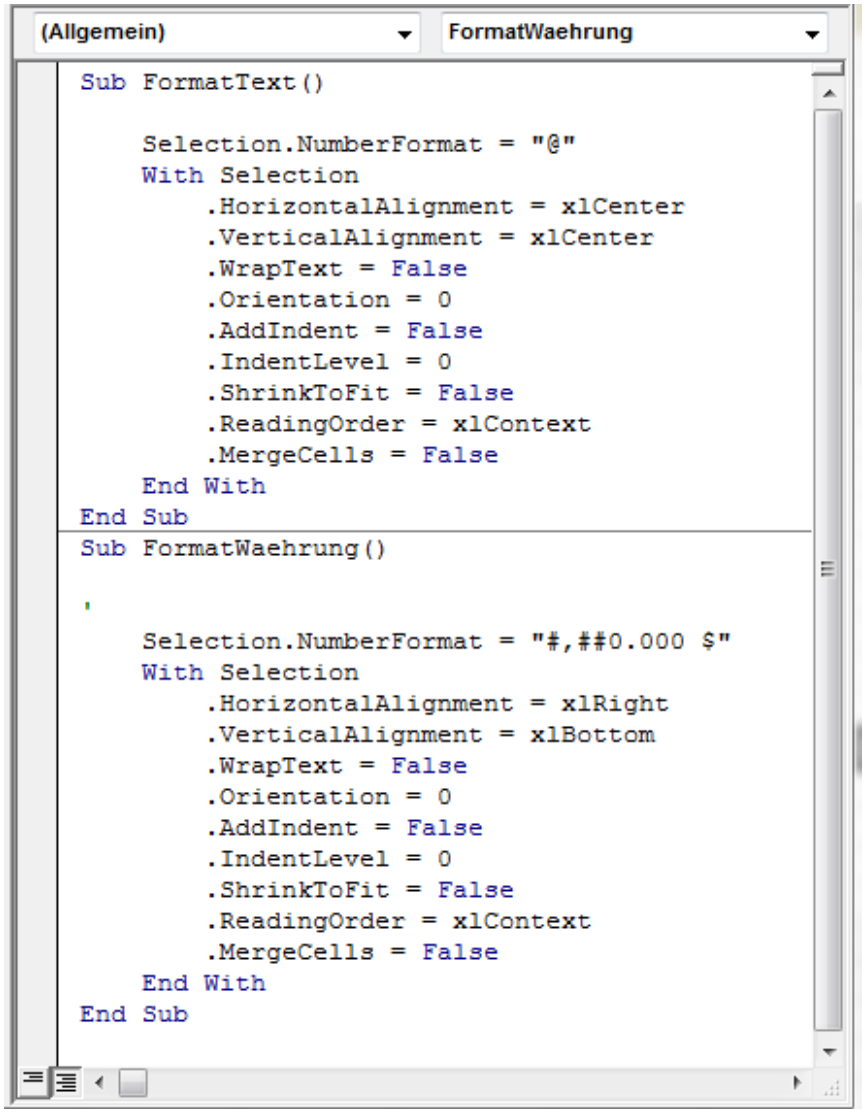
- befindet sich am linken, oberen Rand des Codefensters.
- enthält das Element (Allgemein). Module, die bei der Erstellung eines Makros erzeugt wurden, enthalten nur diesen Eintrag.
- zeigt die Steuerelemente einer benutzerdefinierten Form an.
- zeigt das aus dem Projekt-Explorer gewählte Excel-Objekt an.

Prozedur-Kombinationsfeld ...

- befindet sich am rechten oberen Rand des Codefensters.
- zeigt alle Prozeduren zu den, im linken Kombinationsfeld gewählten Objekt an.
- listet alle Ereignisse zu dem gewählten Objekt auf. Ereignisse reagieren auf Benutzeraktionen wie zum Beispiel „Mausklick“.
- hat immer den Eintrag (Deklaration). Dieser Eintrag enthält Anweisungen, die das gesamte Modul betreffen. Die Anweisungen befinden sich meist am Anfang des Moduls im Codebereich.

Code starten

- Die Einfügemarke befindet sich zwischen den Schlüsselwörtern Sub und End Sub.
- Mit Hilfe von <F5> oder *Ausführen – Makro ausführen* wird das Makro vollständig durchlaufen.

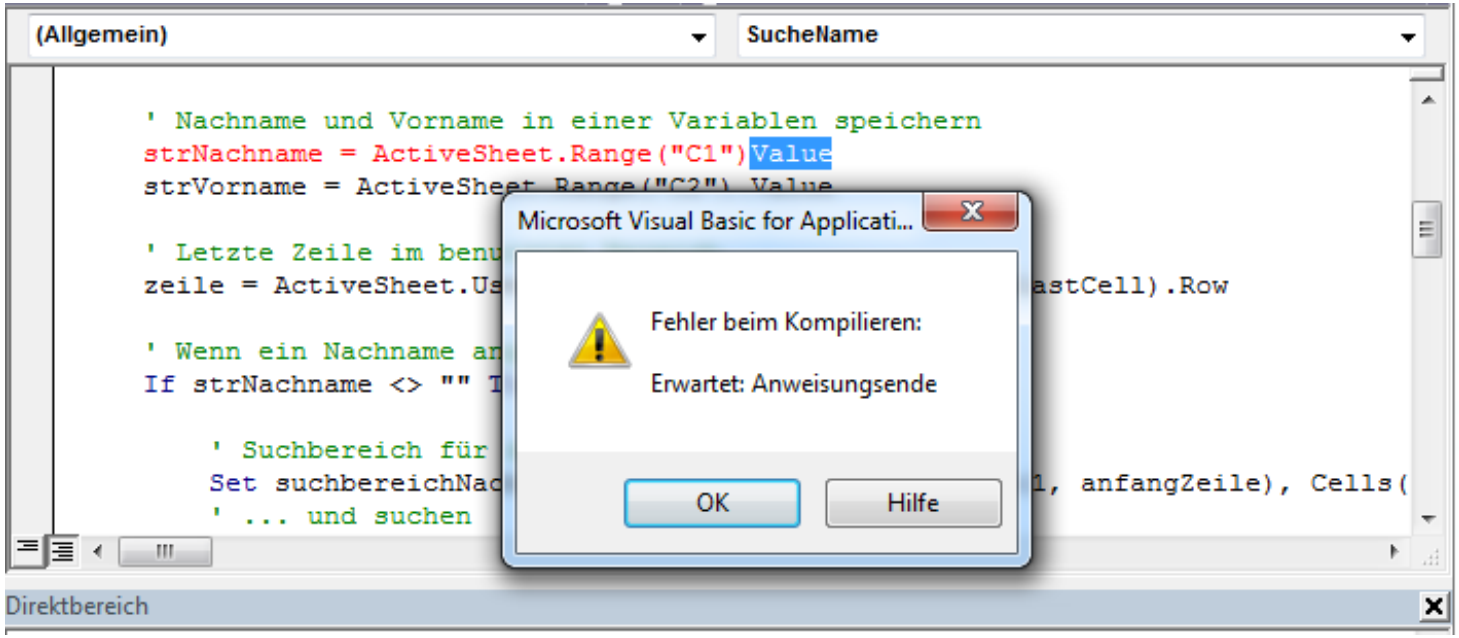


Fehler in VBA

- Syntaxfehler entstehen beim Schreiben des Programmcodes. Die Fehler entsprechen Grammatikfehlern. Diese Fehler werden vom Programm angezeigt.
- Logische Fehler führen zu einem falschen Ergebnis. Meist wurde bei der Umsetzung ein Denkfehler gemacht.
- Laufzeitfehler treten bei der Ausführung des Programms auf. Zum Beispiel das angegebene Netzlaufwerk ist nicht vorhanden.

Syntaxfehler ...

- sind Grammatikfehler in der Syntax von VBA.
- werden in VBA rot gekennzeichnet.
- halten das Programm an.



Automatische Syntaxüberprüfung einschalten

- Klick auf das Menü *Extras* – *Optionen* im VBA-Editor.
- Die Registerkarte Editor ist aktiv.
- Das Kontrollkästchen Automatische Syntaxüberprüfung wird aktiviert.

Deklaration von Variablen erzwingen

- Klick auf das Menü *Extras* – *Optionen* im VBA-Editor.
- Die Registerkarte Editor ist aktiv.
- Das Kontrollkästchen Variablendeklaration erforderlich wird aktiviert.
- In jedes neue Modul wird automatisch die Anweisung Option Explicit an den Anfang gesetzt.

Logische Fehler ...

- werden durch falsch definierte Anforderungen erzeugt.
- entstehen durch ein fehlerhaftes Design.
- werden durch eine falsche Kommunikation zwischen Entwickler und Auftraggeber verursacht.
- können durch ein Debuggen des Programms gefunden werden.

Ausführen bis Cursor-Position

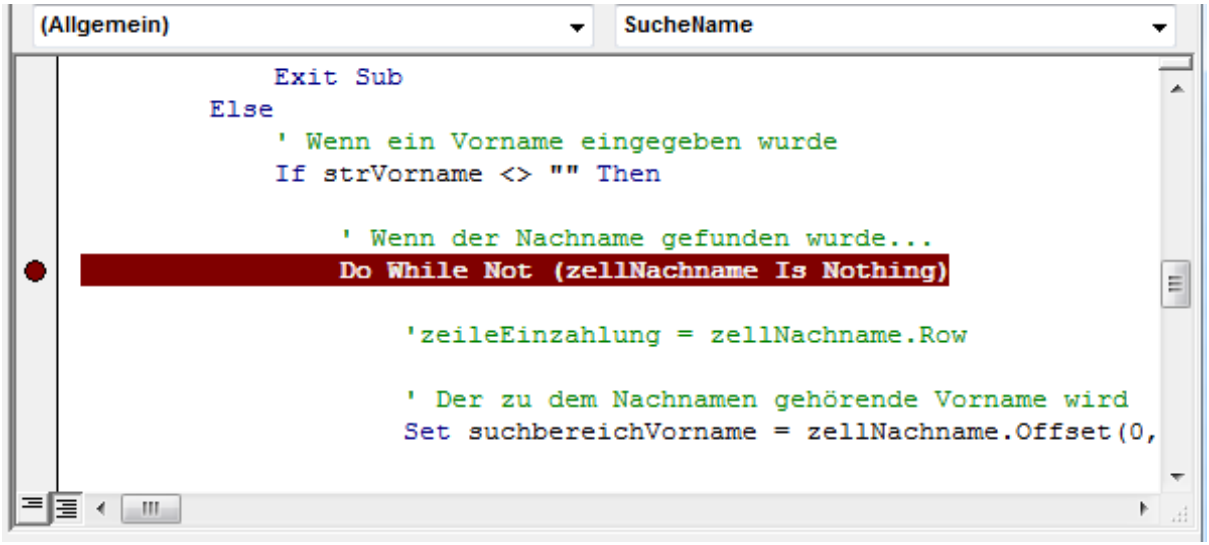
- Die Einfügemarke steht in einer bestimmten Anweisung. Bis zu diesem Punkt tritt kein Fehler auf.
- Das Programm wird mit Hilfe von <STRG>+<F8> oder *Debuggen – Ausführen bis Cursor-Position* gestartet.
- Das Programm wird bis zur Einfügemarke vollständig durchlaufen. An der Einfügemarke wird das Programm automatisch gestoppt.

Haltepunkte ...

- stoppen das Programm in einer bestimmten Zeile.
- ermöglichen die Untersuchung einer bestimmten Zeile.
- Ab einem Haltepunkt kann das Programm Schritt für Schritt durchlaufen werden.
- können nur auf ausführbare Anweisungen gesetzt werden.

… setzen

- Mit Hilfe von <F9> (*Debuggen – Haltepunkte ein / aus*) können Haltepunkte in der aktuellen Programmzeile gesetzt werden. Die Einfügemarke kennzeichnet die aktuelle Zeile.
- Ein Haltepunkt wird mit einem braunen Punkt am linken Rand gekennzeichnet. Die Programmzeile wird braun unterlegt.

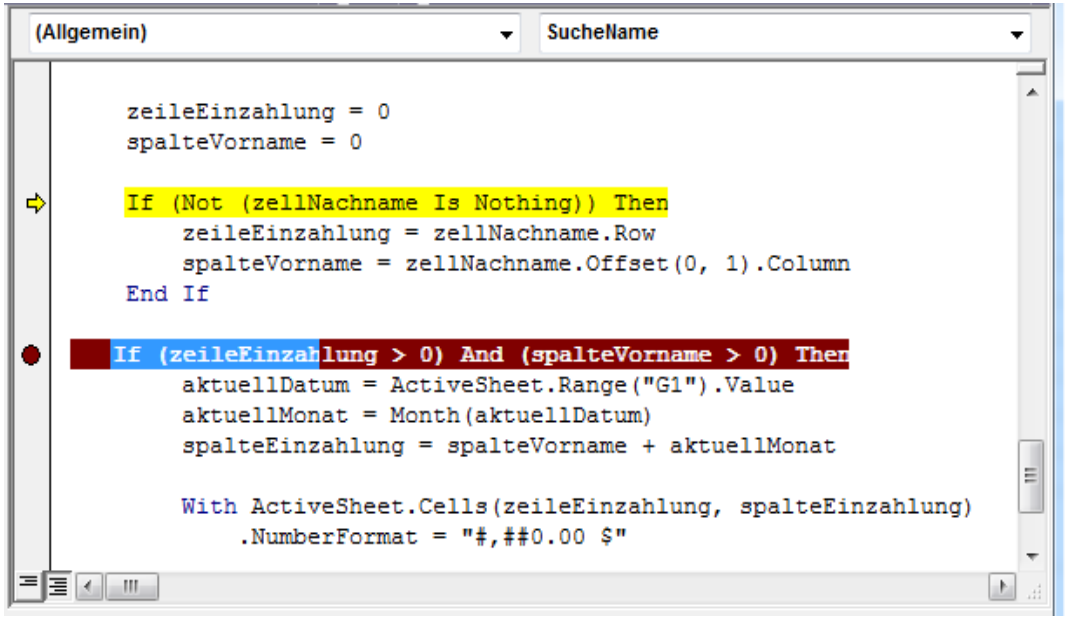


… löschen

- Mit Hilfe von <F9> (*Debuggen – Haltepunkte ein / aus*) können Haltepunkte in der aktuellen Programmzeile gelöscht werden. Die Einfügemarke kennzeichnet die aktuelle Zeile.
- *Debuggen – Alle Haltepunkte löschen* löscht alle Haltepunkte in einem Programm.

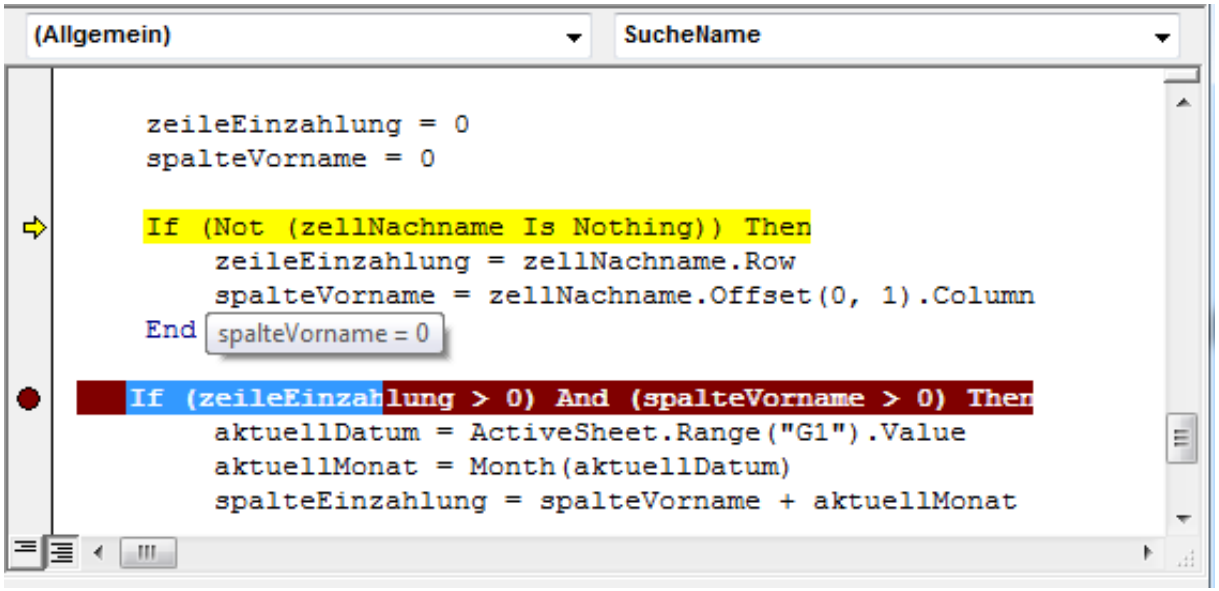
Einzelschrittmodus nutzen

- Mit Hilfe von <F8> wird das Programm Zeile für Zeile durchlaufen.
- Die aktuelle Zeile wird gelb hinterlegt und am linken Rand des Codefenster mit einem gelben Pfeil gekennzeichnet.



Wert einer Variablen anzeigen

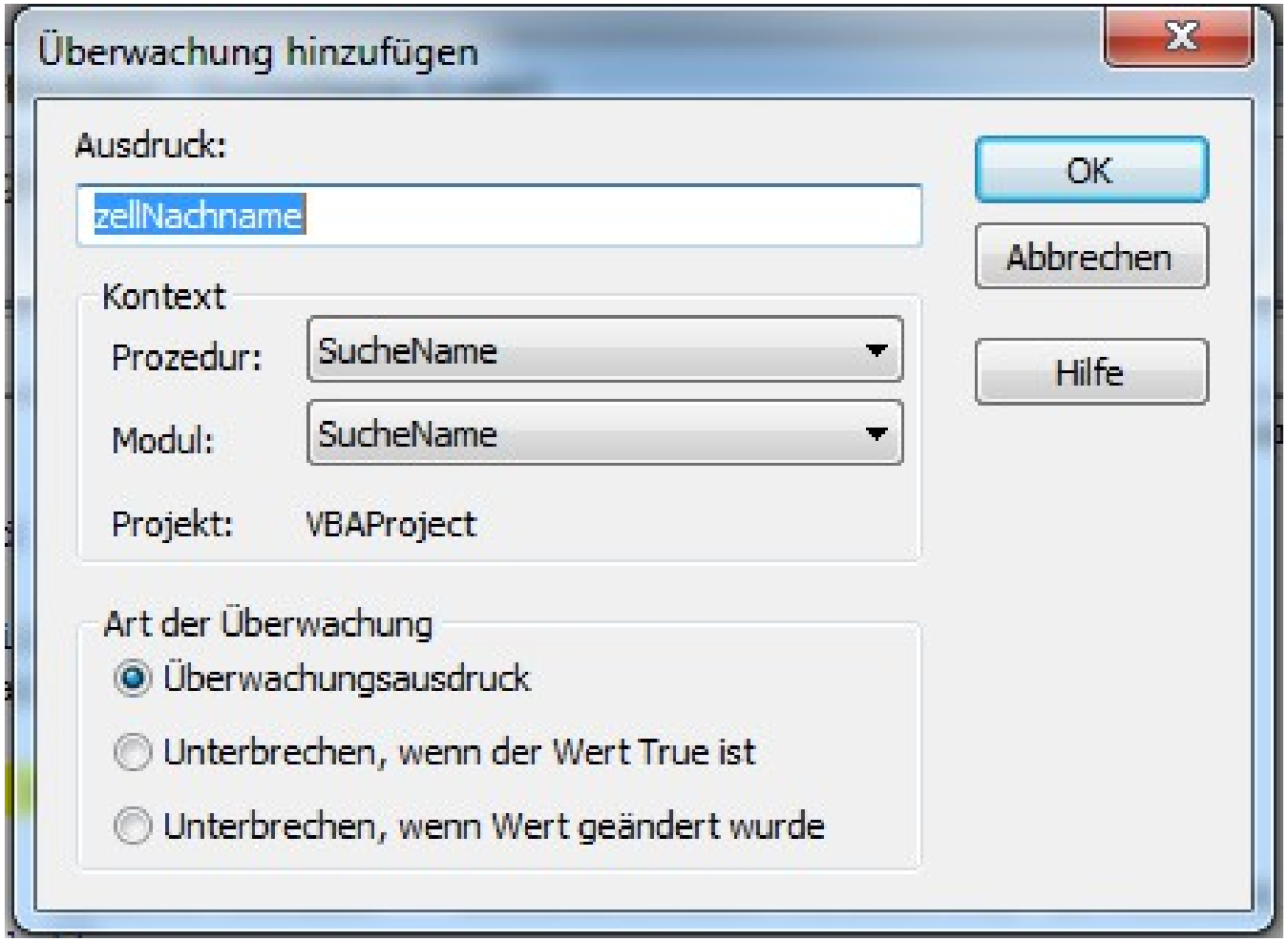
- Voraussetzung: Das Programm wird im Einzelschrittmodus durchlaufen.
- Sobald der Mauszeiger über einer Variablen liegt, wird der Wert der Variablen als Quick-Tipp angezeigt.



Wert einer Variablen ständig überprüfen

- Mit Hilfe der linken Maustaste wird die zu überprüfende Variable im Codebereich markiert.
- Anschließend wird der Menübefehl *Überwachung hinzufügen* im Kontextmenü der markierten Variablen ausgeführt. Das Kontextmenü selber wird mit Hilfe der rechten Maustaste geöffnet.
- Das Dialogfenster Überwachung hinzufügen wird geöffnet.

Dialogfenster „Überwachung hinzufügen”



Erläuterung

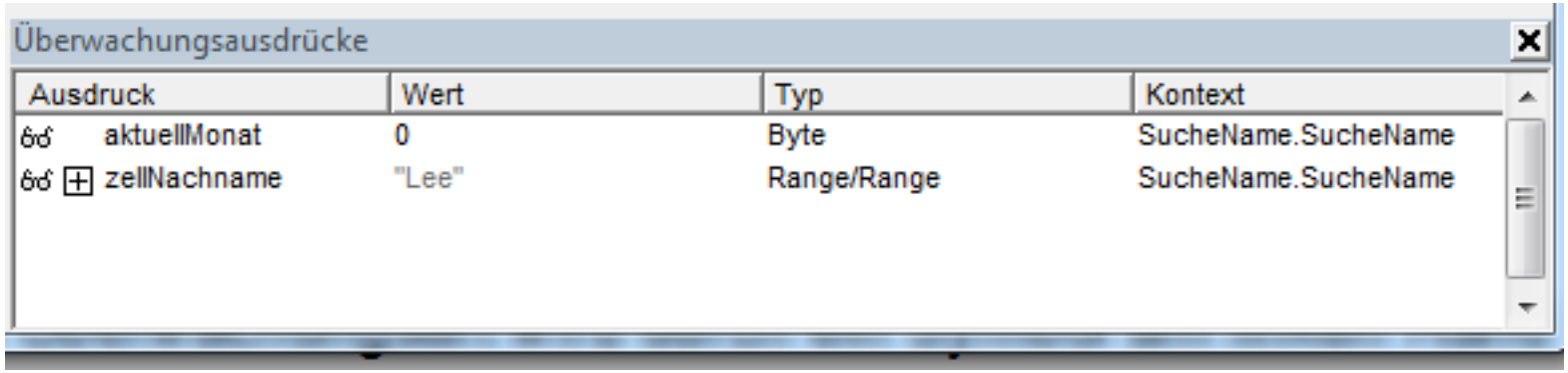
- Im oberen Textfeld wird der zu überwachende Variable angezeigt.
- Die Prozedur und das Modul, in dem die Variable definiert ist, werden darunter angezeigt.
- Die Art der Überwachung wird mit Hilfe von Optionsfeldern ausgewählt.
- Das Fenster wird mit Hilfe der Schaltfläche *OK* geschlossen.

Überwachungsarten

- Überwachungsausdruck ist die Standardeinstellung. Die ausgewählte Variable und deren Wert wird im Dialogfenster Überwachungsausdrücke angezeigt.
- Unterbrechen, wenn der Wert True ist. Das laufende Programm wird unterbrochen, wenn der Ausdruck wahr liefert. Diese Option kann nicht für Strings (Zeichenketten) genutzt werden.
- Unterbrechen, wenn der Wert geändert wurde. Das laufende Programm wird unterbrochen, wenn der Wert der Variablen sich verändert.

Überwachungsfenster

- Sobald eine Variable überwacht wird, wird das Fenster Überwachungsausdrucke angezeigt.
- *Ansicht – Überwachungsfenster* zeigt das geschlossene Fenster an.



Überwachungsausdrücke

- Die Überwachungsart als Symbol.
- Der Name des zu überwachenden Ausdrucks. Mit einem Klick auf das Kreuz werden weitere Informationen wie Eigenschaften eines Objekts ein- oder ausgeblendet.
- Der Wert der Variablen.
- Der Datentyp der Variablen.
- Wo befindet sich die Variable?

… entfernen

- Mit Hilfe eines Mausklicks auf das Symbol am rechten Rand wird die gesamte Zeile markiert.
- <ENTF> löscht die markierte Überwachung.

… hinzufügen

- Der zu überwachende Ausdruck wird markiert.
- Mit Hilfe der gedrückten Maustaste wird der Ausdruck in das Überwachungsfenster gezogen werden.
- Sobald die Maustaste losgelassen wird, wird der Ausdruck im Fenster angezeigt.

Überwachung bearbeiten

- Die Variable wird mit Hilfe der Maus aktiviert.
- Der Menübefehl *Debuggen - Überwachung bearbeiten* wird ausgewählt.
- Das Dialogfenster Überwachung bearbeiten wird geöffnet.
- Die vorhandenen Einstellungen können verändert und mit *OK* gespeichert werden.
- Mit Hilfe der Schaltfläche *Löschen* kann eine Überprüfung entfernt werden.

Arbeitsmappe mit einem Makro speichern

- *Datei – Speichern unter*.
- Als Dateityp wird der Eintrag Excel-Arbeitsmappe mit Makros (*.xlsm) genutzt.
- Für die Speicherung wird ein aussagekräftiger Name eingegeben.
- Ein Ordner wird als Speicherplatz ausgewählt.