R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – Automatisierung von Arbeitsschritten
Entfernen von leeren Zeilen

Arbeitsschritte

- Eine leere Zeile wird markiert.
- Die markierte Zeile wird gelöscht.
- Frage:
  - Muss für jede zu löschende Zeile ein Makro erstellt werden?
  - Wie können alle leere Zellen automatisiert gelöscht werden?

Arbeitsschritte automatisieren

- Voraussetzung: Die Entwicklertools sind eingeblendet.
- Die Arbeitsschritte werden einmalig mit Hilfe eines Makros aufgezeichnet.
- Nach Beendigung der Aufzeichnung werden die Arbeitsschritte automatisiert über ein Symbol oder Tastenkürzel gestartet.

Entwicklertools einblenden

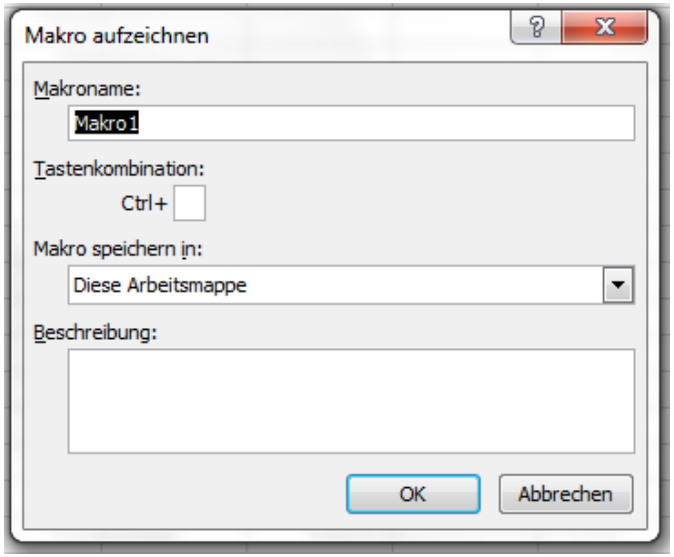
- *Datei – Optionen*.
- *Menüband anpassen*.
- In der rechten Liste Menüband anpassen werden alle Hauptregisterkarten angezeigt. Das Menüband Entwicklertools ist standardmäßig ausgeblendet (Kästchen ist leer).
- Durch einen Mausklick links von der Bezeichnung des Menübandes wird dieses aktiviert. In dem Kästchen wird ein Häkchen angezeigt.

Arbeitsschritte aufzeichnen

- Das Menüband Entwicklertools ist eingeblendet.
- Durch einen Mausklick wird die Aktion *Makro aufzchn.* in der Gruppe Code gestartet.
- Es öffnet sich der Dialog Makro aufzeichnen. In dem Dialog werden die Grundeinstellungen der Aufzeichnung festgelegt. *OK* schließt den Dialog.
- Hinweis: Die Aufzeichnung endet nicht automatisch.

Dialog „Makro aufzeichnen“

- In dem obersten Textfeld wird ein eindeutiger Name für die Aufzeichnung eingegeben.
- Zum Starten des Makros kann ein Buchstabe eingegeben werden. Hinweis: Einige Kombinationen wie <CTRL>+<C> sind belegt.
- Mit Hilfe des Kombinationsfeldes wird der Speicherort des Makros festgelegt. Standardmäßig wird ein Makro in der aktuellen Arbeitsmappe gespeichert.
- In dem unteren Textfeld kann eine Beschreibung für das Makro eingegeben werden.



Aufzeichnung beenden

- Das Menüband Entwicklertools ist eingeblendet.
- Mit einem Mausklick auf die Aktion *Aufzeichnung beenden* wird die momentan laufende Aufzeichnung beendet.
- Hinweis: Alle Arbeitsschritte zwischen dem Beginn und dem Ende der Aufzeichnung sind in der Programmiersprache VBA gespeichert.

Visual Basic for Application (VBA) ...

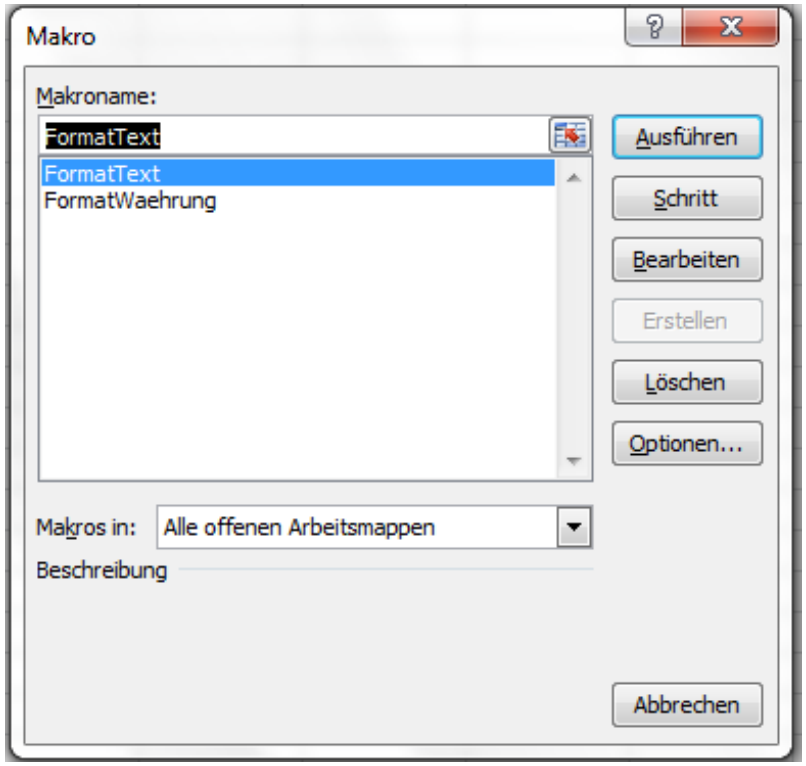
- automatisiert eine Folge von Arbeitsschritten.
- ist eine Programmiersprache, die in jeder Office-Anwendung eingebettet ist.
- erweitert die Funktionalität von Office-Anwendungen.
- passt eine Anwendung entsprechend der Wünsche des Benutzers an.

Start des Makros ...

- mit Hilfe des Symbols *Makros* im Bereich Code des Menübandes Entwicklertools.
- mit Hilfe von <CTRL> und der eingegebenen Taste.
- über ein Symbol in einem benutzerdefinierten Menüband.
- Nach dem Start werden die aufgezeichneten Arbeitsschritte abgearbeitet.

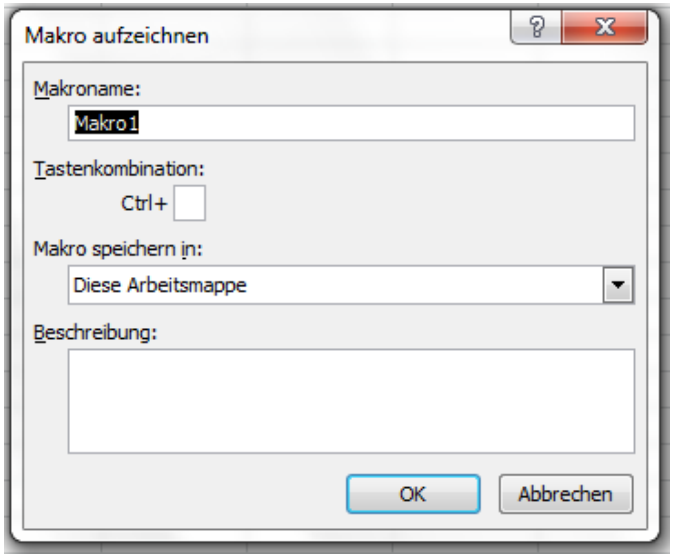
Entwicklertools - Makros

- Mit einem Mausklick wird ein Makro aus der Liste ausgewählt.
- Die Schaltfläche *Ausführen* startet das gewählte Makro.



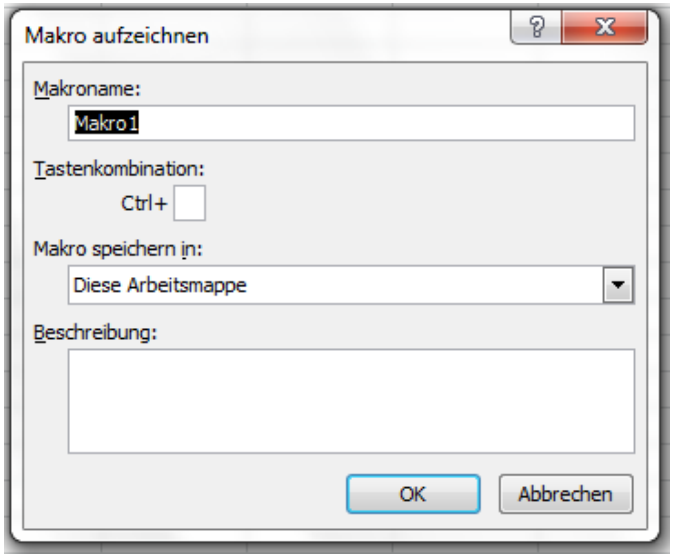
… mit Hilfe der Tastenkombination starten

- <CTRL>+<gewählte Taste> startet das Makro.



Tastenkombination ändern

- Mit Hilfe des Symbols *Makros* im Bereich Code des Menübandes Entwicklertools wird eine Liste aller vorhandenen Makros geöffnet.
- Mit Hilfe der Schaltfläche *Optionen* werden die Voreinstellungen angezeigt und können verändert werden.



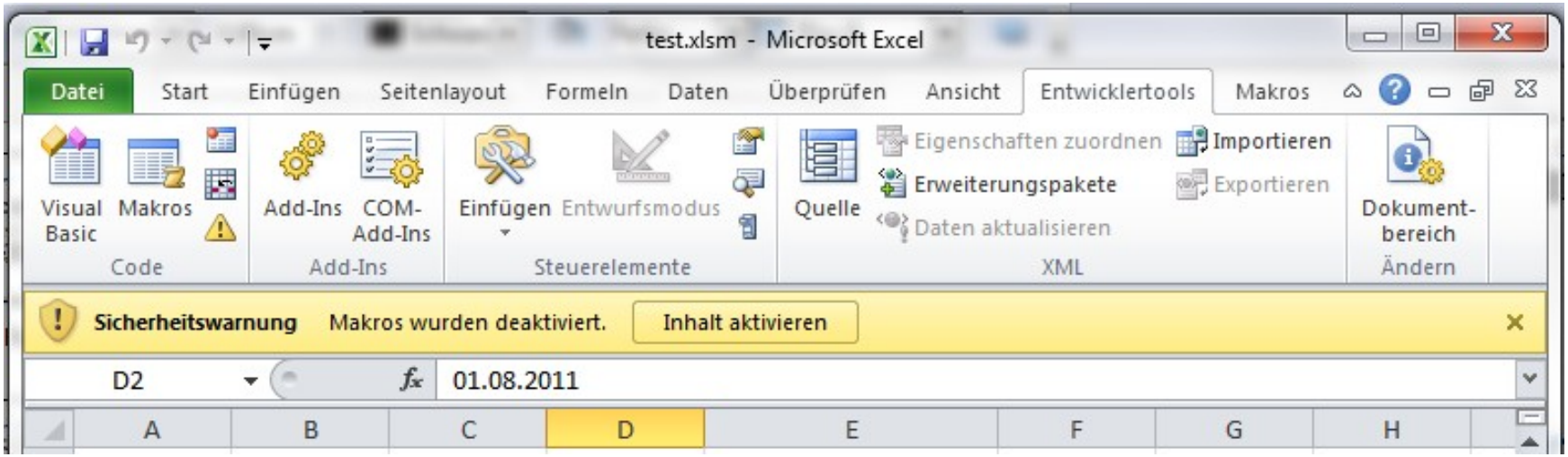
… über ein Menüband starten

- Voraussetzungen: Das Makro ist als Icon in einem Menüband sichtbar.
- Mit einem Klick auf das passende Icon wird das Makro gestartet.

Arbeitsmappen 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.

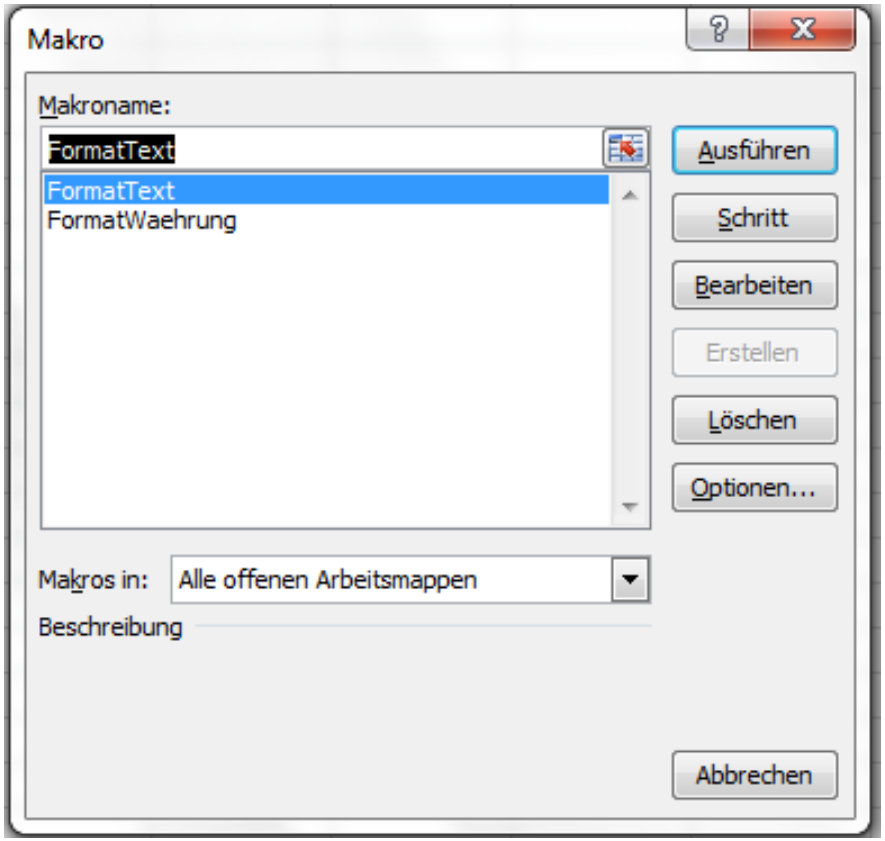
Arbeitsmappe mit einem Makro öffnen



- *Datei – Öffnen*.
- Beim erstmaligen Öffnen wird die Sicherheitswarnung angezeigt.
- Durch ein Klick auf die Schaltfläche *Inhalt aktivieren* werden die Makros aktiviert.

Makros bearbeiten

- Das Menüband Entwicklertools ist aktiv.
- In der Gruppe Code wird auf das Symbol *Makros* geklickt.
- Die Schaltfläche *Bearbeiten* öffnet den VBA-Editor. Alle Schritte des Makros werden in VBA-Code dargestellt.



Makro: Eine leere Zeile wird gelöscht

```
Sub LeereZeileLoeschen()
    Rows("18:18").Select
    Selection.ClearContents
    Selection.Delete Shift:=xlUp
End Sub
```

- Rows bezeichnet eine Zeile. Die Zeile wird mit Hilfe der Angaben in den runden Klammern eindeutig identifiziert. Mit Hilfe von .Select wird die Zeile ausgewählt.
- Durch die Methode .ClearContents werden die Formeln in dem selektierten Bereich gelöscht.
- Alle Zeilen werden durch die Methode .Delete gelöscht und alle nachfolgenden Zeilen werden nach oben verschoben (Shift:=xlUp).

Nachteile des aufgezeichneten Codes

- Es wird immer nur die aufgezeichnete Zeile gelöscht.
- Der Nutzer des Makros ist dafür verantwortlich, dass die, im Makro angegebene Zeile leer ist.

Ist die Zeile vollständig leer?

```
Sub LeereZeileLoeschen()
    Dim zelle As Range
    Dim txtZeile As String

    Rows("18:18").Select

    For Each zelle In Selection
        txtZeile = txtZeile & zelle.Value
    Next zelle

    If txtZeile = "" Then
        Selection.Delete Shift:=xlUp
    End If

End Sub
```

Variablen ...

- sind Platzhalter für Zahlen und Zeichen im Speicher.
- können einen beliebigen Standard-Datentyp annehmen.
- haben einen eindeutigen Namen. Der Name sollte selbsterklärend sein.
- bekommen mit Hilfe des Gleichheitszeichen einen Wert zugewiesen.

... definieren

Dim txtZeile As String

Dim variablenname As Datentyp

- Jede Variable hat einen eindeutigen Namen (quelle), der mit einem Buchstaben beginnt.
- Jede Variable ist von einem bestimmten Typ. Der Typ wird mit Hilfe des Schlüsselwortes As zugewiesen. In diesem Beispiel kann die Variable Text (String) speichern.
- Mit Hilfe des Schlüsselwortes Dim wird der Nutzer der Variablen festgelegt. In diesem Beispiel kann die Variable nur in der Prozedur LeereZeileLoeschen genutzt werden.

Datentypen ...

- legen das Format des zu speichernden Wertes fest.
- legen Regeln für die Verwendung der Variablen fest.
- beschreiben die maximale Größe des Platzhalters.
- legen den Speicherbedarf fest.
- werden in der Hilfe unter *Visual Basic-Sprachverzeichnis - Datentypen* aufgelistet.

… für Zeichenfolgen

	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.

Hinweise

ausgabeText = "Guten Tag"

- Zeichenfolgen werden durch Anführungsstriche begrenzt.
- Telefonnummern, Postleitzahlen werden in diesem Format gespeichert.
- Zeichenfolgen können Zahlen enthalten, mit denen nicht gerechnet wird.

Variablen nutzen

txtZeile = txtZeile & zelle.Value

- Mit Hilfe des Gleichheitszeichen wird der definierten Variablen ein Wert zugewiesen.
- Der Variablen wird ein konstanter Wert zugewiesen: txtZeile = "leer".
- Der Variablen wird ein berechneter Wert zugewiesen: txtZeile = txtZeile & zelle.Value. In diesem Beispiel wird der Text aus der angegebenen Zelle mit dem vorhandenen Variablenwert verknüpft. Der Operator „Kaufmännisches Und“ verbindet verschiedene Textelemente.

Objektvariablen ...

- sind Platzhalter für ein bestimmtes Excel-Objekt.
- können auf einen bestimmten Objekttyp verweisen.
- vom Typ
  - Workbook können einen Verweis auf eine Arbeitsmappe speichern.
  - Worksheet sind Platzhalter für eine bestimmtes Tabellenblatt in der angegebenen Arbeitsmappe.
  - Range verweist auf einen definierten Zellbereich.

Objektvariablen definieren

Dim zelle As Range

Dim variablenname As Objekttyp

- Der Name (zelle) der Objektvariablen ist frei wählbar.
- Mit Hilfe des Schlüsselwortes As wird der Objektvariablen ein bestimmter Objekttyp (hier: Range) zugewiesen.
- Die Variable ist nur in dem Bereich bekannt, wo sie definiert ist. Das Schlüsselwort Dim regelt den Zugriff auf die Variable. In diesem Beispiel ist nur ein lokaler Zugriff möglich.

Hinweise zum Namen der Variablen

- Der Variablenname beginnt immer mit einem Buchstaben.
- Der Variablenname sollte nur aus den Zeichen A..Z, a..z, 0..9 und den Unterstrich zusammengesetzt werden.
- Der Name sollte sprechen. Das heißt, der Name spiegelt den Typ oder die Nutzung des gespeicherten Wertes / Verweises wieder.

Zuweisung an eine Objektvariable

```
Set zelle = Range("A1")
```

- Die Zuweisung muss mit dem Schlüsselwort `Set` eingeleitet werden.
- Mit Hilfe des Gleichheitszeichens wird der Variablen ein Verweis auf ein bestimmtes Objekt zugewiesen.
- Das zugewiesene Objekt und die Variable sollten vom gleichen Objekttyp sein.
- In diesem Beispiel ist der Variablen `zelle` die Zelle „A1“ zugewiesen. D. h. der Platzhalter `zelle` kann auf die Zelle „A1“ zugreifen und diese verändern.

Range nutzen

```
Set zelle = Range("A1")
```

- In diesem Beispiel ist die Variable ein Platzhalter für die Zelle A1.
- Das Objekt `Range` kann eine Zelle, eine Zeile, eine Spalte oder einen Zellbereich beschreiben.

Alle Zellen eines Range durchlaufen

```
Dim zelle As Range
Dim txtZeile As String

Rows("18:18").Select                              ' Auswahl einer Zeile

For Each zelle In Selection                  ' Für jede Zelle in der Auswahl
    txtZeile = txtZeile & zelle.Value
Next zelle
```

Eine Liste vollständig durchlaufen

```
For Each zelle In Selection
    txtZeile = txtZeile & zelle.Value
Next zelle
```

```
Für jedes objekt in objektliste
    ' Anweisung
nächstes objekt
```

- Mit Hilfe einer For Each - Schleife wird eine Liste von Elementen vollständig durchlaufen.
- Mit Hilfe des Schlüsselwortes Next wird das nächste Listenelement aus der angegebenen Auflistung geholt.

Beispiel

```
For Each zelle In Selection
    txtZeile = txtZeile & zelle.Value
Next zelle
```

```
Für jedes objekt in objektliste
    ' Anweisung
nächstes objekt
```

- In diesem Beispiel soll die Auflistung Selection (der ausgewählte Bereich) vom Datentyp Range durchlaufen werden.
- Die Variable zelle ist ein Platzhalter für ein Element in der Auflistung. Die Variable und die Liste haben den gleichen Datentyp. Der Inhalt des Elements wird mit einem vorhandenen Text verknüpft.

Bedingte Anweisung

```
Sub LeereZeileLoeschen()
    Dim txtZeile As String

     If txtZeile = "" Then
        Selection.Delete Shift:=xlUp
    End If

End Sub
```

Erläuterung

```
If txtZeile = "" Then
    Selection.Delete Shift:=xlUp
End If
```

```
Wenn Bedingung = True Dann
    Selection.ClearContents
Ende Wenn
```

- Eine bedingte Anweisung beginnt mit dem Schlüsselwort If.
- Dem Schlüsselwort If folgt eine Bedingung. In diesem Beispiel wird abgefragt, ob die Variable txtZeile ein Leerstring ist.
- Wenn die Bedingung erfüllt ist, werden die Anweisungen zwischen Then und End If ausgeführt.

Bedingungen ...

- sind Ausdrücke, die wahr oder falsch sind.
- vergleichen Werte.
- nutzen Vergleichsoperatoren.
- können verknüpft werden.
- werden häufig in runde Klammern gesetzt. Die runden Klammern dienen der besseren Lesbarkeit.

Vergleichsoperatoren

Operator	Erläuterung
=	ist gleich
<>	ungleich
<	kleiner als
<=	kleiner gleich
>	größer
>=	größer gleich

… für Zahlen nutzen

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

- Dezimalzahlen vom Typ Double oder Single sollten nicht auf Gleichheit geprüft werden. Näherungswerte müssen nicht exakt den angegebenen Wert entsprechen.
- Ganzzahlen werden als Byte, Integer oder Long definiert.

… für Datumswerte nutzen

Operator	Erläuterung	Beispiel
=	ist gleich	#2/12/2011# = #6/11/2011# ≈ *Falsch*
<>	ungleich	#2/12/2011# <> #6/11/2011# ≈ *Wahr*
<	kleiner als	#2/12/2011# < #6/11/2011# ≈ *Wahr*
<=	kleiner gleich	#2/12/2011# <= #6/11/2011# ≈ *Wahr*
>	größer	#2/12/2011# > #6/11/2011# ≈ *Falsch*
>=	größer gleich	#2/12/2011# >= #6/11/2011# ≈ *Falsch*

- Datumswerte beginnen und enden in VBA mit einem Hash-Zeichen.
- Datumswerte werden als konstanter Wert in der Form #monat/tag/jahr# angegeben.
- Variablen für Datums- und Zeitwerten sind vom Datentyp Date.

… für Texte nutzen

Operator	Erläuterung	Beispiel
=	ist gleich	*"abc" = "ABC" ≈ Falsch*
<>	ungleich	*"abc" <> "ABC" ≈ Wahr*
<	kleiner als	*"abc" < "ABC" ≈ Falsch*
<=	kleiner gleich	*"abc" <= "ABC" ≈ Falsch*
>	größer	*"abc" > "ABC" ≈ Wahr*
>=	größer gleich	*"abc" >= "ABC" ≈ Wahr*

- Texte beginnen und enden immer mit den Anführungszeichen.
- Variablen für Texte sind vom Datentyp String.
- Es werden die ASCII-Codierungen der Buchstaben verglichen. (A = 65; a = 97).

Alle leere Zeilen in einer Datei löschen

```
Sub LeereZeileLoeschen()
    Dim txtZeile As String
    Dim zeilenNr As Long
    Dim zeilenNr_Ende As Long

    Dim arbeitsmappe As Workbook
    Dim arbeitsblatt As Worksheet
    Dim zelle As Range

    Set arbeitsmappe = ThisWorkbook
    Set arbeitsblatt = arbeitsmappe.ActiveSheet

    zeilenNr = 1
    zeilenNr_Ende = arbeitsblatt.UsedRange.SpecialCells(xlCellTypeLastCell).Row
```

Alle leere Zeilen in einer Datei löschen

```
    Do While zeilenNr <= zeilenNr_Ende
        txtZeile = ""

        For Each zelle In arbeitsblatt.Rows(zeilenNr).Cells
            txtZeile = txtZeile & zelle.Value
        Next zelle

        If txtZeile = "" Then
            arbeitsblatt.Rows(zeilenNr).Delete Shift:=xlUp
        End If

        zeilenNr = zeilenNr + 1
    Loop
End Sub
```

Variablen für Ganzzahlen definieren

Dim zeilenNr As Long

Dim variablenname As Datentyp

- Jede Variable hat einen eindeutigen Namen (zeilenNr), der mit einem Buchstaben beginnt.
- Der Datentyp wird mit Hilfe des Schlüsselwortes As zugewiesen. In diesem Beispiel kann die Variable Ganzzahlen (Long) speichern. Der Datentyp legt den Speicherbedarf sowie den Zahlenraum fest. Zum Beispiel eine Variable vom Datentyp Byte kann Zahlen von 0 bis 255 speichern.

Datentypen 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)

… für Dezimalzahlen

	Speicher-bedarf in Bytes	Datenbereich
As Single	4	-3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte
As Double	8	-1,79769313486232E308 bis -4,94065645841247E-324 für negative Werte; 4,94065645841247E-324 bis 1,79769313486232E308 für positive Werte

… für Darstellung von Währungen

	Speicher-bedarf in Bytes	Datenbereich
As Currency	8	-922.337.203.685.477,5808 bis 922.337.203.685.477,5807

Genauigkeit von Dezimalzahlen

- As Single beschreibt ein Wert von einfacher Genauigkeit. Der Platzhalter enthält eine Näherung an eine reelle Zahl.
- As Double beschreibt ein Wert von doppelter Genauigkeit. Der Platzhalter enthält eine Näherung an eine reelle Zahl.
- As Currency besitzt 15 Stellen vor dem Dezimalkomma und vier Stellen nach dem Dezimalkomma.

Hinweis

- Falls einer Variablen vom Datentyp „Ganzzahl“ eine Dezimalzahl zugewiesen wird, wird diese automatisch gerundet. Mit dieser gerundeten Zahl wird weiter gerechnet.
- Falls einer Variablen vom Datentyp „Dezimalzahl“ eine Ganzzahl zugewiesen wird, wird diese als Dezimalzahl in der Variablen gespeichert.

Variablen zurücksetzen

txtZeile = ""

zeilenNr = 0

- Variablen vom Datentyp „Text“ haben standardmäßig den Wert "" (leerer String). Durch die Zuweisung eines Leerstrings wird die Variable auf den Standardwert zurückgesetzt.
- Eine Variable vom Datentyp „Zahl“ hat standardmäßig den Wert 0. Durch die Zuweisung einer Null wird eine Variable vom Typ „Ganzzahl“ oder „Dezimalzahl“ zurückgesetzt.

Objekttypen für Variablen

```
Dim arbeitsmappe As Workbook
Dim arbeitsblatt As Worksheet
Dim zelle As Range
```

- Workbook verweist auf eine Arbeitsmappe.
- Worksheet verweist auf ein Tabellenblatt.
- Range verweist auf eine Zelle, Spalte, Zeile oder einen definierten Zellenbereich.

Verweis auf eine Arbeitsmappe

```
Set arbeitsmappe = ThisWorkbook
Set arbeitsmappe = Application.Workbooks("LeereZeile.xlsm")
```

- ThisWorkbook verweist auf die Arbeitsmappe, in der das Makro oder der VBA-Code ausgeführt wird.
- Die Liste Workbooks enthält alle geöffneten Arbeitsmappen. In diesem Beispiel wird die geöffnete Arbeitsmappe mit dem Namen „LeereZeile.xlsm“ genutzt.

Verweis auf ein Tabellenblatt

```
Set arbeitsblatt = arbeitsmappe.ActiveSheet
Set arbeitsblatt = arbeitsmappe.Worksheets("Tabelle1")
```

- In der gewählten Arbeitsmappe gibt es ein aktives Tabellenblatt (ActiveSheet). Ein aktives Tabellenblatt ...
  - liegt im Vordergrund der Arbeitsmappe.
  - wird durch eine Beschriftung in Fett auf den Tabellenreiter gekennzeichnet.
- Die Liste Worksheets besteht aus allen Tabellenblätter der angegebenen Arbeitsmappe. In diesem Beispiel verweist die Variable auf das Tabellenblatt mit dem Namen „Tabelle1“ genutzt. Die Groß- und Kleinschreibung wird bei der Angabe des Namens beachtet.

Verweis auf Zellen

```
Set zelle = ThisWorkbook.ActiveSheet.Range("A1")
Set zelle = ThisWorkbook.ActiveSheet.Range("A1:D4")
Set zelle = ThisWorkbook.ActiveSheet.Cells(1, 3)
Set zelle = ThisWorkbook.ActiveSheet.Range(Cells(1, 3), Cells(1, 6))
```

- Mit Hilfe von Range kann eine Zelle, eine Zeile, eine Spalte oder ein Zellbereich angegeben werden.
- Der Auflistung Cells wird eine Zeilen- und eine Spaltennummer übergeben. Mit Hilfe der Zeilen- und Spaltennummer kann eine Zeile eindeutig identifiziert werden.
- In einem Range kann eine Auflistung von Zellen, getrennt durch Kommata angegeben werden.

Verweis auf bestimmte Zellen

```
zeilenNr_Ende = arbeitsblatt.UsedRange.SpecialCells(xlCellTypeLastCell).Row
```

- Die Objektvariable arbeitsblatt verweist auf ein bestimmtes Tabellenblatt.
- In diesem Tabellenblatt wird eine bestimmte Zelle im genutzten Bereich (UsedRange) gesucht.
- Es wird eine Zelle mit einem speziellen Merkmal gesucht (SpecialCells). Das gesuchte Merkmal wird als vordefinierte Konstante der Methode in den runden Klammern übergeben. In diesem Beispiel wird die letzte Zelle (xlCellTypeLastCell) in dem genutzten Bereich gesucht.
- Mit Hilfe der Eigenschaft .Row wird die Zeilennummer der letzten Zelle im genutzten Bereich ermittelt.

Schleifen nutzen

```
    Do While zeilenNr <= zeilenNr_Ende
        txtZeile = ""

        For Each zelle In arbeitsblatt.Rows(zeilenNr).Cells
            txtZeile = txtZeile & zelle.Value
        Next zelle

        zeilenNr = zeilenNr + 1
    Loop
End Sub
```

Anweisungen x mal wiederholen

```
Do While zeilenNr <= zeilenNr_Ende


 Loop
```

- Tue etwas solange die Bedingung erfüllt ist, …
- In diesem Beispiel steht im Schleifenkopf Do While die Bedingung für den Abbruch in der Schleife.
- Mit Hilfe des Schlüsselwortes Loop wird ein neuer Schleifendurchlauf gestartet.
- Die Variablen in der Bedingung müssen in der Schleife so gesetzt werden, dass die Bedingung nicht erfüllt ist. Andernfalls läuft die Schleife endlos.

Variablen setzen

```
zeilenNr = 1
zeilenNr_Ende = arbeitsblatt.UsedRange.SpecialCells(xlCellTypeLastCell).Row

Do While zeilenNr <= zeilenNr_Ende
      zeilenNr = zeilenNr + 1
 Loop
```

- Vor Beginn der Schleife wird die Zählvariable zeilenNr sowie der Endwert zeilenNr_Ende auf einen definierten Wert gesetzt.
- In der Schleife selber wird die Zählvariable bei jedem Durchlauf um eins hochgezählt (zeilenNr = zeilenNr + 1). Durch diese Anweisung hat nach einer bestimmten Anzahl von Durchläufen die Zählvariable einen größeren Wert als der Endwert.

Alle Zellen durchlaufen

```
For Each zelle In  arbeitsblatt.Rows(zeile).Cells
        txtZeile = txtZeile & zelle.Value
Next zelle
```

- Mit Hilfe von der For Each – Schleife kann eine Objektliste vollständig durchlaufen werden. Jede Objektauflistung in Excel endet mit einem „s“.
- In diesem Beispiel wird die Liste Cells durchlaufen. In dieser Liste befinden sich alle Zellen des Tabellenblattes oder eines bestimmten Bereichs. Hier werden die Zellen einer bestimmten Zeile (.Rows(zeile)) angesprochen.

Leere Zellen mit Hilfe einer Funktion ermitteln

```
Sub LeereZeileLoeschen()
    Dim zeilenNr As Long
    Dim lastZeile As Long
    Dim lastSpalte As Long

    Dim arbeitsmappe As Workbook
    Dim arbeitsblatt As Worksheet
    Dim zeile As Range

    Set arbeitsmappe = ThisWorkbook
    Set arbeitsblatt = arbeitsmappe.ActiveSheet

    zeilenNr = 1
    lastZeile = arbeitsblatt.UsedRange.SpecialCells(xlCellTypeLastCell).Row
```

Leere Zellen mit Hilfe einer Funktion ermitteln

```
Do While zeilenNr <= lastZeile
        lastSpalte = arbeitsblatt.Rows(zeilenNr).SpecialCells(xlCellTypeLastCell).Column
        Set zeile = arbeitsblatt.Range(Cells(zeilenNr, 1), Cells(zeilenNr, lastSpalte))

        If Application.WorksheetFunction.CountBlank(zeile.Cells) = zeile.Cells.Count Then
                arbeitsblatt.Rows(zeilenNr).Delete Shift:=xlUp
        End If

        zeilenNr = zeilenNr + 1
    Loop
End Sub
```

Anzahl der Elemente in einer Auflistung

zeile.Cells.Count

- Mit Hilfe der Methode Count wird die Anzahl der Elemente in einer Liste von Objekten ermittelt.
- In diesem Beispiel werden die Zellen in einem bestimmten Bereich gezählt.

Anzahl der Elemente in einer Auflistung

Application.WorksheetFunction.CountBlank(zeile.Cells)

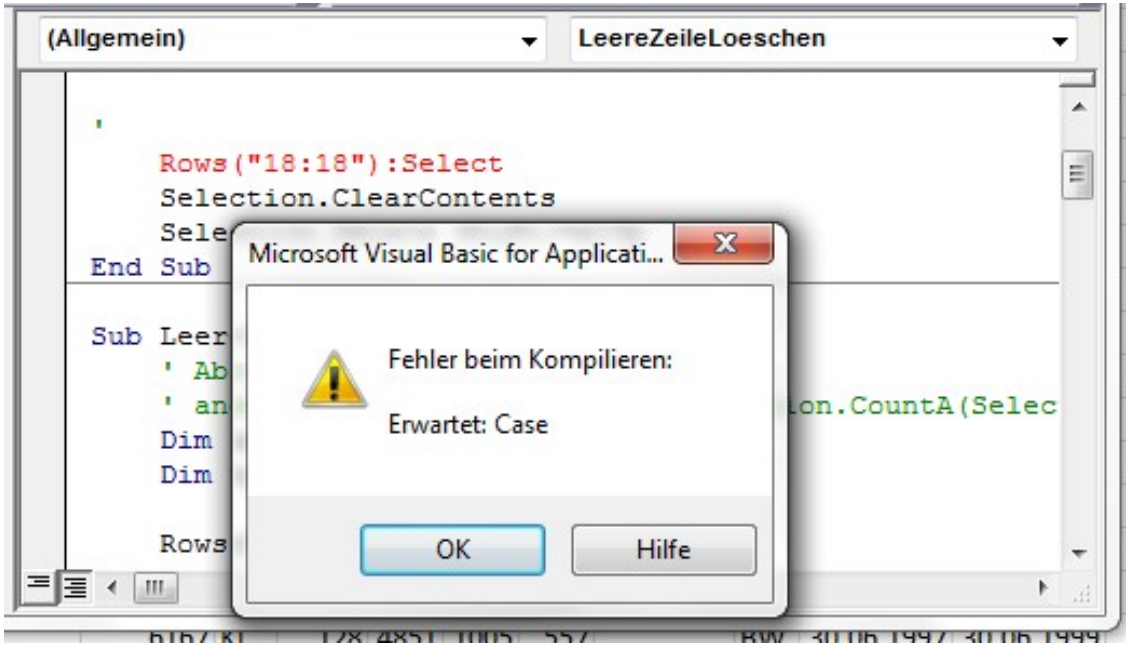
- Das Objekt Application beschreibt die Excel-Anwendung.
- Die Anwendung selber hat als Kind-Objekt WorksheetFunction. Dieses Objekt wird als Container für alle Arbeitsblatt-Funktionen genutzt.
- In diesem Beispiel wird die Funktion CountBlank zum Zählen von leeren Zellen in einem bestimmten Bereich genutzt. Der Bereich wird der Funktion in den runden Klammern als Eingabeparameter übergeben.

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.
- An den Anfang jedes neuen Moduls wird automatisch die Anweisung Option Explicit 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

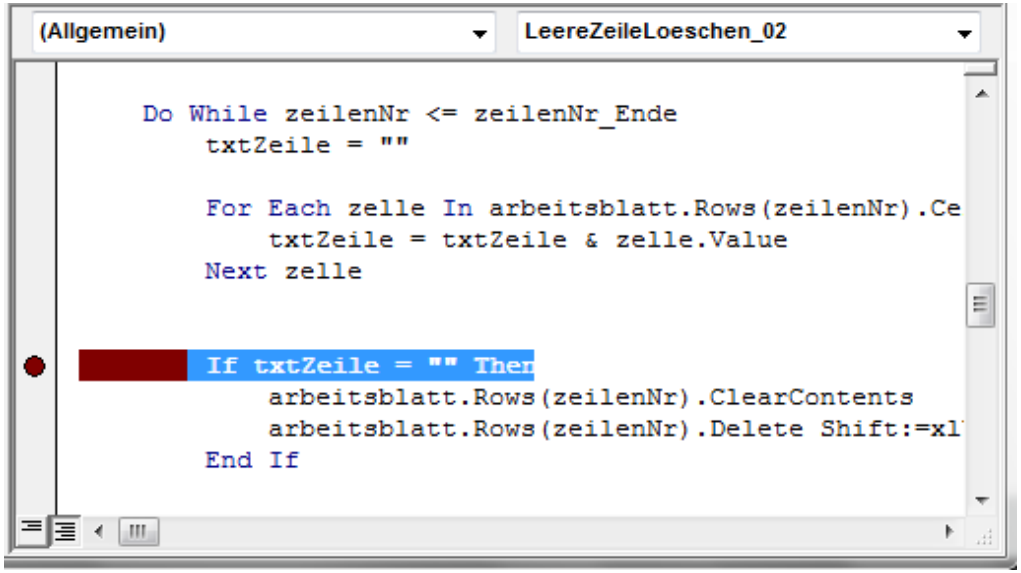
- Die Einfügemarke steht in einer bestimmten Anweisung. Bis zu diesem Punkt tritt kein Fehler auf.
- Das Programm wird mit Hilfe von <STRG>+<F8> oder *Debuggen – Ausführen* bis Cursor-Position gestartet.
- Das Programm wird bis zur Einfügemarke vollständig durchlaufen. An der Einfügemarke wird das Programm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

Haltepunkte ...

- stoppen das Programm in einer bestimmten Zeile.
- ermöglichen die Untersuchung einer bestimmten Zeile.
- Ab dem 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.

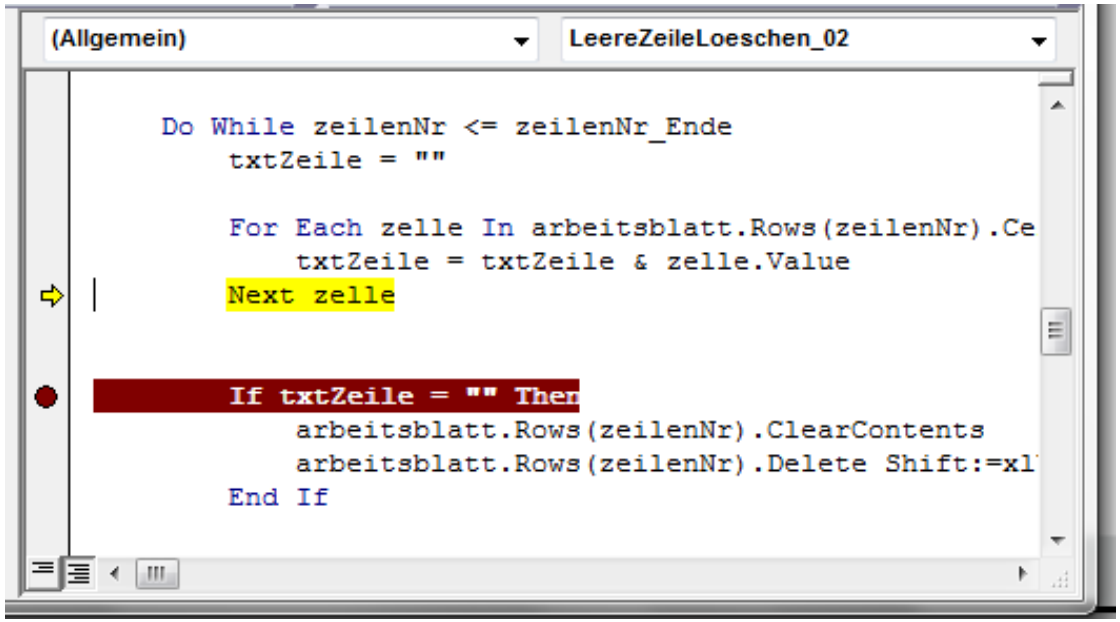


… löschen

- 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.

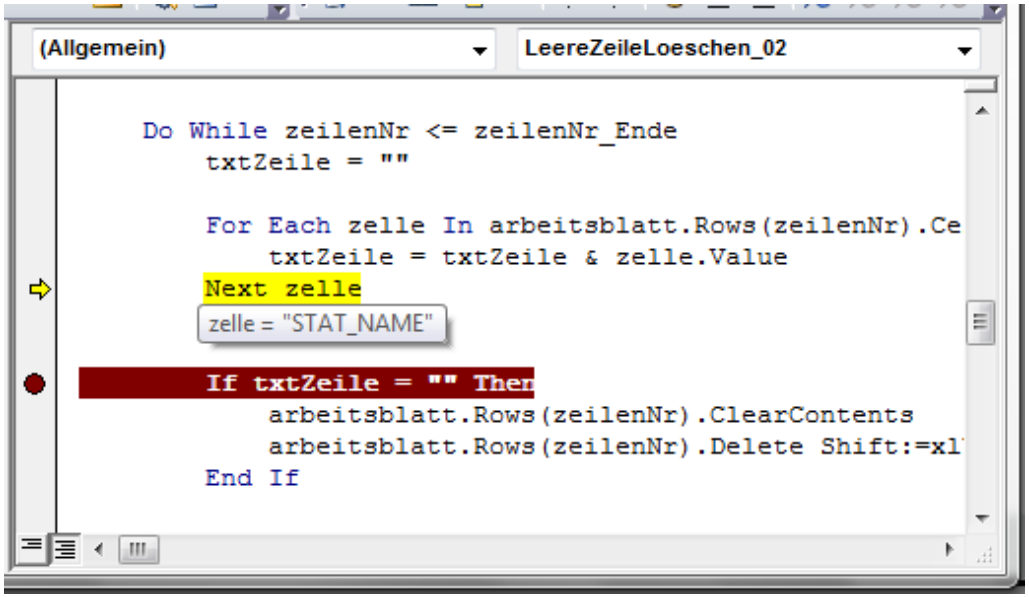
Einzelschrittmodus nutzen

- 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.



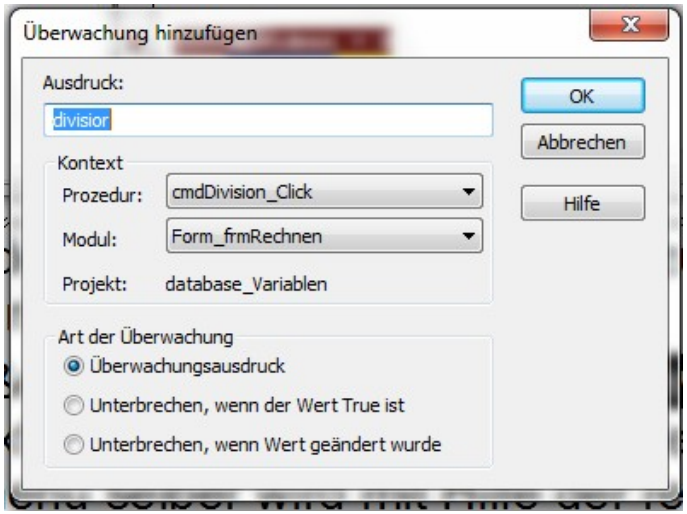
Wert einer Variablen anzeigen

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



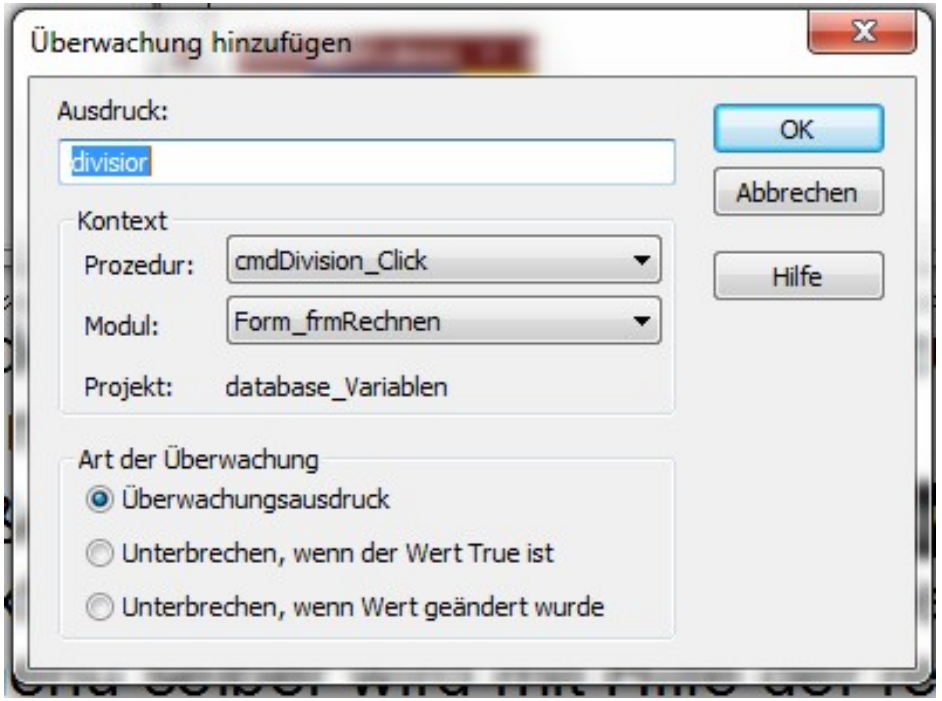
Wert einer Variablen ständig überprüfen

- 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“

- Im oberen Textfeld wird der zu überwachende Ausdruck angezeigt.
- Der zu überwachende Ausdruck ist in der angegebenen Prozedur definiert. Die Prozedur befindet sich in dem angegebenen Modul.
- Die Art der Überwachung wird mit Hilfe von Optionsfeldern ausgewählt.
- Das Fenster wird mit Hilfe der Schaltfläche *OK* geschlossen.

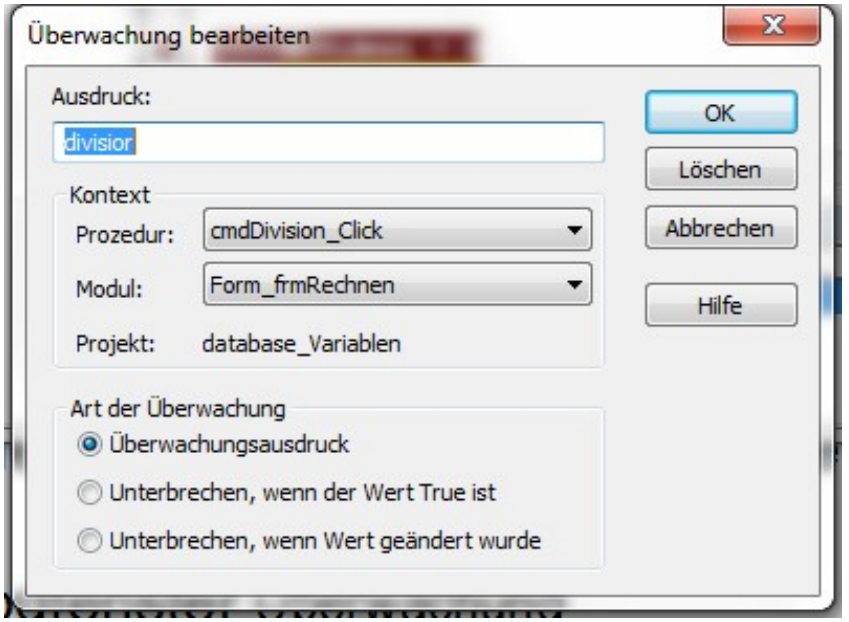


Überwachungsarten

- 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.

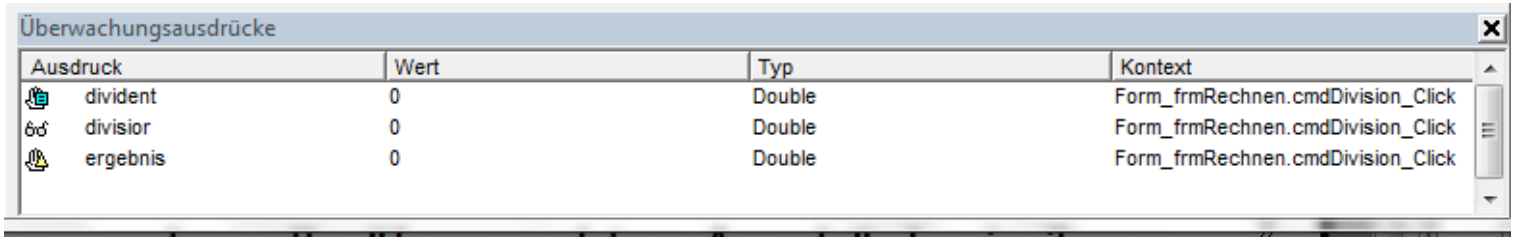
Überwachung bearbeiten

- 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.



Überwachungsausdrücke anzeigen

- *Ansicht – Überwachungsfenster* blendet das entsprechende Fenster ein.
- In dem Fenster werden alle überwachten Ausdrücke in ihren Kontext angezeigt.
- Der Wert des überwachten Ausdrucks und deren Datentyp werden angezeigt.
- Die Überwachungsart wird durch ein Symbol am linken Rand angezeigt.



Mit dem Fenster arbeiten

- Mit Hilfe eines Mausklicks auf das Symbol am rechten Rand wird die gesamte Zeile markiert.
- <ENTF> löscht die markierte Überwachung.
- Ein markierter Ausdruck kann aus dem Codebereich mit Hilfe der gedrückten Maustaste in das Überwachungsfenster gezogen werden. Sobald die Maustaste losgelassen wird, wird der Ausdruck an der Position eingefügt.