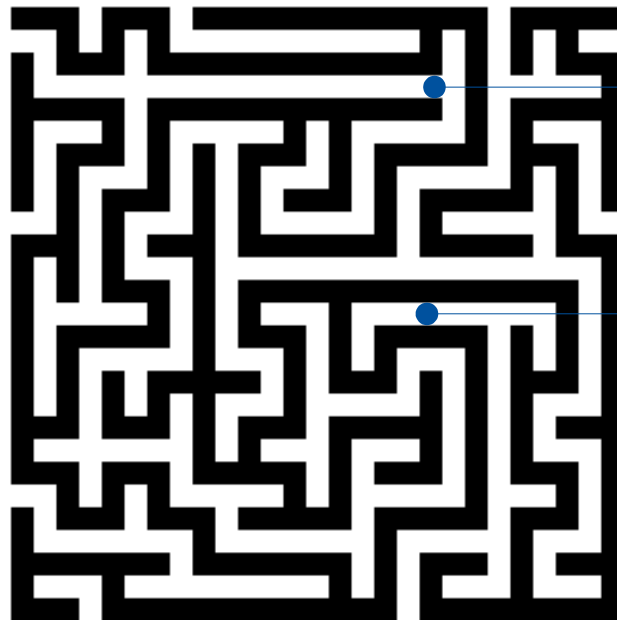


Access 2010 – Programmierung

Bedingte Anweisungen



Gehe nach links
oder rechts

Gehe solange
geradeaus...

Beispiel für bedingte Anweisungen

- In einem Formular gibt der Kunde für die verschiedenen Produkte eine Bestellmenge ein.
- Die Produkte werden mit Hilfe eines Kontrollkästchens ausgewählt.
- In Abhängigkeit der Bestellmenge ...
 - wird der Gesamtpreis für den Artikel berechnet.
 - wird ein Rabatt auf den Gesamtpreis vergeben.

Lösung

- An das Ereignis „Beim Klicken“ der Schaltfläche wird eine Prozedur angehängt.
- In dieser Prozedur wird zuerst abgefragt, welches Kontrollkästchen gewählt ist.
- Anschließend wird die Bestellmenge passend zum Produkt ausgelesen.
- Der Preis für die Bestellung wird berechnet. Ab einer bestimmten Bestellmenge wird ein Rabatt gewährt.
- Das Ergebnis der Berechnung wird in dem entsprechenden Textfeld ausgegeben.

Bedingte Anweisungen werden genutzt...

- Welches Optionsfeld ist in einer Optionsgruppe aktiv? Mit Hilfe der Eigenschaft « Optionsgruppe.Value » kann in VBA abgefragt werden, welches Optionsfeld in einer Gruppe aktiv ist.
- Welches Kontrollkästchen ist aktiv?
- Welches Listenfeld ist bei einer Mehrfachauswahl aktiv?
- Ist der Inhalt eines Textfeldes numerisch? Es können alle Funktionen genutzt werden, die mit „Is...“ beginnen.

Code-Schnipsel

```
Private Sub cmdBerechnen_Click()
```

```
    Dim menge As Integer
```

```
    Dim rabatt As Double
```

```
    Dim endpreis As Currency
```

```
    If chk_DINA4.Value = True Then
```

```
        ...
```

```
    End If
```

```
End Sub
```

Deklaration
von Variablen

Bedingte Anweisung:
Welches Kontrollkästchen
ist gesetzt?

Code-Schnipsel

```
If chk_DINA4.Value = True Then
    menge = CInt(txt_Menge_DINA4)

    If (menge > 10) And (menge < 50) Then
        rabatt = 0.02
    ElseIf (menge >= 50) Then
        rabatt = 0.04
    Else
        rabatt = 0
    End If

    endpreis = (2.49 * menge)
    endpreis = endpreis - (endpreis * rabatt)
    txt_Preis_DINA4 = endpreis

End If
```

Berechnung
des Rabatts

Berechnung des
Gesamtpreises pro
Produkt

Bedingte Anweisungen

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

```
If chk_DINA4.Value = True Then  
    menge = Cint(txt_Menge_DINA4)  
End If
```

Erläuterung

- Die Bedingung beginnt mit dem Kopf « If (Bedingung) Then » und endet mit dem Schlüsselwort « End If».
- Bedingungen sind Ausdrücke, die einen boolschen Wert zurückliefern.
- 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	$3 = 4 \approx \text{falsch}$
<>	ungleich	$3 <> 4 \approx \text{richtig}$
<	kleiner als	$3 < 4 \approx \text{richtig}$
<=	kleiner gleich	$3 <= 4 \approx \text{richtig}$
>	größer	$3 > 4 \approx \text{falsch}$
>=	größer gleich	$3 >= 4 \approx \text{falsch}$

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 Näherungswert.

Textvergleiche als Bedingung

```
Private Sub cmdSenden_Click()  
    Dim strPostleitzahl  
  
    If txtPlz.Value <> "" Then  
        strPostleitzahl = txtPlz.Value  
  
        If (strPostleitzahl Like "30####") Then  
            Else  
                txtOrt.Value = ""  
                txtPlz.Value = ""  
                txtPlz.SetFocus  
            End If  
        End If  
    End Sub
```

Textvergleiche mit Hilfe von Platzhaltern

"Meier" Like "M*"	Das Sternchen ersetzt eine beliebige Anzahl von Zeichen in einer Zeichenfolge. Falls der Vergleichstext mit M beginnt, wird wahr geliefert.
"Meier" Like "M??er"	Das Fragezeichen ersetzt exakt ein Zeichen. In diesem Beispiel muss der Vergleichstext mit M beginnen und auf „er“ enden. Zwischen Beginn und Ende befinden sich zwei beliebige Buchstaben.
"30165" Like "30####"	Das Hash-Zeichen ersetzt eine Zahl. In diesem Beispiel wird nach einem Text gesucht, der mit 30 beginnt. Dem Text folgen drei weitere beliebige Zahlen.
"301-b-H34" Like "3##-?-H*"	Die Platzhalter können beliebig miteinander kombiniert werden.

Textvergleiche mit Hilfe einer Liste von Zeichen

"Meier" Like "M[ea]?er"	Eine Liste von Zeichen wird durch die eckigen Klammern begrenzt. Eine der, in der Liste angegebenen Buchstaben, muss an der zweiten Position vorkommen.
"Meier" Like "[B-F]?er"	Eine Liste von Zeichen wird durch die eckigen Klammern begrenzt. Als erster Buchstabe kann das B, C, D, E oder F genutzt werden.
"Meier" Like "[B-FM-N]?er"	Eine Liste von Zeichen wird durch die eckigen Klammern begrenzt. Als erster Buchstabe kann das B, C, D, E, F, M oder N genutzt werden.
"Meier" Like "M[!ea]?er"	Das Ausrufezeichen steht für das Wort „nicht“. An der zweiten Stelle darf kein e oder a stehen.

Ist die Variable undefiniert?

```
Dim eingabe As String

If IsNull(eingabe) Then
    eingabe = Input("Bitte Benutzername eingeben")
End If
```

- Mit Hilfe der Funktion « IsNull() » wird überprüft, ob eine Variable, Ausdruck oder Objekt definiert ist oder nicht.
- Falls die Variable oder der Ausdruck keinen definierten Wert besitzt, wird der boolesche Wert « True » zurückgeliefert.
- Variablen liefern immer den Wert « False ». Jede Variable in VBA hat in Abhängigkeit ihres Datentyp einen Standardwert.

Funktionen mit dem Präfix „Is...“

- überprüfen ein Aspekt einer Variablen oder Ausdruck.
- beantworten Fragen, die mit Ja oder Nein beantwortet werden können.
- bekommen in runden Klammern immer die zu überprüfende Variable oder den zu überprüfenden Ausdruck übergeben.
- liefern einen booleschen Wert zurück. Falls die Aussage zutrifft, wird « True » zurückgeliefert, andernfalls « False ».

Leerer String?

```
Dim benutzer As String
```

```
If (benutzer = "") Then
```

```
    eingabe = Input("Bitte Benutzername eingeben")
```

```
End If
```

- Ist der String leer?
- Ein leerer String wird durch « "" » symbolisiert. Zwei Anführungszeichen, aneinander gereiht, kennzeichnen einen Leerstring.
- Der String ist definiert, hat aber keinen Inhalt.

Verknüpfungsoperatoren für Bedingungen

Operator	Erläuterung	Beispiel
And	und	$(3 > 4) \text{ And } (3 < 10) \approx \textit{richtig}$ $(5 > 6) \text{ And } (5 < 4) \approx \textit{falsch}$ Alle Bedingungen müssen zutreffen.
Or	oder	$(3 > 4) \text{ Or } (3 < 10) \approx \textit{richtig}$ $(5 > 6) \text{ Or } (5 < 4) \approx \textit{richtig}$ Eine der Bedingungen muss zutreffen.
Not	nicht	Negation einer Bedingung $\text{Not}(3 > 4) \approx \textit{richtig}$ $\text{Not}(3 < 4) \approx \textit{falsch}$

Beispiel: Verknüpfung mit AND

```
If (menge > 10) And (menge < 50) Then  
    rabatt = 0.02  
End If
```

- Die Variable „menge“ hat einen Wert größer als 10 und kleiner als 50.
- Wenn die Menge beide Bedingungen erfüllt, wird ein Rabatt von 0.02 gewährt.

Beispiel: Verknüpfung mit Not

```
If Not (IsNull(txtVon.Value)) Then  
    beginn = CInt(txtVon.Value)  
End If
```

- « IsNull(txtVon.Value) » liefert False zurück, wenn das Textfeld einen definierten Wert hat.
- Der Rückgabewert wird negiert. True (Wahr) wird zu False (falsch) und umgekehrt. D. h. wenn das Textfeld einen definierten Inhalt hat, wird der Wert in dem Textfeld txtVon in ein Integer umgewandelt und der Variablen beginn zugewiesen.

Beispiel: Oder-Verknüpfung

```
Private Sub txtOperandLinks_KeyPress(KeyAscii As Integer)

    ' Zahlen 0 (Ascii-Code 48) bis 9 (Key-Ascii-Code 57)
    If (KeyAscii < 48) Or (KeyAscii > 57) Then
        KeyAscii = 0
    End If

End Sub
```

- Wenn die gedrückte Taste kleiner als 48 oder größer als 57 ist, wird der Tastendruck zurückgesetzt.
- Eine der Bedingungen muss zutreffen.

Weiteres Beispiel

```
Private Sub fraBestellung_AfterUpdate()  
    If fraBestellung.Value = 1 Or 2 Then  
        ausgabe = "Ihre Bestellung wird bearbeitet"  
    End If  
  
    MsgBox (ausgabe)  
End Sub
```

- Die Optionsgruppe „fraBestellung“ wird mit Hilfe der verknüpften Bedingung überprüft.
- Wenn die Option 1 oder 2 aktiv ist, wird der Variablen „ausgabe“ der Text „Ihre Bestellung wird bearbeitet“ übergeben.
- Sind die Bedingungen fehlerfrei?

Sind die Bedingungen fehlerfrei?

- Welche Bedingungen werden in dem Beispiel genutzt?
« If (fraBestellung.Value = 1) Or (2) Then ».
- Erste Bedingung « (fraBestellung.Value = 1) »:
Wenn das Optionsfeld den Wert eins hat, ist diese Bedingung wahr.
- Zweite Bedingung « (2) »:
Diese Bedingung ist immer wahr. Eine wahre Bedingung hat einen Wert ungleich 0. Ob diese Bedingung den Wert 2 oder 3 hat, spielt keine Rolle.
- Durch die zweite Bedingung werden immer die Anweisungen zu « If » ausgeführt. Resultat: Der String wird bei jedem Optionswert ausgegeben.

Lösung

```
Private Sub fraBestellung_AfterUpdate()  
    If (fraBestellung.Value = 1) Or ( fraBestellung.Value = 2) Then  
        ausgabe = "Ihre Bestellung wird bearbeitet"  
    End If  
  
    MsgBox (ausgabe)  
End Sub
```

Die Bedingung trifft nicht zu ...

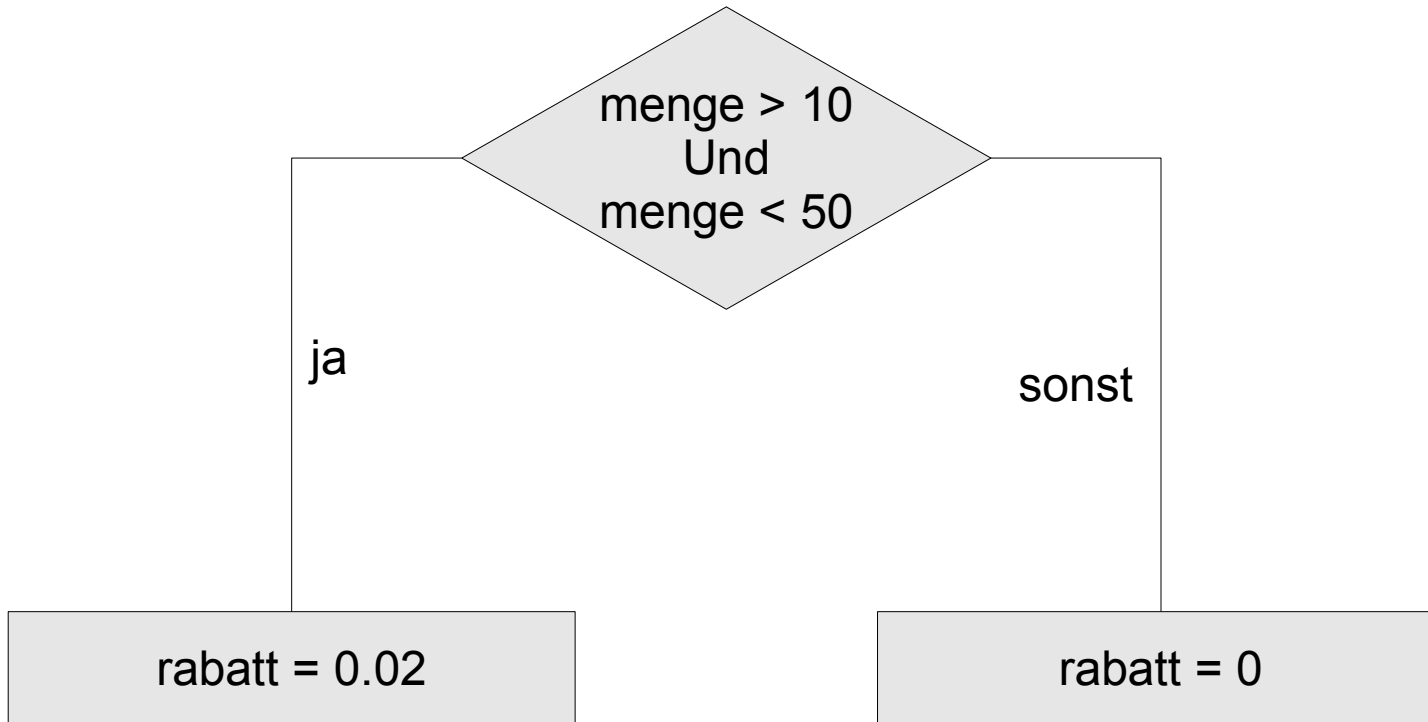
```
If (Bedingung) Then  
    Anweisung  
Else  
    Anweisung  
End If
```

```
If (menge > 10) And (menge < 50) Then  
    rabatt = 0.02  
Else  
    rabatt = 0  
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?
- « Else » folgt am Ende einer if-Anweisung. « Else » steht immer in Verbindung mit « if ».
- « Else » ist optional.

Grafische Darstellung



Fallunterscheidung

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

```
If (menge > 10) And (menge < 50) Then
    rabatt = 0.02
Elseif (menge >= 50) Then
    rabatt = 0.04
Else
    rabatt = 0
End If
```

Erläuterung

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

Auswahlanweisung

Select Case variable

Case wert:

Anweisung

Case wert, wert, wert:

Anweisung

Case min To max:

Anweisung

Case Is operator wert:

Anweisung

End Select

Select Case menge

Case 10 To 50:

rabatt = 0.02

Case Is > 50:

rabatt = 0.04

Case Else:

rabatt = 0

End Select

Erläuterung

- Die Auswahlanweisung bietet für ein Element verschiedene Möglichkeiten kann.
- Es werden verschiedene Werte für ein bestimmtes Element abgefragt und behandelt.
- Eine Auswahlanweisung beginnt mit « Select Case » und endet mit « End Select ».
- Das zu untersuchende Element wird im Kopf « Select Case element » angegeben. Für „element“ werden verschiedene Werte untersucht.
- Mit Hilfe einer Auswahlanweisung können Texte und Zahlen untersucht werden.

Auswahlmöglichkeiten ...

- beginnen mit dem Schlüsselwort « Case ».
- Dem Schlüsselwort folgt ein oder mehrere Vergleichswerte. Der Datentyp des Vergleichswertes muss mit Datentyp des zu überprüfenden Elements übereinstimmen.
- Ein Doppelpunkt schließt die Case-Anweisung ab.

Vergleich mit einem Wert

« Case wert: »

« Case 5: »

- Wenn der Variablenwert exakt dem Wert entspricht, werden die zu dem Fall gehörenden Anweisungen ausgeführt.
- In diesem Fall wird ein Wert mit dem Wert des zu überprüfenden Elements verglichen.

Liste von Vergleichswerten

« Case wert, wert: »

« Case "q", "Q", "x", "X": »

- Der Wert des zu überprüfenden Elements wird mit Hilfe einer Liste verglichen.
- Die Listenelemente werden durch Kommata getrennt.
- Wenn ein Wert in der Liste zutrifft, werden die dazugehörigen Anweisungen ausgeführt.
- Die Liste entspricht einer Oder-Verknüpfung von Werten.

Intervall von Vergleichswerten

« Case min To max: »

« Case 10 To 50: »

- Startwert To Endwert.
- Wenn der Variablenwert innerhalb der angegebenen Unter- und Obergrenze liegt...
- In diesem Beispiel werden die dazugehörigen Anweisungen ausgeführt, wenn das zu überprüfende Element einen Wert größer gleich 10 hat und kleiner gleich als 50 ist.

Vergleichsoperatoren nutzen

« Case Is > wert: »

« Case Is > 50: »

- Das Schlüsselwort « Is » ist ein Platzhalter für das zu untersuchende Element.
- In dem Vergleichsausdruck kann jeder Vergleichsoperator genutzt werden.

Case Else ...

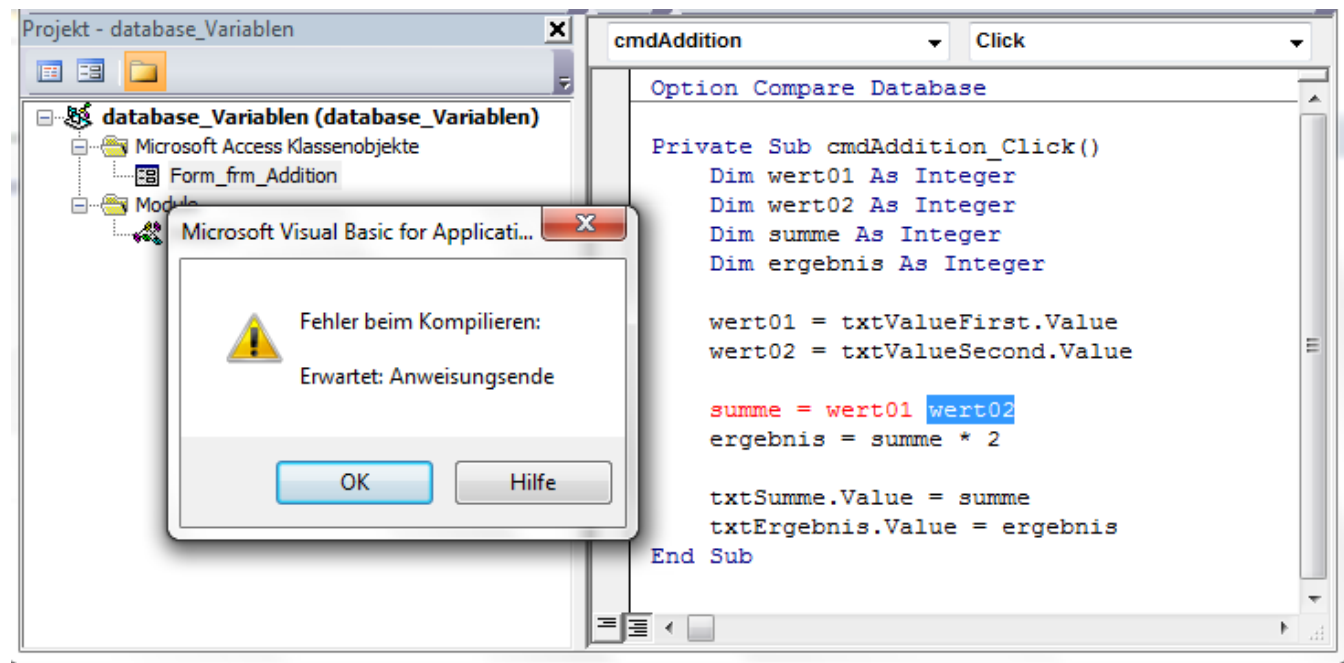
- beschreibt den Standardfall.
- tritt ein, wenn alle anderen Möglichkeiten nicht zutreffen.
- entspricht dem else-Zweig einer if-Anweisung.
- ist optional.

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 jeden 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

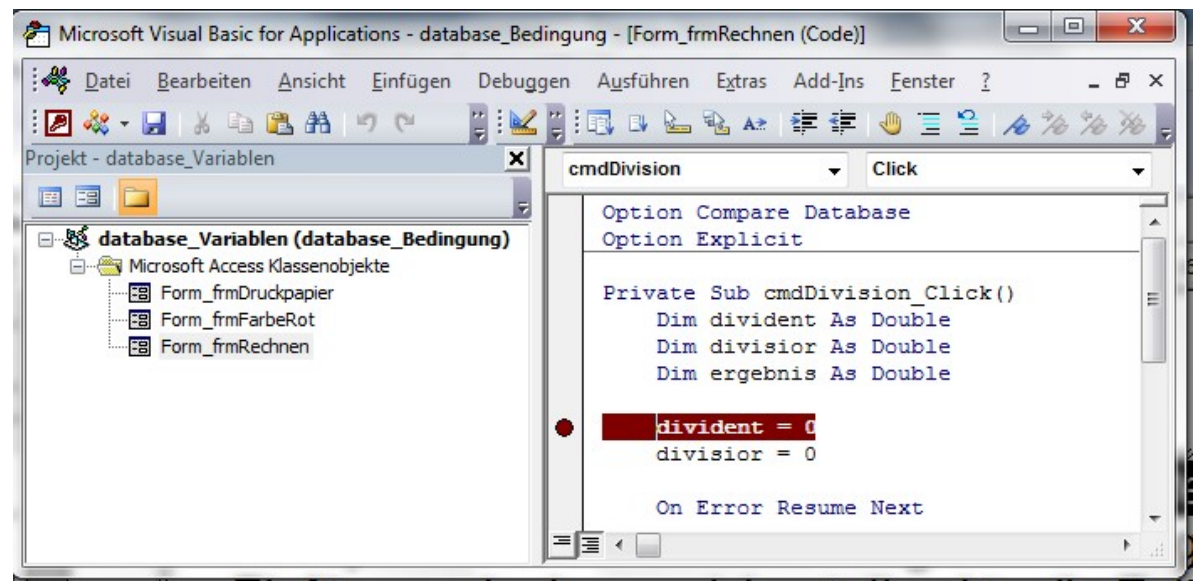
- Die Einfügemarke steht in einer bestimmten Anweisung. Bis zu diesem Punkt tritt kein Fehler auf.
- Das Programm wird mit Hilfe von <STRG>+<F8> oder *Debuggen – Ausführen bis Cursor-Position* gestartet.
- Das Programm wird bis zur Einfügemarke vollständig durchlaufen. An der Einfügemarke wird das Programm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

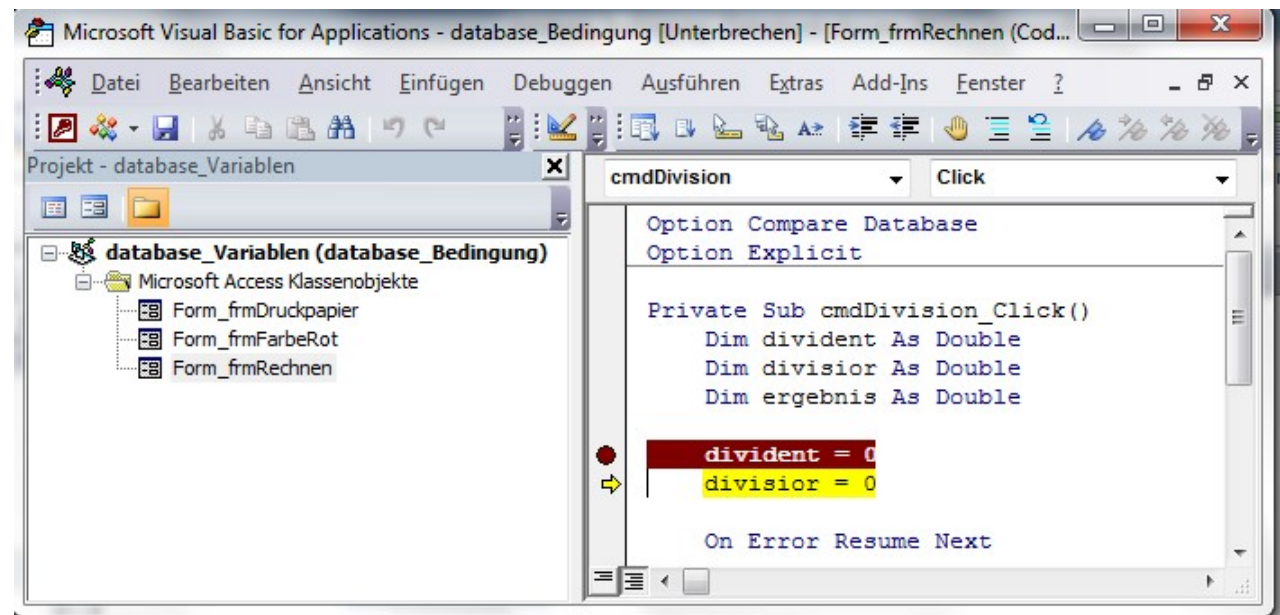


... löschen

- 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.

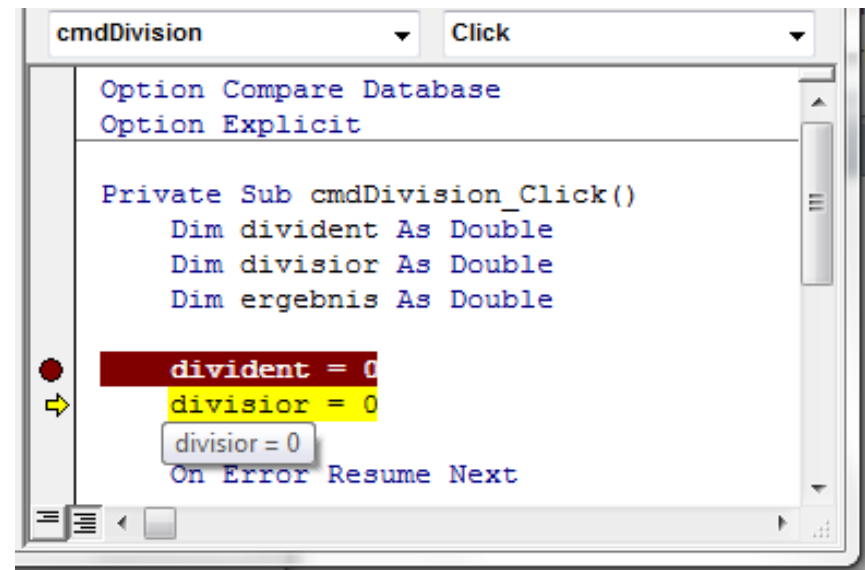
Einzelmodus nutzen

- 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.



The screenshot shows a code editor window titled 'cmdDivision' with a 'Click' event handler. The code is as follows:

```
Option Compare Database
Option Explicit

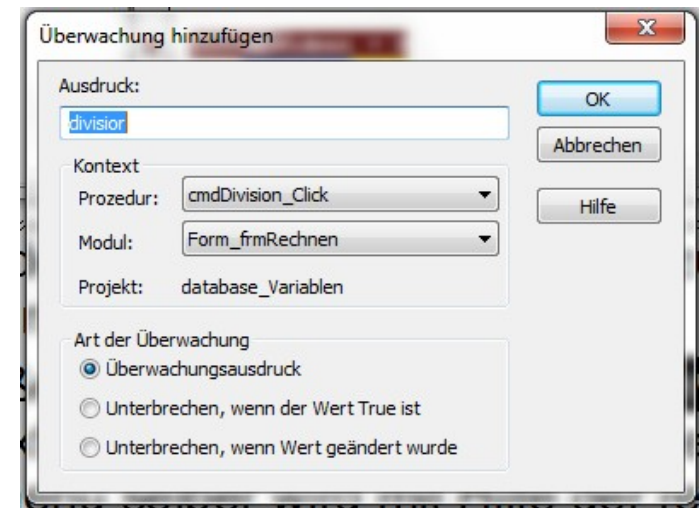
Private Sub cmdDivision_Click()
    Dim dividant As Double
    Dim divisor As Double
    Dim ergebnis As Double

    dividant = 0
    divisor = 0
    divisor = 0
    On Error Resume Next
```

The line `divisor = 0` is highlighted in yellow. A red mouse cursor is positioned over it, and a tooltip box displays the value `divisor = 0`.

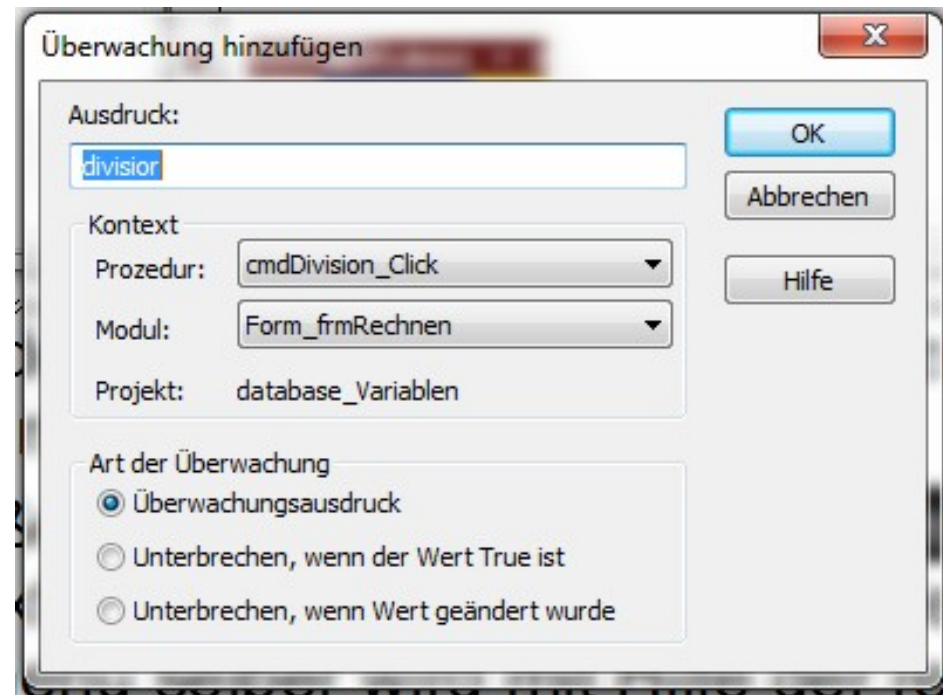
Wert einer Variablen ständig überprüfen

- 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.

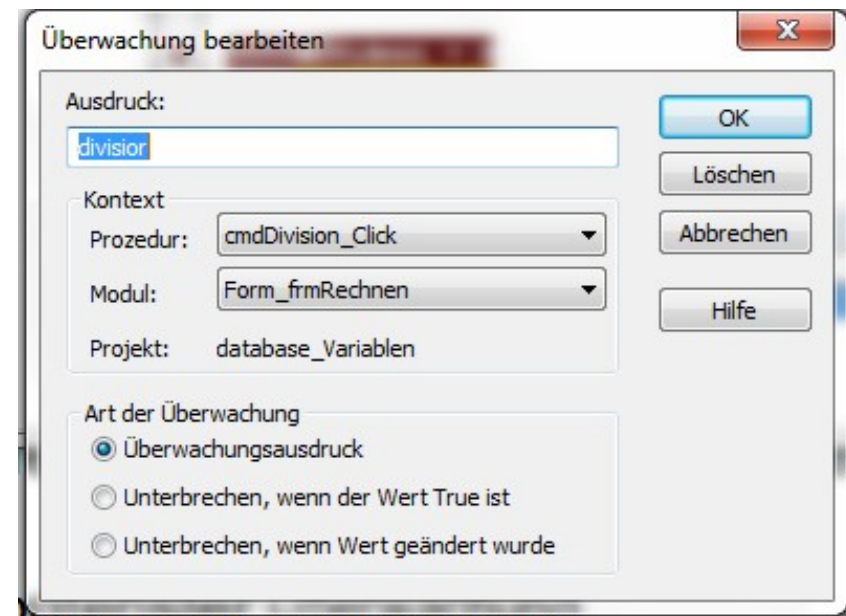


Überwachungsarten

- 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.

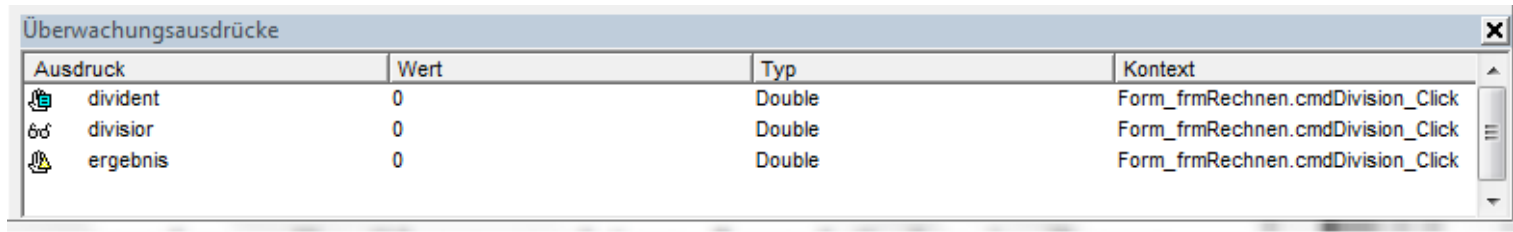
Überwachung bearbeiten




- 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

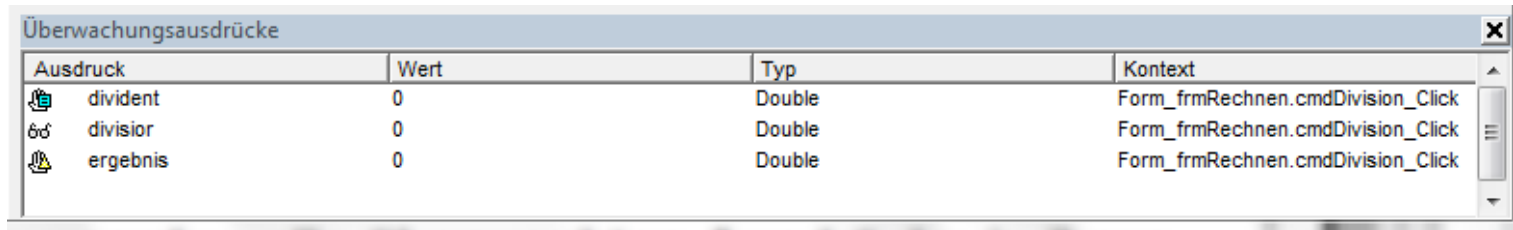
- *Ansicht – Überwachungsfenster* blendet das entsprechende Fenster ein.
- In dem Fenster werden alle überwachten Ausdrücke in ihren Kontext angezeigt.
- Der Wert des überwachten Ausdrucks und deren Datentyp werden angezeigt.
- Die Überwachungsart wird durch ein Symbol am linken Rand angezeigt.






Ausdruck	Wert	Typ	Kontext
 dividend	0	Double	Form_frmRechnen.cmdDivision_Click
 divisor	0	Double	Form_frmRechnen.cmdDivision_Click
 ergebnis	0	Double	Form_frmRechnen.cmdDivision_Click

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.



Ausdruck	Wert	Typ	Kontext
 divident	0	Double	Form_frmRechnen.cmdDivision_Click
 divisor	0	Double	Form_frmRechnen.cmdDivision_Click
 ergebnis	0	Double	Form_frmRechnen.cmdDivision_Click

Alle Variablen einer Prozedur überwachen

- Im Einzelschrittmodus werden alle Variablen einer Prozedur im Lokalfenster angezeigt.

The screenshot shows the VBA editor for a procedure named `cmdDivision` with the `Click` event selected. The code is as follows:

```
Option Compare Database
Option Explicit

Private Sub cmdDivision_Click()
    Dim dividant As Double
    Dim divisor As Double
    Dim ergebnis As Double

    dividant = 0
    divisor = 0

```

The line `dividant = 0` is highlighted in yellow, and a yellow arrow cursor points to it, indicating the current execution step.

Below the code editor is the **Locals** window, which displays the current state of local variables:

Expression	Value	Type
Me		Form_frmRechnen/Form_frmRechnen
dividant	0	Double
divisor	0	Double
ergebnis	0	Double

Laufzeitfehler ...

- sind Fehler, die nach dem Starten des Codes auftreten.
- können einen Programmabsturz verursachen. Das Programm wird unkontrolliert beendet.
- ist zum Beispiel eine „Division durch Null“, ein falscher Umgang mit Datentypen, Endlosschleifen etc..

... abfangen

```
Private Sub cmdDivision_Click()  
    Dim dividend As Double  
    Dim divisor As Double  
    Dim ergebnis As Double  
    dividend = 0  
    divisor = 0  
  
    On Error Resume Next  
    dividend = Cdbl(txtOperandLinks.Value)  
    divisor = Cdbl(txtOperandRechts.Value)  
  
End Sub
```

... abfangen

```
Private Sub cmdDivision_Click()
```

```
    Dim dividend As Double
```

```
    Dim divisor As Double
```

```
    Dim ergebnis As Double
```

```
    On Error GoTo markeFehler
```

```
    ergebnis = dividend / divisor
```

```
    txtErgebnis.Value = ergebnis
```

```
    Exit Sub
```

```
markeFehler:
```

```
    MsgBox (Err.Number & ": " & Err.Description)
```

```
    txtErgebnis.Value = 0
```

```
End Sub
```

Erläuterung

- « On Error Resume Next » geht zur nächsten Zeile, wenn ein Fehler auftritt. Ohne den Fehler zu beheben, wird die nächste Zeile ausgeführt.
- « On Error GoTo markeFehler » geht zu einer bestimmten Sprungmarke in der Prozedur, wenn ein Fehler auftritt. An der Marke « markeFehler: » wird der Fehler behoben. Meist werden Marken am Ende der Prozedur gesetzt. Eine Zeile vor der Marke wird das Programm mit Hilfe der Anweisung « Exit Sub » vorzeitig verlassen.

Sprungmarken ...

« markeFehler: »

- bestehen aus einem benutzerdefinierten Namen und dem Doppelpunkt.
- sind Lesezeichen in einer Prozedur.
- sind eindeutig in einer Prozedur.
- werden bei der Ausführung des Programms nicht beachtet. Das Programm bricht nicht automatisch vor einer Sprungmarke ab.

Informationen zu Laufzeitfehlern ...

- bietet das Objekt « Err ».
- « Err.Number » gibt die Fehlernummer zurück.
- « Err.Description » enthält den passenden Fehlertext.
- « Err.Clear » entfernt alle Fehlermeldungen.