R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – VBA
Fehler im Programm

Fehler (Bugs)

- Syntaxfehler entstehen beim Schreiben des Programmcodes.
- Logische Fehler können durch Denkfehler bei der Umsetzung der Aufgabe in ein Programm erzeugt werden. Das Programm wird fehlerfrei ausgeführt, aber das Ergebnis ist nicht korrekt.
- Laufzeitfehler treten während der Ausführung des Programms auf. Zum Beispiel eine CD, von der Daten gelesen werden, befindet sich nicht im Laufwerk.

Syntaxfehler

- Die Syntax einer Programmiersprache ist die Gesamtheit der Regeln für die Bildung von Anweisungen aus Operatoren und Operanden sowie die Nutzung von Funktionen.
- ... verhindern die Kompilierung eines Programms.
- ... sind zum Beispiel:
  - Tippfehler bei der Eingabe von Variablennamen oder Schlüsselwörtern.
  - Mischung von numerischen und nicht-numerischen Operatoren.
  - Falsche Parameterübergabe an Funktionen.
  - Syntaxfehler in Schleifen oder bedingten Anweisungen.

Syntaxfehler vermeiden

- Öffnen Sie das Menü *Extras* – *Optionen*.
- Aktivieren Sie die Registerkarte Allgemein.
- Mit Hilfe der Option Automatische Syntaxüberprüfung werden Fehler im Codefenster markiert.
- Die Anweisung Option Explicit wird automatisch mit Hilfe der Option Variablendeklaration erforderlich an den Anfang eines neuen Moduls eingefügt. Variablen müssen deklariert werden. Es werden keine Variablennamen doppelt besetzt.

Syntaxfehler im Codefenster

```
Sub fehler()
    Dim count As Integer

    For count = 1  10
    Next count

End Sub
```

- Die fehlerbehaftete Zeile wird farbig hinterlegt.
- In einem Dialogfenster wird der Fehler mehr oder weniger verständlich erläutert.
  - Die Schaltfläche *Hilfe* ruft eine Hilfeseite auf.
  - Die Schaltfläche *OK* schließt das Dialogfenster.

Laufzeitfehler

- ... sind Bugs, die nach dem Start eines Programms auftreten können.
- ... betreffen immer die Programmlogik.
- ... können Programme zu einem unerwünschten Verhalten oder einem Programmabsturz zwingen.
- ... entstehen, wenn Ausdrücke oder Anweisungen vom Programm nicht korrekt ausgewertet werden.

Beispiele

- Division durch Null.
- Überlauf (zu großer oder zu kleiner Wert für den angegebenen Datentyp).
- Falsche Abbruchbedingung für eine Schleife.
- Verwendung von ungültigen Operatoren.
- Ein- und Ausgabefehler.
- Tippfehler wie "1o" statt 10.

Laufzeitfehler abfangen

```
Sub Fehler()
    Dim ergebnis As Integer

    On Error GoTo ausgabeFehler

    ergebnis = 5 \ 0

    Exit Sub

ausgabeFehler:
   MsgBox Err.Description

End Sub
```

Regeln für die Fehlerbehandlung

- On Error ... Wie reagiert das Programm auf einen Fehler?
- On Error GoTo [marke] springt zu einer bestimmten Position im Programm. Ab dieser Position beginnt die Fehlerbehandlungsroutine.
- On Error Resume Next überspringt den Fehler. Das Programm arbeitet mit der nächsten fehlerfreien Zeile weiter. Fehlerbehaftete Variablen werden aber weiter durch das Programm gezogen.

Informationen zum Fehler

- Das Objekt Err sammelt alle Fehler eines VBA-Programms.
- Die Eigenschaft Err.Number hat als Wert eine eindeutige Fehlernummer.
- Err.Description gibt eine Textbeschreibung des Fehlers zurück.
- Err.Helpfile kann einen Fehler mit einer Hilfeseite verbinden.
- Err.Source enthält einen Hinweis auf den Auslöser des Fehlers.
- Die Methode Err.Clear setzt alle Eigenschaften des Objekts auf die Standardwerte zurück. Die vorhandenen Fehler werden gelöscht.
- Die Methode Err.Raise löst einen Laufzeitfehler aus.

Sprungmarken nutzen

```
    On Error GoTo ausgabeFehler
    txtWert = InputBox("Bitte geben Sie ei
    intWert = CInt(txtWert)
    ergebnis = 5 \ intWert

    MsgBox "5 \ " & intWert & " = " & erg
    Exit Sub

ausgabeFehler:
    MsgBox Err.Description
    intWert = 1
    Resume Next
```

Der Name einer Sprungmarke ist frei wählbar und eindeutig. Dem Namen der Marke folgt immer ein Doppelpunkt. Anschließend folgen die dazugehörigen Anweisungen. Falls kein Fehler auftritt, muss die Prozedur vor der Sprungmarke verlassen werden.

Fortsetzung in der nächsten Zeile

```
On Error GoTo ausgabeFehler
txtWert = InputBox("Bitte geben Sie eine Zahl ein:")
intWert = CInt(txtWert)
ergebnis = 5 \ intWert

MsgBox "5 \ " & intWert & " = " & ergebnis
Exit Sub

ausgabeFehler:
MsgBox Err.Description
intWert = 1
Resume Next
```

Das Programm wird mit der Zeile, die nach der fehlerbehafteten Zeile kommt, fortgesetzt.

Fehlerbehaftete Zeile wiederholen

```
On Error GoTo ausgabeFehler
txtWert = InputBox("Bitte geben Sie eine Zahl ein:")
intWert = CInt(txtWert)
ergebnis = 5 \ intWert

MsgBox "5 \ " & intWert & " = " & erge
Exit Sub

ausgabeFehler:
MsgBox Err.Description
txtWert = InputBox("Bitte geben Sie eine Zahl ein:")
Resume
```

Die Anweisung wird solange wiederholt, bis der Benutzer eine Zahl eingibt, die als Ganzzahl interpretiert werden kann.

Enumeration für eigene Fehlercodes

```
Public Enum myErrorNumber
    ERR_NOTANUMBER
    ERR_ISDECIMAL
End Enum
```

- In einer Aufzählung werden Werte zusammengefasst.
- Jede Variable in einer Enumeration symbolisiert einen bestimmten Wert. Die erste Variable symbolisiert den Wert 0 und so weiter.
- Mit Hilfe des Gleichheitszeichen kann jeder Variablen ein Wert zugewiesen werden.
- ... können nur am Anfang eines Moduls definiert werden.

Funktion für eigene Fehlermeldungen

```
Function RaiseError(ErrNumber As myErrorNumber) As String
    Dim eNumber As Long

    eNumber = vbObjectError + 513 + ErrNumber

    Select Case ErrNumber
        Case ERR_NOTANUMBER
            RaiseError = "Eingabe: Nicht num
        Case ERR_ISDECIMAL
            RaiseError = "Eingabe: Dezimalzahl"
    End Select
End Function
```

In Abhängigkeit einer Enumeration-Variablen wird eine Fehlermeldung ausgegeben.

Systemfehler

```
Function RaiseError(ErrNumber As myErrorNumber) As String
    Dim eNumber As Long

    eNumber = vbObjectError + 513 + ErrNumber

    Select Case ErrNumber
        Case ERR_NOTANUMBER
            RaiseError = "Eingabe: Nicht numeri
        Case ERR_ISDECIMAL
            RaiseError = "Eingabe: Dezimalzahl"
    End Select
End Function
```

Systemfehler liegen im Bereich von 0 bis 512.

Fehlerroutine nutzen

```
txtWert = InputBox("Bitte geben Sie eine Zahl ein:")

If IsNumeric(txtWert) Then
    If (InStr(txtWert, ",") > 0) Or (InStr(txtWert, ".") > 0) Then
        MsgBox RaiseError(ERR_ISDECIMAL)
    Else
        intWert = CInt(txtWert)
        ergebnis = 5 \ intWert

        MsgBox "5 \ " & intWert & " = " & ergebnis
    End If
Else
    MsgBox RaiseError(ERR_NOTANUMBER)
End If
```

Logische Fehler

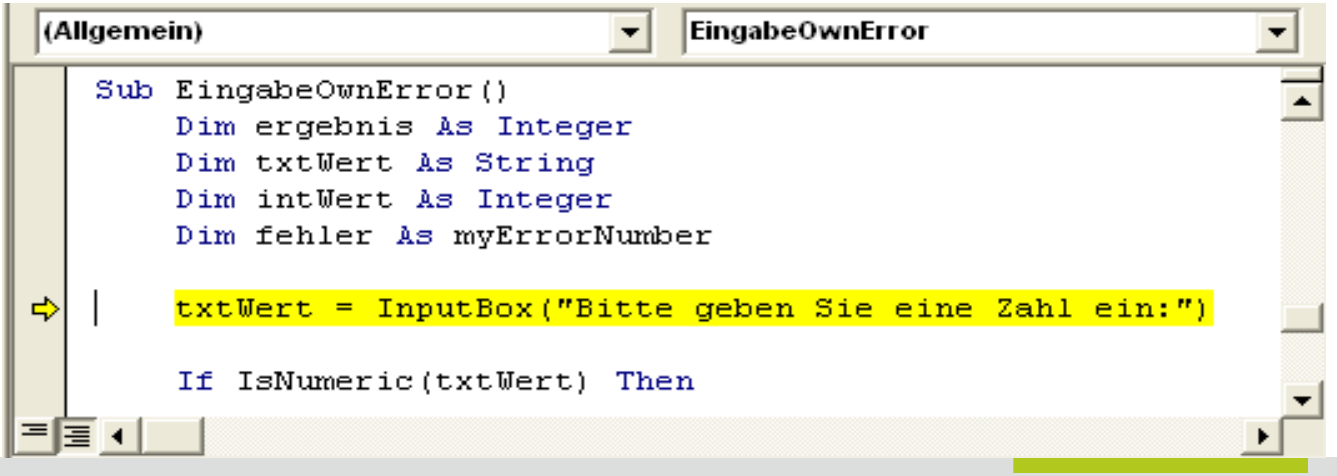
- ... entstehen
  - ... beim Design eines Programms oder
  - ... bei der Definition von Anforderungen an das Programm.
- ... können
  - ... durch fehlendes Fachwissen oder
  - ... Missverständnissen zwischen Nutzern und Entwicklern entstehen.
- ... können durch Debuggen des Programms gefunden werden.

... werden hervorgerufen durch

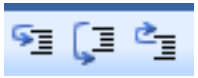
- ... eine falsche Anzahl von Schleifendurchläufen.
- ... durch das Erzeugen von Endlosschleifen.
- ... falsch formulierte Bedingungen in Anweisungen und Schleifen.
- ... eine falsche oder nicht vorhandene Klammerung von komplexen Ausdrücken.
- ... falsch initialisierte Variablen.

Einzelschritt

- Das Programm wird Zeile für Zeile durchlaufen.
- Die momentan auszuführende Zeile wird farbig unterlegt und mit einem Pfeil am Rand gekennzeichnet. Der Pfeil kann mit Hilfe der gedrückt gehaltenen Maustaste verschoben werden. Das Programm führt ab der gekennzeichneten Zeile das Programm neu aus.
- Legen Sie den Mauszeiger auf einen Bezeichner oder Ausdruck. Der aktuelle Wert wird in einem ToolTip angezeigt.

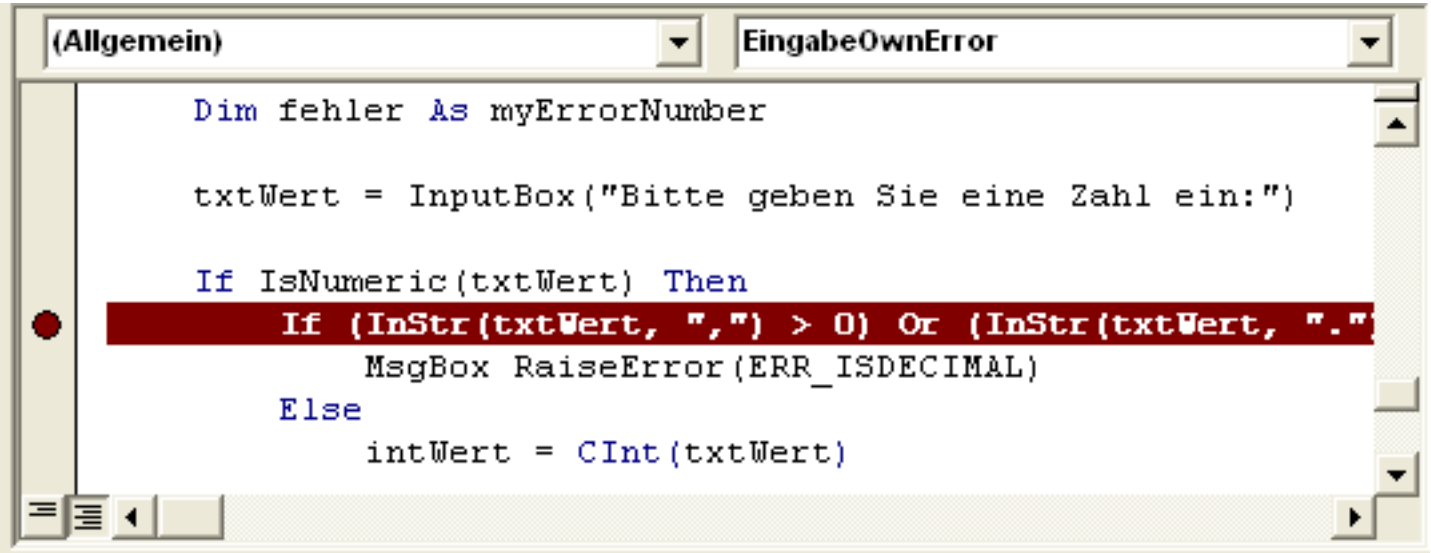


Einzelschritt starten

- *Debuggen – Einzelschritt* oder <F8>.
- Jede Zeile muss mit <F8> quittiert werden.
- Andere Möglichkeit  in der Symbolleiste Debuggen:
  - Mit Hilfe des linken Icons wird der Einzelschritt-Modus gestartet.
  - Mit Hilfe des mittleren Icons werden alle Einzelschritte ausgeführt. Aufgerufene Prozeduren oder Funktionen werden in einem Block abgearbeitet.
  - Mit Hilfe des rechten Icons wird die gerade ausgeführte Prozedur oder Funktion in einem Block abgearbeitet. Anschließend wird das Programm angehalten.

Haltepunkte

- ... werden durch einen farbigen Punkt am linken Rand des Codefensters dargestellt. Die dazugehörige Zeile wird in der gleichen Farbe gekennzeichnet.
- ... können nicht auf Deklarationsanweisungen, leeren Zeilen, Sprunganweisungen oder Kommentaren gesetzt werden!



Arbeitsweise

- Das Programm wird mit <F5> gestartet.
- Das Programm wird bis zum ersten Haltepunkt automatisch abgearbeitet.
- Ab dem Haltepunkt kann das Programm
  - ... im Einzelschritt
  - ... oder mit <F5> bis zum nächsten Haltepunkt durchlaufen werden.

Haltepunkte setzen und löschen

- *Debuggen – Haltepunkt ein / aus* oder <F9>
  - ... setzt einen Haltepunkt in der aktuellen Zeile.
  - ... entfernt einen vorhanden Haltepunkt.
- *Debuggen – Alle Haltepunkte löschen* entfernt alle Haltepunkte aus dem aktuellen Projekt.

Verschiedene Fenster nutzen

- Voraussetzung: Das Programm wird im Einzelschritt durchlaufen.
- *Ansicht* –
  - *Direktfenster*. Geben Sie den gewünschten Befehl ein. Schließen Sie die Zeile mit <RETURN> ab. Der Befehl wird direkt ausgeführt und das Ergebnis angezeigt.
  - *Lokalfenster*. Alle im aktuellen Block, gültigen Variablen werden angezeigt.
  - *Aufrufliste.* In welcher Zeile wird eine Prozedur aufgerufen?
  - *Überwachungsfenster*. Kontrolle von Variablen, Audrücken, etc.

Direktfenster

- ... kann den Programmablauf zum Teil protokollieren.
- Variablenwerte können mit Hilfe der Anweisung Debug.Print in das Direktfenster geschrieben werden.
- Variablenwerte können über das Direktfenster verändert werden.
- Im Direktfenster können Funktionen ausprobiert werden.

Anweisungen im Direktfenster

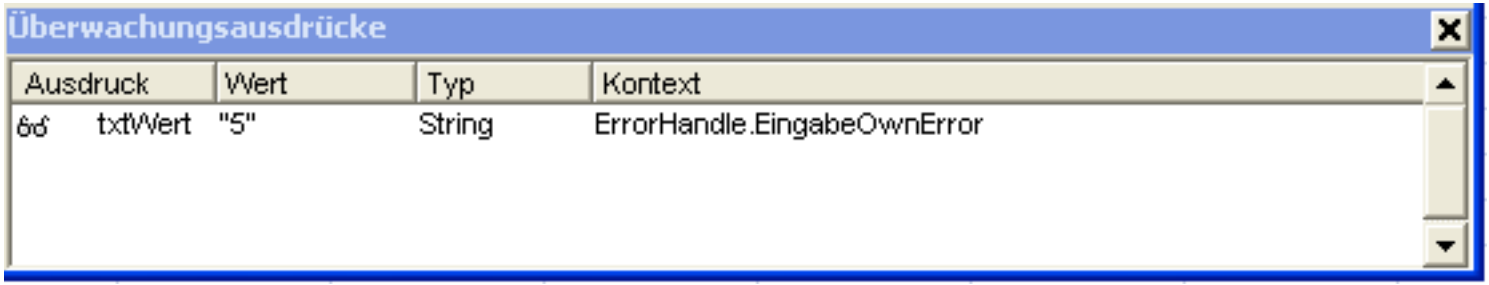
- Klicken Sie mit den Mauszeiger auf eine freie Stelle im Direktfenster.
- Geben Sie die Anweisung ?Variable ein. Schließen Sie die Anweisung mit <ENTER> ab. Der Wert der Variablen wird in der nächsten Zeile angezeigt.
- Die Anweisung variable = wert verändert den Wert der Variablen.
- Die Anweisung ?Funktion(para, para, ...) führt eine bestimmte Funktion aus. Das Resultat der Funktion wird in der nachfolgenden Zeile angezeigt. Vordefinierte Funktionen können getestet werden.

Aktueller Wert einer Variablen

- Markieren Sie gewünschte Variable mit Hilfe der Maus.
- Klicken Sie auf *Debuggen – Aktuellen Wert anzeigen*.
- In dem sich öffnenden Dialogfenster wird
  - ... die Prozedur angezeigt, in der sich die Variable befindet.
  - ... der Variablennamen angezeigt.
  - ... der aktuelle Wert der Variablen angezeigt.

Überwachungsfenster

- Eine Variable oder ein Ausdruck wird zeilenweise angezeigt.
- Folgende Informationen werden im Fenster angezeigt:
  - Der Name der Variablen.
  - Der aktuelle Wert der Variablen.
  - Der Datentyp der Variablen.
  - Wo befindet sich der zu überwachende Ausdruck.
- Das Icon am Anfang der Zeile zeigt die Art der Überwachung an. Hier wird der Überwachungsausdruck genutzt.

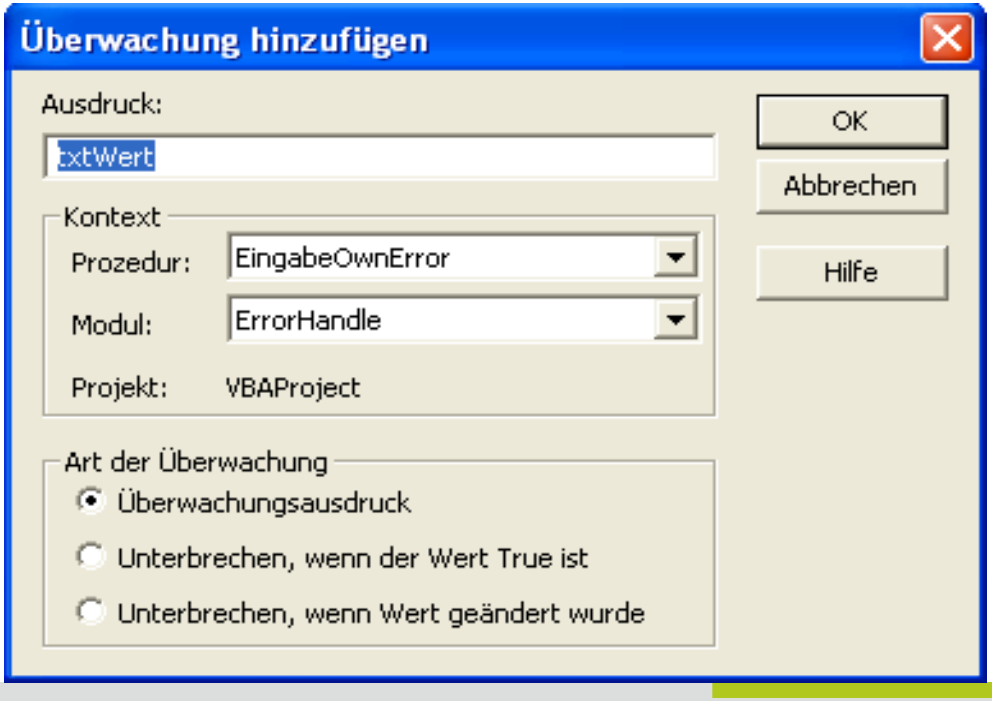


Zeilen bearbeiten

- Mit einem Klick auf das Icon wird eine Zeile vollständig markiert.
- <ENTF> löscht die Zeile aus dem Überwachungsfenster.
- Mit Hilfe der rechten Maustaste wird das dazugehörige Kontextfenster geöffnet. Klicken Sie auf den Befehl *Überwachung bearbeiten*, um die Überwachung zu verändern.
- Markieren Sie den Wert einer Variablen im Überwachungsfenster. Geben Sie mit Hilfe der Tastatur einen neuen Wert ein und bestätigen diesen mit <RETURN>.

Variable dem Überwachungsfenster hinzufügen

- Markieren Sie die zu überwachende Variable.
- Öffnen Sie mit der rechten Maustaste das dazugehörige Kontextmenü und wählen den Befehl *Überwachung hinzufügen*.
- Schließen Sie das Dialogfenster mit *OK*.



Überwachungsarten

- Überwachungsausdruck
  - Der Wert der Variablen wird im Überwachungsfenster angezeigt.
  - Das Programm wird nicht automatisch aufgrund des Überwachungsausdrucks unterbrochen.
- Unterbrechen, wenn der Wert True ist
  - Falls die boolsche Variablen den Wert true hat, wird das Programm unterbrochen.
  - Falls die numerische Variable einen Wert ungleich null hat, wird das Programm unterbrochen.
- Unterbrechen, wenn der Wert sich geändert hat
  - Sobald sich der Wert ändert, wird das Programm unterbrochen.