Programmieren in Anwendungen

Annette Bieniusa

Technische Universität Kaiserslautern

bieniusa@cs.uni-kl.de

07.05.2015

Überblick

Aufwärmübung

Arbeit mit dem Dateisystem

Ereignisorientierte Programmierung

UserForm-Dialoge

Aufwärmübung

Aufwärmübung!

Aufgabe

Schreiben Sie eine Funktion, die das Maximum dreier Zahlen berechnet!

Variante 1

```
Function Max3 (x as Integer, y as Integer, z as Integer)
    as Integer
   If x >= y Then
      If x >= z Then Max3 = x Else Max3 = z
    Else
      If y >= z Then Max3 = y Else Max3 = z
    End If
End Function
```

Variante 2

```
Function Max2 (a as Integer, b as Integer) as Integer
    If a >= b Then Max2 = a Else Max2 = b
End Function
Function Max3 (x as Integer, y as Integer, z as Integer)
    as Integer
    Max3 = Max2(x,Max2(y,z))
End Function
```

Variante 3: In Excel

```
Function Max3 (x as Integer, y as Integer, z as Integer)
    as Integer
   Max3 = WorksheetFunction.Max(x, y, z)
End Function
```

Bitte testen Sie Ihre Implementierung mindestens mit folgenden Testfällen:

- Max3(1,2,3)
- Max3(2,3,1)
- Max3(3,1,2)
- Max3(1,1,1)

Arbeit mit dem Dateisystem

Arbeit mit dem Dateisystem

- Zugriff auf beliebige Dokumente und Dateien über das Dateisystem
- Abhängig vom Betriebssystem!
- Im Folgenden behandeln wir die Varianten für MS Windows
- Für MacOS und Office 2011: `http://msdn.microsoft.com/en-us/library/jj614412(v=office.14).aspx`

Aufsuchen von Dateien

- Die `Dir`-Funktion liefern den Datei- oder Ordnernamen, der mit dem Pfadnamen und evtl. Attribut übereinstimmt.
- Platzhalter: `*` und `?`
- Wird keine passende Datei gefunden, wird der leere String zurückgegeben.
- Beim ersten Aufruf der Dir-Funktion müssen Sie einen Pfadnamen angeben. Um das nächste Element abzurufen, können Sie die Funktion danach ohne Parameter aufrufen.

```
Dir [(Pfadname(, Attribut))] As String
Dir("C:\VBATestDateien\*.doc")
```

Beispiel: Suche nach Dateien

```
Sub DateiSuchenUndEinfuegen()
  Dim Dateiname As String
 Const Pfad As String = "C:\MeineVBADateien\*.docm"

  Dateiname = Dir(Pfad)
  Do While Dateiname <> ""
    Selection.TypeText (Dateiname & vbCrLf)
    Dateiname = Dir
  Loop
End Sub
```

Filterattribute von Dateien

Konstante	Beschreibung
`vbNormal`	ohne Attribute
`vbReadOnly`	ohne Attribute + schreibgeschützt
`vbHidden`	ohne Attribute + versteckt
`vbDirectory`	ohne Attribute + Verzeichnisse
`vbVolume`	Datenträger

Weitere Operationen im Dateisystem

- Das `FileSystem`-Objekt stellt wichtige Methoden zur Verfügung.
- Bestimmen der Dateigrösse: `FileSystem.FileLen(Pfadname)`
- Kopieren einer Datei: `FileSystem.FileCopy(Dateiname)`
- Löschen einer Datei: `FileSystem.Kill(Dateiname)`
- Letzter Zeitpunkt von Änderungen:
  `FileSystem.FileDateTime(Pfadname)`
- Weitere Funktionen: `http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.filesystem.aspx`

Beispiel: Information zu einer Datei

```
Sub DateiInfo()
  Dim Dateiname As String, InfoText As String
  Dateiname = "Brief.docx"
  FileSystem.ChDir("C:\MeineVBADateien")
  InfoText = Round (FileSystem.FileLen(Dateiname) /
      1024,2) & " KB"
  InfoText = InfoText & vbCrLf & FileSystem.FileDateTime(
      Dateiname)
  MsgBox InfoText
End Sub
```

Das FileDialog Objekt

- Dialogfenster zum Öffnen und Speichern von Dateien bzw. Auswahl von Dateien und Ordnern
- Arten: `msoFileDialogOpen`, `msoFileDialogSaveAs` `msoFileDialogPicker`, `msoFileDialogFolderPicker`

```
Sub DateiAuswahl()
  Dim fd as FileDialog, Zaehler as Integer
  Set fd = Application.FileDialog(msoFileDialogFilePicker)
  'Filter als Standardeinstellung
  fd.Filters.Add "Word-Docs mit Makro","*.docm", 1
  fd.AllowMultiSelect = True
  fd.Show
  For Zaehler = 1 To fd.SelectedItems.Count
    Application.Selection.TypeText(
         fd.SelectedItems(Zaehler) & vbCrLf)
  Next
End Sub
```

Fehlerbehandlung

- Manchmal ist es hilfreich, die automatisch erzeugten Fehlermeldungen abzufangen und durch eigene Fehlermeldungen zu ersetzen.

```
Sub DateiOeffnen()
  On Error Goto EigeneFehlermeldung
    Workbooks.Open("C:\Text.xlsx")
    'Weitere Anweisungen
    Exit Sub 'beende Ausfuehrung hier!

  EigeneFehlermeldung: 'Sprungmarke
    Call MsgBox("Datei nicht gefunden!", vbCritical +
        vbOkOnly, "Fehler!")
End Sub
```

Fehlerkodes

```
On Error GoTo ErrorHandler
    Throw New DivideByZeroException() 'Fehler wird
        ausgeloest
ErrorHandler:
    If (TypeOf Err.GetException() Is DivideByZeroException
        ) Then
    ' Code zur Fehlerbehandlung hier
    End If
```

- Weitere Information über die Art des Fehlers: `Err.Number`
- Fehlerbeschreibung von VBA: `Err.Description`

Sprungmarken

- Auch mehrfache `On Error`- Anweisungen mit verschiedenen Sprungmarken möglich
- Sprungmarken müssen innerhalb der Prozedur definiert sein

```
Public Sub VerschiedeneFehler()
On Error GoTo Fehler1
'Code
On Error GoTo Fehler2
'Code
Exit Sub
Fehler1:
MsgBox "Fehler im ersten Teil"
Exit Sub
Fehler2:
MsgBox "Fehler im zweiten Teil"
End Sub
```

Ereignisorientierte Programmierung

Ereignisorientierte Programmierung

- *Ereignisse* treten z.B. beim Arbeiten mit Steuerelementen auf.
- Ereignisse können durch den Benutzer direkt (beispielsweise durch Anklicken von Buttons, Wechsel zwischen Dokumenten), aber auch durch das System selbst angestossen werden (z.B. Öffnen oder Speichern von Dokumenten).
- Ereignisprozeduren sind Makros, die als Reaktion auf bestimmte Ereignisse ausgeführt werden.

```
' Hebt bei Markieren einer Zelle die gesamte Zeile und
    Spalte hervor.
Sub Worksheet_SelectionChange(ByVal Target As Range)
    Cells.Interior.ColorIndex = xlColorIndexNone
    ActiveCell.EntireRow.Interior.ColorIndex = 15
    ActiveCell.EntireColumn.Interior.ColorIndex = 15
End Sub
```

Beispiel: Durchlauf von Arbeitsblättern

```
Sub Workbook_Open()
    Application.OnKey Key:="{PgUp}", Procedure:="SheetsUp"
    Application.OnKey Key:="{PgDn}", Procedure:="
        SheetsDown"
End Sub

Sub SheetsUp()
    Dim i As Integer
    i = ActiveSheet.Index + 1
    If i <= Sheets.Count Then Sheets(i).Select
End Sub

Sub SheetsDown()
    Dim i As Integer
    i = ActiveSheet.Index - 1
    If i >= 1 Then Sheets(i).Select
End Sub
```

Beispiel: Automatisierte Speicherabfrage

```
Sub Workbook_Open()
    Application.OnTime EarliestTime:=Now + TimeValue("
        00:10:00"), Procedure:="SaveWorkbook"
End Sub

Sub SaveWorkbook()
    If MsgBox("Save workbook?", vbYesNo) = vbYes Then
        ActiveWorkbook.Save
    Application.OnTime EarliestTime:=Now + TimeValue("
        00:10:00"), Procedure:="SaveWorkbook"
End Sub
```

UserForm-Dialoge

Dialoge und Formulare

Integrierte Dialoge für häufig verwendete Abfragen, z.B. Druck- oder Speicherdialog

Vordefinierte VBA-Dialoge für einfache Benutzerein- und ausgaben, z.B. `InputBox` und `MsgBox`

Benutzerdefinierte Dialoge (UserForm-Dialoge) für komplexere Interaktion

Dokumente als Formulare durch direktes Einbetten von Steuerelementen

Grundlagen zu UserForm-Dialogen

- Vielzahl von Steuerelementen, z.B. Listenfelder, Schaltflächen, Eingabefelder
- UserForm-Diloge werden durch Drag&Drop in der VBA-Entwicklungsansicht zusammengestellt.
- Flexible Reaktion auf Benutzereingaben durch Ereignisprozeduren, z.B. Hinzufügen von Werten

```
'Anzeigen eines Formulars
frmName.Show

'Beim Aktivieren eines Formulars
Sub frmName_Activate()
...
End Sub

'Ereignisprozeduren erstellen
Sub frmName_Ereignisname
Sub SteuerelementName_Ereignisname

'Ressourcenfreigabe
Unload frmName
```

Steuerelemente in UserForm-Dialogen

Textfelder (Lables) `lblName`

- Werte zuweisen bzw. auslesen: `frmName.lblName.Value`
- Ereignis bei Änderung: `lblName_Change`

Schaltflächen (CommandButton) `cmdName`

- Beschriftung: `frmName.cmdName.Caption`
- Aktivierung / Deaktivierung:

  `frmName.cmdName.Enabled = True/False`
- Ereignis bei Betätigen: `frmName.cmdName_Click`

Steuerelemente in UserForm-Dialogen II

Listenfeld (List) `lstName`

- Elemente hinzufügen: `frmName.lstName.AddItem "Wert"`
- Elemente entfernen: `frmName.lstName.RemoveItem Indexwert`
- Indes des ausgewählten Elements: `frmName.lstName.ListIndex`
- Text des ausgewählten Elements: `frmName.lstName.Name`
- Ereignis bei Auswahl: `lstName_Change`

Kontrollfeld (Checkbox) `chkName`

- Auswahl prüfen, liefert `True` oder `False`: `frmName.chkName.Value`
- Ereignis bei Anklicken: `chkName_Click`

und viele mehr!

Beispiel: Buchstaben zählen

```
'Bestimmen der Anzahl
Private Sub cmdStart_Click()
  Dim satz, buchstabe As String
  Dim zahl As Integer

  'Einlesen der Eingabe
  satz = txtEingabeText.Text
  buchstabe = txtEingabeBuchstabe.Text

  'Test auf Vollstaendigkeit
  If satz = "" Then
    MsgBox "Sie haben keinen Satz eingegeben!", vbCritical, "Achtung!"
    Exit Sub
  ElseIf buchstabe = "" Then
    MsgBox "Sie haben keinen Buchstaben angegeben!", vbCritical, "Achtung!"
    Exit Sub
  End If

  'Falls Gross-Kleinschreibung ignoriert werden soll,
  'wird die Eingabe in Kleinschreibung umgewandelt
  If chkGrossKlein.Value Then
    buchstabe = LCase(buchstabe)
    satz = LCase(satz)
  End If
  'Bestimmen der Anzahl = upper bound eines Arrays
  zahl = UBound(Split(satz, buchstabe))
  'Ausgabe des Ergebnisses
  lblAusgabe.Caption = "Der Satz besitzt " & zahl & " mal den Buchstaben " &
       buchstabe
  cmdReset.Visible = True
End Sub
```

Beispiel: Buchstaben zählen - Teil 2

```
'Zuruecksetzen der Felder
Private Sub cmdReset_Click()
  lblAusgabe.Caption = ""
  txtEingabeBuchstabe.Text = ""
  txtEingabeText.Text = ""
  chkGrossKlein.Value = False
  cmdReset.Visible = False
End Sub
```