

Programmieren in Anwendungen

Annette Bieniusa

Technische Universität Kaiserslautern

bieniusa@cs.uni-kl.de

15.05.2014

Überblick

Hinweise zu Übungsblatt 1

Arbeit mit dem Dateisystem

Sicherheitsaspekte

Ereignisorientierte Programmierung

UserForm-Dialoge

Hinweise zu Übungsblatt 1

Hinweise zu Übungsblatt 1

Aufgabe 1

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

```
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
```

Hinweise zu Übungsblatt 1

Aufgabe 1

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

```
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
```

```
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
```

Hinweise zu Übungsblatt 1

Aufgabe 1

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

```
'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](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
<code>vbNormal</code>	ohne Attribute
<code>vbReadOnly</code>	ohne Attribute + schreibgeschützt
<code>vbHidden</code>	ohne Attribute + versteckt
<code>vbDirectory</code>	ohne Attribute + Verzeichnisse
<code>vbVolume</code>	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
```

Fehlercodes

```
On Error GoTo ErrorHandler
    Throw New DivideByZeroException() 'Fehler wird
        ausgelöst
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
```

Sicherheitsaspekte

Sicherheitseinstellungen für Makros

- ▶ Akzeptieren Sie nur Makros aus Quellen, denen Sie vertrauen!
- ▶ Ab Office 356: Authentifizierung von vertrauenswürdigen Quellen/Programmierern
- ▶ Standardmässig werden dann nur Makros ausgeführt, die aus vertrauenswürdigen Quellen stammen (Überprüfung durch digitale Signaturen).
- ▶ Die Freigabe beliebiger Makros ist nicht empfehlenswert, da hierbei grosse Sicherheitslücken geschaffen werden!

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
```

```
UserFormName.Show
```

```
'Beim Aktivieren eines Formulars
```

```
Sub UserFormName_Activate()
```

```
...
```

```
End Sub
```

```
'Ereignisprozeduren erstellen
```

```
Sub UserFormName_Ereignisname
```

```
Sub SteuerelementName_Ereignisname
```

```
'Ressourcenfreigabe
```

```
Unload UserFormName
```

Steuerelemente in UserForm-Dialogen

Textfelder

- ▶ Werte zuweisen bzw. auslesen: `UserFormName.TextfeldName.Value`
- ▶ Ereignis bei Änderung: `TextfeldName_Change`

Schaltflächen

- ▶ Beschriftung: `UserFormName.SchaltflaecheName.Caption`
- ▶ Aktivierung / Deaktivierung:
`UserFormName.SchaltflaecheName.Enabled = True/False`
- ▶ Ereignis bei Betätigen: `UserFormName.SchaltflaecheName_Click`

Steuerelemente in UserForm-Dialogen II

Listenfeld

- ▶ Elemente hinzufügen: `UserFormName.ListenfeldName.AddItem "Wert"`
- ▶ Elemente entfernen:
`UserFormName.ListenfeldName.RemoveItem Indexwert`
- ▶ Index des ausgewählten Elements:
`UserFormName.ListenfeldName.ListIndex`
- ▶ Text des ausgewählten Elements:
`UserFormName.ListenfeldName.Name`
- ▶ Ereignis bei Auswahl: `ListenfeldName_Change`

Kontrollfeld

- ▶ Auswahl prüfen, liefert `True` oder `False`:
`UserFormName.KontrollfeldName.Value`
- ▶ Ereignis bei Anklicken: `KontrollfeldName_Click`