

Programmieren in Anwendungen

Annette Bieniusa

Technische Universität Kaiserslautern

bieniusa@cs.uni-kl.de

08.05.2014

Überblick

Wiederholung: Visual Basic for Applications (VBA)

Datentypen

Ausdrücke

Kontrollstrukturen

Prozeduren und Funktionen

Im Dialog mit dem Benutzer

Objekte in VBA

VBA Objekte in Word und Excel

Themen

Visual Basic for Applications(VBA)

- ▶ Wiederholung grundlegender Konzepte der imperativen Programmierung anhand der neuen Sprache VB (Visual Basic)
- ▶ Einführung in die ereignisorientierte Programmierung
- ▶ Anwendungsbeispiele mit VBA in Word und Excel

Statistik und Grafiken mit R

- ▶ Einführung in die Statistiksoftware R
- ▶ Wiederholung grundlegender Konzepte der Statistik und Datenanalyse
- ▶ Datenvisualisierung und Datenanalyse in R an ausgewählten Fallstudien

Visual Basic for Applications (VBA)

- ▶ Skriptsprache zur Automatisierung und Anpassung von Microsoft Office Programmen
- ▶ Modul-orientiert und prozedural
- ▶ Einsatzfelder
 - ▶ Automatisiertes Erzeugen von Dokumenten wie Serienbriefen
 - ▶ Benutzerdefinierte Dialogfenster oder Fehlermeldungen

Literaturhinweis

VBA-Programmierung - Integrierte Lösungen mit Office 2010, 1. Auflage, Okt 2010 (erhältlich im Rechenzentrum bzw. HERDT-Verlag)

Beispiel: Berechnung von Zeiträumen

```
'Dauer des Semesters
Sub AnzahlTage()
    Dim Heute As Date, Semesterende As Date, Ausgabe As
        String
    Heute = Date
    Semesterende = DateValue("25.07.2014")
    Ausgabe = "Bis zum Semesterende sind es noch " &
        DateDiff("d", Heute, Semesterende) & " Tage."
    MsgBox Ausgabe
End Sub
```

Datentypen

- ▶ Der Datentyp definiert die Art sowie den Wertebereich einer Variablen oder Konstanten.
- ▶ Abhängig vom Datentyp wird Speicherplatz für Variablen bzw. Konstanten reserviert.
- ▶ Bei impliziten Deklarationen wird in VBA der Datentyp `Variant` verwendet.

Übersicht: Datentypen

Integer	%	Ganze Zahlen	[-32.768,32.767]
Long	&	Ganze Zahlen	[-2.147.483.648, 2.147.483.647]
Single	!	Fliesskommazahlen	7 Ziffern Genauigkeit
Double	#	Fliesskommazahlen	15 Ziffern Genauigkeit
Currency	@	Fliesskommazahlen	15 Vor- und 4 Nachkommastellen
String	\$	Zeichenketten	bis zu 65.535 Zeichen
Date		Datum und Uhrzeit	
Boolean		Logische Werte	True und False
Variant		Beliebig Wert / Objekt	

Datentypen in Variablendeklarationen

```
Dim VarName1 As Datentyp1, VarName2 As Datentyp 2, ...  
Dim VarName1Typkennzeichen, VarName2Typkennzeichen, ...
```

Beispiele:

```
Dim Alter As Integer, Name as String  
Dim Kundennummer&, Rechnungsbetrag@
```

- ▶ Numerische Variablen werden mit 0 initialisiert.
- ▶ `Variant`-Variablen werden mit `Empty` initialisiert.
- ▶ String-Variablen werden mit `""` (leerer String) initialisiert.

Umgang mit Datentypen

```
Sub ErmittleMonat1()  
    Dim strWert As String  
  
    strWert = "24.12.2012"  
    MsgBox "Monat: " & Mid(strWert, 4, 2)  
End Sub
```

Umgang mit Datentypen

```
Sub ErmittleMonat1()  
    Dim strWert As String  
  
    strWert = "24.12.2012"  
    MsgBox "Monat: " & Mid(strWert, 4, 2)  
End Sub
```

Funktioniert nicht bei "strWert = "1.1.2012"!

Umgang mit Datentypen

```
Sub ErmittleMonat1()  
    Dim strWert As String  
  
    strWert = "24.12.2012"  
    MsgBox "Monat: " & Mid(strWert, 4, 2)  
End Sub
```

Funktioniert nicht bei "strWert = "1.1.2012"!

```
Sub ErmittleMonat2()  
    Dim datWert As Date  
  
    datWert = #12/24/2012#  
    MsgBox "Monat: " & Month(datWert)  
End Sub
```

Ausdrücke

Wiederholung: Variablen

- ▶ Mit Hilfe von Variablen kann man (temporär) Werte speichern und diese in den Prozeduren verwenden.
- ▶ Der Wert einer Variablen kann durch eine *Zuweisung* verändert werden.
- ▶ Variablen werden über *Bezeichner (Variablennamen)* referenziert.
- ▶ Die *Deklaration* einer Variablen ist das Vereinbaren einer Variablen vor ihrem ersten Gebrauch.

```
Dim Variablenname As Datentyp
```

- ▶ Beispiele:

```
Dim Anzahl As Integer  
Dim AusgabeText As String  
Dim Alter As Integer, Temperatur As Integer  
Dim Gestern As Date
```

Konstanten

- ▶ Konstanten werden ebenfalls über Bezeichner referenziert, sind aber **unveränderlich**.
- ▶ Einer Konstanten wird bereits während der Deklaration ein Wert zugewiesen, der später nicht mehr verändert werden kann.

```
Const Konstantenname As Datentyp = Ausdruck
```

- ▶ Beispiele:

```
Const Pi as Double = 3.14159
```

```
Const Programmname as String = "Mein Programm"
```

- ▶ VBA stellt eine Vielzahl von Konstanten zur Verfügung (siehe z.B. Abschnitt Meldungsfenster)
- ▶ Wichtig String-Konstanten: Zeilenumbruch `vbCrLf` , Tabulator `vbTab`

Operatoren

- ▶ Ein Ausdruck ist eine Kombination aus Werten, Variablen, Konstanten und Operatoren.
- ▶ Arithmetische Operatoren: +, -, *, /, \, Mod, ^
- ▶ Vergleichsoperatoren: <, <=, >, >=, =, <>
- ▶ Logische Operatoren: Not, And, Or
- ▶ Verkettungsoperator: & (Konkatenation von Strings)

Vergleiche mit String-Mustern

- ▶ Der Vergleichsoperator Like wird verwendet, um Strings mit String-Mustern zu vergleichen.

Zeichen	Bedeutung	Beispiel
?	Ein einzelnes Zeichen	"Hallo"Like "H?lo"-> false
*	Kein oder mehrere Zeichen	"Haut"Like "H*t"-> true
[Liste]	Ein Zeichen der Liste	"X" Like "[A-Z]"-> true
[!Liste]	Ein Zeichen nicht in der Liste	"K" Like "[!a-m]"-> false

Kontrollstrukturen

Wiederholung: Verzweigungen

- ▶ Bei Verzweigungen werden Programmteile abhängig von einer Bedingung ausgewertet.

```
If Ausdruck Then
    ...
Else
    ...
End If
```

```
If Alter >= 18 Then
    MsgBox "Normaltarif"
Else
    MsgBox "Jugendtarif"
End If
```

Schleifen

- ▶ Mit Schleifen wird ein Anweisungsblock wiederholt ausgeführt.
- ▶ Zählergesteuerte Wiederholung

```
For Zaehler = Start To Ende [Step d]  
    Anweisungsblock  
Next
```

- ▶ Beispiel

```
For X = 1 To 20 Step 5  
    Debug.Print "X ist " & X  
Next
```

- ▶ Schrittweite wird durch `step` angepasst, ohne Angabe wird Schrittweite 1 verwendet.

Schleifen unter Bedingungen

- ▶ Kopfgesteuerte bedingte Wiederholung

```
Do While/Until Ausdruck
    Anweisungsblock
Loop
```

```
X = 1
Do Until X >= 20
    Debug.Print "X ist " & X
    X = X + 5
Loop
```

```
Y = 1
Do While Y < 20
    Debug.Print "Y ist " & Y
    Y = Y + 5
Loop
```

- ▶ Fussgesteuerte bedingte Wiederholung

```
Do
    Anweisungsblock
Loop While/Until
    Ausdruck
```

```
X = 1
Do
    Debug.Print "X ist " & X
    X = X + 5
Loop Until X >= 20
```

- ▶ Fussgesteuerte Schleifen werden immer mind. einmal ausgeführt!

Prozeduren und Funktionen

Übersicht: Prozeduren

- ▶ Prozeduren bestehen aus einer Folge von Anweisungen (z.B. Zuweisungen von Variablen, Prozeduraufrufen, Verzweigungen, Schleifen, ...).
- ▶ Wichtige Form der Abstraktion beim Programmieren!

Sub-Prozeduren

- ▶ Sub-Prozeduren geben keinen Wert zurück.
- ▶ Syntax von einfachen Sub-Prozeduren

```
Sub Prozedurname ()  
...  
End Sub
```

- ▶ Syntax von Prozeduraufrufen

```
Prozedurname  
Call Prozedurname
```

Prozeduren mit Parametern

- ▶ Syntax von Sub-Prozeduren mit Parametern

```
Sub Prozedurname ([ByVal|ByRef] Parametername [As Type  
],...)  
...  
End Sub
```

- ▶ Syntax von Prozeduraufrufen

```
Prozedurname (Ausdruck, ...)  
Call Prozedurname (Ausdruck, ...)
```


Prozeduren mit Parametern und Call-by-Value

- ▶ Der Wert des Ausdrucks wird als Kopie an den die Prozedur weitergegeben. Die Prozedur kann den übergebenen Wert verändern, ohne dass sich der ursprüngliche Wert ändert (**call by value**).

```
Sub IncrVal (ByVal i as Integer)
    i = i + 1
End Sub
...
Dim j as Integer, k as Integer
j = 10
Call IncrVal(j)
k = j 'k hat hier den Wert 10
```

Prozeduren mit Parametern und Call-by-Reference

- ▶ Alternativ kann ein Parameter eine Referenz auf die Variable erhalten, die den Wert enthält (**call by reference**).
- ▶ Die Prozedur arbeitet dann nicht mit einer Kopie, sondern kann die Variable selbst direkt verändern.

```
Sub IncrRef (ByRef i as Integer)
    i = i + 1
End Sub
...
Dim j as Integer, k as Integer
j = 10
Call IncrRef(j)
k = j 'k hat hier den Wert 11
```

Funktionen

- ▶ Funktionen können, wie Prozeduren, mehrere Anweisungen ausführen und geben **immer** einen Wert an das aufzurufende Programm zurück.
- ▶ Syntax von Funktionen

```
Function Funktionsname ([ByVal|ByRef] Parametername [As  
    Type],...) [As Type]  
...  
Funktionsname = Rueckgabewert  
...  
End Function
```

- ▶ Wenn kein Rückgabewert spezifiziert wird, wird ein Standardwert entsprechend dem Datentypen der Funktion zurückgegeben.
- ▶ Syntax von Funktionsaufrufen

```
Funktionsname (Ausdruck , ...)
```
- ▶ Während Prozeduraufrufe eigenständige Anweisungen sind, sind Funktionsaufrufe Ausdrücke.

Beispiele: Prozeduren und Funktionen

```
Sub AbsatzFormatieren()  
    ActiveDocument.Selection.Range.Bold = True  
    ActiveDocument.Selection.Range.Italic = True  
End Sub
```

```
Function Max(x as Integer, y as Integer) as Integer  
    If x < y Then  
        Max = y  
    Else  
        Max = x  
    End Function
```

Aufruf von Prozeduren und Funktionen mit Parametern

- ▶ Wenn der Aufruf in einer Zeile steht, sind verschiedene Varianten möglich:

```
ProcName A,B,C  
Call ProcName(A,B,C)
```

- ▶ Aus der Praxis: Verwenden Sie in diesem Fall `call` zum Aufruf von Methoden und klammern Sie die Argumente!
- ▶ Sonst wird die Variante mit Klammern und ohne `call` verwendet (`ProcName(A,B,C)`).
- ▶ Hat eine Prozedur nur *ein* Argument und wird dieses eingeklammert ohne `call` zu verwenden, wird zunächst das Argument ausgewertet. Dabei kann aus einem Call-by-Ref unbeabsichtigt ein Call-by-Value werden, oder auch ein Typkonversionsfehler passieren.

```
Call IncrRef(j)   'Semantik von Call-By-Ref  
IncrRef(j)       'Semantik von Call-By-Val  
IncrRef j        'Semantik von Call-By-Ref
```

Im Dialog mit dem Benutzer

Wiederholung: Meldungsfenster

- ▶ Meldungsfenster können dazu genutzt werden, dem Benutzer Informationen mitzuteilen und auch abzufragen. Sie bestehen aus dem *Meldungstext* und standardmäßig der Schaltfläche "OK". Sie kann optional mit einem Titel, Informationssymbolen und weiteren Schaltflächen ergänzt werden.

- ▶ Einfaches Meldungsfenster

```
MsgBox("Meldungstext")
```

- ▶ Meldungsfenster mit Titel und Informationssymbol

```
MsgBox("Meldungstext", vbInformation, "Titel")
```

Schaltflächen

Mögliche Kombinationen von Schaltflächen

Konstante	Wert	Schaltfläche
<code>vbOkOnly</code>	0	OK
<code>vbOkCancel</code>	1	OK und Abbrechen
<code>vbAbortRetryIgnore</code>	2	Abbrechen, Wiederholen und Ignorieren
<code>vbYesNoCancel</code>	3	Ja, Nein und Abbrechen
<code>vbYesNo</code>	4	Ja und Nein
<code>vbRetryCancel</code>	5	Wiederholen und Abbrechen

Rückgabewerte von Schaltflächen

Konstante	Wert	gewählte Schaltfläche
vbOk	1	OK
vbCancel	2	Abbrechen
vbAbort	3	Abbrechen
vbRetry	4	Wiederholen
vbIgnore	5	Ignorieren
vbYes	6	Ja
vbNo	7	Nein

Eingabedialoge

- ▶ Die Funktion `InputDialog` erzeugt Eingabedialog mit Text und einer Eingabezeile.
- ▶ Der vom Anwender eingegebene Wert wird als String zurückgeliefert. Bei Betätigen der Schaltfläche `Abbrechen` ist es der leere String `""`.
- ▶ Der optionale Parameter `Default` legt den Wert fest, der standardmässig im Eingabefeld angezeigt wird.

```
InputDialog (Text, [Title], [Default])
```

Objekte in VBA

Objekte in VBA

- ▶ Alle Elemente in MS Office, wie Dokumente, Tabellen, Graphiken, etc., sind Objekte.
- ▶ Typische Objekte in Excel sind Arbeitsmappen (`Workbook`), Tabellenblätter (`Worksheet`), Diagramme (`Charts`) und Zellen (`Range`, `Cell`).
- ▶ Das gerade aktive Objekt wird mittels `ActiveXXX` referenziert (z.B. `ActiveWorkbook` oder `ActiveCell`).
- ▶ Eine vollständige Übersicht listen die Developer Referenzen (z.B. [http://msdn.microsoft.com/en-us/library/office/ff846392\(v=office.14\).aspx](http://msdn.microsoft.com/en-us/library/office/ff846392(v=office.14).aspx)).

Objekte

- ▶ Objekte sind Programmeinheiten, die Daten sowie die Prozeduren zum Verarbeiten dieser Daten enthalten.
- ▶ Objekte haben
 - ▶ einen Zustand (definiert durch Eigenschaften/Properties),
 - ▶ ein Verhalten (definiert durch objektspezifische Prozeduren/Methoden) und
 - ▶ eine Identität (wodurch es sich von Objekten des gleichen Typs unterscheidet).

Klassen

- ▶ Eine Klasse ist eine Art Modell für Objekte des gleichen Typs.
- ▶ Sie dient als Bauplan für die einzelnen Objekte und definiert deren Eigenschaften und Methoden.
- ▶ Beispiel: `Document` bezeichnet die Klasse für Word-Dokumente, `ActiveDocument` eine Objekt-Instanz der Klasse `Document`.

Zugriff auf Methoden und Eigenschaften

- ▶ Objektverweise sind Variablen, die eine Referenz auf ein Objekt enthalten.

- ▶ Deklaration von Objektvariablen

```
Dim Objektvariable as Objekttyp
```

- ▶ Zuweisung auf Objektvariable:

```
Set Objektvariable = Objekt
```

- ▶ Zugriff auf Eigenschaften:

```
Objektvariable.Eigenschaftsname
```

- ▶ Aufruf von Methoden:

```
Objektvariable.Methodenname
```

```
Objektvariable.Methodenname(Parameter)
```

- ▶ Löschen von Objektvariablen

```
Set Objektvariable = Nothing
```

Auflistungen (Collections)

- ▶ Auflistungen sind spezielle Objekte, die aus einer Menge von Objekten des gleichen Typs bestehen.
- ▶ Beispiel: Die Auflistung `Worksheets` in Excel enthält alle aktuell geöffneten Tabellenblätter.
- ▶ Auf einzelne Objekte in einer Auflistung wird über einen Indexwert zugegriffen (beginnend mit 1).

```
Sub Blaetter()  
    Dim Anzahl As Integer, Index as Integer  
    Anzahl = Worksheets.Count  
    For Index = 1 To Anzahl  
        Debug.Print Worksheets(Index).Name  
    End Sub
```

- ▶ Häufig kann auch der Name des Objekts zum Zugriff verwendet werden.

```
Workbooks("Mappe1.xlsm").Worksheets("Tabelle2").Range(  
    "A1:B5")
```


VBA Objekte in Word und Excel

Beispiel: Dokumente in Word

```
Sub DokumentOeffnenSchliessenErstellen()  
  Dim Brief As Document  
  Set Brief = Documents.Open(ActiveDocument.Path & "\  
    Testbrief.docx")  
  Brief.Close  
  Set Brief = Nothing 'gibt den Speicherplatz wieder  
    frei  
End Sub
```

Activate	Aktiviert das Dokument
Close	Schliesst und speichert das Dokument
Path	Pfad zum Speicherort des Dokuments
Printout	Druckt das Dokument
Save	Speichert das Dokument
SaveAs	Speichert das Dokument unter einem neuen Namen

Weitere Methoden/Eigenschaften: Verwendung von Templates,
Schreib-/Passwortschutz

Textbereiche

- ▶ Ein `Range` Objekt referenziert einen zusammenhängenden Bereich eines Dokuments.
- ▶ Es wird über einen Start- und einen Endcharacter definiert.
- ▶ Ein `Range` Objekt kann auch nur die Eingabemarke definieren.

```
Sub TestRangeObjects()  
    Dim rngIns As Range, rngPar1 As Range, rngPar2 as  
        Range  
    Dim doc As Document  
    Set rngPar1 = doc.Paragraphs(1).Range  
    MsgBox "Der 1. Absatz beginnt mit dem Wort " &  
        rngPar1.Words.First & " ."  
    Set rngPar2 = doc.Range(Start:=doc.Paragraphs(2).  
        Range.Start, End:=doc.Paragraphs(3).Range.End)  
    Set rngIns = doc.Range(Start:=0, End:=0)  
    rngIns.InsertBefore "Hello "  
End Sub
```

Markierte Textbereiche

- ▶ Das `Selection` Objekt referenziert den markierten Bereich des aktuellen Dokuments.

```
Sub TestSelectionObject()  
    Dim rngParagraph As Range  
    Selection.Font.Bold = True  
  
    Select Case Selection.Type  
        Case wdSelectionNormal  
            MsgBox "Sie haben folgenden Text markiert: "&  
                Selection.Text  
        Case wdSelectionIP  
            MsgBox "Sie haben nichts markiert"  
  
        Set rngParagraph = ActiveDocument.Paragraphs(2).Range  
        rngParagraph.Select  
        Selection.Font.Italic = True  
    End Sub
```

Zugriff auf Dokumenteninhalte

Characters	Auflistung einzelner Zeichen
Sentences	Auflistung von Sätzen
Paragraphs	Auflistung von Absätzen
Words	Auflistung von Worten
Start/ End	Start- bzw. Endposition
Text	Textinhalt

Beispiel: Arbeitsmappen und Tabellen in Excel

- ▶ Arbeitsmappen (Workbooks) beinhalten Tabellenblätter (Worksheets) und Diagramme (Charts).
- ▶ UsedRange referenziert den verwendeten Bereich eines Tabellenblatts.
- ▶ CurrentRegion bezeichnet einen Bereich gefüllter Zellen, die von leeren Zellen umgeben sind.
- ▶ Mittels Cells lassen sich einzelne Zellen referenzieren.

```
Sub ArtikelSuchen()  
    Dim ArtNr As String, Zaehler As Integer  
    ArtNr = InputBox ("Geben Sie eine Artikelnummer ein: ",  
        "Artikel suchen")  
    ThisWorkbook.Sheets("Artikel").Activate  
    For Zaehler = 1 To Range("A1").CurrentRegion.Rows.Count  
        If Cells(Zaehler,1).Value = ArtNr Then  
            MsgBox "Artikel wurde gefunden"  
            Exit Sub  
        End If  
    Next  
    MsgBox "Artikel nicht vorhanden"  
End Sub
```

Zellbereiche

- ▶ Ein `Range` Objekt bezeichnet in Excel einen zusammenhängenden Zellbereich.

```
Worksheets(1).Range("A4").Value  
Worksheets("Messreihe").Range("B2:B8").Count  
Range("A1:A5", "B1:B5").Count  
    'verwendet implizit das aktuelle Tabellenblatt  
Range(Cells(3,3), Cells(4,6)).Select  
    'entspricht Range("C3:F3").Select
```

Zusammenfassung

- ▶ Ausdrücke: Variablen, Konstanten, zusammengesetzte Ausdrücke mit Operatoren, Funktionsaufrufe
- ▶ Anweisungen (Statements): Prozeduraufrufe, Kontrollstrukturen (Verzweigungen, Schleifen)
- ▶ Meldungs- und Eingabefenster
- ▶ Objekte mit Eigenschaften und Methoden
- ▶ VBA Objekte in Word und Excel (Auswahl durch `Selection` und `Range`)