

Programmieren in Anwendungen

Annette Bieniusa

Technische Universität Kaiserslautern

bieniusa@cs.uni-kl.de

24.04.2014

Überblick

Organisatorisches

Einführung in VBA

Was ist VBA?

Entwicklungsumgebung

Arbeiten mit Makros

Administratives

- ▶ Homepage mit Informationen und Material
http://softech.informatik.uni-kl.de/Homepage/PIA_SS14
- ▶ Vorlesung: donnerstags, 17:15 - 19:00, in Raum 52-207
- ▶ Saalübung: freitags, Fr 13:45h - 15:15h, in Raum 48-379
- ▶ **Wenn möglich, Raumwechsel in Terminalraum 32-411**
- ▶ Flexibler Wechsel zwischen Vorlesung und Übung wegen der vielen Feiertage im Sommersemester
- ▶ Bitte im KIS anmelden, um aktuelle Informationen per Email zu erhalten!

Fragen über Fragen

- ▶ Vorwissen: Welche Veranstaltungen haben Sie bisher besucht?
 - ▶ Informatik-Vorlesungen
 - ▶ Programmierkurse
 - ▶ Einführung in Statistik oder Wahrscheinlichkeitsrechnung

Fragen über Fragen

- ▶ Vorwissen: Welche Veranstaltungen haben Sie bisher besucht?
 - ▶ Informatik-Vorlesungen
 - ▶ Programmierkurse
 - ▶ Einführung in Statistik oder Wahrscheinlichkeitsrechnung
- ▶ Können Sie selbständig arbeiten?

Fragen über Fragen

- ▶ Vorwissen: Welche Veranstaltungen haben Sie bisher besucht?
 - ▶ Informatik-Vorlesungen
 - ▶ Programmierkurse
 - ▶ Einführung in Statistik oder Wahrscheinlichkeitsrechnung
- ▶ Können Sie selbständig arbeiten?
- ▶ Abschluss: Klausur oder Projektausarbeitung?

Fragen über Fragen

- ▶ Vorwissen: Welche Veranstaltungen haben Sie bisher besucht?
 - ▶ Informatik-Vorlesungen
 - ▶ Programmierkurse
 - ▶ Einführung in Statistik oder Wahrscheinlichkeitsrechnung
- ▶ Können Sie selbständig arbeiten?
- ▶ Abschluss: Klausur oder Projektausarbeitung?
- ▶ Was erwarten Sie von dieser Veranstaltung?

Themen

Visual Basic for Applications(VBA)

- ▶ Wiederholung grundlegender Konzepte der imperativen Programmierung anhand der Programmiersprache 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
- ▶ Basiert auf der Syntax von Visual Basic (nicht mehr kompatibel seit VB.NET)
- ▶ Proprietärer Microsoft-Standard
- ▶ Modul-orientiert und prozedural

Literaturhinweis

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

Typische Einsatzgebiete

- ▶ Automatisiertes Erzeugen von Dokumenten wie Serienbriefen
- ▶ Benutzerdefinierte Dialogfenster oder Fehlermeldungen
- ▶ Dokumentstatistiken erstellen
- ▶ Daten aus anderen Anwendungen einbinden (insbesondere Access-Datenbanken)
- ▶ Einbinden von Funktionalität spezifischer Office-Anwendungen (integrierte Lösungen)
 - ▶ Umsatz- und Budgetzahlen aus einer Access-Datenbank werden in Excel ausgewertet und visualisiert.
 - ▶ Umfangreiche Excel-Tabellen können über Word kompakt gedruckt werden.

Microsoft Office-Anwendungen



Word Textverarbeitung,
Seriendruck



Excel Tabellenkalkulation,
Formeln, Diagramme

Access Relationale
Datenbanken,
Formulare



PowerPoint Folienpräsentation,
Begleitmaterial

Outlook E-Mail,
Terminplanung,
Adressbücher



Die VBA-Entwicklungsumgebung

The screenshot displays the Microsoft Word VBA development environment. The main window is titled "Normal.dotm - Modul1 (Code)" and contains the following VBA code:

```
Sub AnzahlDokumente()  
End Sub
```

The interface includes a menu bar with options like "Word", "Datei", "Bearbeiten", "Ansicht", "Einfügen", "Format", "Debuggen", "Ausführen", "Extras", "Fenster", and "Hilfe". The status bar at the bottom right shows "Mi. 22:41" and the user name "Annette Bienius".

On the left side, there is a "Projekt" (Project) pane showing a tree view of the project structure:

- Normal (Normal.dotm)
 - Microsoft Word Objekte
 - ThisDocument
 - Module
 - Modul1
- Project (Dokument6)
 - Microsoft Word Objekte
 - ThisDocument
 - Verweise
 - Reference to Normal

Below the Project pane is the "Eigenschaften" (Properties) window, which is currently empty.

At the bottom of the screen, there is a "Direkt" (Direct) window, which is also empty.

Elemente der VBA-Entwicklungsumgebung

Editor, Werkzeuge, Projektverwaltung

Projekt-Explorer gibt eine hierarchisch geordnete Übersicht von allen Elementen eines Projekts.

Eigenschaftenfenster listet die Eigenschaften und aktuellen Eigenschaftswerte des markierten Elements.

Code-Fenster/Modulfenster dient zum Eingeben und Bearbeiten des Programmcodes.

Direktfenster erlaubt es einzelne Anweisungen direkt auszuführen.

Lokal-Fenster zeigt die Werte aller lokaler Variablen im Debug-Modus an.

Module

- ▶ VBA-Programme bestehen aus einer Kombination von Modulen.
- ▶ Jedes Modul enthält Deklarationen von Konstanten, Variablen und Prozeduren.

Standardmodule sind für ein gesamtes Projekt gültig.

Klassenmodule definieren ein neues Objekt mit zugehörigen Eigenschaften und Methoden.

Dokumentenmodule sind an ein konkretes Dokument gebunden.

UserForm-Dialoge gehören zu einem benutzerdefinierten Formular und beinhalten die zugehörigen Ereignisprozeduren.

Organisation von Modulen

- ▶ Module können an ganze Dateien oder an einzelne Dateielemente gebunden werden.
- ▶ Gruppieren Sie Prozeduren, die zusammen gehören, in jeweils eigene Module!
- ▶ Benennen Sie Module mit aussagekräftigen Namen (z.B. Verwaltung oder Dokumentstatistik) statt der automatisch erzeugten Namen Modul1, Modul2, etc. !
- ▶ Module können exportiert und importiert werden. Beim Exportieren werden sie in Textdateien mit der Dateiendung *.bas gespeichert.

Sub-Prozeduren

- ▶ Sub-Prozeduren bestehen aus einer Folge von Anweisungen (z.B. Zuweisungen von Variablen, Prozeduraufrufen, Verzweigungen, Schleifen, ...)
- ▶ Syntax von einfachen Sub-Prozeduren

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

- ▶ Syntax von Prozeduraufrufen (ohne Parameter)

```
Prozedurname  
Call Prozedurname
```


Prozedurnamen und andere Bezeichner

- ▶ Bezeichner sind frei wählbare Namen, mit denen Prozeduren, Variablen und Konstanten benannt werden.
- ▶ Sie dürfen nicht mit Schlüsselwörtern übereinstimmen.
- ▶ Sie müssen mit einem Buchstaben beginnen und können bis zu 255 Buchstaben, Zahlen oder Unterstriche enthalten, jedoch keine Punkte oder Leerzeichen.
- ▶ VBA ist nicht *case sensitive*, d.h. Groß- und Kleinschreibung werden nicht unterschieden.
- ▶ Bezeichner sollten eindeutig und aussagekräftig sein!

Beispiel: Meldungsfenster mit allgemeiner Information

```
'Information zu Datum und Anwendung
Sub AllgemeineInfo()
    MsgBox "Heute ist " & Date & " und Sie arbeiten mit " &
        Application.Name
End Sub
```

- ▶ Kommentare beginnen mit '.
- ▶ Mit & werden Zeichenketten (Strings) zusammengefügt (*Konkatenation*).
- ▶ `Date` liefert das Datum des heutigen Tages.
- ▶ Auf Eigenschaften von Objekten (wie z.B. `Application`) wird mit `.` zugegriffen.

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"
```

- ▶ Andere Symbole: vbCritical, vbQuestion, vbExclamation

Meldungsfenster mit Rückgabewert

- ▶ Über *Schaltflächen (Buttons)* kann der Benutzer die Anwendung interaktiv steuern.

```
Ergebniswert = MsgBox ("Text", Buttons + Symbole, "  
Titel")
```

- ▶ Mit Rückgabewert werden die Parameter in Klammern angegeben!
- ▶ Um Schaltflächen mit Symbolen zu kombinieren, werden diese mit + addiert.

Übersicht: 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

Die Verwendung der Konstanten ist aussagekräftiger als die der Werte:

```
MsgBox "Das ist ein Test", 0, "Das ist der Titel"
```

```
MsgBox "Das ist ein Test", vbOkOnly, "Das ist der Titel"
```

Übersicht: 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

Beispiel: Verwendung von Schaltflächen

```
'Abfrage von Essensgewohnheiten
Sub Essensgewohnheiten()
    Wahl = MsgBox("Waren Sie heute in der Mensa essen?",
        vbYesNo + vbQuestion, "Essensgewohnheiten")
    If Wahl = vbYes Then
        MsgBox "Ein Mensabesucher!"
    Else 'Wahl = vbNo
        MsgBox "Ein Hobbykoch!"
    End If
End Sub
```

Verzweigungen

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

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

```
If Temperatur >= -3 Then
    MsgBox "Gefrierschrank
           defekt"
End If
```

- ▶ Zweiseitige Auswahl

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

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


Verzweigungen

Mehrstufige Auswahl

```
If Ausdruck1 Then
    ...
ElseIf Ausdruck2 Then
    ...
ElseIf Ausdruck3 Then
    ...
Else
    ...
End If
```

```
If Temperatur < 0 Then
    MsgBox "Frost!"
ElseIf Temperatur < 20 Then
    MsgBox "
        Durchschnittstemperatur
    "
ElseIf Temperatur < 30 Then
    MsgBox "Sommerfeeling"
Else
    MsgBox "Hitzewelle!"
End If
```

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 Datentype
```

- ▶ Beispiele:

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

Deklaration und Sichtbarkeit von Variablen

- ▶ Variablen, die im Deklarationsteil zu Beginn eines deklariert werden, sind im gesamten Modul gültig.
- ▶ Variablen, die am Anfang einer Prozedur deklariert werden, sind nur innerhalb der Prozedur gültig.
- ▶ VBA erlaubt die implizite und explizite Deklaration von Variablen.

Implizite Deklaration

- ▶ Variablen müssen vor ihrer ersten Verwendung nicht deklariert werden.

Explizite Deklaration

- ▶ Variablen müssen vor ihrer ersten Verwendung deklariert werden.
- ▶ Angabe von `Option Explicit` zu Beginn des Moduls erforderlich.

Beispiel: Verwendung von Variablen

```
'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 Ende der Vorlesungszeit sind es
        noch " & DateDiff("d", Heute, Semesterende) & "
        Tage."
    MsgBox Ausgabe
End Sub
```

- ▶ `Date` ist der Datentyp für ein Datum
- ▶ `DateValue` berechnet für eine Datum im String-Format den Wert im Datumsformat
- ▶ `DateDiff` liefert die Differenz in Tagen ("d") für zwei Datumswerte.

Zusammenfassung

- ▶ Allgemeine Informationen zu VBA und Microsoft Office
- ▶ VBA-Entwicklungsumgebung
- ▶ Interaktion mit dem Benutzer über Meldungsfenster
- ▶ Programmieren in VBA
 - ▶ Module
 - ▶ Prozeduren
 - ▶ Anweisungen (einfache Prozeduraufrufe, Kontrollstrukturen: Verzweigungen, Zuweisungen)
 - ▶ Variablen

In der nächsten Vorlesung:

- ▶ Funktionen
- ▶ Datentypen
- ▶ Ausdrücke (Konstanten, Operatoren)
- ▶ Kontrollstrukturen: Fallauswahl und Schleifen

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

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

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 ErmittlerMonat1()  
    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
```