Einführung in die Excel-Makroprogrammierung

J. Abulawi

Themenübersicht

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

1. **Einführung – Was ist VBA?**
2. **Der VBA-Editor**
3. **Einfache Anweisungen**
4. **Variablen und Datentypen**
5. **Kontrollstrukturen**
6. **Objektorientierung**
7. **Beispiele...**

Diese Unterlagen sind nur für Studierende der HAW zum internen Gebrauch gedacht.

Die in diesen Unterlagen enthaltenen Informationen stammen zum Teil aus folgenden Quellen:

- Microsoft Visual Basic-Hilfe
- http://userpage.fu-berlin.de/~agnschum/download/VBA_Skript.pdf
- http://www.redmonds.de/Excel_1997_programmierung_kap.pdf
- http://www.ti5.tu-harburg.de/manual/vba5/httoc.htm
  -> Elektronische Fassung des Titels: VBA 5 in 21 Tagen,
  ISBN: 3-8272-2008-4

Nicht hier verwendete, aber trotzdem interessante Links:

http://www.reichelt-edv.de/
-> hier gibt es viele VBA-Programme als Download

http://www.roehrenbacher.at/erc/tt/tt_archiv_2.htm
-> hier gibt es verschiedene Tipps und Tricks zu VBA/Excel

1. Einführung

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

VBA = Visual Basic for Applications

- Spezielle Programmiersprache für Host-Anwendungen wie MS Office, CATIA …
- Befehlsatz basiert auf Visual Basic for Windows
- Erzeugbare Code-Strukturen:
  - Sub-Routinen
  - Funktionen

Historie:

BASIC = **B**eginner‘s **A**ll **P**urpose **S**ymbolic **I**nstruction **C**ode

- Entwicklung in den **frühen 60ern**
- einfach zu erlernende Programmiersprache
- sehr begrenzter Befehlssatz

Visual Basic für Windows

Weiterentwicklung (**1992**) als voll funktionale Programmiersparche

Parallelentwicklung:

Makrorekorder ohne Editiermöglichkeit für Officeprogramme

⇒ VBA Kombination von Makrorekorder und VB-Editor/-Befehlen

⇒ nur lauffähig mit entsprechender Host-Anwendung

1. Einführung

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

VBA-Makroprogrammierung

Wozu?

- Automatisierung von Routinevorgängen für den Eigenbedarf
- Programmierung von Anwendungen für Dritte

Wie?

- Aufzeichnung von Vorgängen in der Host-Anwendung
- Manuelle, strukturierte Programmierung
- Kombination von Aufzeichnung und manueller Programmierung

VBA = objekt- und ereignisorientierte Interpretersprache

Kein Compiler

Hauptprogramm = Host-Anwendung

Modulare Programmierung

Variablendeklaration optional

Schnelle Programmerstellung

Einfaches Austesten

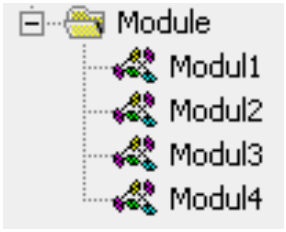
bei Fehlern:

Programmabbruch und Sprung in den Editor möglich

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

VBA – Eigenschaften

- objekt- und ereignisorientiert
- Interpretersprache (ohne Compiler)
- Hauptprogramm = Host-Anwendung
- Programmierung in Modulen →
- ein Befehl pro Zeile
- Zeilenende = Befehlsende



bei Fehlern:

Programmabbruch und Sprung in den Editor möglich

Interpreter:

= Simultandolmetscher

Das VBA-Programm wird nicht erst von einem Compiler in eine für das Betriebssystem verständliche Maschinensprache übersetzt und dann ausgeführt.

Stattdessen wird es Anweisung für Anweisung vom Interpreter in die Maschinensprache übersetzt und gleichzeitig ausgeführt

Modulare Programmierung:

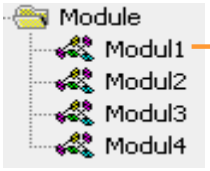
Bei der VBA-Makroprogrammierung entstehen keine eigenständigen, ausführbaren Programme.

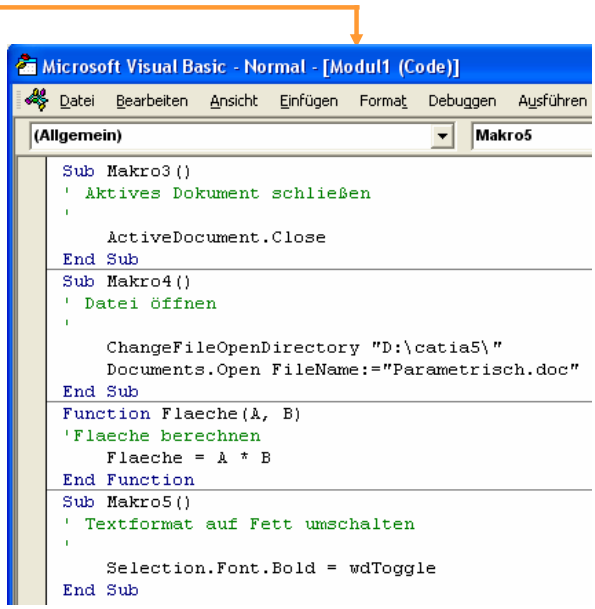
Die erzeugten Makros und Funktionen werden in Office in Modulen abgelegt, die Bestandteil des aktiven Dokuments sind.

1. Einführung

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Speicherung aller Makros in Modulen:





Funktionen, Prozeduren und Makros werden als Teil des Projektes in sogenannten Modulen gespeichert.

Bevor man mit dem Programmieren beginnt, muss zunächst ein neues Modul erzeugt werden.

(Option: Einfügen Modul in der Visual Basic Entwicklungsumgebung).

Beim Aufzeichnen eines Makros oder beim Erstellen über die Makroverwaltung wird automatisch ein Modul eingefügt, wenn noch keins vorhanden ist.

Grundsätzlicher Makroaufbau

Sub-Prozedur:

Anfang →
Kommentar →
Anweisung →
Ende →

```
Sub Makro4()
  ' Schließt Mappe 1
  Workbooks(1).Close
End Sub
```

Funktion:

```
Function Flaeche(A,B)
  ‘ Rechteckfläche berechnen
  Flaeche = A * B
End Function
```

Argumente

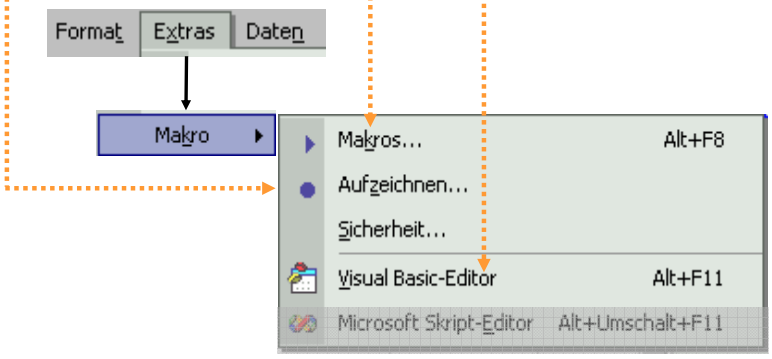
Rückgabewert = Funktionsname

2. Der VBA-Editor

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Wie wird ein Makro erzeugt?

- Manuelle Programmierung
- Makro-Rekorder



Makroaufzeichnung:

- nur Objekte der Host-Applikation (Excel, Word oder CATIA…)
- Protokollierung aller Objekteigenschaften (auch nicht geänderte)
- **keine**:
  - **Benutzerdialoge**
  - **Kontrollstrukturen** (Fallunterscheidung, Schleifen etc.)
  - **VB-Specials** (Fehlerhandling…)
- in Excel: **Umschaltung abs./rel. Bezüge** möglich
- **nur Prozeduren** „Sub ….. End Sub“

Manuell erstelltes Makro:

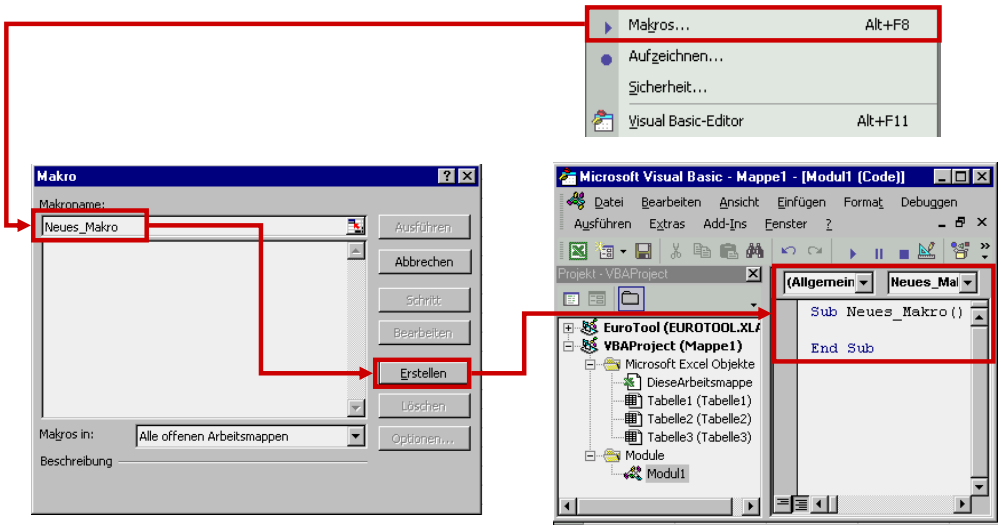
- Funktionen und Prozeduren erstellbar
- „echtes“ Computerprogramm
- auch nur lauffähig mit Host-Applikation
- volle VBA-Funktionalität

Microsoft Skript Editor:

HTML-Editor (nicht unser Thema…)

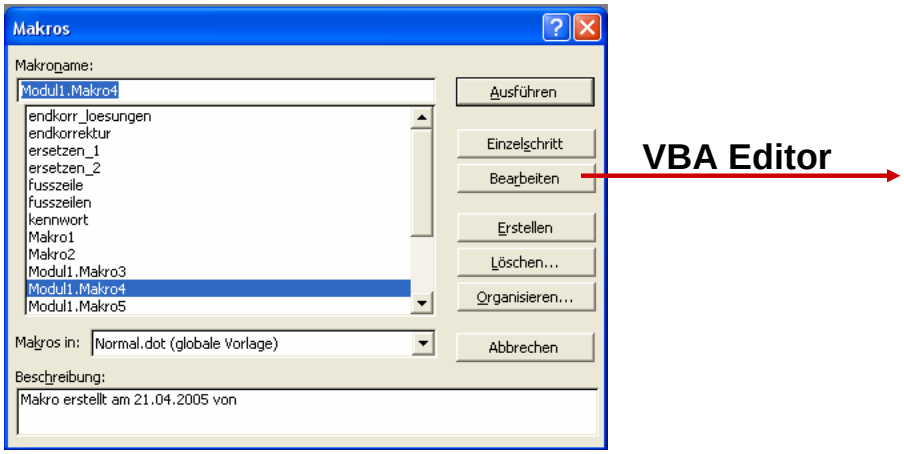
2. Der VBA-Editor

Ein neues Makro erstellen ...



Vorhandene Makros bearbeiten

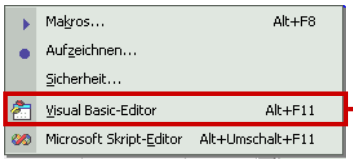
Makroverwaltung öffnen mit Extras -> Makro -> Makros...



2. Der VBA-Editor

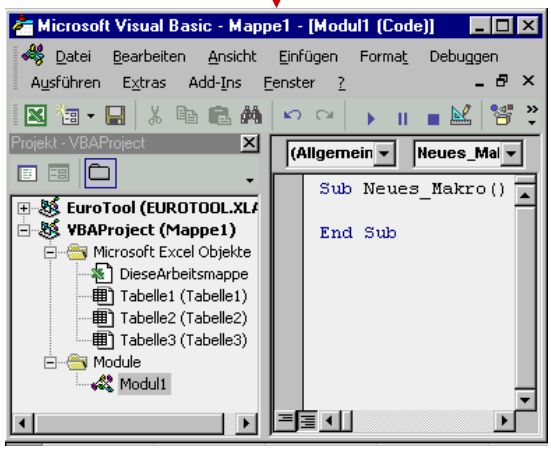
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Aufruf des Editors



Editor-Komponenten

- **Code-Fenster**
- **Projekt-Explorer**
- **Eigenschaftsfenster**
- **Programmierhilfen**
  - Objektkatalog
  - Debugger
  - Hilfefunktion



Operatoren

- Arithmetisch: **+ - * / ^ \ mod**
- Addition von Zeichenketten: **+ &**
- Vergleichsoperatoren: **= < > <= >= <>**
- Prüfung, ob zwei Objekte auf dasselbe Objekt verweisen: **is**
- Vergleich von Zeichenmustern: **like**
- Logische Operatoren: **and or not xor imp eqv**

Operatoren:

• Arithmetisch ^ * / + - \ (abgerundete Ganzzahldivision)
mod (Rest einer Ganzzahldivision)

• Zeichenketten + & ("15" & 3 ergibt "153")

• Vergleichsoperatoren = < > <= >= <>

• **is** *(Wenn zwei Objekte auf dasselbe Objekt verweisen, ist das Ergebnis True; andernfalls ist Ergebnis False.)*

• **like** zum Vergleich von Zeichenmuster: "abc" like "a*"

• Logische Operatoren **and or not xor imp** (Implikation) **eqv** (Äquivalenz)

3. Einfache Anweisungen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Zuweisungen

Wertübergabe **=**

Beispiele:
```
Wert = 293.5*2
Text = “Anfang” & “Ende”
```

Parameterübergabe in Prozeduren **:=**

Beispiele:
```
Formula1 := “$B$1”
FileName := ”Test.xls"
```

Zuweisung eines Objektverweises **Set**

Beispiele:
```
Set Auswahl = Selection
Set Dok1 = ActiveDocument
```

3. Einfache Anweisungen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Mitteilungen an den Benutzer

`Msgbox` *`Mitteilung,Typ,Titel`*

Beispiel 1:

```
MsgBox "Dies ist neu", vbInformation, "Mitteilungsmakro"
```

Das Argument ***Typ*** hat die folgenden Einstellungen:

Konstante	Wert	Beschreibung
vbOKOnly	0	Nur die Schaltfläche **OK**.
VbOKCancel	1	Schaltflächen **OK** und **Abbrechen**.
VbAbortRetryIgnore	2	Schaltflächen **Abbruch**, **Wiederholen, Ignorieren**.
VbYesNoCancel	3	Schaltflächen **Ja**, **Nein** und **Abbrechen**.
VbYesNo	4	Schaltflächen **Ja** und **Nein**.
VbRetryCancel	5	Schaltflächen **Wiederholen** und **Abbrechen**.
VbCritical	16	Meldung mit Stop-Symbol.
VbQuestion	32	Meldung mit Fragezeichen-Symbol.
VbExclamation	48	Meldung mit Ausrufezeichen-Symbol.
VbInformation	64	Meldung mit Info-Symbol.

3. Einfache Anweisungen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Mitteilungen an den Benutzer

`Msgbox` *`Mitteilung,Typ,Titel`*

Beispiel 2:

```
MsgBox ”Vorsicht!", vbExclamation, ”Warnung"
```

Das Argument ***Typ*** ergibt sich:

Die erste Gruppe von Werten (0 - 5)
beschreibt die Anzahl und den Typ der im Dialogfeld angezeigten Schaltflächen.

Die zweite Gruppe (16, 32, 48, 64) beschreibt die Symbolart.

Die dritte Gruppe (0, 256, 512) legt die Standardschaltfläche fest.

Die vierte Gruppe (0, 4096) legt fest, in welcher Form das Dialogfeld gebunden ist.

Verwenden Sie beim Addieren der Zahlen zu einem Gesamtwert für das Argument ***Typ*** nur eine Zahl aus jeder Gruppe.

Anmerkung Diese Konstanten sind durch VBA festgelegt. Daher können die Namen an einer beliebigen Stelle im Code anstelle der tatsächlichen Werte verwendet werden.

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Kommunikation mit den Benutzer

```
Variable = Msgbox(Mitteilung,Wert,Titel)
```

Beispiel 1:

```
Antwort = MsgBox ("Gleichung unlösbar", _
  vbOKCancel, "Abfrage")
```

Antwort: **1** **2**

Rückgabewerte der Msgbox

Konstante	Wert	Beschreibung
vbOK	1	**OK**
vbCancel	2	**Abbrechen**
vbAbort	3	**Abbruch**
vbRetry	4	**Wiederholen**
vbIgnore	5	**Ignorieren**
vbYes	6	**Ja**
vbNo	7	**Nein**

3. Einfache Anweisungen

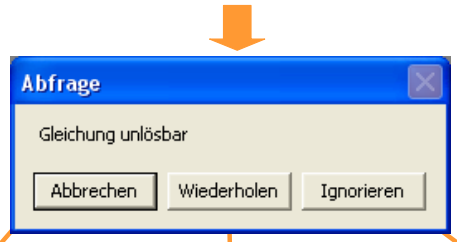
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Kommunikation mit den Benutzer

```
Variable = Msgbox(Mitteilung,Wert,Titel)
```

Beispiel 2:

```
Antwort = MsgBox ("Gleichung unlösbar", _
  vbAbortRetryIgnore, "Abfrage")
```



Antwort: **3** **4** **5**

Syntax

MsgBox(*prompt*[, *buttons*] [, *title*] [, *helpfile*, *context*])

Wenn sowohl ***helpfile*** als auch ***context*** angeben werden, kann der Benutzer F1 (Windows) oder HILFE (Macintosh) drücken, um das Hilfethema für **context** anzuzeigen. Einige Host-Anwendungen, zum Beispiel Microsoft Excel, fügen dem Dialogfeld automatisch die Schaltfläche **Hilfe** hinzu.

Wenn im Dialogfeld die Schaltfläche **Abbrechen** angezeigt wird, hat das Drücken von ESC dieselbe Wirkung wie das Klicken auf **Abbrechen**.

Wird im Dialogfeld die Schaltfläche **Hilfe** angezeigt, wird für das Dialogfeld eine kontextbezogene Hilfe zur Verfügung gestellt. Ein Wert wird aber nur zurückgegeben, wenn auf eine der anderen Schaltflächen geklickt wird.

Anmerkung Wenn Sie außer dem ersten benannten Argument weitere Argumente angeben möchten, müssen Sie **MsgBox** in einem Ausdruck verwenden. Wenn Sie einige Argumente mit einer bestimmten Position nicht angeben möchten, müssen Sie dennoch das entsprechende Komma als Trennzeichen angeben.

3. Einfache Anweisungen

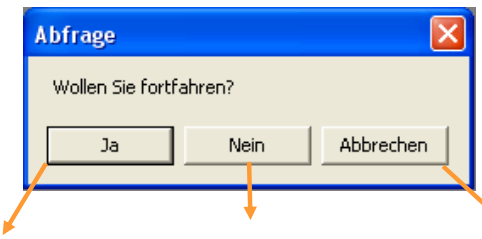
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Kommunikation mit den Benutzer

```
Variable = Msgbox(Mitteilung,Wert,Titel)
```

Beispiel 3:

```
Antwort = MsgBox ("Wollen Sie fortfahren?", _
   vbYesNoCancel, "Abfrage")
```



Antwort: **6** **7** **3**

MsgBox-Funktion (Beispiel)

In diesem Beispiel wird die **MsgBox**-Funktion verwendet, um eine Meldung zu einem schwerwiegenden Fehler in einem Dialogfeld mit den Schaltflächen **Ja** und **Nein** anzuzeigen. **Nein** ist dabei die Voreinstellung. Der von **MsgBox** gelieferte Wert hängt von der Schaltfläche ab, die der Benutzer wählt. Im Rahmen dieses Beispiels wird angenommen, daß `DEMO.HLP` eine Hilfedatei ist, in der ein Thema mit der Kontextnummer 1000 vorhanden ist.

```
Dim Mldg, Stil, Titel, Hilfe, Ktxt, Antwort, Text1
Mldg = "Möchten Sie fortfahren ?"  ‘ Meldung definieren.
Stil = vbYesNo + vbCritical + vbDefaultButton2
                                                    '

Schaltflächen definieren
Titel = "MsgBox-Demonstration”      ' Titel definieren
Hilfe = "DEMO.HLP"                  ' Hilfedatei definieren
Ktxt = 1000                         ' Kontext für Thema def.
Antwort = MsgBox(Mldg, Stil, Titel, Hilfe, Ktxt)
                                        ' Meldung anzeigen.
If Antwort = vbYes Then             ' Benutzer hat "Ja” gewählt
    Text1 = "Ja"                    ' Operation ausführen.
Else                                ' Benutzerwahl "Nein"
    Text1 = "Nein"                  ' Operation ausführen.
End If
```

3. Einfache Anweisungen

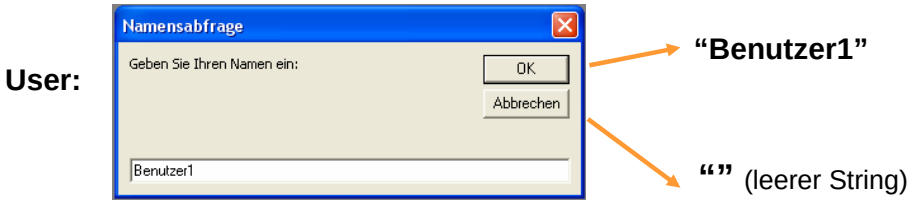
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Benutzereingaben

```
Textvariable = InputBox(Mitteilung,Titel,Eingabe)
```

Beispiel:

```
User = InputBox ("Geben Sie Ihren Namen ein:", _
   "Namensabfrage",User)
```



HAW / Fb F+F / J. Abulawi SS 2005 Excel-Makros 19/38

Zeigt eine **Eingabeaufforderung** in einem Dialogfeld an, wartet auf die Eingabe eines Textes oder auf das Klicken auf eine Schaltfläche und **gibt einen Wert vom Typ String** zurück, der den Inhalt des Textfeldes angibt.

Syntax

InputBox(*prompt*[, *title*] [, *default*] [, *xpos*] [, *ypos*] [, *helpfile*, *context*])

prompt Erscheint als Meldung im Dialogfeld. Die Maximallänge von ***prompt*** ist - je nach Breite der verwendeten Zeichen - etwa 1024 Zeichen. Wenn ***prompt*** aus mehreren Zeilen besteht, müssen Sie die Zeilen mit einem Wagenrücklaufzeichen (**Chr(**13**)**), einem Zeilenvorschubzeichen (**Chr(**10**)**) oder einer Kombination aus Wagenrücklaufzeichen und Zeilenvorschubzeichen (**Chr(**13**)** & **Chr(**10**)**) trennen.

title Optional. Eine Zeichenfolge, die in der Titelleiste des Dialogfeldes angezeigt wird. Wird ***title*** nicht angegeben, erscheint der Anwendungsname in der Titelleiste.

default Optional. Eine Zeichenfolge, die als Voreinstellung im Textfeld angezeigt wird, wenn der Benutzer keine Eingabe vorgenommen hat. Wird ***default*** nicht angeben, erscheint das Textfeld leer.

xpos Optional. = den horizontaler Abstand (in Twips) des linken Rands des Dialogfeldes vom linken Rand des Bildschirms festlegt. Default: horizontal zentriert.

ypos Optional. = vertikaler Abstand (in Twips) des oberen Rands des Dialogfeldes vom oberen Rand des Bildschirms festlegt. Default: ein Drittel unterhalb des oberen Bildschirmrands (bezogen auf die gesamte Bildschirmhöhe) angezeigt.

3. Einfache Anweisungen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

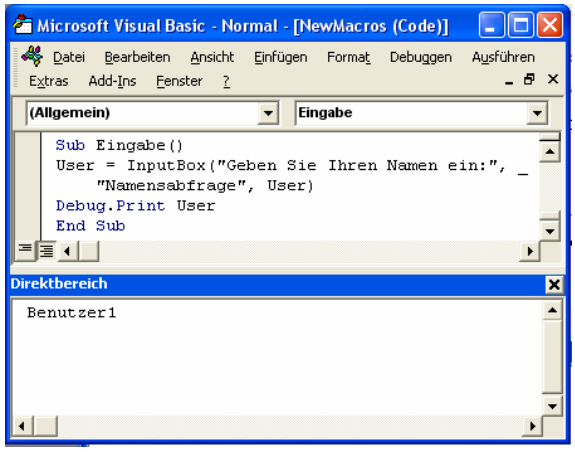
Ausgaben im Direktfenster des VBA-Editors

`Debug.Print` ***`Variable`***

Beispiel:

```
Debug.Print User
```

Ansicht
-> **Direktfenster**
(STRG+G)



Shortcuts für das Direktfenster

EINGABETASTE	Ausführen einer markierten Code-Zeile.
STRG+C	Kopieren des markierten Textes in die Zwischenablage
STRG+V	Einfügen des Inhalts der Zwischenablage an der Einfügemarke.
STRG+X	Ausschneiden des markierten Textes in die Zwischenablage.
STRG+L	Anzeigen des Dialogfelds **Aufrufeliste** (nur im Haltemodus).
F5	Fortsetzen der Ausführung einer Anwendung.
F8	Zeilenweises Ausführen von Code (Einzelschritt).
UMSCHALT+F8	Prozedurweises Ausführen von Code (Prozedurschritt).
ENTF oder RÜCKTASTE	Löschen des markierten Textes, ohne ihn in die Zwischenablage zu kopieren.
F2	Anzeigen des Objektkatalogs.
STRG+EINGABETASTE	Einfügen eines Wagenrücklaufzeichens.
STRG+POS1	Positionieren des Cursors am Anfang des Direktfensters.
STRG+ENDE	Positionieren des Cursors am Ende des Direktfensters.
UMSCHALT+F10	Anzeigen des Kontextmenüs.
ALT+F5	Ausführen des Fehlerbehandlungs-Codes oder Zurückgeben des Fehlers an die aufrufende Prozedur.
ALT+F8	Springt in den Fehlerbehandlungs-Code oder gibt den Fehler an die aufrufende Prozedur zurück.

4. Variablen und Datentypen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Variablen

- **ein- oder mehrdimensional**
- **Deklaration: optional**
- **Variant = Standardtyp für nicht deklarierte Variablen**

Befehl zum Erzwingen der Variablendeklaration:

```
Option Explicit
```

Syntax der Variablendeklaration:

```
Dim Variable As Datentyp
```


Variant:

- kann Zeichenfolgen, Datums- Zeit- boolesche oder numerische Werte enthalten und automatisch umwandeln.
- benötigt 16 Bytes Speicherplatz pro Variable
- langsamer Zugriff

Gefahr:

Tippfehler bei Variablennamen werden vom System nicht erkannt

Sie werden als neue Variablen betrachtet

Die Suche nach solchen Fehlern ist sehr aufwendig

Deshalb:

am Modulanfang immer Variablendeklaration mit „Option Explicit“ erzwingen

4. Variablen und Datentypen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Numerische Datentypen

- **Byte**
- **Integer**
- **Long**

} **ganzzahlig**

- **Single**
- **Double**
- **Currency**

Sonstige Datentypen

- **Boolean**
- **String**
- **Date**
- **Object**
- **Variant**

Datentypen:

Byte	**0 ...**	**255**	
Integer	**- 32.768 ...**	**32.767**	
Long	**- 2.147.483.648 ...**	**2.147.483.647**	**+/- 2 Mio**
Single	**-3,4E38 ..**	**3,5E38**	**8 Stellen**
Double	**-1,8E308 ...**	**1,8E308**	**16 Stellen**
Currency	**Festkomma: 15 Stellen vor + 4 Stellen nach Komma**		
Boolean	**True - False**		
String	**< 65.400 Zeichen**		
Date	**Datum und Uhrzeit mit versch. Formatierungen**		
Object	**Adressen, die auf Objekte in der Anwendung verweisen können**		

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Gültigkeitsbereiche von Variablen

Deklaration im Modulkopf	Bekanntheit der Variable	Lebensdauer
`Public Var As Byte`	in allen Modulen	bis zum
`Dim Var As Byte`	nur im aktuellen	Neustart des
`Private Var As Byte`	Modul	Moduls
Deklaration in Prozedur/Funktion		
`Dim Var As Byte`	nur in der aktuellen	bis zum Ende der Prozedur/Funktion
`Static Var As Byte`	Prozedur/Funktion	bis zum Neustart des Moduls

Variablen, die nur in einer Prozedur oder Funktion bekannt sein sollen, deklariert man innerhalb derselben mit dem Schlüsselwort DIM.

Globale Variablen, die in allen Prozeduren/Funktionen eines Moduls gültig sein sollen, **deklariert man im Deklarationsteil eines Moduls** (Abschnitt vor der 1. Prozedur/Funktion).

Modulübergreifende Variablen, die darüber hinaus in allen Modulen Verwendung finden, muss man mit dem Schlüsselwort **Public** deklarieren.

Nicht deklarierte Variablen sind standardmäßig **nur lokal gültig**.

Auf Modulebene deklarierte Variablen erhalten ihren Wert, bis das Modul zurückgesetzt bzw. neu gestartet wird.
Option: Ausführen/Zurücksetzen

In der Funktion/Prozedur deklarierte Variablen entsprechen nicht den auf Modulebene deklarierten globalen Variablen (auch bei Namensgleichheit).

Sie behalten ihren Wert nur bis zur Beendigung der Prozedur bzw. Funktion, wenn sie nicht STATIC sind - dann werden sie erst wieder beim Zurücksetzen des Moduls neu initialisiert

Auswahlbefehl - vollständige Alternative

```
If Bedingung1_erfuellt Then
      Anweisung1
  ElseIf Bedingung1_erfuellt Then
      Anweisung2
  Else
      Anweisung3
End If
```


If...Then...Else-Anweisung

Syntax

If *Bedingung* **Then** [*Anweisungen*] [**Else** *elseAnweisungen*]

Alternativ können Sie die Block-Syntax verwenden:

If *Bedingung* **Then**
    [*Anweisungen*]

[**ElseIf** *Bedingung-n* **Then**
    [*elseifAnweisungen*] . . .

[**Else**
    [*elseAnweisungen*]]

End If

Auswahlbefehl - Fallauswahl

```
Select Case Monat
  Case 1, 3, 5, 7,8,10,12
      ZahlDerTage =31
  Case 2, 4, 6, 9, 11
      ZahlDerTage =30
  Else
      MsgBox "falsche Monatsangabe"
End Select
```


Führt Anweisungen aus, abhängig vom Wert eines Ausdrucks.

Select Case Testausdruck

[Case Ausdrucksliste-n

[Anweisungen-n]] ...

[Case Else

[elseAnw]]

End Select

Testausdruck beliebiger numerischer oder Zeichenfolgenausdruck.

Ausdrucksliste-n Erforderlich, wenn der Case-Abschnitt verwendet wird. Eine durch Kommas getrennte Liste in einer oder mehreren der folgenden Formen: Ausdruck, Ausdruck To Ausdruck, Is Vergleichsoperator Ausdruck. Das Schlüsselwort To gibt einen Bereich von Werten an. Bei diesem Schlüsselwort muß der kleinere Wert immer links von To stehen. Verwenden Sie das Schlüsselwort Is in Kombination mit Vergleichsoperatoren (außer Is und Like), um einen Bereich von Werten anzugeben. Wenn Sie das Schlüsselwort Is nicht angeben, wird es automatisch eingefügt.

Anweisungen-n Optional. Eine oder mehrere Anweisungen, die ausgeführt werden, wenn Testausdruck mit irgendeinem Teil in Ausdrucksliste-n übereinstimmt.

elseAnw Optional. Eine oder mehrere Anweisungen, die ausgeführt werden, wenn Testausdruck mit keinem der Ausdrücke im Case-Abschnitt übereinstimmt.

5. Kontrollstrukturen

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Indizierte Schleife

```
For Index = 1 To 10 Step 1
   Anweisungen
Next Index
```

Anweisungen werden hier genau 10x durchlaufen

Die Angabe der Schrittweite `Step` ist optional

Standard-Schrittweite = 1

For *Zähler* **=** *Anfang* **To** *Ende* [**Step** *Schritt*]
[*Anweisungen*]
[**Exit For**]
[*Anweisungen*]
Next [*Zähler*]

Das Argument *Schritt* ist entweder positiv oder negativ. Sein Wert legt die Schleifenausführung folgendermaßen fest:

Wert	Schleifenausführung, wenn
Positiv oder 0	*Zähler* <= *Ende*
Negativ	*Zähler* >= *Ende*

Innerhalb einer Schleife kann eine beliebige Anzahl von **Exit For**-Anweisungen an beliebiger Stelle als alternative Möglichkeit zum Verlassen der Schleife verwendet werden. **Exit For** wird oft in Zusammenhang mit der Auswertung einer Bedingung (zum Beispiel **If...Then**) eingesetzt und überträgt die Steuerung an die unmittelbar auf **Next** folgende Anweisung.

Sie können **For...Next**-Schleifen verschachteln, indem Sie eine **For...Next**-Schleife innerhalb einer anderen verwenden. Das Argument *Zähler* muß für jede Schleife einen eindeutigen Variablennamen erhalten.

Zyklus über alle Elemente einer Auflistung

```
For Each Objekt In Auflistung
   Anweisungen
Next
```

Anweisungen werden für jedes Objekt der Auflistung durchlaufen

For Each *Element* **In** *Gruppe*
[*Anweisungen*]
[**Exit For**]
[*Anweisungen*]
Next [*Element*]

Das Innere des **For…Each**-Blocks wird ausgeführt, wenn sich mindestens ein Element in *Gruppe* befindet. Nachdem mit der Ausführung der Schleife begonnen wurde, werden alle Anweisungen in der Schleife für das erste Element in *Gruppe* ausgeführt. Enthält *Gruppe* mehrere Elemente, so werden die Anweisungen in der Schleife für jedes Element wiederholt. Wenn alle Elemente in *Gruppe* abgearbeitet, verläßt das Programm die Schleife und setzt die Ausführung mit der Anweisung fort, die auf die **Next**-Anweisung folgt.

Wenn Sie *Element* in einer **Next**-Anweisung nicht angeben, wird die Ausführung trotzdem so fortgesetzt, als ob *Element* enthalten wäre. Eine **Next**-Anweisung vor der zugehörigen **For**-Anweisung führt zu einem Fehler.

Sie können die **For...Each...Next**-Anweisung nicht mit einem Datenfeld eines benutzerdefinierten Typs verwenden, da eine Variable vom Typ **Variant** keinen benutzerdefinierten Typ beinhalten darf

Pre-Check Schleifen

= Überprüfung VOR Durchführung der Anweisungen

```
Do While Zahl > 10
Zahl = Zahl - 1
Wert = Wert + 1
Loop
```

```
Do Until Zahl = 10
Zahl = Zahl + 1
Wert = Wert + 1
Loop
```

Wiederholung der Anweisungen,

solange die Bedingung WAHR ist

bis die Bedingung WAHR ist

Do...Loop-Anweisungen dienen dazu, einen Block von Anweisungen eine unbestimmte Anzahl von Wiederholungen ausführen zu lassen.

Die Anweisungen werden wiederholt, solange eine Bedingung dem Wert **True** entspricht oder bis eine Bedingung dem Wert **True** entspricht.

Sie können eine **Do...Loop**-Anweisung unter Verwendung der **Exit Do**-Anweisung verlassen. Verwenden Sie z.B. die **Exit Do**-Anweisung im **Do**-Anweisungsblock einer **If...Then...Else**-Anweisung oder einer **Select Case**-Anweisung, wenn Sie eine Endlosschleife verlassen möchten. Entspricht die Bedingung dem Wert **False**, so wird die Schleife wie üblich ausgeführt.

Anmerkung

Drücken Sie **ESC** oder **STRG+PAUSE**, um eine Endlosschleife zu beenden.

Post-Check Schleifen

= Überprüfung NACH Durchführung der Anweisungen

```
Do
Zahl = Zahl - 1
Wert = Wert + 1
Loop While Zahl > 10
```

```
Do
Zahl = Zahl + 1
Wert = Wert + 1
Loop Until Zahl = 10
```

Wiederholung der Anweisungen,

solange die Bedingung WAHR ist

bis die Bedingung WAHR ist

Prüfung der Bedingung, nachdem die Schleife mindestens einmal durchlaufen wurde

Was bedeutet objektorientiert ?

Objekt = Element, das mittels Programmcode manipuliert werden kann

- **hat bestimmte Eigenschaften und Methoden**
- **kann bestimmte Ereignisse auslösen**

Eigenschaft = Merkmal eines Objekts

Methode = Anweisung zur Erzeugung oder Veränderung eines Objekts

Klasse = Gruppe von Objekten, die gleiche Eigenschaften und Methoden haben können

Objekthierarchie = hierarchische Struktur der Objekte einer Applikation

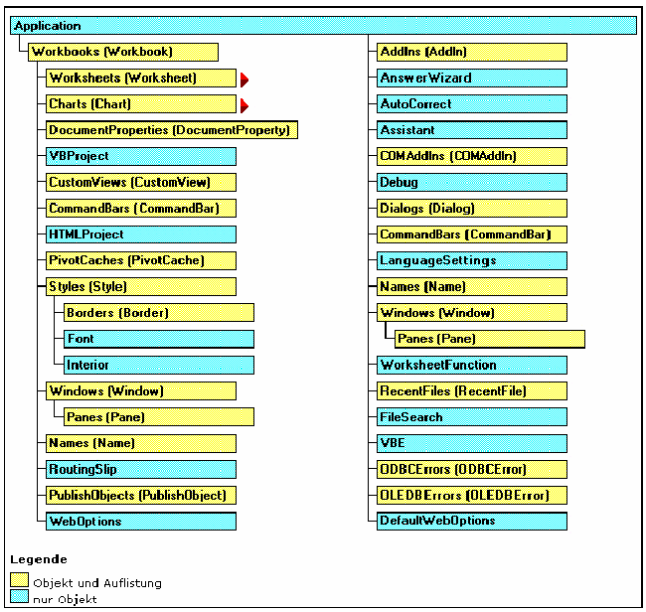
Objektorientierte Prinzipien

1. Bildung von **Klassen von Objekten** („Objekttypen“)
2. **Kapselung** (encapsulation): Objekte haben Eigenschaften, die nur mit den dafür vorgesehenen Methoden verändert werden können.
3. **Vererbung** (inheritance): Bildung von Klassenhierarchien, Subklassen erben Eigenschaften und Verhalten ihrer Oberklassen und können erweiterte Funktionalität bekommen (Spezialisierung)
4. **Polymorphismus**: Subklassen können geerbte Funktionen überschreiben (method overloading)

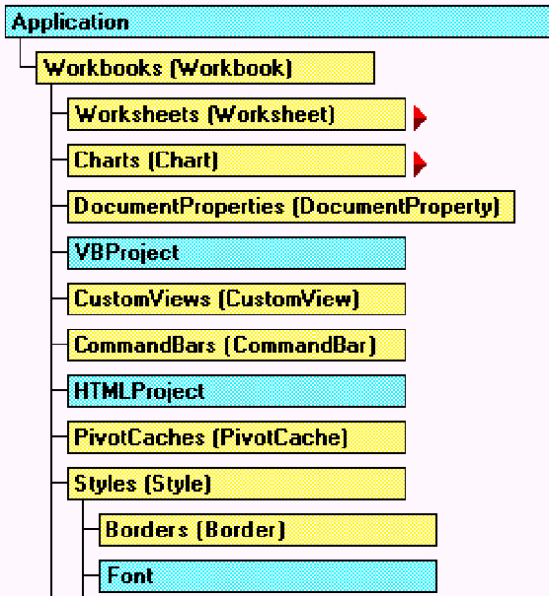
Verwendung der Objekte der VBA-Anwendung

- VBA stellt die Objekte der jeweiligen Anwendung zur Verfügung.
- Die Objekte der Anwendung sind in einer Objekthierarchie gegliedert.
- Um auf ein Objekt zuzugreifen, ist die Kenntnis der Position eines Objektes in der Hierarchie notwendig.

Excel-Objekthierarchie



Excel-Objekthierarchie



"Markenzeichen" objektorientierter Programme

Objektname.Methode

`Sheets.Add`

"Markenzeichen" objektorientierter Programme

Objektname.Methode **Argumente der Methode**

```
ActiveWorkbook.SaveAs Filename:="D:\Mappe2.xls"
```

"Markenzeichen" objektorientierter Programme

Objektname.Methode **Argumente der Methode**

```
ActiveWorkbook.SaveAs Filename:="D:\Mappe2.xls"
```

Objektname.Objektname.Eigenschaft = Zuweisung

```
Selection.EntireColumn.Hidden = True
```

7. Beispiel

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Excel - Prozedur:

In einer Excel-Tabelle werden alle Zellen orange hinterlegt,
deren Wert kleiner als 5 ist.

```
Sub MarkAlleUnterSchwellwert()
AnzZeilen = ActiveSheet.UsedRange.Rows.Count
AnzSpalten = ActiveSheet.UsedRange.Columns.Count
For spalte = 1 To AnzSpalten
    For zeile = 1 To AnzZeilen
       If Cells(zeile, spalte) < 5 Then
          Cells(zeile, spalte).Interior.ColorIndex = 40
       End If
    Next
Next
End Sub
```

Noch Fragen?

Vielen Dank für Ihre Aufmerksamkeit!