

---

# Anwendungsentwicklung

Eine Einführung unter VBA und MS Excel

Interaktive Folien zur *Programmierung*

## Deklarativ Anwendungen entwickeln

 [Tabellenkalkulation](#)

 [Elementare Was-Wenn-Analyse](#)

 [Lineare Optimierung](#)

## Prozedural Anwendungen entwickeln

⇒ [Modularisierung](#)

⇒ [Datentypen und Ablaufstrukturen](#)

 [HEXAGON](#), [WORT](#)

⇒ [Benutzerschnittstelle](#)

 [HEXAGONDIALOG](#)

⇒ [Fallbeispiel](#)

 [GEWINNVERTEILUNG](#)

⇒ [Datenfeld](#)

 [DATENFELD](#), [BINÄRSUCHE](#)

 [Algorithmen und Datenstrukturen](#)

## Datenbankanwendungen entwickeln

 [Dateiverwaltung](#)

 [Datenentwurf](#)

 [Datenbankverwaltung](#)

 [Anwendungsentwicklung](#)

---

# Organisation

---

## Veranstaltungen

- Vorlesung Montag (und selten Donnerstag)
- Übungen verteilt auf die ganze Woche

## 3. Stock WWZ

- Sprechstunden
- Laborbenutzung
- Drucken
- Internet
- Tutoren

## Merkblätter

- Grundstudium
- Hauptstudium
- Kreditpunkte

## Verkauf

- Unterrichtsmaterial und Druckerkärtchen im Sekretariat
- Bücher bei Karger Libri (Hörerschein im Sekretariat)

## Web Site

<http://www.wwz.unibas.ch/wi/>

# Stufen der Anwendungsentwicklung

Vorlagen (engl. templates) anpassen



Eigene Tabellenblätter entwickeln



Tasten- und Mausaktionen als  
Makroprogramme → aufzeichnen



Makroprogramme verbessern



Eigene Programme entwickeln

Programm := strukturierte Menge von Anweisungen an einen Computer

## Ausnahmen ...

- [Hannes Keller](#) - Tiefseetaucher, Mathematiker und Paradiesvogel aus Zürich - verkaufte eine halbe Million *Switch* an Vobis
- [Team Brendel](#) aus Basel erzielte - vor allem mit ihrer Adressverwaltung - mehrere Millionen Franken Umsatz pro Jahr

## ... und die Regel

- Schweiz als Software-Importland (aber als attraktiver Markt)
- Druck der grossen USA-Anbieter

## Gründe

- Kauf des Bekannten (nicht unbedingt des Besten)
- Kleines Marketingpotential
- Fehlende kritische Grösse
- Fehlende Risikofinanzierung
- Ausbildung?
- Sprache?
- ...

# Deklarativ und prozedural entwickeln

**Deklarativ Anwendungen entwickeln** heisst ...

das **Was** spezifizieren und das **Wie** der **Software** überlassen!

- **Formeln** mit vordefinierten Funktionen erstellen
- **Tabellenblätter** erstellen

**Prozedural Programme entwickeln** heisst ...

das **Wie** **selbst** entwerfen und implementieren!

- **Programme** erstellen

	<i>schnell</i>	<i>einfach</i>	<i>flexibel</i>	<i>umfassend</i>
<b>Deklarativ</b> entwickeln	.....	.....	.....	.....
<b>Prozedural</b> entwickeln	.....	.....	.....	.....

## Programmmentwicklung

⇒ entwurfssprachlich an einem Beispiel

- mit VBA unter MS Excel

## Syntax und Semantik von VBA

- **Operator** wie +
- **Datentyp** wie Integer
- **Objekt** wie Tabellenblatt
- **Konstante** wie PI
- **Variable** wie Umsatz
- **Anweisung** wie MsgBox  
(MsgBox gibt eine “Message” in einer “Box” aus)

Unsere VBA-Erfahrungen lassen sich  
auf das ganze MS Office übertragen

# Vorlagen anpassen

Anpassen (engl. customizing)

- eigene oder fremde **Vorlagen** (engl. templates) anpassen
- **Makros** aufzeichnen und in VBA anpassen →

## Vorlagen anpassen

### Neue Microsoft Excel-Vorlagen

**Rechnung1**

**Private und geschäftliche Vorlagen**

**KREDITDATEN**

**HAUSHALTSPLANER**

**Vesuvio Kaffee**  
Alte Straße 00  
00000 Köln  
(0000) 00000 Fax (0000) 00000000

Laufende Nr. 1003

Anpassen...

**Rechnung**

Kunde Name R. Suyama 19.10.95

**Anpassen von Vorlagen und Erhalten von Tips**

Anzahl	Be		TOTAL
2	1 Pf. Spezialröstung	11,00 DM	22,00 DM
5	Logo-Kaffeetassen	5,00 DM	25,00 DM
	Zwischensumme		47,00 DM
	Versandkosten		7,00 DM
	Steuern MwSt.		3,85 DM
	<b>SUMME</b>		<b>57,85 DM</b>

**Verknüpfen von Zellen mit einer Datenbank mit Hilfe des Vorlagen-Assistenten**

Vorlagen in anderen Teilen vom MS Office?



---

# Makros automatisieren Routineaufgaben

---

## Zweck

Unterstreiche den Text der laufenden Zeile

## Beispiel

Dies ist ein Beispiel

---

**Makro** := Automatisierung einer Aufgabe durch Zusammenfassung mehrerer **Aktionen** unter einem Namen

## Skizze eines Makroprogramms

**Macro** Unterstreiche\_die\_laufende\_Zeile  
    Springe an den Anfang der nächsten Zeile  
    **For** Zeilenlänge **Times**  
        Schreibe “-”  
    Springe an den Anfang der nächsten Zeile  
**End**



Man erzählt sich, dass alles mit einem Programmfehler begann ...

## Computer“defekt” macht 18jährige zu Frührentnern

“47 Jahre vor Erreichung des Rentenalters haben jetzt alle volljährig gewordenen Bürger Neuköllns die Broschüre “Leben in unserer Mitte - Vorbereitung aufs Rentenalter” vom Bezirksamt zugesandt bekommen. Normalerweise bekommen die Broschüre Personen, die vor dem Eintritt in das Rentenalter stehen, während den volljährig gewordenen ein Gruss des Bürgermeisters zugeht. Wie der Bezirk mitteilte, hat der Computer, in dem alle Namen gespeichert sind, wegen eines **Defekts** die beiden Namenslisten zusammengefasst.”

## Weitere Flops ...

- Ein Programm verrechnete sich bei der Schätzung des Steueraufkommens einer Stadt in den USA. Dies führte zu einer zu niedrigen Ansetzung der Steuerrate und so zu einem hohen Defizit in der Staatskasse.
- Die Versenkung des britischen Zerstörers “Sheffield” im Falklandkrieg soll durch ein fehlerhaftes Programm zur Steuerung von Abwehrraketen verursacht worden sein.

## Einige Gründe ...

- Zu grosse **Programme**

Die Programme zur Steuerung des Jungfernflugs des amerikanischen Space Shuttle umfassten eine halbe Million Zeilen.

- Zu grosse **Entwicklungsgruppen**
- Unsorgfältige **Tests**

# Modularisierung am Beispiel *Bibliothek*

**Modul** := entwurfs- oder programmiersprachlicher Baustein, der Aufgaben zusammenfasst, die eng ..... sind

## Modulhierarchie

### **Bibliotheksverwaltung**

Ausleihverkehr buchen

Stammdaten verwalten

Dokumente verwalten

Benutzer verwalten

Statistiken erstellen

Dokumentestatistiken

...

**Monatliche Ausleihstatistik →**

...

Benutzerstatistiken

---

# Entwicklungsrisiken - Die 12 Murphy-Gesetze

---

- *Wenn etwas schiefgehen kann, geht es bestimmt schief.*
- *Nichts ist so einfach, wie es aussieht.*
- *Alles wird teurer als zunächst geplant.*
- *Jede Arbeit erfordert mehr Zeit, als man hat.*
- *Es ist ein Naturgesetz, dass nichts jemals richtig klappt.*
- *Bevor man eine Sache anfangen kann, muss immer erst etwas getan werden.*
- *Alles lässt sich solange verbessern, bis es endlich zusammenbricht.*
- *Jede neue Erklärung schafft neue Verwirrung.*
- *Wer anderer Meinung ist, kann nicht logisch denken.*
- *Je wichtiger eine Entscheidung, desto grösser die Kommission.*
- *Mit zunehmender Dringlichkeit einer Entscheidung sinkt die Zahl der dafür Zuständigen.*
- *Sie können manche Leute immer und alle Leute manchmal zum Narren halten - und das genügt.*

---

# Phasen der Programmentwicklung

---

## I Spezifikation

Präzise Aufgabenstellung

## II Modularisierung

Aufteilung des Projekts in möglichst unabhängige Einheiten mit klarer Zielsetzung (Module)

## III Entwurfs sprachliche Algorithmisierung

Schrittweise Verfeinerung der Modul-Einrückungsliste in Verarbeitungsrezepte (Algorithmen)

## IV Programmiersprachliche Übersetzung

Übersetzung der Algorithmen in Programme, die von einem Computer ausgeführt werden können

## V Test

Test der einzelnen Module und ihres Zusammenwirkens

# I Grobspezifikation

Programmieren Sie ein Modul, das eine Ausleihestatistik erstellt und auf dem Bildschirm als Histogramm (..... -diagramm) ausgibt.

## Ungenau Spezifikation

AUSLEIHESTATISTIK FÜR DEN MONAT F E B R U A R

Fachgebiet	Ausgeliehene Titel
BWL	*****
VWL	*****
RECHT	*****
SOZ./PSYCH.	*****
ÜBRIGE	*****

Aufgabe : Offene Fragen?



Benutzereingaben?  
Ausgabeberechnung?

---

# I Feinspezifikation

---

## Zweck

Histogramm der Ausleihhäufigkeiten in den Fachgebieten BWL, VWL, Recht, Soziologie/Psychologie und Übrige am .....  
ausgeben

## Ausgabe

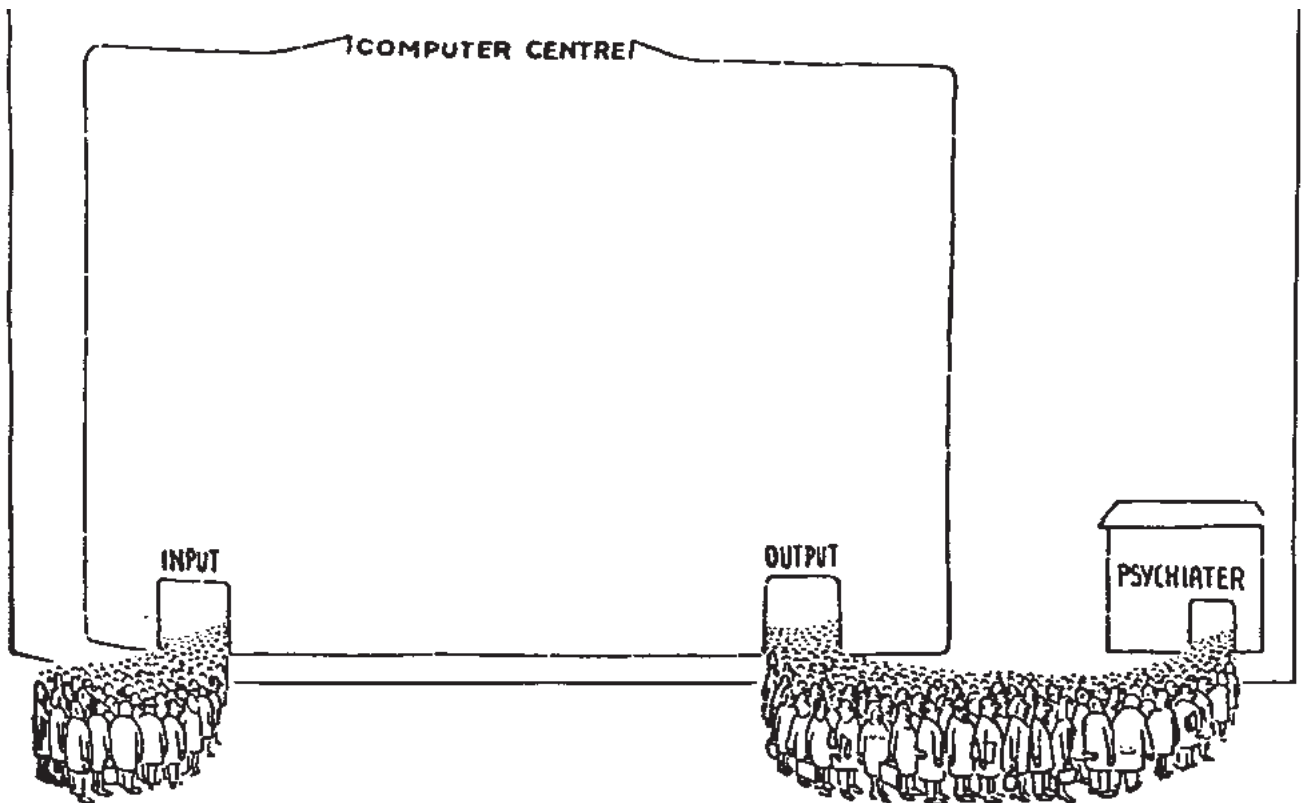
Histogramm der ..... Ausleihhäufigkeiten der Fachgebiete BWL, VWL, Recht, Soziologie/Psychologie und Übrige. Die maximale Balkenlänge wird der ..... Häufigkeit zugeordnet. Die Häufigkeiten werden durch proportionale Balken dargestellt.

## Eingabe

- a) ..... Ausleihhäufigkeiten der folgenden Fachgebiete  
BWL, VWL, Öffentliches Recht, Privates Recht, Soziologie,  
Psychologie, Übrige
- b) Ausleihemonat im Format .....



## II Modularisierung nach dem EVA-Prinzip



**E**ingabe + **V**erarbeitung + **A**usgabe

---

## II Modularisierung nach dem EVA-Prinzip

---

**E**ingabe

Ausleihhäufigkeiten und -monat einlesen



**V**erarbeitung

Ausgabe des Histogramms vorbereiten



**A**usgabe

Histogramm auf dem Bildschirm ausgeben

# III Umgangssprachlicher Entwurf (Entwurfscodes)

## Histogramm

### Daten eingeben

Programmtitel ausgeben

Für jede Ausleihhäufigkeit:

Prompt "Ausleihhäufigkeit"

.....

.....

Prompt "Ausleihemonat"

Monatszahl lesen

Monatszahl prüfen

### Daten verarbeiten

Histogramm vorbereiten

**Maximale Häufigkeit ermitteln →**

### Daten ausgeben

.....

.....

Für jede Ausgabehäufigkeit:

.....

.....

.....

## IV Programmiersprachliche Implementierung

### ① Datenfeld “Ausgabehäufigkeit”

Der Befehl **Ausgabehäufigkeit(i)** greift auf die **i-te** Zelle des Datenfelds mit dem Namen “Ausgabehäufigkeit” zu.

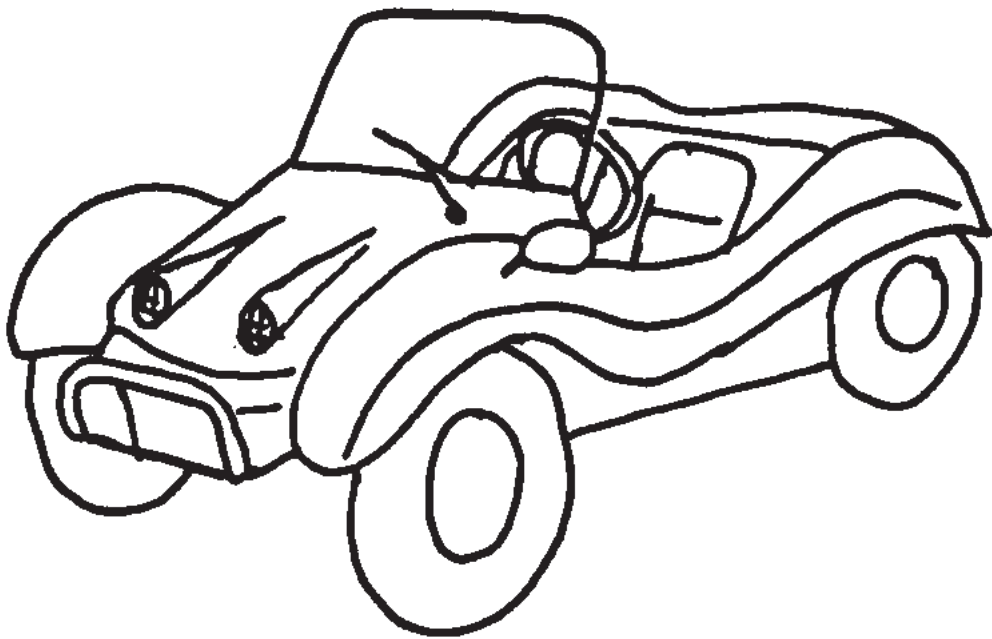
<b>Ausgabehäufigkeit</b>	200	100	500	300	400
<b>i</b>	1	2	3	4	5
<b>Fach</b>	BWL	VWL	Recht	Psych./Soz.	Übrige

### ② Algorithmus “Maximum ermitteln”

```
Maximum = Ausgabehäufigkeit(1)
For i = 2 To 5
    If Ausgabehäufigkeit(i) > Maximum Then
        Maximum = Ausgabehäufigkeit(i)
    End If
Next i
```

- 1) Weise einer Speicherzelle mit dem Namen *Maximum* die erste Ausgabehäufigkeit zu
- 2) Lies nacheinander alle übrigen Elemente des Datenfelds “Ausgabehäufigkeit”. Überschreibe jeweils den Inhalt von Maximum mit dem laufenden Element, sobald dieses *grösser* als der Inhalt von Maximum ist.
- 3) Nach dem Lesen aller Elemente des Datenfelds steht in der Speicherzelle Maximum die *maximale* Häufigkeit.

# Vielfalt von Programmiersprachen

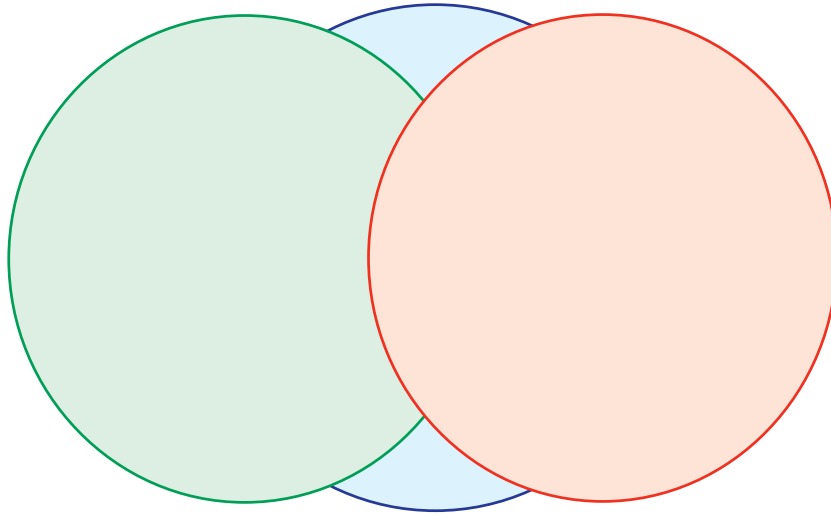


VW-Buggy / BASIC:

Einstiegsdroge für ein falsches Verhaltensmuster

Programmiersprachen ...

ein Turm zu Babel

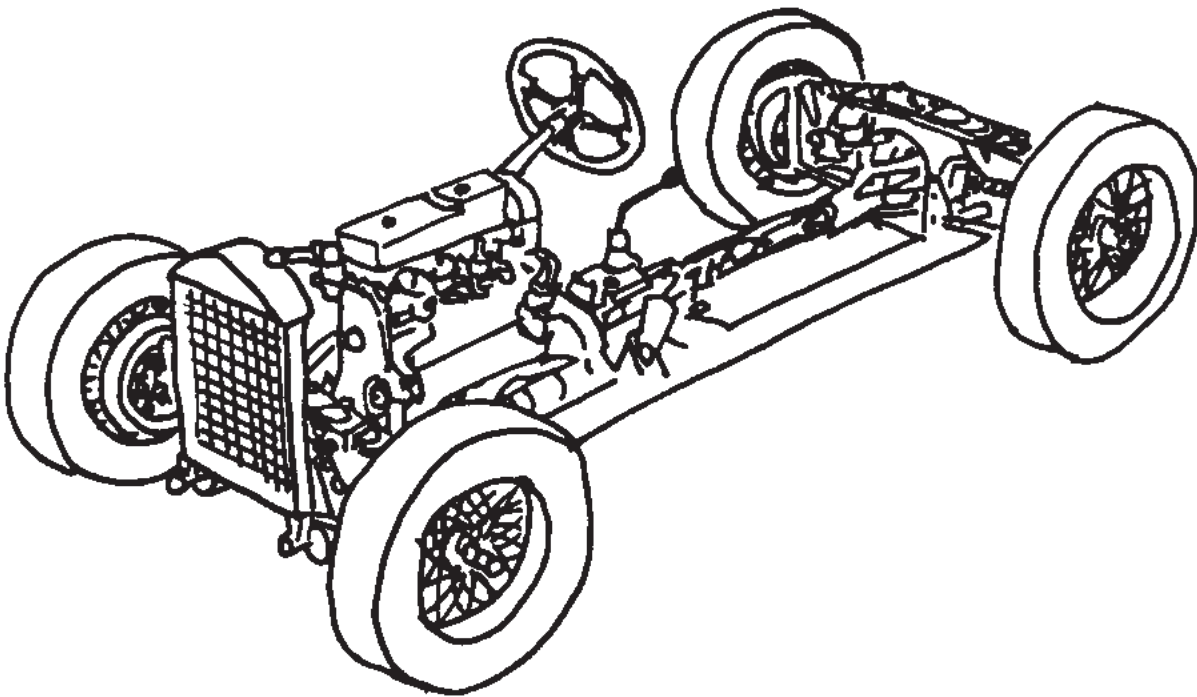


## ① Endbenutzerwerkzeuge

- **MS Excel** und **MS Access** zur ...  
komfortable Erstellung von Tabellenblättern und Datenbanken

## ② Programmierwerkzeuge

- **Visual Basic für Applikationen** (VBA) zur ...  
komfortable Programmierung *unselbständiger* Anwendungen  
in Excel, Access, Word, Project und Powerpoint aus MS Office
- **Visual Basic** zur ...  
komfortable Programmierung *selbständiger* Anwendungen
- **C++** zur ...  
Programmierung *effizienter* selbständiger Anwendungen



Fahrzeug-Chassis / Assemblersprachen:

Die Basis für alles andere, aber selbst nur für  
Spezialisten und Bastler brauchbar

## Maschinensprachen (1:1)

Bsp. 1100010100110001

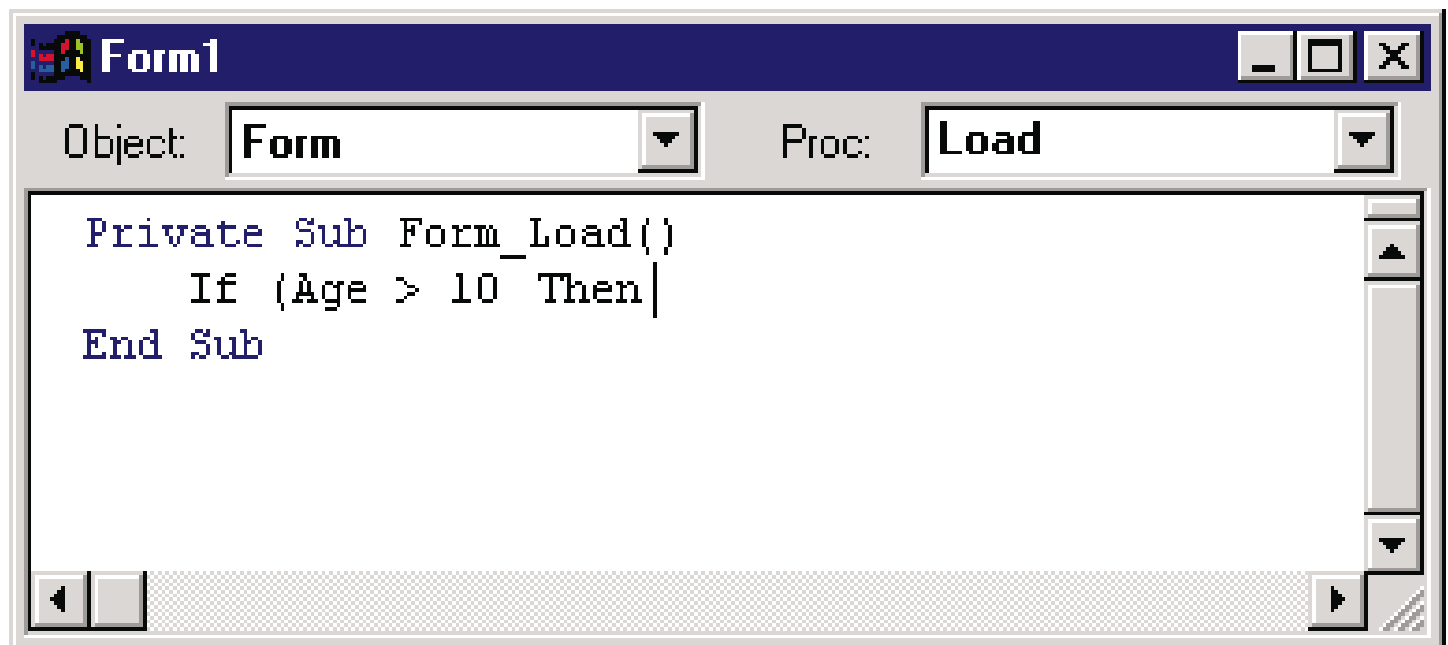
## Assemblersprachen (1:1)

Bsp. ADD Reg3, Reg3

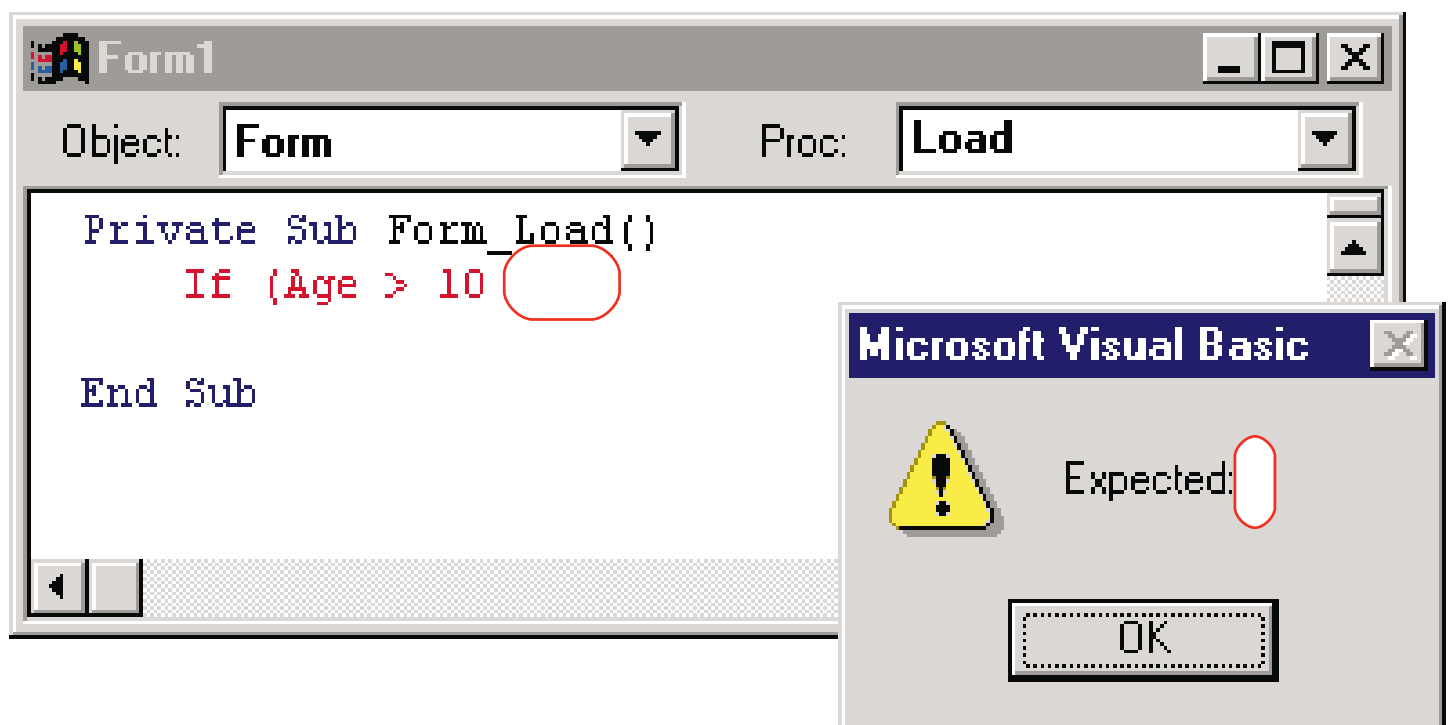
## Hochsprachen (1:n)

- *Allgemeine* - (Bsp. C++, VBA)
- *Dedizierte* - (Bsp VBScript)

# Programme gehorchen Grammatikregeln



*Wo ist der Syntaxfehler ?*



*Wie zeigt VBA den Fehler ?*



---

# Programmbeispiele

---

⇒ [Quadrat.xls](#) (Excel Spreadsheet)

Anweisung, Subroutine, Datentyp, Konstante, Variable, Grafikausgabe

## [Wort.xls](#)

Funktion, Objekt, Wiederholung, Entscheidung, Textausgabe

## [QuadratDialog.xls](#)

Schaltflächen

## [Gewinnverteilung.xls](#)

Dialog, Objekte

## [sucheBinär.xls](#)

Datenfeld, Suchalgorithmen

---

# Programmbeispiel **Quadrat**

---

Aus Linien ein Quadrat zeichnen

- ⇒ Programm in **Unterprogramme** aufteilen
- ⇒ Unterprogramm in **Anweisungen** aufteilen
- ⇒ Unterprogramm mit **Argumenten** flexibilisieren

“ProjektQuadrat”

“Ausgabeblatt”

“ModulQuadrat”

“Quadrat”

Projekt enthält

**Tabellenblätter** enthalten

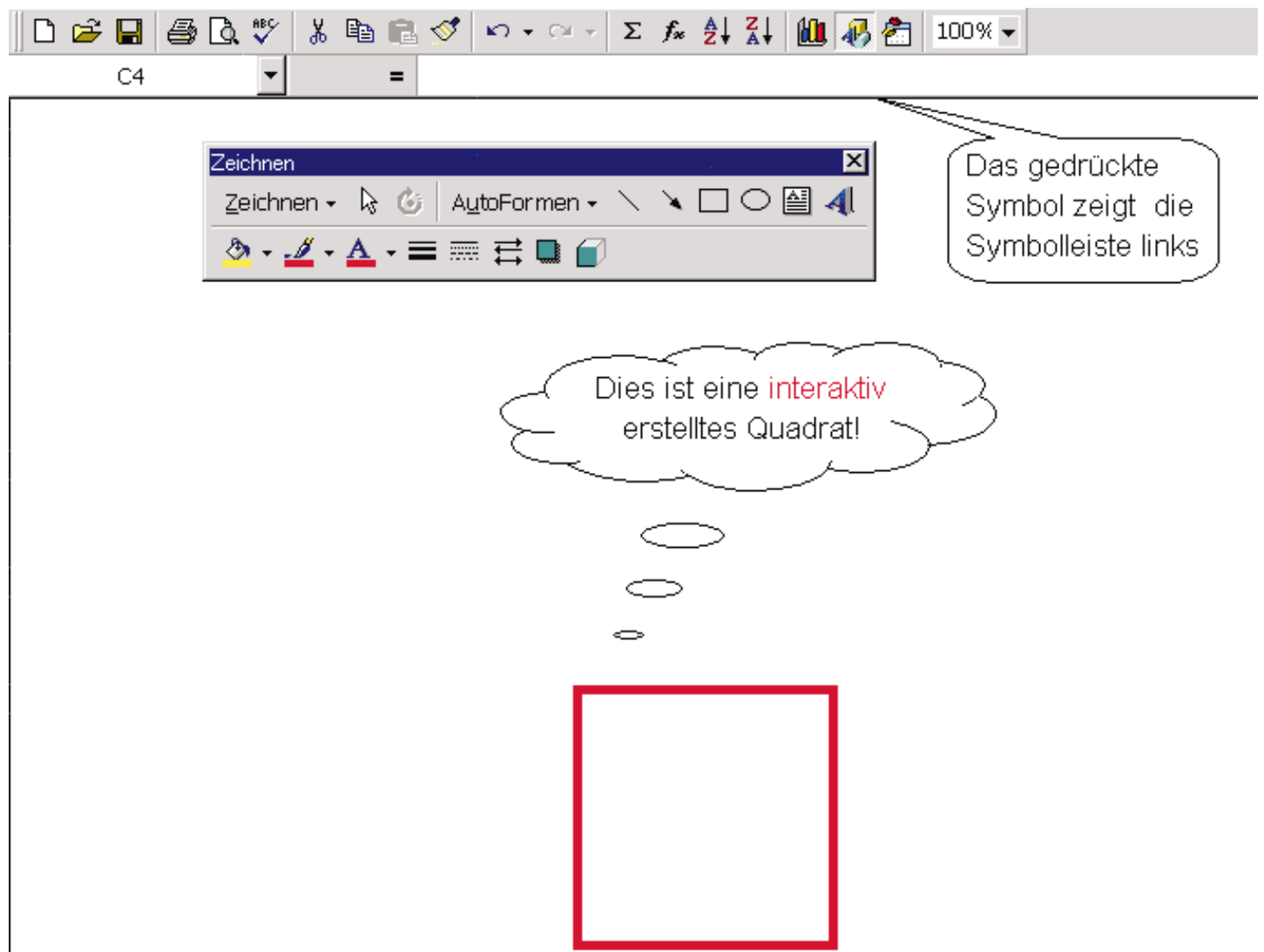
**Module** enthalten

**Unterprogramme** sind

**Subroutinen** → oder

**Funktionen** →

# Linie interaktiv zeichnen



# Koordinatensystem von VBA

*Masseinheit*

Punkt

*Ursprung*

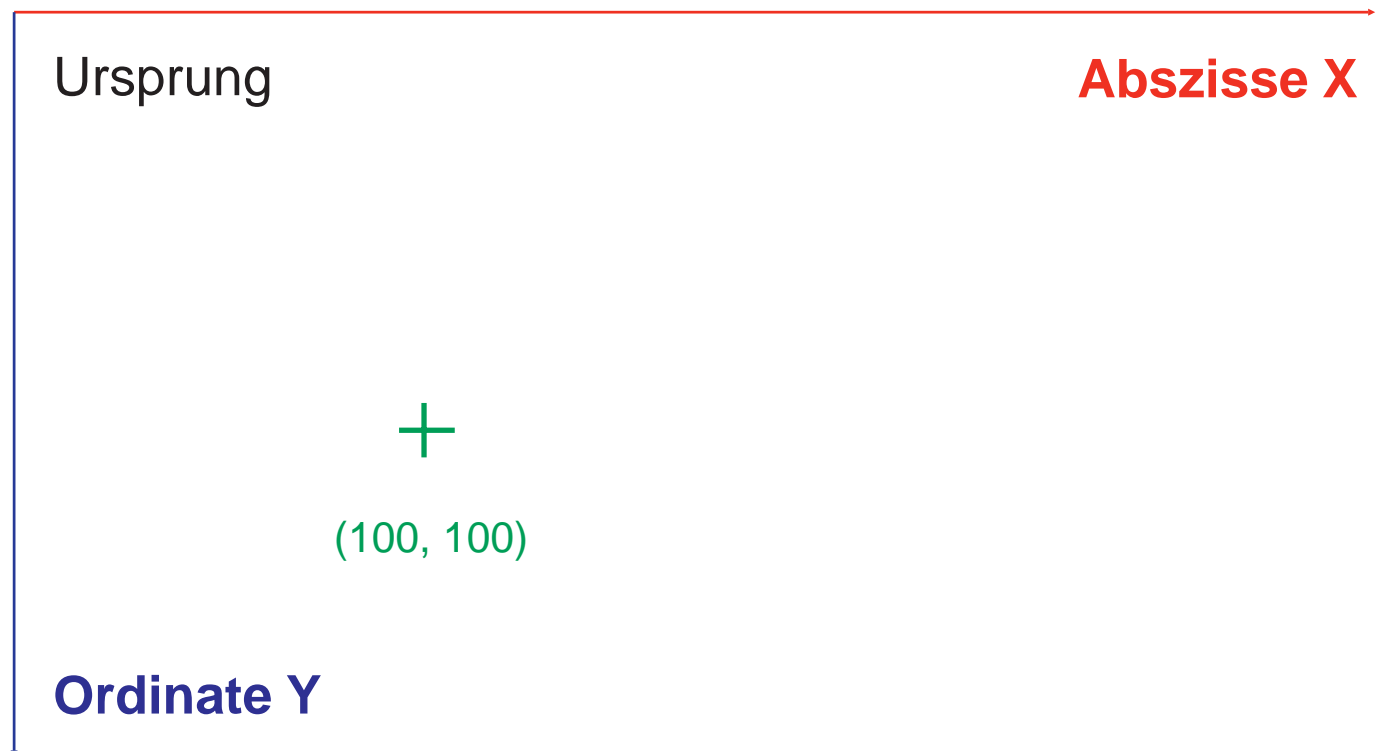
Punkt links oben mit den Koordinaten (0,0)

*Punkt (Koordinatendarstellung)*

100 Punkte vom Ursprung nach ..... und

100 Punkte vom Ursprung nach .....

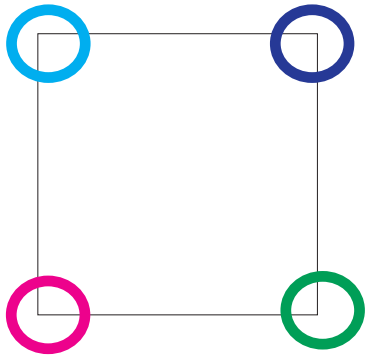
(0, 0)



# Entwurf Quadrat

## Aufgabe

Zeichne ein Quadrat mit der linken oberen Ecke auf (100, 100)



## Definition

**Baustein** QuadratFesterLänge

Zeichne

Gerade (100,100, 200,100)

Gerade (200,100, 200,200)

Gerade (200,200, 100,200)

Gerade (100,200, 100,100)

## Fragen

- **Baustein** programmiersprachlich?
- Gerade programmiersprachlich?

# Gerade programmgesteuert zeichnen

## Aufgabe

Gerade vom Punkt (100,100) bis zum Punkt (200,100) zeichnen



## Anweisung

```
Worksheets("Ausgabeblatt")  
    .Shapes.AddLine.(100,100, 200,100)  
    .Visible = True
```

*aus den folgenden Elementen:*

Wo? `Worksheets("Ausgabeblatt")`

auf dem Tabellenblatt "Ausgabeblatt"

Was? `.Shapes.AddLine(100,100, 200,100)`

aus den möglichen Grafikobjekten (shapes) eine Gerade zwischen den Punkten (100,100) und (200,100)

Wie? `.Visible = True`

die sichtbar ist

"Ausgabeblatt" ist eine Textkonstante,  
die das Tabellenblatt eindeutig benennt

# Vereinfachte Syntax

---

## Statt ...

```
Worksheets("Ausgabebblatt"1)  
    .Shapes.AddLine.(100,100, 200,100)  
    .Visible = True
```

## einfacher ...<sup>2</sup>

```
Ausgabebblatt3  
    .Shapes.AddLine.(100,100, 200,100)  
    .Visible = True
```

<sup>1</sup> Register in der Tabellenblattansicht

<sup>2</sup> Die Vereinfachung funktioniert nur, falls das Argument von Worksheets eine Konstante ist

<sup>3</sup> Wert der Eigenschaft Name im Projekt-Explorer



---

# Variable und Konstante

---

## Beispiel

Ausgabeblatt

```
.Shapes.AddLine.(100,100, 200,100)  
    .Visible = True
```

## Variable

**Variable** := Paar aus ...

- ⇒ fester **Name** eines Speicherplatzes (z.B. `Visible`)
- ⇒ änderbarer **Wert** dieses Speicherplatzes (z.B. `True`)

## Konstante

**Konstante** (z.B. `True`, `200`, `Ausgabeblatt`) ...

- ⇒ bezeichnet keinen Speicherplatz
- ⇒ kann ihren **Wert** nicht ändern

---

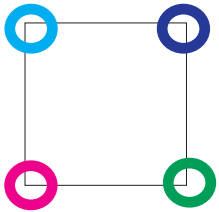
# Subroutine QuadratFesterLänge

---

## Aufgabe

Subroutine, die ein Quadrat zeichnet.

## Definition



```
Sub QuadratFesterLänge()  
  With Ausgabeblatt.Shapes  
    .AddLine(100,100, 200,100).Visible = True  
    .AddLine(200,100, 200,200).Visible = True  
    .AddLine(200,200, 100,200).Visible = True  
    .AddLine(100,200, 100,100).Visible = True  
  End With  
End Sub
```

Unbenannte Konstanten sind .....  
Vordefinierte benannte Konstante ist .....

## Vorteile

- Der Aufruf QuadratFesterLänge ist *verständlicher*
- *Bibliotheken* eigener und fremder Unterprogramme sparen Arbeit

## Fragen

- Shapes?, AddLine?, Visible?
- With ... End With?

---

# Vereinbarung und Aufruf einer Subroutine

---

## *Vereinbarung*

**Sub** Quadrat ()

...

**End Sub**

## *Aufruf*

...

Quadrat

...

Aufruf darf *keine* () enthalten!

# Eine Subroutine ist ein Unterprogramm

**Unterprogramm** := formal und inhaltlich zusammengehörende Anweisungen, die sich unter einem **Namen** aufrufen lassen

*Subroutine Quadrat*

Spezialfall eines Unterprogramms

*Syntax*

```
Sub Name ()  
    Anweisungen  
End Sub
```

*Zweck*

- Abkürzen
- Details verbergen
- Getestetes zur Verfügung stellen

# Umgebung - Tabellenblatt- und Modulsicht

## Tabellenblatt für die Eingabe und Ausgabe

erscheint nach einem **Doppelklick** auf \*.xls im Explorer



## Programmierungsumgebung für die Verarbeitung

erscheint nach **Alt / F11** im Tabellenblatt

```
Sub QuadratFesterLänge()  
  With Ausgabebblatt.Shapes  
    .AddLine (100,100, 200,100).Visible = True  
    .AddLine (200,100, 200,200).Visible = True  
    .AddLine (200,200, 100,200).Visible = True  
    .AddLine (100,200, 100,100).Visible = True  
  End With  
End Sub
```



## Projektexplorer

### Projekt ...

- listet die Projektelemente auf (Ctrl / R)

## Programmeditor

### (Deklarationen) klicken ...

- bietet ein Menü der Unterprogramme

### (Allgemein) klicken ...

- bietet ein Menü der Formularobjekte  
(z.B. seiner Textfelder und Schaltflächen)

---

# Projektextplorer für Quadrat.xls (Ctrl / R)

---

## ProjektQuadrat

### Objekte

ArbeitsmappeQuadrat

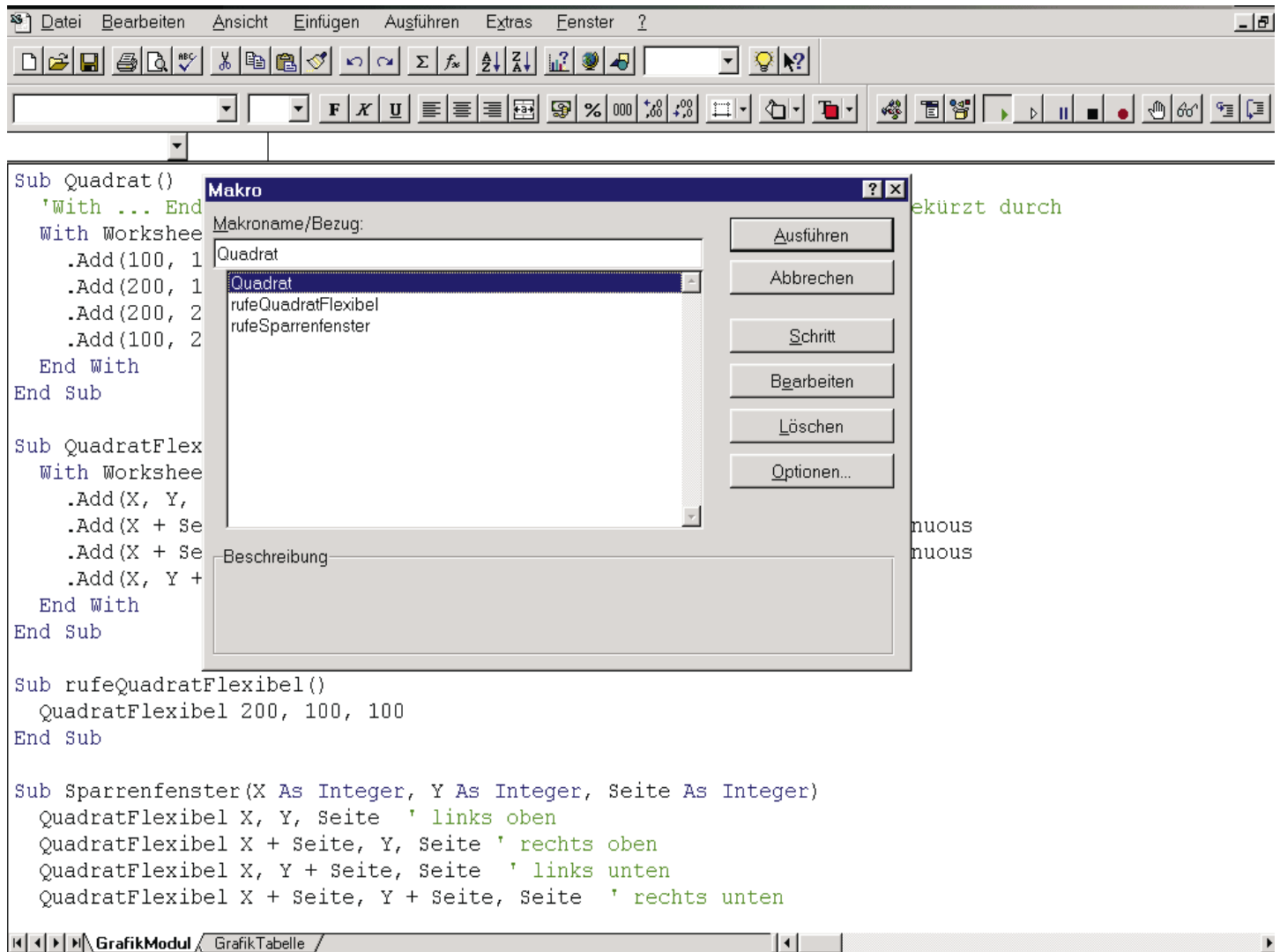
Ausgabeblatt

### Module

ModulQuadrat

Quadrat

# °Umgebung - F5 führt aus



**Welches Unterprogramm will ich ausführen?**

Ein Unterprogramm nennt man auch **Prozedur**



# °Umgebung - F8 testet zeilenweise

1. Einzelschrittmodus

2. Fenster anordnen

3. Einzelschritt

The screenshot shows the VBA environment with the following components:

- Testfenster (Test Window):** Displays the VBA code for the 'Quadrat' subroutine. The code is as follows:

```
Sub Quadrat()  
    'Alle Linien des Tabellenblatts "GrafikTabelle" löschen  
    Worksheets("GrafikTabelle").Lines.Delete  
    'With ... End With führt für das selbe Objekt mehrere Ar  
    With Worksheets("GrafikTabelle").Lines  
        .Add(100, 100, 200, 100).Border.LineStyle = xlContinuc  
        .Add(200, 100, 200, 200).Border.LineStyle = xlContinuc  
        .Add(200, 200, 100, 200).Border.LineStyle = xlContinuc  
        .Add(100, 200, 100, 100).Border.LineStyle = xlContinuc  
    End With  
End Sub  
  
Sub QuadratFlexibel(X As Integer, Y As Integer, Seite As I  
    Worksheets("GrafikTabelle").Lines.Delete  
    With Worksheets("GrafikTabelle").Lines  
        .Add(X, Y, X + Seite, Y).Border.LineStyle = xlContinuc  
        .Add(X + Seite, Y, X + Seite, Y + Seite).Border.LineSt  
        .Add(X + Seite, Y + Seite, X, Y + Seite).Border.LineSt  
        .Add(X, Y + Seite, X, Y).Border.LineStyle = xlContinuc  
    End With  
End Sub
```
- Ausgabefenster (Output Window):** An empty window for displaying output.
- VBA Toolbar:** The 'F8' button (Einzelschrittmodus) is highlighted with a red arrow.

# Überblick Datentypen und Ablaufstrukturen

## Einführung

- ✓ Modularisierung
- ✓ Subroutine - das einfachste Modul
- ✓ Entwicklungsumgebung von MS Excel

 [Web Quiz](#) "Einführung"

## Datentypen und Ablaufstrukturen



**Programm**

=

**Daten**

+

**Abläufe**

# Datentyp eines Speicherinhalts

**Speicherinhalte** müssen vor ihrer  
Verarbeitung interpretiert werden

Die Bitfolge 1010000 bedeutet je nach Interpretation 80 oder "P"



**Operationen** müssen ebenfalls  
interpretiert werden

+ wird je nach Interpretation des Speicherinhalts anders ausgeführt  
(  $1 + 2 = \dots\dots\dots$  und "Apfel" + "saft" =  $\dots\dots\dots$  )



Datentypen erleichtern die Interpretation

**Datentyp** := Vorschrift, die den **Wertebereich** und  
die **Operationen** eines Speicherinhalts festlegt

Bsp. Variable `Umsatz` ist vom Datentyp "**Gleitkommazahl**"

# Datentypen

Name	Bedeutung	Speicher	Wertebereich
<b>EINFACHE DATENTYPEN</b>			
<b>Integer</b>	kurze Ganzzahl	2 Bytes	-32.768 bis 32.767
<b>Long</b>	lange Ganzzahl	4 Bytes	ca. -10 Stellen bis +10 Stellen
<b>Single</b>	einfach genaue Gleitkommazahl	4 Bytes	ca. -3.4E38 bis -1.4E-45 ca. 1.4E-45 bis 3.4E38
<b>Double</b>	doppelt genaue Gleitkommazahl	8 Bytes	ca. -1.8E308 bis -4.9E-324 ca. 4.9E-324 bis 1.8E308
<b>Currency</b> (Geld)	Geldbetrag mit hoher Genauigkeit	8 Bytes	Festkommazahl (15 und 4 Stellen): #####.####
<b>Boolean</b>	Wahrheitswert	2 Bytes	True oder False
<b>Object</b>	Zeiger auf ein → Objekt	4 Bytes	unsichtbare Speicheradresse
<b>Variant</b>	beliebiger Inhalt	16 Bytes + 1 Byte für jedes Zch.	numerische Werte im Bereich Double oder ein beliebiger Text
<b>ZUSAMMENGESETZTE DATENTYPEN</b>			
<b>String</b>	Zeichenfolge	1 Byte / Zeichen	0 bis Milliarden von Zeichen
<b>Datenfeld</b>	benutzerdefinierte Folge von Elementen <i>gleichen</i> Typs	elementabhängig	elementabhängig
<b>Date</b>	Datum	8 Bytes	01. Jan. 100 bis 31. Dez. 9999
<b>Type</b>	benutzerdefinierte Folge von Elementen <i>ungleichen</i> Typs	elementabhängig	elementabhängig

## Aufgabe

Welche *Operationen* sind auf dem Datentyp `Integer` definiert ?

.....

.....

.....

---

# Datentypen **sparen, beschleunigen und sichern**

---

Datentypen **verbessern** die ...

- **Speichereffizienz**

`Integer` braucht 2 Bytes, `Double` 8 Bytes.

- **Laufzeiteffizienz**

`Integer`- schneller als `Double`-Operationen

Datentypen **vermeiden** ...

- **Syntaxfehler**

`TemporärVar` **statt** `TemporäreVar`

- **Laufzeitfehler**

```
Dim Parfumname As String  
Parfumname = 4711
```

---

# Jede Variable benötigt einen Datentyp

---

Speicherplätze müssen vor ihrer  
Verarbeitung interpretiert werden



Jede **Variable** hat deshalb einen **Datentyp**

```
Umsatz As Integer  
Name As String
```



Beginne jedes Modul mit **Option Explicit!**

`Option Explicit` schreibt Datentyp-Vereinbarungen vor

*Vereinbarte Variablen sind leichter  
lesbar, sicherer und effizienter*

---

## Variablen entsprechen **Zellen**

---

	A	B
1		
2		

### *Zelle*

**B2** ist eine “Schublade” für sich ändernde Zahlen- oder Textwerte.

### *Variable*

**Umsatz** ist eine “Schublade” für den sich ändernden Umsatz

# Konstanten und Variablen **vereinbaren**

**Vereinbaren** := einem Speicherplatz einen **Namen**, einen **Datentyp** und eventuell einen **Wert** zuordnen

## Variable

```
Dim1 Umsatz As Integer
```

```
Dim Name As String
```

```
Dim MeineZelle As Range (.....)
```

## Benannte Konstante

```
Const Grenze As Integer = 1000
```

1000 ist eine unbenannte und Grenze eine benannte Konstante

<sup>1</sup> Dim kommt von **dimensionieren**



# Einer Variablen einen Wert **zuweisen**

**Zuweisen** := einem Speicherplatz einen Wert geben

= trennt die linke und die rechte Seite

Umsatz = 15000

Name = "Fridolin"

Name = InputBox("Name?")

Linke und rechte Seite müssen sich vertragen

Umsatz = 15000 ok?

Name = 4711 ok?

Umsatz = "Fünfzehntausend" ok?

# Vereinbaren, initialisieren und aggregieren

## ① Vereinbaren

```
Dim Umsatz As Integer
```

## ② Initialisieren

```
Umsatz = 0
```

**Initialisieren** := einer Variablen einen .....wert zuweisen

## ③ Aggregieren

```
Umsatz = Umsatz + 10000
```

(Zähle zum aktuellen Wert vom Umsatz 10000 dazu und überschreibe den Wert der Variable `Umsatz` mit dem Ergebnis)

...

```
Umsatz = Umsatz - 1000
```

...

## Aufgabe 4.1 (Zuweisung)

**Sub** ..... ( )

```
Dim Alpha As String
Dim Omega As String
Dim Tmp As String
```

```
Alpha = "Alpha"
```

```
Omega = "Omega"
```

```
Tmp = Alpha
```

```
Alpha = Omega
```

```
Omega = Tmp
```

**End Sub**

Wert unmittelbar vor dem Verlassen der Subroutine ?

Alpha .....

Omega .....

Tmp ist die Abkürzung für .....

---

# Argument

---

Sub QuadratFesterLänge () flexibilisieren



## Aufruf

Quadrat 200, 100, 100 zeichnet vom Punkt (200,100) aus ein Quadrat der Seitenlänge 100

## Vereinbarung

```
Sub Quadrat(X As Integer, Y As Integer, Seite As Integer)
  With Ausgabeblatt.Shapes
    .AddLine(X, Y, X+Seite, Y).Visible = True
    .AddLine(X+Seite, Y, X+Seite, Y+Seite).Visible = True
    .AddLine(X+Seite, Y+Seite, X, Y+Seite).Visible = True
    .AddLine(X, Y+Seite, X, Y).Visible = True
  End With
End Sub
```

Unbenannte Konstanten: .....  
Vordefinierte benannte Konstante: .....  
Argumentwerte: .....  
Argumentvariablen: .....  
Datentypen: .....

## Vorteil

Quadrat(X, Y, Seite) ist *flexibler* als Quadrat()

---

# Argumentvereinbarung ist **sicher**

---

*Weshalb?*

Quadrat 200,100? ←

Quadrat "200", "oha lätz", 100?

Ein Unterprogramm akzeptiert i.d.R. nur ...

- ... eine bestimmte ..... von Argumenten
- ... Argumente mit einem bestimmten .....

*Wie?*

...

```
Quadrat(X As Integer, Y As Integer, Seite As Integer)
```

...

*Syntax*

**Sub** Name ( **Argumentvariable** **As** **Datentyp**, ... )

Anweisungen

**End Sub**

# Syntax von Vereinbarung und Aufruf

## Vereinbarung

Sub Quadrat (X As Integer, ...)

## Aufruf

Quadrat 200, 100, 100

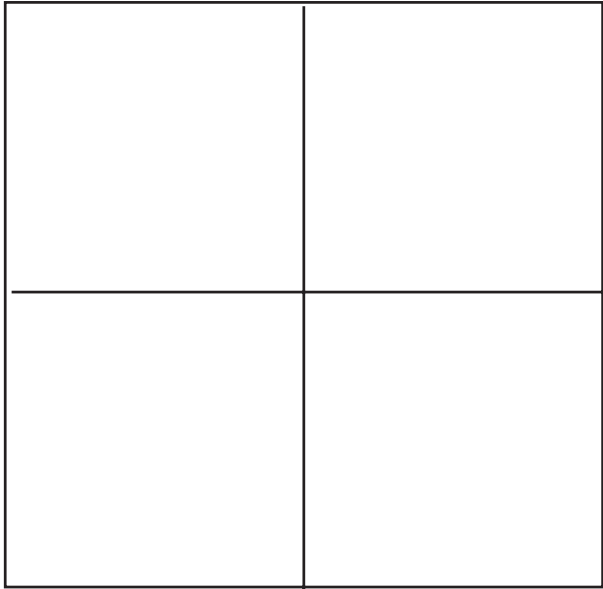
Die Vereinbarung erfolgt **mit**, der Aufruf **ohne** Klammern

---

# Hausaufgabe **Quadrat**

---

Zeichnen Sie das folgende “Sparrenfenster”, indem Sie die Subroutine `Quadrat` viermal aufrufen.



# Prospektiv programmieren

Quadrat anderen Projekten zugänglich machen



Projektabhängigkeiten eliminieren

Quadrat ()



Quadrat (  
X As Integer,  
Y As Integer,  
Seite As Integer )



Quadrat (  
Blattname As String,  
X As Integer,  
Y As Integer,  
Seite As Integer )



Im Konsumentenprogramm auf Quadrat.xls verweisen

(*Extras/Verweise*)



## Aufgabe 4.2 (🖱️ HEXAGON)

### Lernziele

- ⇒ Entwicklungssicht (Projekt-Explorer, Editor, Ausführung, Test)
- ⇒ Makro und Unterprogramm (Anweisung, Unterprogramm, Subroutine, Datentyp, Konstante, Variable, Argument, Grafikausgabe)

### Wiederholungsfragen

1. Was ist eine *Variable*?

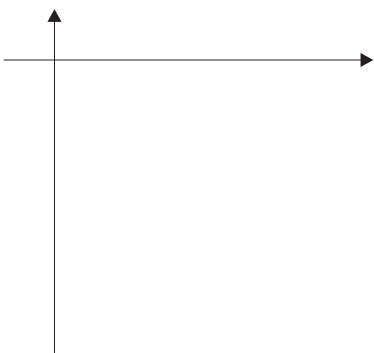
- a) Name eines änderbaren Speicherinhalts
- b) Zelladresse
- c) Vereinbarung eines Namens

2. Was ist ein *Makro*?

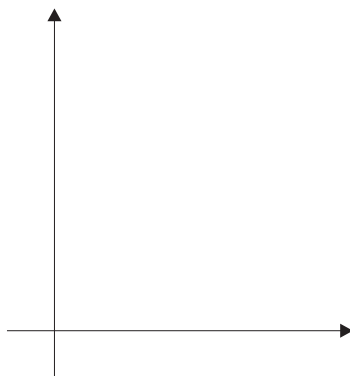
- a) automatische Vereinbarung einer Variablen
- b) Verknüpfung von Unterprogrammen in einem Modul
- c) Automatisierung mehrerer Endbenutzeraktionen durch Aufruf unter einem Namen

3. Welches *Koordinatensystem* verwendet MS Excel?

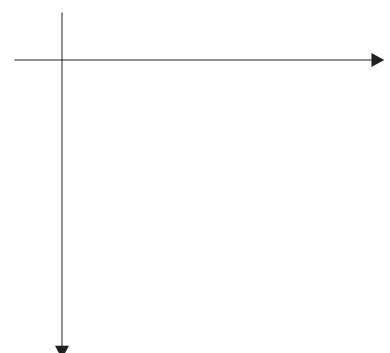
a)






b)



c)



## Vertiefungsfragen

- a) Laden Sie  [MakroSkelett.xls](#) und die Symbolleiste Zeichnen mit dem Menüpunkt “Ansicht/Symbolleiste/Zeichnen”.
- Wählen Sie aus Leiste das Symbol *Linie* und zeichnen Sie ein Rechteck.
  - Löschen Sie das Rechteck und zeichnen Sie erneut ein Rechteck. Zeichnen Sie aber dieses Mal ihre Aktionen als *Makro* auf (Menüpunkt “Extras/Makro aufzeichnen”).
  - Wählen Sie den Menüpunkt “Extras/Makro/Makros/Bearbeiten” und passen Sie das aufgezeichnete Makro im erscheinenden Editor so an, dass ein Quadrat statt eines Rechtecks gezeichnet wird.
  - Vergleichen Sie das aufgezeichnete Makro mit der Skript-Subroutine [QuadratFesterLänge](#).
- b) Laden Sie  [Quadrat.xls](#) und erfolgen Sie den Ablauf des Programms mit dem Debugger:
- Gehen Sie in die Programmiersicht (Alt/F11).
  - Klicken Sie auf die Zeile `Sub QuadratFesterLänge()`. Mit F8 können Sie dann den Programmablauf Zeile für Zeile verfolgen. (Ordnen Sie Ihre Fenster so an, dass Sie das Excel-Tabellenblatt und den Programmcode gleichzeitig sehen können)
  - Wählen Sie den Menüpunkt “Ansicht/Projekt-Explorer” und interpretieren Sie den Inhalt des öffnenden Fensters.
- c) Laden Sie  [HexagonSkelett.xls](#).
- Wählen Sie den Menüpunkt “Extras/Makro/Makros”, dann “testeGleichseitigesDreieck” und schliesslich “Bearbeiten”. Setzen Sie in der Prozedur `testeGleichseitigesDreieck` für den Aufruf von `gleichseitigesDreieck` verschiedene Werte ein und führen Sie die Prozedur mit F5 aus. Beobachten Sie die Wirkung auf dem Tabellenblatt. Welche Bedeutung haben die Argumente der Prozedur `gleichseitigesDreieck`?

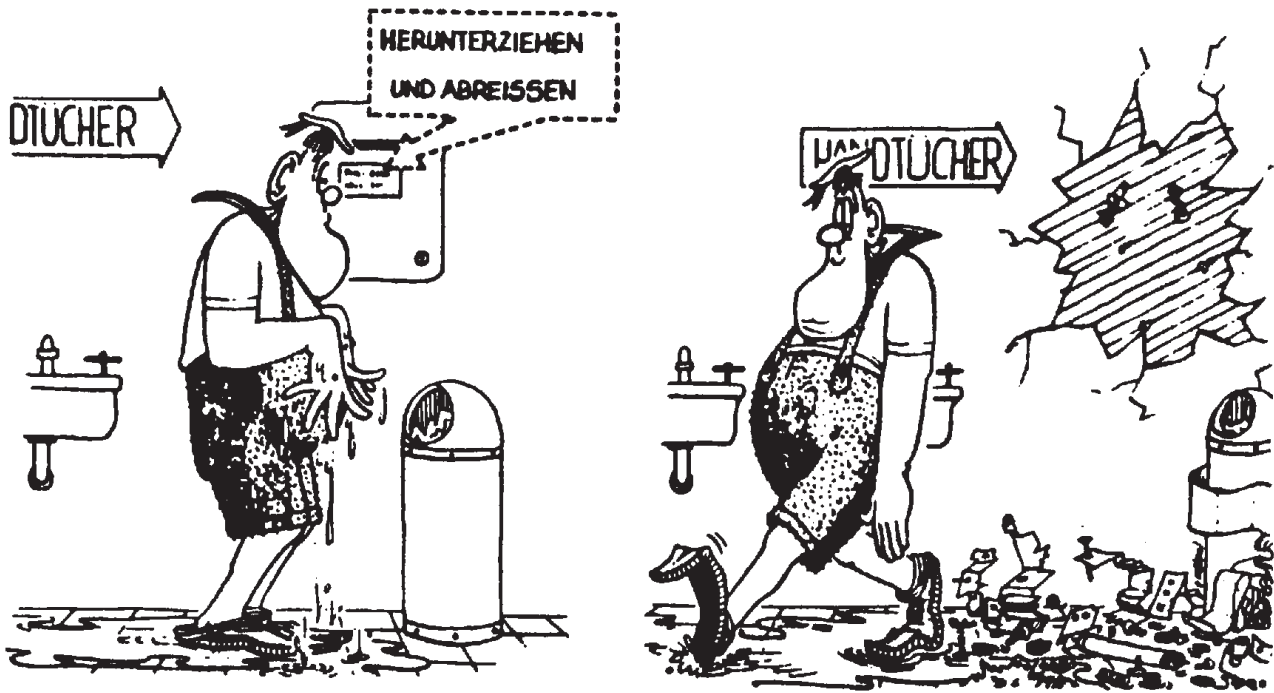
- Schreiben Sie eine Prozedur, welche durch mehrmaliges Aufrufen der Prozedur `gleichseitigesDreieck` ein Hexagon mit dem Mittelpunkt (200,100) und der Seitenlänge 70 zeichnet.

### *Zusatzaufgabe (für Studierende mit Vorkenntnissen)*

Falls Sie bisher die Subroutine `gleichseitigesDreieck` sechsmal aufgerufen haben, schreiben Sie eine Prozedur `HexagonMitSchleife`, welche die Mehrfachaufrufe durch eine Schleife ersetzt.

✓ [Hexagon.xls](#)

# Anweisungen



MS Office

Anwendungsspaket MS Excel

Arbeitsmappe Quadrat.xls

Modul ModulQuadrat

Unterprogramm Quadrat

Anweisung

Vereinbarungs - Dim

Ausführbare - AddLine

# Vereinbarungs- und ausführbare Anweisungen

**Vereinbarungs**anweisung :=  
Befehl, der Speicherplatz reserviert

**Ausführbare** Anweisung := Befehl,  
der *keinen* Speicherplatz reserviert

## Ausführbare Anweisungen

- **Ausdruck** (Formel) auswerten  $(10 + 2) / 3$
- **Zuweisung** ausführen  $x = 4$
- **Steueranweisung** ausführen If ... Then  $\rightarrow$
- ...

---

# System- und benutzerdefinierte Anweisungen

---

## Systemdefinierte Anweisung

=	Weise zu
+	Addiere
...	

## Benutzerdefinierte Anweisung

Quadrat	Zeichne ein Quadrat
...	

---

# Steueranweisungen

---

## Auswahl -

If Bedingung Then  
    Anweisungen  
Else  
    Anweisungen  
End If

*Aufgabe:* “Kräht der Hahn auf dem Mist, so ändert sich das Wetter oder es bleibt, wie es ist”

## Schleifen -

Do While Ausführungsbedingung  
    Anweisungen  
Loop

*Aufgabe:* Mit einer Variablen `Zähler` von 0 bis 3 zählen

# Wiederholungsanweisung

*Wie berechne ich die Summe  $1 + 2 + \dots + 1000$ ?*

Zahl = 1

Summe = 0

Do While Zahl <= 1000

Summe = Summe + Zahl

Zahl = Zahl + 1

Loop

*Verallgemeinern Sie :*

.....

Do While .....

.....

Loop

Initialisierung, Schleifenbedingung, Wiederholung



---

## Aufgabe 4.4a (Schleifenfehler)

---

Addieren Sie die Zahlen von 1 bis und mit 1000!

Zahl = 1

Summe = 1

Do While **Zahl <= 1000**

    Summe = Summe + Zahl

    Zahl = Zahl + 1

Loop

Wo ist der Fehler?

---

## Aufgabe 4.4b (While)

---

*Suchen Sie den Fehler!*

Zahl = 1

Summe = 0

Do While **Zahl <= 1000**

    Summe = Summe + Zahl

    Zahl = Zahl - 1

Loop

---

## Aufgabe 4.5 (While und If)

---

### Aufgabe

Das folgende Programmstück soll aus den *Stundenzahlen* mehrerer Arbeiter die *Lohnsumme* berechnen. Der *Stundenlohn* ist für die ersten 44 *Stunden* Fr. 30.-. Überstunden werden mit Fr. 40.- entlöhnt. Die Eingabe von 0 oder einer negativen Zahl führt zum Programmabbruch.

### Ergänzen Sie!

```
Lohnsumme = .....  
Stunden = InputBox("Stundenzahl (0 bricht ab): ")  
Do While Stunden > 0  
    Lohnsumme = Lohnsumme + Stunden * 30  
    If Stunden > 44 Then  
        Lohnsumme := .....  
    End If  
    .....  
Loop
```

Suchen Sie nach alternativen Lösungen !

# Zuweisung und Vergleich

## Zuweisungen schreiben

Name = "Fridolin" (Name "Fridolin")

## Vergleiche lesen nur

If Name = "Fridolin" Then ...

Do While Name = "Fridolin" ...

### Unterschied

Die Zuweisung ..... die linke Seite

Der Vergleich belässt hingegen beide Seiten.

# Überblick Objekte und Funktionen

✓ 1 Einführung (Modularisierung und Subroutine)

✓ 2 Datentypen und Ablaufstrukturen

 [Web Quiz](#) "Datentypen und Ablaufstrukturen"

**Objekt**



**Daten**

+

**Methoden** zu deren Manipulation

**Funktion**



**Unterprogramm**, das in einer Formel vorkommt

---

# Was ist ein Objekt?

---

Beispiel: Schaltfläche

Ein **Objekt** hat ...

- ✓ eine **Identität** durch einen eindeutigen Namen
- ✓ **Eigenschaften** wie Grösse oder Position
- ✓ ein **Verhalten**, zum Beispiel eine Reaktion auf einen Linksklick

# Objekte von QuadratDialog.xls

## *Prozedur und Dialog*

Quadrat zeichnen

Hilfe

Folien



Abszisse

x:

150

OK

Abbrechen

## Objekte

- **Schaltfläche** “Quadrat zeichnen”
- **Eingabefeld** “Abszisse”
- **Zellen** “Prozedur und Dialog”, “Hilfe” und “Folien”

## Definitionselemente von Schaltfläche

- ✓ eindeutiger **Name** “Quadrat zeichnen”
- ✓ **Eigenschaften** Position, Grösse und Schriftformat
- ✓ **Ereignisprozedur** “Quadrat\_nachKlick”

# MS Office stellt Objektklassen bereit

**Objektklasse** = Muster, nach dem der Entwickler Objekte erstellen kann (Bsp. nach der Objektklasse Flächendiagramm ein bestimmtes Flächendiagramm erstellen)

## Objekt

⇒ Mitglied (Instanz) einer Objektklasse

Bsp. *DiagrammUmsatz1999*

## Eigenschaft

⇒ Merkmal, das Objekte der gleichen Klasse beschreibt

Bsp. DiagrammUmsatz1999.*Title* = "Umsatzstatistik 1999"

## Methode

⇒ Aktion, die ein Objekt oder eine Eigenschaft manipuliert

Bsp. DiagrammUmsatz1999.*PrintOut(greyed)*



# Objekthierarchie

---

MS **Excel** zählt etwa 100 Objektklassen  
und 1000 Eigenschaften und Methoden

Anwendungspaket   MS Excel

Anwendung (Projekt)   Gewinnverteilung

Objektklasse   Schaltfläche

**Eigenschaft**   Bezeichnung

**Methode**   Formeln auswerten

---

# Definitionen

---

**Anwendungspaket** = Bibliothek vordefinierter Objektklassen

Bsp. *MS Excel* erstellt Arbeitsmappen, Tabellenblätter, ...

**Anwendung** = Programm aus vor- und benutzerdefinierten Objekten

Bsp. *Gewinnverteilung* aus Dialogblatt, Eingabeformular ...

## ① *Objektklasse*

Zusammenfassung von Eigenschaften und Methoden zur Erzeugung und Manipulation von Objekten des gleichen Typs

Bsp. Objektklasse *Worksheet* vereinigt Eigenschaften und Methoden zur Erstellung und Manipulation von Tabellenblättern

## ② *Eigenschaft*

Objektmerkmal, dessen Wert abgefragt und verändert werden kann

Bsp. *Farbe* des Objekts "TabelleblattXY" ist rot

## ③ *Methode*

Unterprogramm, das ein Objekt ergibt oder manipuliert

Bsp. *TabelleblattXY.Columns("A:C").Calculate* berechnet die Formeln des Bereichs A:C von TabelleblattXY

## ④ *Benutzerdefiniertes Unterprogramm*

benutzt Objektklassen, Methoden und Eigenschaften

Bsp. *Quadrat*

---

## Aufgabe 5.1 (Objektterminologie)

---

Unterscheiden Sie zwischen ...

- Anwendungspaket (application)
- Anwendung
- Objektklasse (class)
- **Objekt** (object)
- **Eigenschaft** (property)
- **Methode** (method)

MS Word .....

Tabellenblatt .....

Tabellenblatt "Gewinnverteilung" .....

Tabellenblatt schliessen .....

Grösse eines Tabellenblatts .....

ERdreistufig.xls .....

# Durch Punkte hierarchisch zugreifen

```
Application.                1)
    Workbooks (1) .         1)
        Worksheets ("XY") .  1)
            Range ("YZ") .   1)
                Value = 3    2)
```

1) Die **Methoden** Application, Workbooks, Worksheets wählen im Applikationspaket MS Excel aus ...

- *mehreren* Arbeitsmappen (workbooks, \*.XLS)
- *mehreren* Tabellenblättern (worksheets)
- *mehreren* Zellbereichen (ranges)
- *das* Objekt Arbeitsmappe **1**
- *das* Unterobjekt Tabellenblatt **XY**
- *das* Unterobjekt Bereich **YZ**

(Eine Methode ergibt ein Objekt, kann aber  
- anders als Objekte und Eigenschaften - Argumente haben)

2) = weist der Eigenschaft **Value** des Objekts YZ den Wert 3 zu

# Eigenschaftswerte ...

---

## ① zur Entwurfszeit setzen

bereits beim Dialogentwurf die Eigenschaftswerte zuweisen

Bsp. im Register den Namen des Tabellenblatts setzen

## ② zur Laufzeit setzen und lesen

### a) durch Aufruf **lesen**

```
MsgBox  
    Workbooks("XY.xls")  
        .Worksheets(1)  
            .Name
```

(MsgBox zeigt den Namen des Tabellenblatts 1 an)

### b) durch Zuweisung **ändern**

```
Workbooks("XY.xls")  
    .Worksheets(1)  
        .Name = "Blatt"
```

## Aufgabe 5.3 (Objektterminologie und VBA)

a) Was tut der folgende Code?

b) Identifiziere die ...

- **Listenschleife**
- **Auswahanweisung**
- **Objekte**
- **Eigenschaften**
- **Methoden**
- **Variablen**
- **Konstanten**

```
Dim Zelle As Range, leer As Integer  
leer = 0
```

```
For Each Zelle In Range("Bereich")
```

```
    If Zelle.Value = "" Then
```

```
        leer = leer + 1
```

```
    End If
```

```
Next Zelle
```

```
MsgBox "Leere Zellen: " & leer
```

Eine **Listenschleife** geht durch alle Listenelemente (z.B. alle Bereichszellen oder alle Tabellenblätter einer Arbeitsmappe)

---

## °Listenschleife **For Each ... Next**

---

### Beispielmuster

```
For Each Element In Liste
  If ...
    ...
    Exit For      ' springt auf Anweisung nach Next
  End If
Next Box
```

### Syntax

```
For Each ..... In .....
  Anweisungen
  [ Exit For ]
Next Objekt
```

### Listenbeispiele

- Liste aller offenen Tabellenblätter (worksheets)
- Liste aller Zellen eines Tabellenblattes

---

# Objektmodelle in MS Office

---

## Objektmodell des **jeweiligen** Applikationspakets

- MS *Access* Objects
- MS *Excel* Objects
- MS *Outlook* Objects
- MS *PowerPoint* Objects
- MS *Word* Objects

## Übergreifende Objektmodelle

- MS *Forms* Objects
- MS *Office* Objects
- *VBA* Objects

Wie finde ich mich im Objektdschungel zurecht?



# Ausschnitt aus dem Objektmodell von *Excel*

Objektklassen, Methoden, Eigenschaften

**Application** (Applikationspaket)

**Workbooks** (Arbeitsmappen)

**Worksheets** (Tabellenblätter)

**Range** (Zellbereich, z.B. Zelle, Spalte oder Zeile)

**Charts** (Diagramme)

...

**VBProject**

**Styles** (Formate)

**Borders** (Objektrahmen)

**Font** (Schriftart)

**Interior** (Innenbereich eines Objekts)

**Windows**

**Panes** (Fensterausschnitte)

**Names** (Namen von Zellbereichen)

...

**Dialogs** (vordefinierte Formulare)

...

**CommandBars** (Symbol- und Menüleisten) →

...

...s: Liste von Objekten möglich

Siehe Hilfe **VB für MS Excel** / Referenz zu **VB** /  
**Arbeiten mit VB** / *MS Excel-Objekte*

# °Objektklasse **CommandBars**

## Menüleisten

Menü

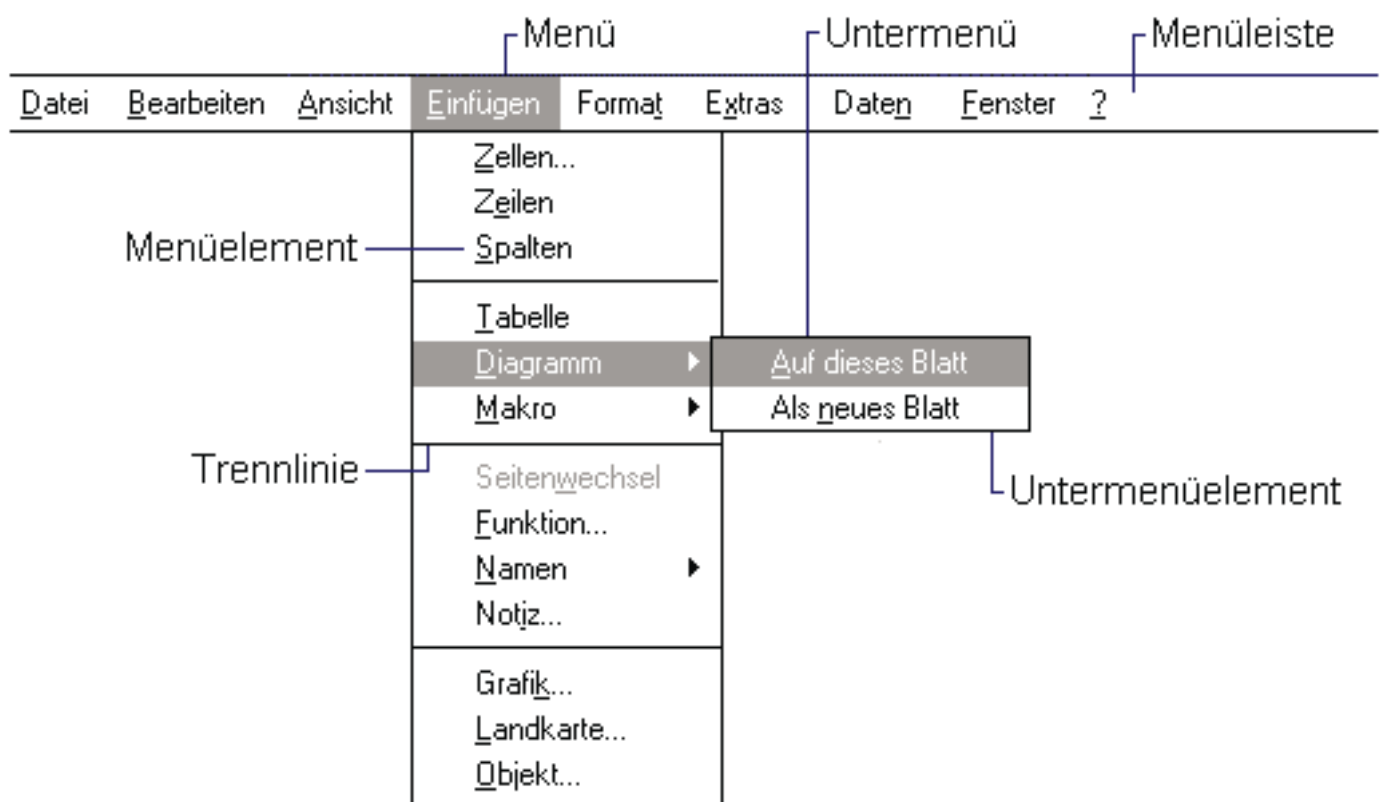
Menüelement

Trennlinie

Untermenü

Menüelement

Trennlinie



---

# Subroutine und Funktion

---

## Subroutine

Unterprogramm, das *keinen* Wert zurückgibt

`Worksheets("XY").Columns("A:C").Calculate`

## Funktion →

Unterprogramm, das einen *einzigsten* Wert zurückgibt

`SUMME(3;2) + 5 = .....`

Funktionen können im Gegensatz zu Subroutine in einer **Formel** (einem Ausdruck) vorkommen

---

# Programmbeispiele

---

## Quadrat.xls (Excel Spreadsheet)

Anweisung, Subroutine, Datentyp, Konstante, Variable, Grafikausgabe

### ⇒ Wort.xls

Funktion, Objekt, Wiederholung, Entscheidung, Textausgabe

## QuadratDialog.xls

Dialog

## Gewinnverteilung.xls

Dialog, Objekte

## sucheBinär.xls

Datenfeld, Suchalgorithmen

---

## Beispiel Wort

---

- **Funktion**
- While und If
- Cells-Eigenschaft
- Textfunktionen

Arbeitsmappe Wort.xls

⇒ Tabellenblatt **Ausgabeblatt**

⇒ Programm **ModulWort** (Programmierungsumgebung)

## Eingabe

Oh Klotilde, Du allein sollst und musst die Meine sein

## Ausgabe

Oh Klotilde, Du allein sollst und musst die Meine sein  
Klotilde, Du allein sollst und musst die Meine sein  
Du allein sollst und musst die Meine sein  
allein sollst und musst die Meine sein  
sollst und musst die Meine sein  
und musst die Meine sein  
musst die Meine sein  
die Meine sein  
Meine sein  
sein

## Verarbeitung

Cells(**2**, **2**) ergibt .....

. . .

Cells(**11**, **2**) ergibt .....

# WORT.XLS - Funktionsaufruf

```
Sub WortFürWortLöschen()
    Dim Zeile As Integer, Text As String
    ' -- Initialisierung
    Zeile = 2
    Text="Oh Klotilde, Du allein sollst und musst die Meine sein"
    ' -- Zeile 2
    Ausgabeblatt.Cells(Zeile, 2) = Text    1)
    Do While einWortWeniger(Text)        2)
        Zeile = Zeile + 1
        ' -- Zeile 3 etc.
        Ausgabeblatt.Cells(Zeile, 2) = Text
    Loop
End Sub
```

## 1) Eigenschaft

`Cells(X, Y)` ergibt den Wert der Zelle von Zeile `X` und Spalte `Y`.

## 2) Schleife

Die Schleife wird solange wiederholt, bis `Text` leer ist →

Der Aufruf von *Subroutinen* erfolgt **ohne**,  
der Aufruf von *Funktionen* **mit** Klammern.



---

# Arten von Funktionen

---

Eine Funktion ergibt einen *einzigsten* Wert

## ① **Vor**definierte Funktion

`Cells (Zeile, Spalte)` ergibt den Inhalt (Wert) eines Objekts der Klasse *Zelle* mit dem Bezeichner `(Zeile, Spalte)`

## ② **Benutzer**definierte Funktion

`einWortWeniger (Text) ...`

trifft zu, falls das erste Wort von `Text` gelöscht werden kann →

---

# Funktionsdefinition

---

## ① *Vordefinierte Funktionen*

<b>InStr</b> (Text, String)	Position von String in Text
<b>Right</b> (Text, N)	N rechte Zeichen von Text
<b>Len</b> (Text)	Zeichenzahl von Text

## ② *Benutzerdefinierte Funktion*

**einWortWeniger** (Text) ist wahr, falls eine Leerstelle existiert, die Text in einen linken und rechten Teil trennt

```
Function einWortWeniger (Text As String) As Boolean1)
    Dim PosLeer As Integer
    PosLeer = InStr(Text, " ")
    If PosLeer <> 0 Then
        Text = Right(Text, Len(Text) - PosLeer)
        einWortWeniger = True
    Else
        einWortWeniger = False
    End If
End Function
```

**Function** Name ( Argumente ) **As** Rückgabetyt  
Anweisungen  
**End Function**

1) Funktionen des Datentyps Boolean ergeben True oder False

---

# Wie übergibt ein Aufruf seine Argumente?

---

## Aufruf eines Unterprogramms (engl. call)

- ① *Argument als Adresse (Referenz) übergeben*
- ② *Argument als Kopie (Wert) übergeben*

## Aufruf von einWortWeniger

- ① *Call by Reference*

Oh Klotilde, Du allein sollst und musst die Meine sein  
Oh Klotilde, Du allein sollst und musst die Meine  
Oh Klotilde, Du allein sollst und musst die  
...

Änderung des **Originals**

- ① *Call by Value*

Oh Klotilde, Du allein sollst und musst die Meine sein  
Oh Klotilde, Du allein sollst und musst die Meine *sein*  
Oh Klotilde, Du allein sollst und musst die *Meine sein*  
...

Änderung nur der jeweiligen **Kopie**

# Variablen**adresse** als Argument übergeben

*Text* vor der Übergabe

**Text** Argument**adresse**

```
Sub WortFürWortLöschen()  
    Dim Zeile As Integer, Text As String  
    Zeile = 2  
    Text = "Oh Klotilde, Du allein sollst und musst die Meine sein"  
    Ausgabeblatt.Cells(Zeile, 2) = Text  
    Do While einWortWeniger(Text)  
        Zeile = Zeile + 1  
        Ausgabeblatt.Cells(Zeile, 2) = Text  
    Loop  
    MsgBox "Was vom Text noch bleibt ... : " & Text  
End Sub  
  
Function einWortWeniger(Text As String) As Boolean  
    Dim PosLeer As Integer  
    PosLeer = InStr(Text, " ")  
    If PosLeer <> 0 Then  
        Text = Right(Text, Len(Text) - PosLeer)  
        einWortWeniger = True  
    Else  
        einWortWeniger = False  
    End If  
End Function
```

Ein Argument **by reference** übergeben heisst  
die **Adresse** einer Variablen übergeben

Voreinstellung in Visual Basic!

# Variablen**kopie** als Argument übergeben

*Text* vor der Übergabe

**Text** Argument**kopie**

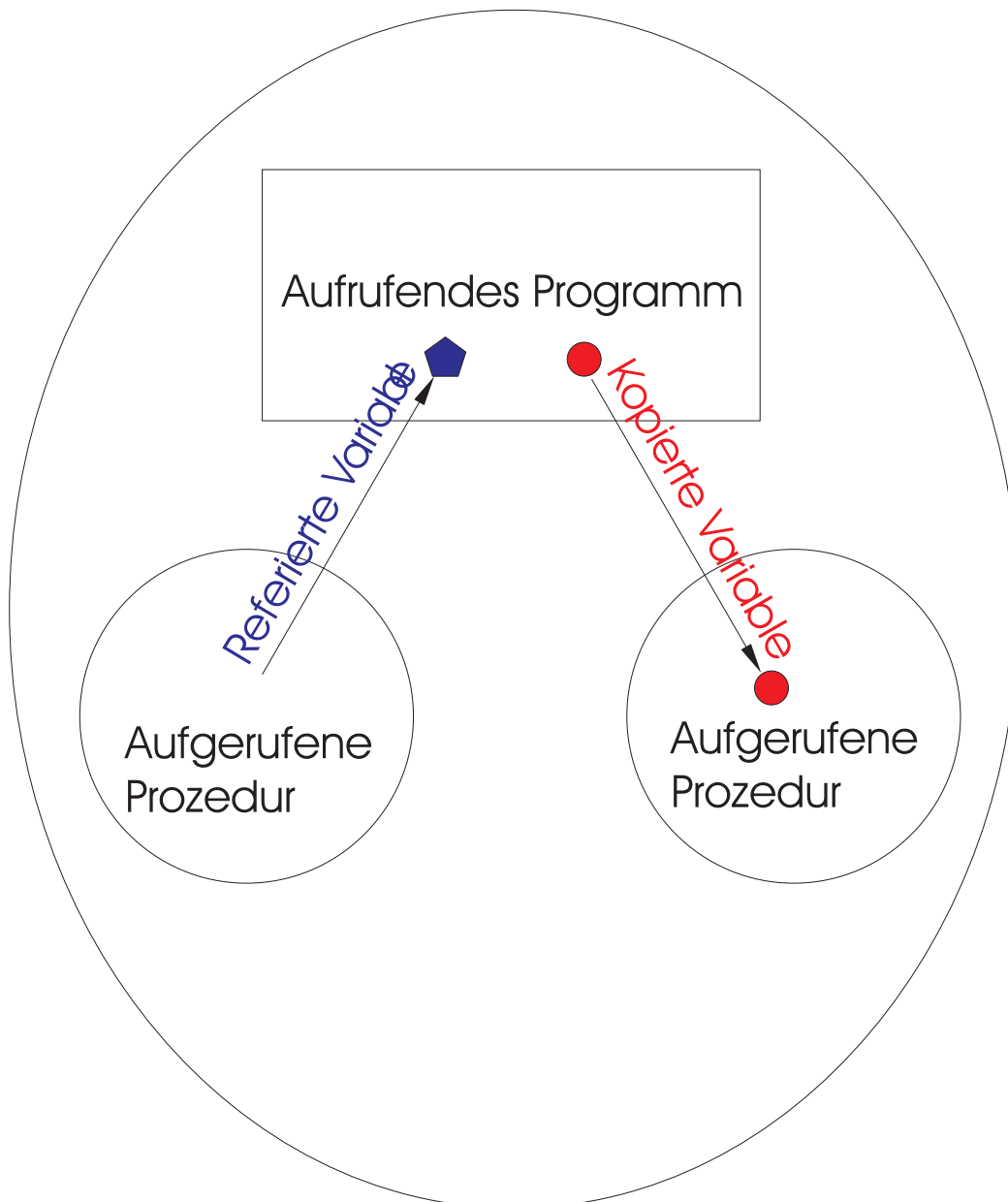
**Text** nach der Rückgabe (identisch mit dem Text vor der Übergabe)

```
Sub WortFürWortLöschen()  
    Dim Zeile As Integer, Text As String  
  
    Zeile = 2  
    Text = "Oh Klotilde, Du allein sollst und musst die Meine sein"  
  
    Ausgabebblatt.Cells(Zeile, 2) = Text  
    Do While einWortWeniger(Text)  
        Zeile = Zeile + 1  
        Ausgabebblatt.Cells(Zeile, 2) = Text  
    Loop  
    MsgBox "Unveränderter Text: " & Text  
End Sub  
  
Function einWortWeniger(ByVal Text As String) As ...  
    Dim PosLeer As Integer  
  
    PosLeer = InStr(Text, " ")  
  
    If PosLeer <> 0 Then  
        Text = Right(Text, Len(Text) - PosLeer)  
        einWortWeniger = True  
    Else  
        einWortWeniger = False  
    End If  
End Function
```

Ein Argument **by value** übergeben  
heisst seine **Kopie** übergeben

# By Reference oder By Value?

Übergabeart	<b>By Reference</b>	<b>By Value</b>
Syntax	(Voreinstellung)	<b>ByVal</b> in der Definition
Übergabe	Adresse	Kopie
Speichereffizienz	+	-
Laufzeiteffizienz	+	-
Sicherheit	- Aufrufumgebung wird beeinflusst	+ Aufrufumgebung bleibt <i>unbeeinflusst</i>



---

## Aufgabe 6.1 (☞ WORTEXPERIMENT)

---

### *Lernziele*

- ⇒ Funktion
- ⇒ Do While ... Loop
- ⇒ If ...Then ... Else ... End If
- ⇒ Objekte Tabellenblatt und -zelle
- ⇒ Textausgabe mit der Cells-Eigenschaft
- ⇒ vordefinierte Textfunktionen

### *Wiederholungsfragen*

1. Welchen Zweck verfolgen Eigenschaften und Methoden?
  - a) Eine Eigenschaft erzeugt die Methoden eines Objekts.
  - b) Eine Eigenschaft ist ein Objektmerkmal, und eine Methode ist ein vordefiniertes Unterprogramm, das Objekte erzeugt oder manipuliert.
  - c) Eine Eigenschaft erzeugt oder manipuliert ein Objekt und eine Methode legt fest, wie diese Manipulation abläuft.
2. Was ist richtig?
  - a) Formeln können Prozeduren und Funktionen enthalten.
  - b) Eine Funktion gibt einen wiederverwendbaren Wert zurück, eine Prozedur gibt keinen Rückgabewert zurück.
  - c) Es gibt keinen Unterschied zwischen Funktionen und Prozeduren.

### *Vertiefungsfragen*

- a) Betrachten Sie die folgende Funktion:

```
Function plus2(n As Integer) As Integer
    n = n + 2
    plus2 = n
End Function
```

Das folgende Unterprogramm ruft `plus2` auf:

```
Sub rufePlus2()
    Dim n As Integer
    n = 3
    n = plus2(n) * n
End Sub
```

- Welchen Wert hat `n` nach der Auswertung der Formel?
- Welchen Wert hätte `n` am Ende der Prozedur, wenn die Formel `n = n * plus2(n)` lauten würde?
- Verändern Sie die Funktion so, dass `n` in `rufePlus2()` durch den Aufruf von `plus2(n)` nicht verändert wird.
- Implementieren Sie die beiden Programme auf einem Tabellenblatt. Fügen Sie am Schluss von `rufePlus2()` eine Zeile ein, die `n` einer Tabellenzelle zuweist.

b) Laden Sie  [WortExperimentSkelett.xls](#).

- Welche Prozeduren und Funktionen enthält das Programm?
- Führen Sie die Subroutine `WortFürWortLöschen` aus.
- Kommentieren Sie jede einzelne Codezeile. Was passiert in welcher Zeile?. Welche Objekte, Methoden und Eigenschaften werden verwendet?
- Fügen Sie in die Schleife eine `MsgBox`-Anweisung, welche die Zeilennummer und den Text ausgibt.

c) Verändern Sie das Programm so, dass die Variable *by value* übergeben werden. Beschreiben Sie die Auswirkung.

✓ [WortExperiment.xls](#)



# Tabellen- und Programmierfunktionen

	<i>Excel</i>	<i>VBA</i>
<i>Sprache</i>	Deutsch	Englisch
<i>Funktionen</i>	<b>Tabellenkalkulations-</b>	<b>Programmier</b> funktionen

## ① *Tabellenkalkulationsfunktionen*

**MAX**(A1:A5) ergibt den grössten Wert des Zellbereichs A1:A5

## ② *Benutzerdefinierte Funktionen*

 Maximum sucht das Maximum einer Liste von Zahlen

## ③ *Durch VBA verwendete Tabellenkalkulationsfunktionen*

### **Application**

```
.MAX ("TabellenblattXY")  
    .Range ("A1:A5")
```

Maximum des Zellbereichs .....  
auf dem Tabellenblatt mit dem Namen .....

Vor Tabellenkalkulationsfunktionen steht in VBA  
**Application** (definiert durch *Applikationspaket* Excel)

---

# Benutzerdefinierte Tabellenfunktionen

---

*Der Tabellenkalkulationsteil kann auch benutzerdefinierte VBA-Funktionen verwenden:*

## ① *Entwurfssprachliche Definition*

```
Function Rabatt(Menge As Integer, Preis As Double)
    If Menge >= 100 Then
        Rabatt = Menge * Preis * 0.1
    Else
        Rabatt = 0
    End If
End Function
```

## ② *Aufruf*

*a) aus dem Tabellenblatt*

**=Rabatt**(20; 5)

*b) aus dem Programm*

```
...
Netto = Brutto - Rabatt(20, 5)
...
```

# Überblick Benutzerschnittstelle

- ✓ 1 Einführung (Modularisierung, Subroutine)
- ✓ 2 Datentypen und Ablaufstrukturen
- ✓ 3 Objekte
- ✓ 4 Funktionen

 [Web Quiz](#) “Objekte und Funktionen”

**Benutzerschnittstelle**

=

**Objekte**

+

**Ereignisse**

+

**Ereignisprozeduren**

# Programmierung des GUI

GUI zur **Entwurfszeit**

**G**..... **U**..... **I**.....



① GUI-**Objekte** interaktiv entwerfen

*Schaltfläche zeichnen*



② GUI-Objekten **Ereignisse** zuordnen

*Der Schaltfläche das Ereignis "Mausklick" zuordnen*

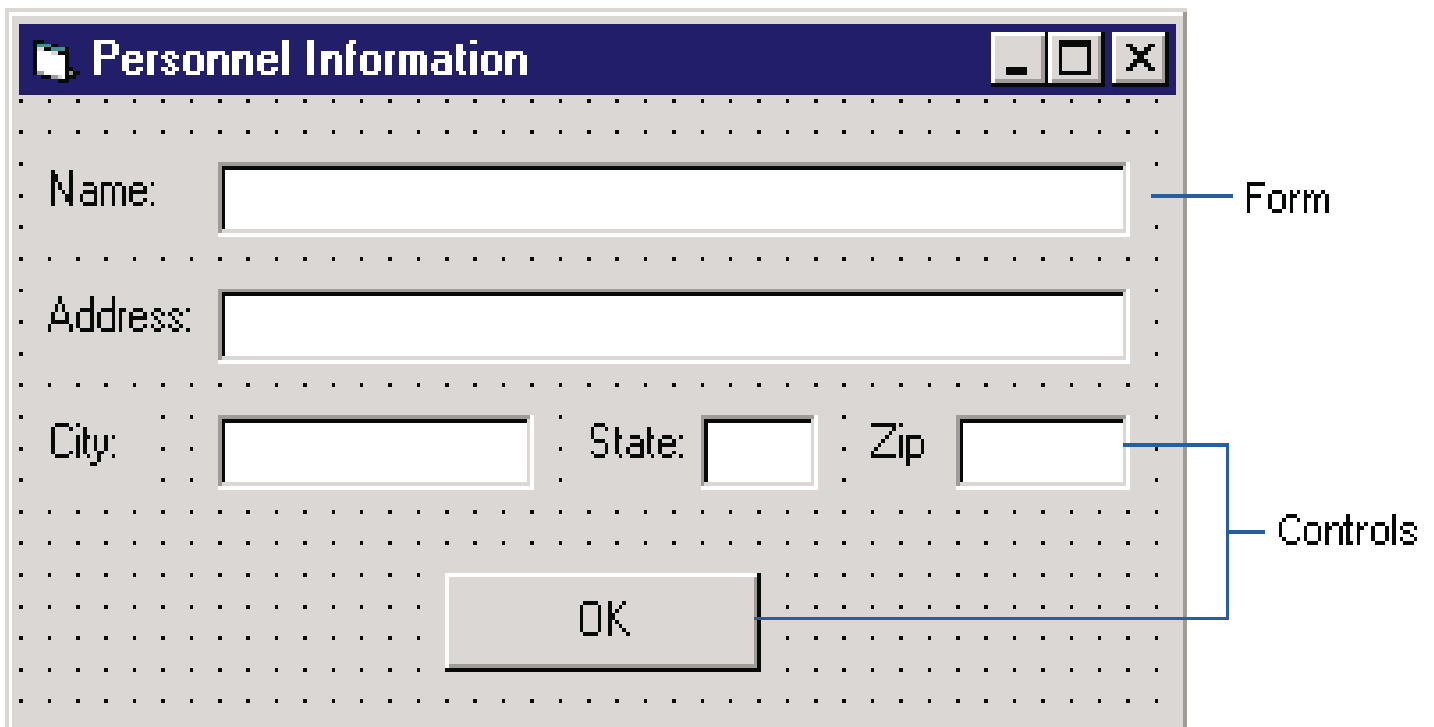


③ **Reaktionen** auf die Ereignisse programmieren

*Mit der Subroutine *Abbrechen* auf "Mausklick" antworten*



GUI zur **Laufzeit**



## ① GUI-Objekte interaktiv entwerfen

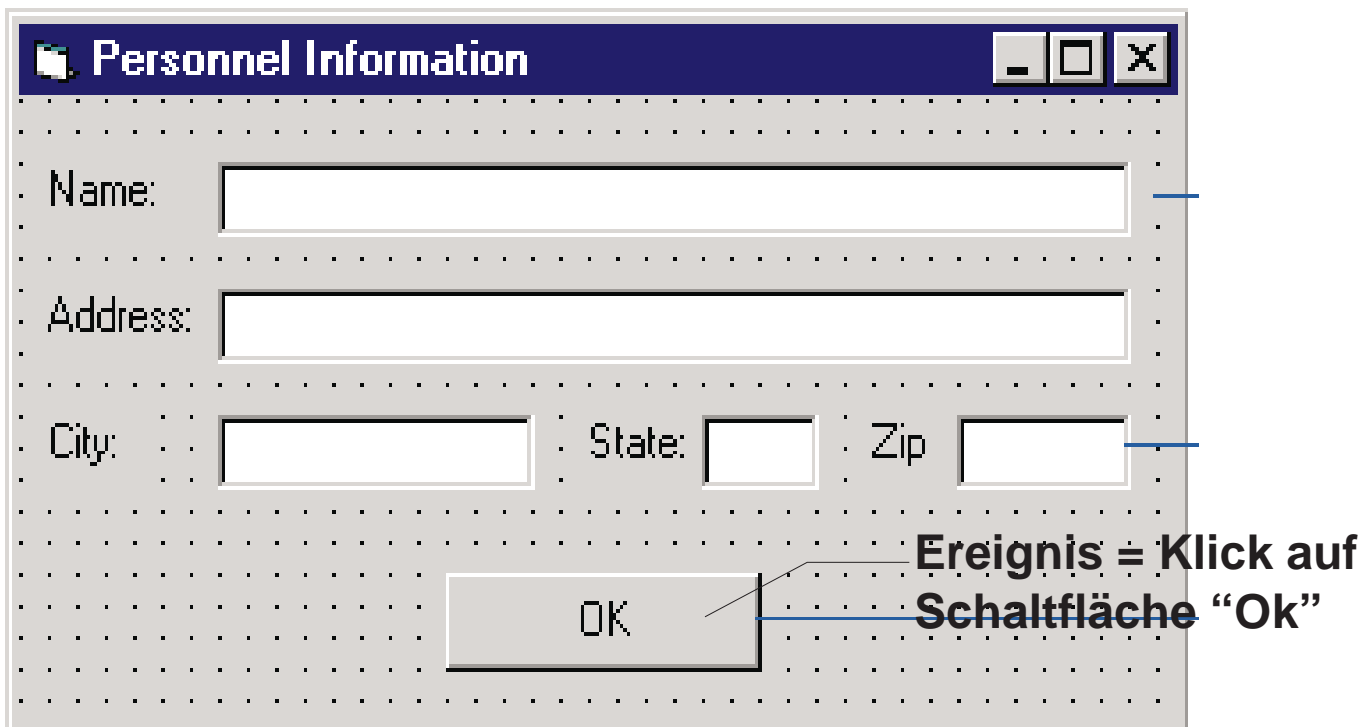
- Formular "Personnel Information" zeichnen
- Bezeichnungsfeld "Name" schreiben
- Textfeld "InhaltName" zeichnen
- Schaltfläche "OK" zeichnen

## ② GUI-Objekten Ereignisse zuordnen

## ③ Reaktionen auf die Ereignisse programmieren

Ein Formular besteht aus **Steuerelementen** (engl. controls)

# Ereignisorientierte Programmierung



① GUI-Objekte entwerfen ✓

② GUI-Objekten Ereignisse zuordnen

- Schaltfläche "Ok" dem Ereignis *OKButton\_Click* zuordnen

③ Reaktionen auf die Ereignisse programmieren

Jedes Steuerelement stellt eine Reihe von **Ereignissen** bereit

# Ereignisprozedur

---

*Ereignisprozedur reagiert auf Mausklick auf "Ok"*

```
Sub Ok_Klick()  
    ...  
    If Name.Text = "" Then  
        MsgBox "Bitte geben Sie Ihren Namen ein"  
    End If  
    ...  
End Sub
```

① GUI-Objekte entwerfen ✓

② VBA-Objekten Ereignisse zuordnen ✓

③ Reaktionen auf die Ereignisse programmieren

- *cmdOKButton\_Click* auf *OKButton\_Click* programmieren

# Objekt und Eigenschaft

The image shows a classic Windows-style dialog box titled "Personnel Information". It contains several text input fields: "Name:", "Address:", "City:", "State:", and "Zip:". Below these fields is an "OK" button. A red arrow points to the "Name:" label and its corresponding text box. Blue lines point to the "Address:", "City:", "Zip:", and "OK" labels and their corresponding controls.

Textfeld "InhaltName" ist ein **Objekt** des Formulars "Personnel...".

"Text" (Inhalt) ist eine **Eigenschaft** von "InhaltName".

"Fritz Meier" ist ein **Wert** der Eigenschaft "Text"

*Eigenschaften (und nicht Objekte) haben Werte!*

`InhaltName.Text = ""`

**Objekt** und **Eigenschaft** durch **Punkt** trennen



---

## °Objekte und Ereignisse

---

Arbeitsmappe bzw. Tabellenblatt wird

... geöffnet → ?

... geschlossen → ?

... aktiviert → ?

... deaktiviert → ?

... neu berechnet → ?

...

Steuerelement wird

... angeklickt → ?

... verlassen → ?

... geändert → ?

...

# Objekte von Arbeitsmappen

---

Arbeitsmappe aus ...

① Blättern (worksheets)

✓ Zellbereich (ranges)

✓ Diagramme (charts)

⇒ Menü- und Symbolleisten (bars)

⇒ Steuerelemente (controls)

② Formularen (user forms)

⇒ Steuerelemente (controls)

③ Modulen ✓

# Vordefinierte Formulare

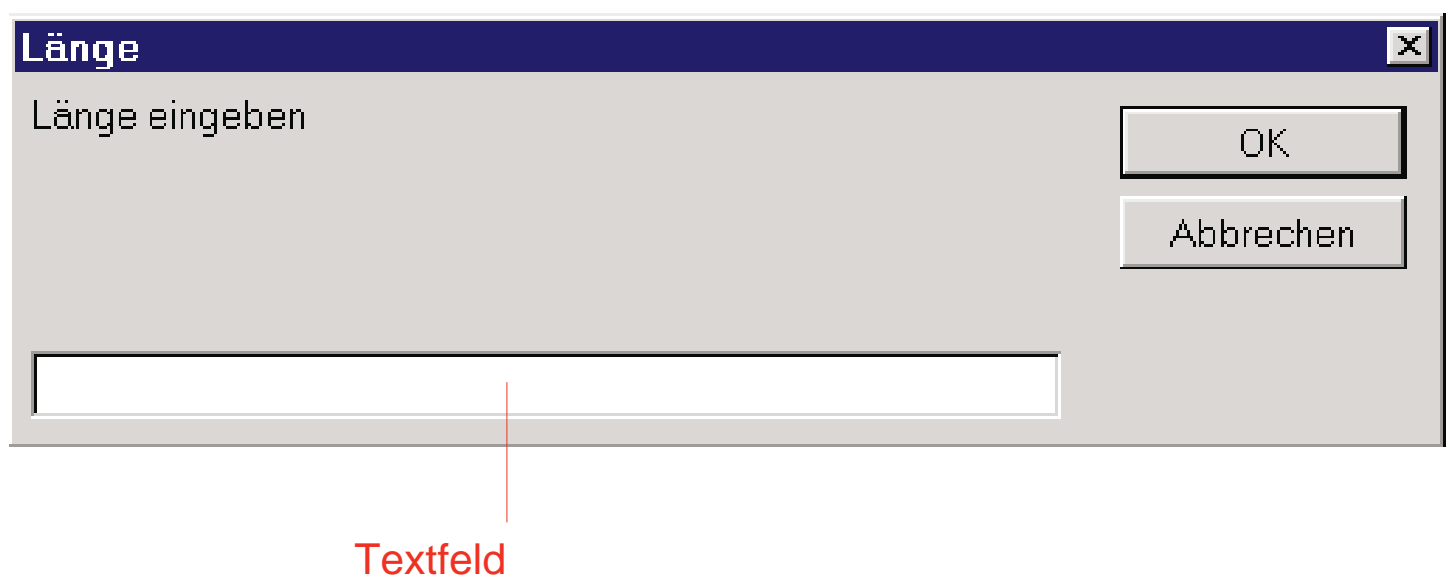
## ① Ausgabefeld

```
MsgBox Prompt := "Dateien schliessen", Title := "Dateien"
```



## ② Eingabefeld

```
Länge = InputBox(Prompt:="Länge eingeben", Title:="Länge")
```



---

# Benannte Argumente

---

## ① Positionsargumente

Quadrat 200,100

## ② Benannte Argumente

MsgBox Prompt := "Schliessen", Title := "Dateien"

statt

MsgBox "Schliessen", 0, "Dateien"

**Benannte Argumente** := Argumente, deren Aufgabe durch einen Namen statt eine Position definiert ist

- verkürzen Unterprogrammaufrufe
- verbessern die Leserlichkeit
- erfordern keine bestimmte Reihenfolge

---

## °Voreingestellte Argumentwerte (defaults)

---

... am Beispiel von MsgBox

<i>Name</i>	<i>Bedeutung</i>	<i>Voreinstellung</i>
<b>Prompt</b>	Meldungstext	""
<b>Buttons</b>	Zahl und Art der Schaltflächen	0
<b>Title</b>	Titelleiste	""
<b>Helpfile</b>	zusammen mit context	weglassen
<b>Context</b>	zusammen mit helpfile	weglassen

# Funktion mit oder ohne Rückgabewert

Mit Rückgabewert **Klammern**

```
Länge = InputBox( Prompt:="Länge eingeben", Title:="Länge" )
```

Ohne Rückgabewert **keine** Klammern

```
MsgBox Prompt := "Schliessen", Title := "Datei"
```

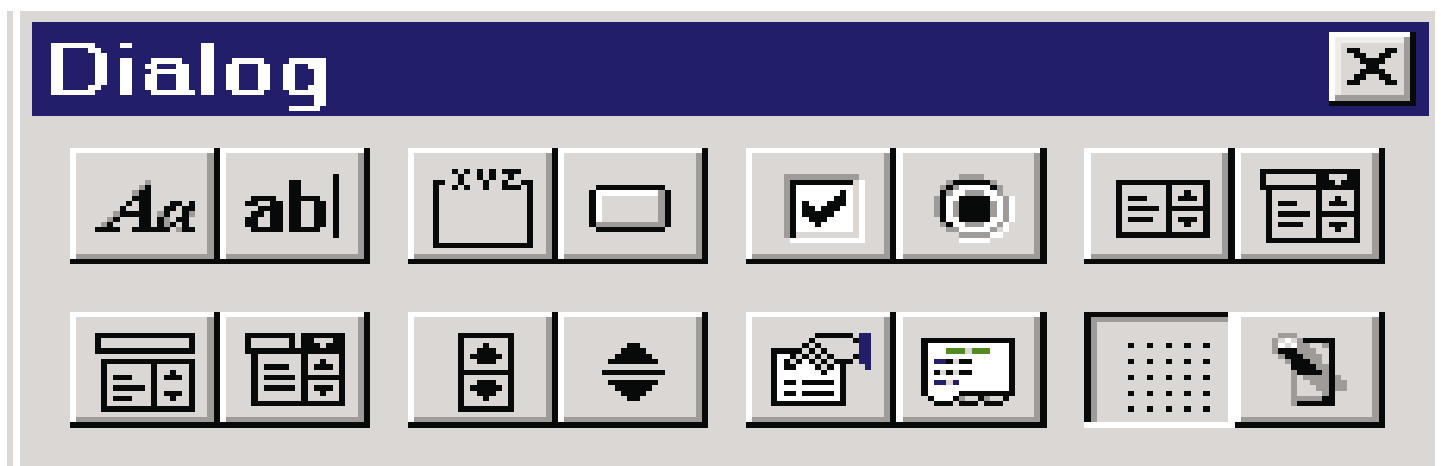
# °Hausaufgabe Steuerelemente I

**Steuerelement** := vordefiniertes elementares GUI-Element

## Arten

- |  |                                   |
|--|-----------------------------------|
| 1 <b>Bezeichnungsfeld</b>                | zur Beschriftung                  |
| 2 <b>Textfeld</b>                        | zur Eingabe                       |
| 3 <b>Gruppenfeld</b>                     | Rahmen für andere Steuerelemente  |
| 4 <b>Schaltfläche</b>                    | zum Anklicken                     |
| 5 <b>Kontrollkästchen</b>                | zum Markieren                     |
| 6 <b>Optionsfeld</b>                     | zum Markieren                     |
| 7 <b>Listenfeld</b>                      | zur Einfach- oder Mehrfachauswahl |
| 8 <b>Dropdown-Listenfeld</b>             | zur Einfachauswahl                |
| 9 <b>Kombinationsfeld Liste/Text</b>     |                                   |
| 10 <b>Kombinationsfeld Dropdown/Text</b> |                                   |
| 11 <b>Bildlaufleiste (Rollbalken)</b>    |                                   |
| 12 <b>Drehfeld</b>                       |                                   |

Welches Steuerelement gehört zu welchem Symbol?



## °Hausaufgabe Steuerelemente II

<i>Englisch</i>	<i>Deutsch</i>
Label	.....
Button	.....
OptionButton	.....
TextBox	.....
CheckBox	.....
ListBox	.....
GroupBox	.....
ScrollBar	.....
DropDown	.....
Spinner	.....

Ordnen Sie den englischen die deutschen Namen zu!

- 1 **Textfeld**
- 2 Gruppenfeld
- 3 **Schaltfläche**
- 5 Kontrollkästchen
- 6 Optionsfeld
- 7 Listenfeld
- 8 Dropdown-Listenfeld
- 9 Bildlaufleiste
- 10 Drehfeld



---

# Programmbeispiele

---

## Quadrat.xls (Excel Spreadsheet)

Anweisung, Subroutine, Datentyp, Konstante, Variable, Grafikausgabe

## Wort.xls

Funktion, Objekt, Wiederholung, Entscheidung, Textausgabe

## ⇒ QuadratDialog.xls

Dialog

## Gewinnverteilung.xls

Dialog, Objekt

## sucheBinär.xls

Datenfeld, Suchalgorithmen

---

# Beispiel QuadratDialog

---

## Tabellenblatt

- Ereignisprozedur zu einer **Schaltfläche**
- Argumente für Quadrat aus dem mit **InputBox** lesen
- Meldung mit **MsgBox** anzeigen

## Programm

- **Ereignisprozedur**

# Elemente eines Excel-Projekts

---

## Projekt

Benutzerschnittstelle

Arbeitsmappe

Tabellenblätter

Formulare (user forms)

Programmierschnittstelle

Module

Variablen und Konstanten

Unterprogramme

Variablen und Konstanten

Verweise auf andere Applikationen

**Projekt-Explorer** in Excel (Ctrl / R)

---

# Projektextplorer für QuadratDialog.xls

---

## ProjektQuadratDialog (QuadratDialog.xls)

### Objekte

ArbeitsmappeQuadratDialog

Tabellenblatt Dialog

### Module

ModulQuadratDialog

### Verweise

Verweis auf Quadrat.xls

---

# Verweis auf externen Code

---

Wer Code **anderer** Anwendungen verwendet ...

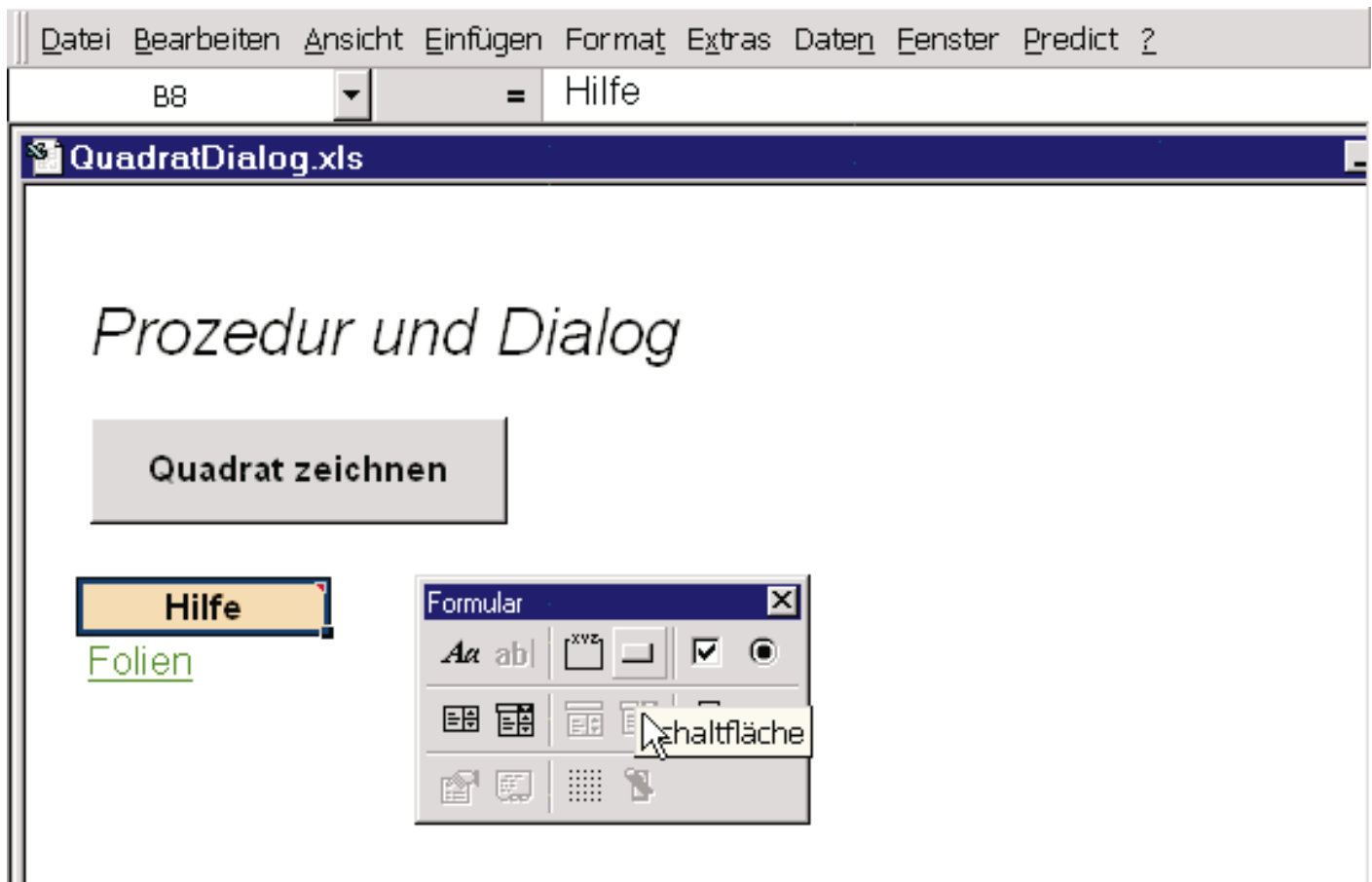


... verweist mit **Extras / Verweise** auf den Code ...

- **anderer Arbeitsmappen** (\*.xls, \*.xla für Add Ins)
- **ausführbarer** Dateien (\*.exe, \*.dll)
- von **ActiveX**-Steuerelementen (\*.ocx)
- von **Bibliotheken** (\*.olb, \*.tlb, \*.dll)

Durch Verweis kann QuadratDialog.xls die Subroutine *Quadrat* aus Quadrat.xls verwenden, ohne sie nochmals zu definieren

# Steuerelement ① - Schaltfläche



Programmierer zeichnet **Schaltfläche**  
interaktiv auf das **Tabellenblatt**

## Steuerelement ② - Eingabe

### *Prozedur und Dialog*

Quadrat zeichnen

Hilfe

Folien



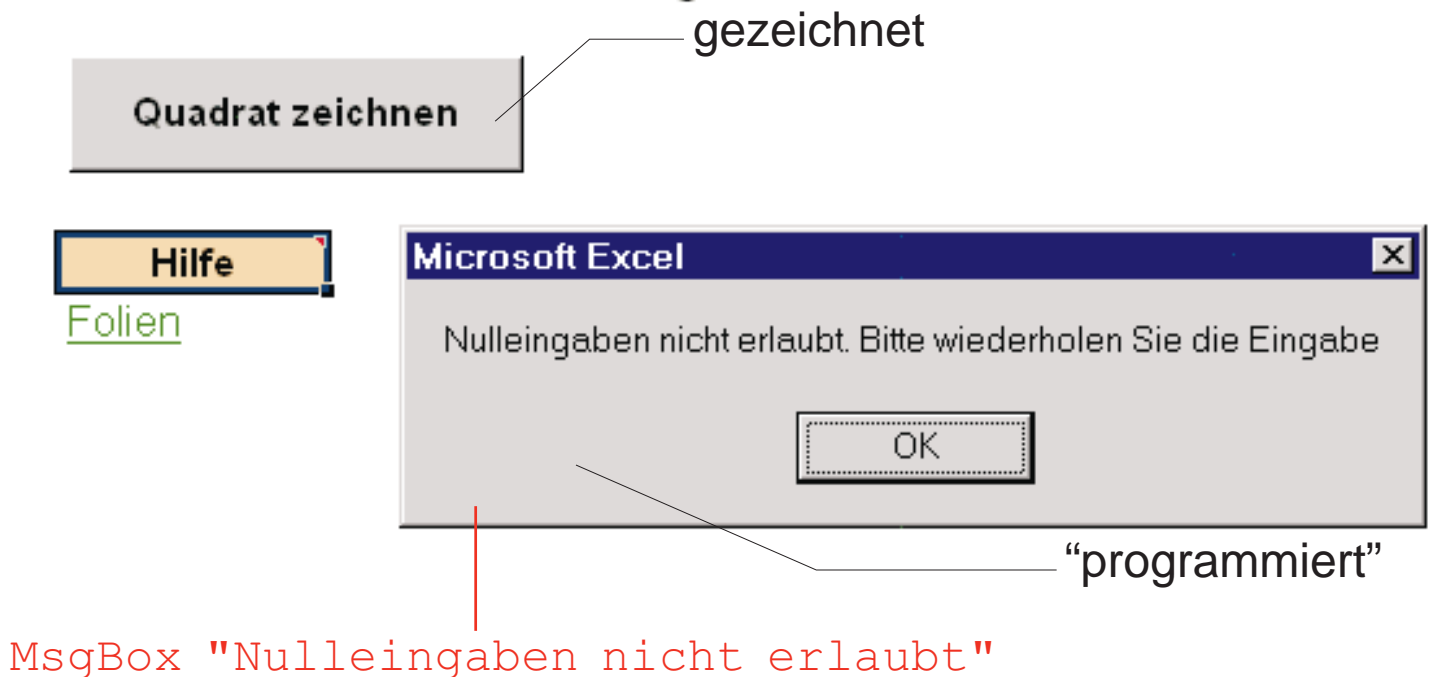
```
X = InputBox("X: ", "Abszisse")
```

Programmierer codiert **Eingabefeld** der Ereignisprozedur im **Editor** (Alt/F11)

Weshalb stehen die Argumente in Klammern ?

## Steuerelement ③ - Ausgabe

### *Prozedur und Dialog*



Programmierer codiert **Ausgabefeld** der Ereignisprozedur im **Editor** (Alt / F11)

Weshalb stehen die Argumente *nicht* in Klammern ?



---

# Ereignisprozedur der Schaltfläche

---

```
Sub Quadrat_Click()  
    Dim X As String, Y As String, Länge As String  
    ' Eingaben lesen  
    X = InputBox(Prompt:="X: ", Title:="Abszisse")  
    Y = InputBox("Y: ", "Ordinate")  
    Länge = InputBox("Seitenlänge: ", "Quadrat")  
    ' Eingaben prüfen  
    If X = "0" Or Y = "0" Or Länge = "0" Then  
        MsgBox "Nulleingaben nicht erlaubt"  
    Else  
        Quadrat "Dialog", CInt(X), CInt(Y), CInt(Länge)  
    End If  
End Sub
```

## Erläuterung

Ganzzahl = **CInt**(Ausdruck)

konvertiert einen Ausdruck eines numerischen oder alphanumerischen Datentyps in den Datentyp Integer

(engl. **C**..... **to Integer**)

Welchen Nachteil hat diese Art der Gültigkeitsprüfung ?

## Aufgabe 7.1 (Logische Operatoren)

Or And Not

***Wird die Auswahlanweisung betreten?***

***Gegeben***

- a) X sei 100
- b) X sei 15

***Entscheidungsanweisungen***

- 1) If X < 10 Or X > 20 Then ...
- 2) If X < 10 And X > 20 Then ...
- 3) If Not (X < 10 And X > 20) Then ...
- 4) If Not (X < 10 Or X > 20) Then ...

## Aufgabe 7.2 (☞ HEXAGONDIALOG)

### Lernziele

- ⇒ Schaltfläche
- ⇒ Ereignisprozedur
- ⇒ Validitätsprüfung
- ⇒ InputBox und MsgBox

### Wiederholungsfragen

1. Was ist richtig?

- a) *By Value* übergibt eine Argumentadresse, so dass die aufgerufene Sub- oder Funktionsprozedur den Inhalt der Variable selbst ändern kann. *By Reference* übergibt eine Kopie, d.h. Wertänderungen bleiben auf das Unterprogramm beschränkt.
- b) *By Reference* übergibt eine Argumentadresse, so dass die aufgerufene Sub- oder Funktionsprozedur den Inhalt der Variable selbst ändern kann. *By Value* übergibt eine Kopie; Wertänderungen beschränken sich auf das aufgerufene Unterprogramm.
- c) *By Value* übergibt eine Argumentadresse, d.h. Wertänderungen bleiben auf das Unterprogramm beschränkt. *By Reference* übergibt eine Kopie so, dass das aufgerufene Unterprogramm den Inhalt der Variable selbst ändern kann.

2. Ein *Steuerelement* ist ...

- a) ein vordefiniertes Element eines Dialogblattes (Formulars).
- b) eine Methode zur Steuerung von Ereignissen.
- c) ein Range-Objekt innerhalb einer Arbeitsmappe.

3. Wann ist die folgende Bedingung erfüllt:

$X="0" \text{ Or } Y="0" \text{ Or } \text{Länge}="0"?$

- a) falls X, Y und Länge 0 sind.
- b) falls eine der Variablen X, Y oder Länge den String "0" enthält.
- c) falls eine der Variablen X, Y oder Länge den Integer-Wert 0 enthält.

### Vertiefungsfragen

Laden Sie  [QuadratDialogSkelett.xls](#).

- a) Gehen Sie mit Alt/F11 in die Entwicklungsumgebung und beschreiben Sie die Struktur des Projektexplorers. Was bedeutet die Kartei Verweise im Projekt QuadratDialog?
- b) Verändern Sie die Aufrufe von `InputBox` so, dass die Eingabefelder bereits eine sinnvolle *Voreinstellung* enthalten (Ctrl/F1 auf `InputBox` führt Sie zu einem Hilfetext).
- c) Verbessern Sie die *Validitätsprüfung* derart, dass jede Eingabe sofort geprüft wird. Bei einem Fehler kann die Eingabe sofort wiederholt werden.

Laden Sie  [HexagonDialogSkelett.xls](#).

- d) Gehen Sie mit Alt/F11 in die Entwicklungsumgebung und beschreiben Sie die Struktur des Projekts HexagonDialog?
- e) Schreiben Sie analog zu `Quadrat_Click()` eine Prozedur `Hexagon_Click()`, welche die Prozedur `HexagonFlexibel` aufruft. Fügen Sie anschliessend eine Schaltfläche "Hexagon zeichnen" hinzu, so dass die Benutzerin nun zwischen "Quadrat zeichnen" und "Hexagon zeichnen" wählen kann.

✓ [QuadratDialog.xls](#)

✓ [HexagonDialog.xls](#)

Wie entdecke und verbessere ich Fehler ?



## Syntaxfehler

entdeckt VBA bei der *Eingabe*

## Logische Fehler

entdeckt man durch systematisches *Testen*

## Laufzeitfehler

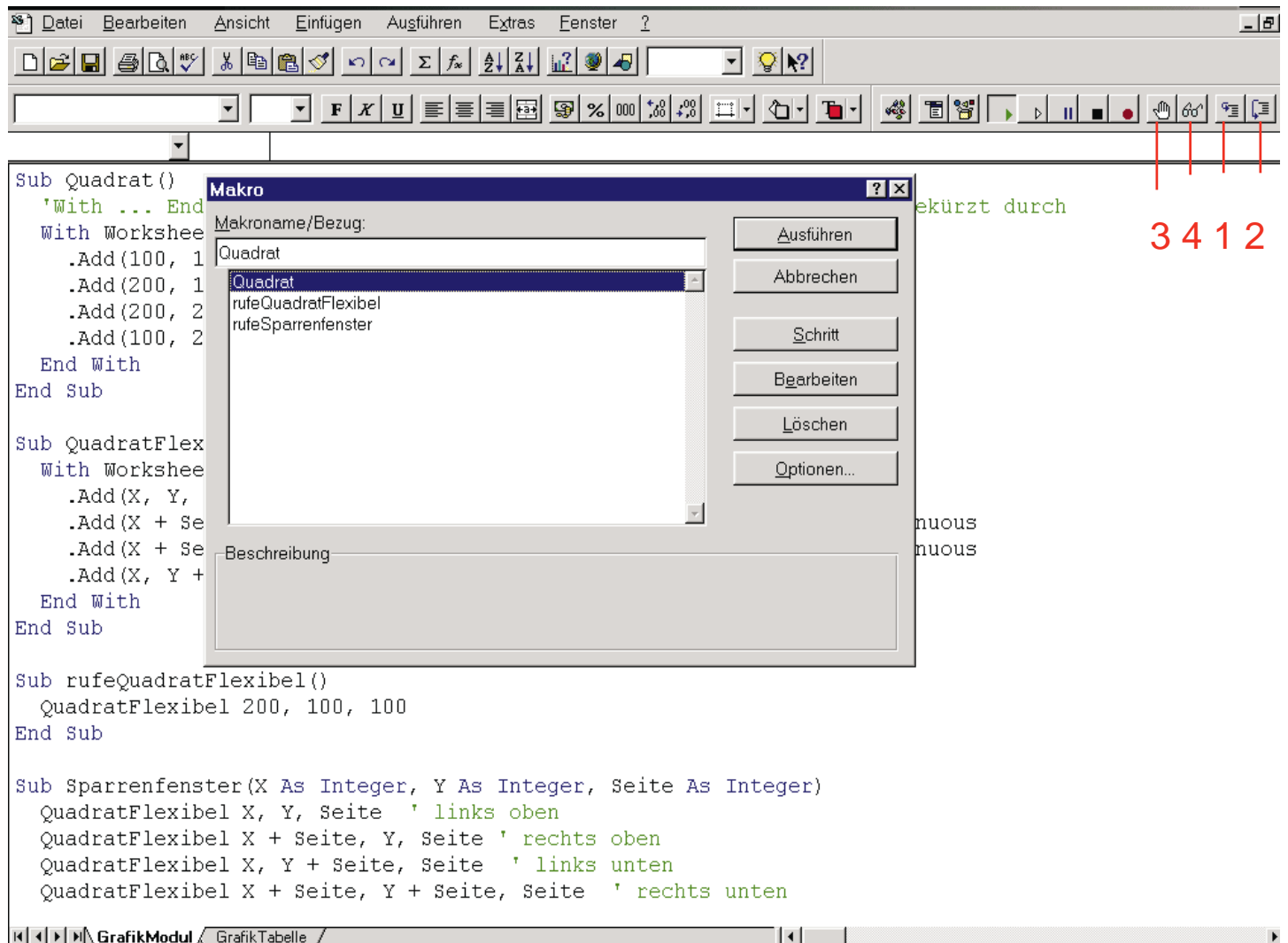
meldet VBA erst bei der *Programmausführung*



Programm **debuggen**

Entdecken und verbessern Sie logische Fehler  
und Laufzeitfehler bereits im **Testfenster**!

# °Überblick



- 1 Unterprogramm unter dem Cursor **zeilenweise** ausführen
- 2 Zeilenweise Ausführung eines **Aufrufs** überspringen
- 3 **Haltepunkt** ein- oder ausschalten
- 4 **Überwachungsausdruck** anfügen oder löschen

# °Schritt - Prozedur - Haltepunkt



—Klicken Sie hier, um Anweisungen und aufgerufene Prozeduren schrittweise auszuführen (links) oder um eine aufgerufene Prozedur zu überspringen (rechts).

Klicken Sie hier, um den markierten Haltepunkt zu löschen.

Wenn Sie die Prozedur ausführen, wird sie am Haltepunkt unterbrochen, und das Testfenster wird angezeigt.

Überwachung

Direkt

Haupt

...

Ausdruck	Wert	Kontext
66 VarAlter	Außerhalb des Kontexts	Modul1.InfoAbrufen
66 VarName	Außerhalb des Kontexts	Modul1.InfoAbrufen

Der Wert jedes Überwachungsausdrucks ändert sich während der schrittweisen Ausführung des Codes.

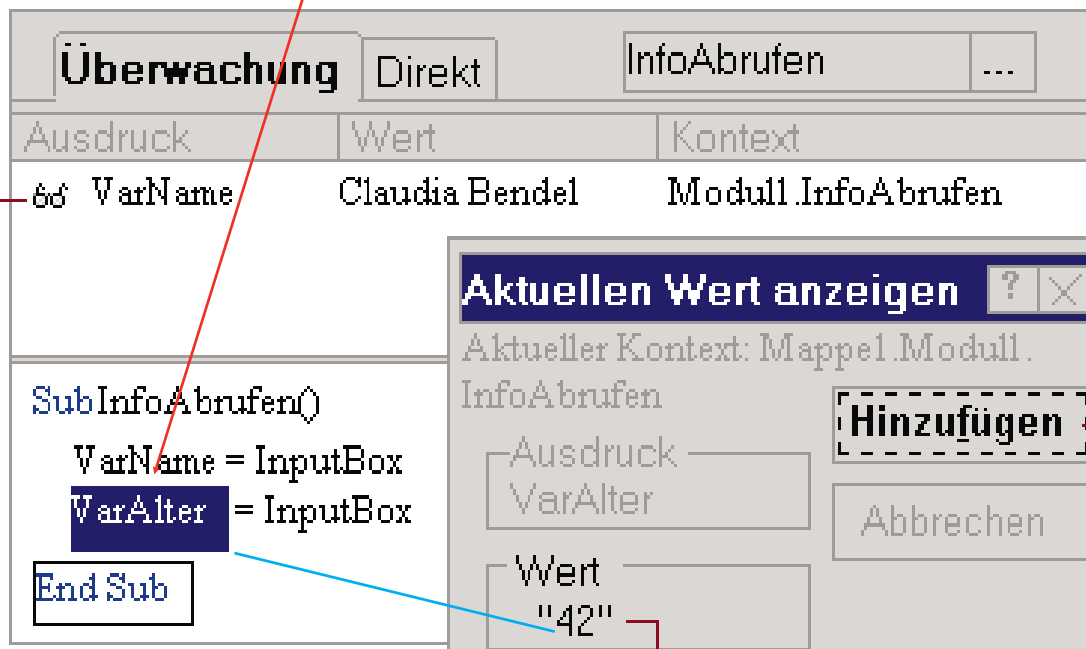
```
Dim VarAlter
Sub Haupt()
  InfoAbrufen
  InfoAnzeigen
End Sub
```

Hilfethemen

Die als nächstes auszuführende Anweisung ist umrahmt.

# °Überwachungsausdrücke

Um einen Überwachungsausdruck zu bearbeiten, doppelklicken Sie auf die entsprechende Zeile im Überwachungsbereich. Sie löschen einen Überwachungsausdruck, indem Sie die Zeile markieren und die ENTF-TASTE drücken.



Klicken Sie hier, um den Ausdruck aus diesem Dialogfeld dem Überwachungsbereich hinzuzufügen.

 Klicken Sie auf die Schaltfläche **Aktuellen Wert anzeigen**, um den Wert eines markierten Ausdrucks zu überprüfen.



## °Prozeduraufrufe rückverfolgen

Der Name der momentan ausgeführten Prozedur wird hier angezeigt.

Klicken Sie im Unterbrechungsmodus hier, um das Dialogfeld **Aufrufe** anzuzeigen.

Um eine Prozedur im Codebereich anzuzeigen, klicken Sie auf den Namen und dann auf **Anzeigen**.

Die momentan ausgeführten Prozeduren werden in der Reihenfolge angezeigt, in der Sie aufgerufen wurden. Die zuletzt aufgerufene Prozedur steht am Anfang der Liste.

The screenshot shows a debugger window with the 'Direkt' tab selected. The call stack on the left lists the following procedures in reverse chronological order of calls: Sub Haupt(), Erste, End Sub, Sub Erste(), Zweite, End Sub, and Sub Zweite(). The 'Aufrufe' (Calls) dialog box is open, displaying the same list of calls. The top entry, 'Mappe1.Modul1.Zweite', is highlighted with a dashed border. The dialog box has an 'Anzeigen' (Show) button and an 'Abbrechen' (Cancel) button. Red arrows point from the text annotations to the corresponding elements in the interface.

---

# Programmbeispiele

---

## Quadrat.xls (Excel Spreadsheet)

Anweisung, Subroutine, Datentyp, Konstante, Variable, Grafikausgabe

## Wort.xls

Funktion, Objekt, Wiederholung, Entscheidung, Textausgabe

## QuadratDialog.xls

Dialog

## ⇒ Gewinnverteilung.xls

Dialog, Objekt

## sucheBinär.xls

Datenfeld, Suchalgorithmen

---

# Betriebliche Anwendung **Gewinnverteilung**

---

Eine **komplexe** Anwendung ...

- über ihre **Datenstrukturen** und
- **Algorithmen** erschliessen

Ein **Formular** ...

- **entwerfen**
- auf dem Tabellenblatt **anzeigen**

°Mit dem **Objektkatalog** ...

- Objekte und Prozeduren finden
- Objekte und Prozeduren einfügen

# Doppelte Buchhaltung

---

## Buchungszyklus

1. Eröffnung BI
2. Laufender Verkehr
3. Abschluss ER und BI

## Konten der Erfolgsverteilung

- **Aktienkapital** (AK) enthält Eigentümeranteile
- **Erfolgsrechnung** (ER) ermittelt Gewinn oder Verlust
- **"Gewinn"verteilung** (GV) verteilt den Erfolg (Gewinn oder Verlust) (Bsp. Gewinnverteilung an Dividende)
- **Reservefonds** enthält nicht ausgeschütteten Gewinn
- **Dividende** enthält Gewinnanteil der Aktionäre
- **Tantième** enthält Gewinnanteil der Verwaltung



Erfolgsverteilung automatisieren

# Beschreibung von Gewinnverteilung.xls

Eigenschaften von Gewinnverteilung.xls		?	×
<u>T</u> itel:	Verbuchung der Gewinnverteilung (GV) der AG		
<u>T</u> hema:	Anwendungsentwicklung mit VBA unter MS Excel		
<u>A</u> utor:	Markus Lusti		
<u>M</u> anager:			
<u>E</u> irma:	WWZ / Wirtschaftsinformatik		
<hr/>			
<u>K</u> ategorie:	Beispiel		
<u>S</u> tichwörter:	Excel VBA GV AG		
<u>K</u> ommentar:	Illustration der Entwicklung mit MS Excel-Objekten unter Visual Basic für Applikationen. Vertiefung der Kapitel "Objekte" und "Benutzerschnittstelle"		

# GV - EVA-Spezifikation

## Eingabe

Aufwand (Unternehmungsaufwand)	15'000.-
Ertrag (Unternehmungsertrag)	20'000.-
Aktienkapital (Grundkapital)	100'000.-
Reserve (Reservefonds)	5'000.-
Tantième in %	10%

## Ausgabe

Buchungssätze des Formats "Soll an Haben, Betrag"

## Verarbeitung (Erfolgsverteilung nach Gesetz und Praxis)

"Aus dem Reingewinn ist jährlich ein Betrag von einem **Zwanzigstel** einem allgemeinen **Reservefonds** zuzuweisen (1), bis dieser Fonds die Höhe von einem **Fünftel** des einbezahlten **Grundkapitals** erreicht hat (2).

Diesem **Reservefonds** sind, auch nachdem er die gesetzliche Höhe erreicht hat, zuzuweisen: ...

3. ein **Zehntel** derjenigen Beträge (3), die aus dem Reingewinn nach der ordentlichen Speisung des **Reservefonds** (1) und nach Bezahlung einer **Dividende** von **fünf vom Hundert** an Aktionäre (4) und **sonstige** Gewinnbeteiligte (5) verteilt werden." (**Gesetz**: OR 671)

Nach der ordentlichen Reservezuweisung (1) erhält der Verwaltungsrat eine **Tantième** in % des Gewinns (5). Nach der zweiten Speisung der **Reserve** (3) sind vom Gewinnrest so viele **ganze Prozent** vom einbezahlten **Grundkapital** wie möglich als **Zusatzdividende** auszuschütten (6). (**Praxis**)

( Die Ziffern in Klammern verweisen auf die folgende Seite )

# GV - Entwurfssprachlicher Algorithmus

ER	Erfolgsrechnung (Konto oder Saldo)
GV	Gewinnverteilung (Konto oder Saldo)
AK	Aktienkapital (Konto oder Saldo)
T%	Tantième in %
<i>kursiv</i>	Buchungssätze

## Algorithmus gemäss OR 671 und Praxis

**Falls** Aufwand < Ertrag **dann** '\_\_\_\_\_ Gewinn  
Gewinn = Ertrag - Aufwand  
*ER an GV, Gewinn*  
**Falls** Reserve < AK / 5 **dann** (2)  
    *GV an Reserve, Gewinn / 20* (1)  
    *GV an Dividende, Aktienkapital / 20* (4)  
    *GV an Tantième, Gewinn \* T% / 100* (5)  
    *GV an Reserve, Gewinn \* T% / 100 / 10* (3)  
    ZusatzDivSatz = GV / 0.011AK  
    Zusatzdividende = AK \* Int (ZusatzDivSatz ) / 100  
    *GV an Dividende, Zusatzdividende* (5,6)  
    *GV an Reserve, Zusatzdividende / 10* (3)  
**Falls** GV > 0 **dann**  
    *GV an Reserve, GV*

**sonst** '\_\_\_\_\_ Verlust  
Verlust = Aufwand - Ertrag  
**Falls** Verlust < Reserve **dann**  
    *Reserve an ER, Verlust*  
**sonst**  
    *Reserve an ER, Reserve*  
    *GV an ER, Verlust - Reserve*

# GV - Aufbau der Arbeitsmappe

## Arbeitsmappe

### ① Tabellenblatt “Dialogblatt”

⇒ Ausgabebereich

⇒ Schaltfläche “Start”

### ② Formular “Eingabeformular” (user form)

⇒ Bezeichnungsfelder

⇒ Textfelder

⇒ Schaltflächen “Verbuchen” und “Abbrechen”

### ③ “Modul” **Verarbeitungsmodul**

⇒ Ereignisprozeduren zu den Schaltflächen

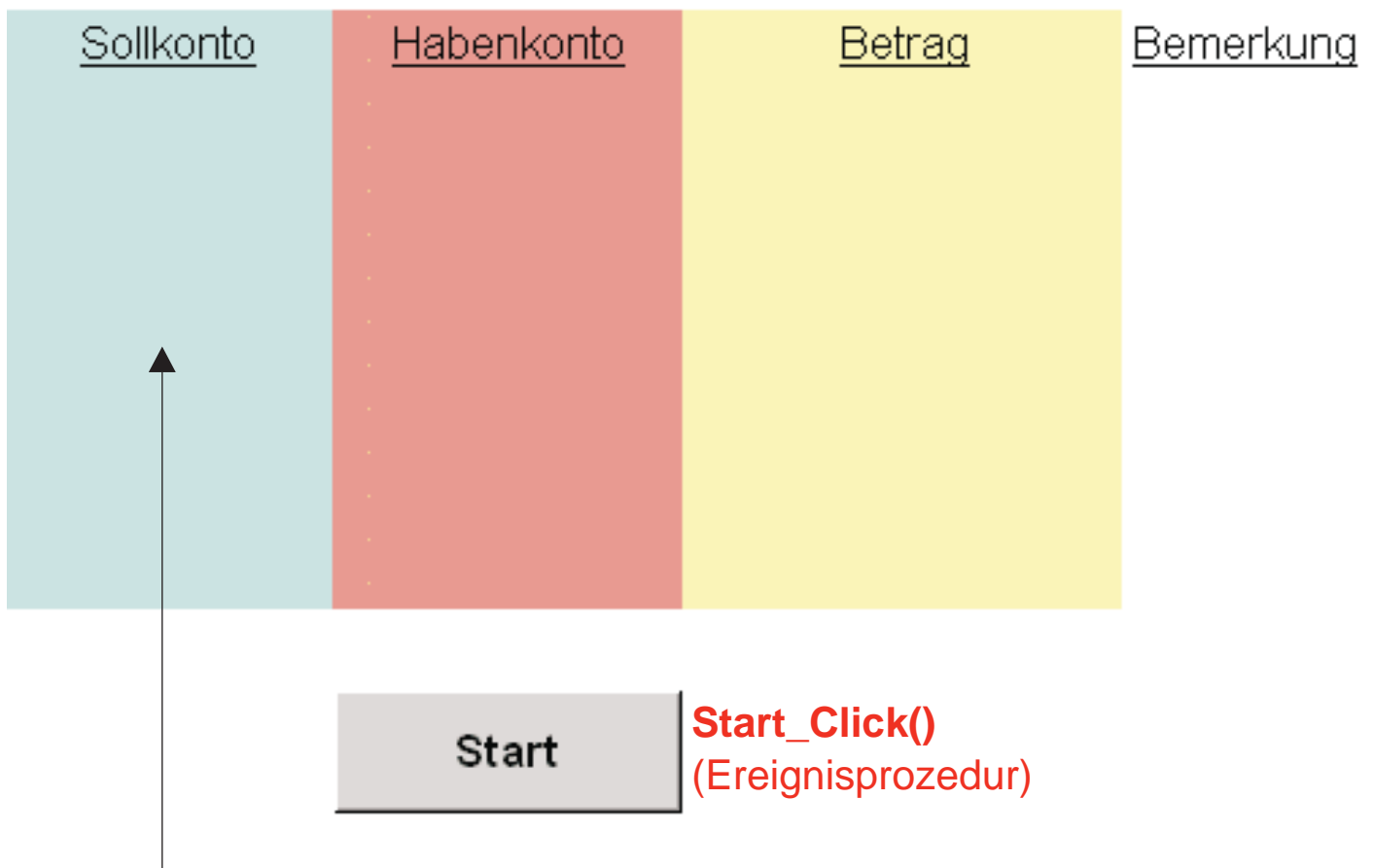
Ein **Formular** ist eine “besondere” Art von Modul



# Ausgabebereich des Dialogblatts

<u>Sollkonto</u>	<u>Habenkonto</u>	<u>Betrag</u>	<u>Bemerkung</u>
ER	GV	SFr. 30,000.00	Gewinn (Reingewinn)
GV	Reserve	SFr. 1,500.00	5% vom Gewinn
GV	Dividende	SFr. 5,000.00	5% vom Aktienkapital von 100000
GV	Tantième	SFr. 3,000.00	10% vom Gewinn
GV	Reserve	SFr. 300.00	10% von der Tantième
GV	Dividende	SFr. 18,000.00	Zusatzdividende = 18% vom Aktienkapital von 100000
GV	Reserve	SFr. 1,800.00	Reserve = 10% von der Zusatzdividende von 18000
GV	Reserve	SFr. 400.00	Gewinnrest

# Dialogblatt entwerfen



Dialogblatt. Range(**Zelle**) . Value = **Wert**  
(Objektwert) (temporäre Variable)

- ✓ Arbeitsmappe beschreiben
- ✓ Auf dem Dialogblatt und Eingabeformular ausgeben
- ⇒ Dialogblatt entwerfen
- Prozeduren, insb. Ereignisprozeduren, programmieren

# Vom Eingabeformular lesen

<u>Sollkonto</u>	<u>Habenkonto</u>	<u>Betrag</u>	<u>Bemerkung</u>
<b>Eingabeformular</b>			
<b>Schaltflächen</b>			

**Ausgabebereich**

Start

**Falldaten eingeben**

Aufwand	<input type="text" value="15000"/>	<b>Verbuchen</b>
Ertrag	<input type="text" value="20000"/>	<b>Abbrechen</b>
Aktienkapital	<input type="text" value="100000"/>	
Reserve	<input type="text" value="5000"/>	
Tantième in %	<input type="text" value="zehn"/>	

**Microsoft Excel**

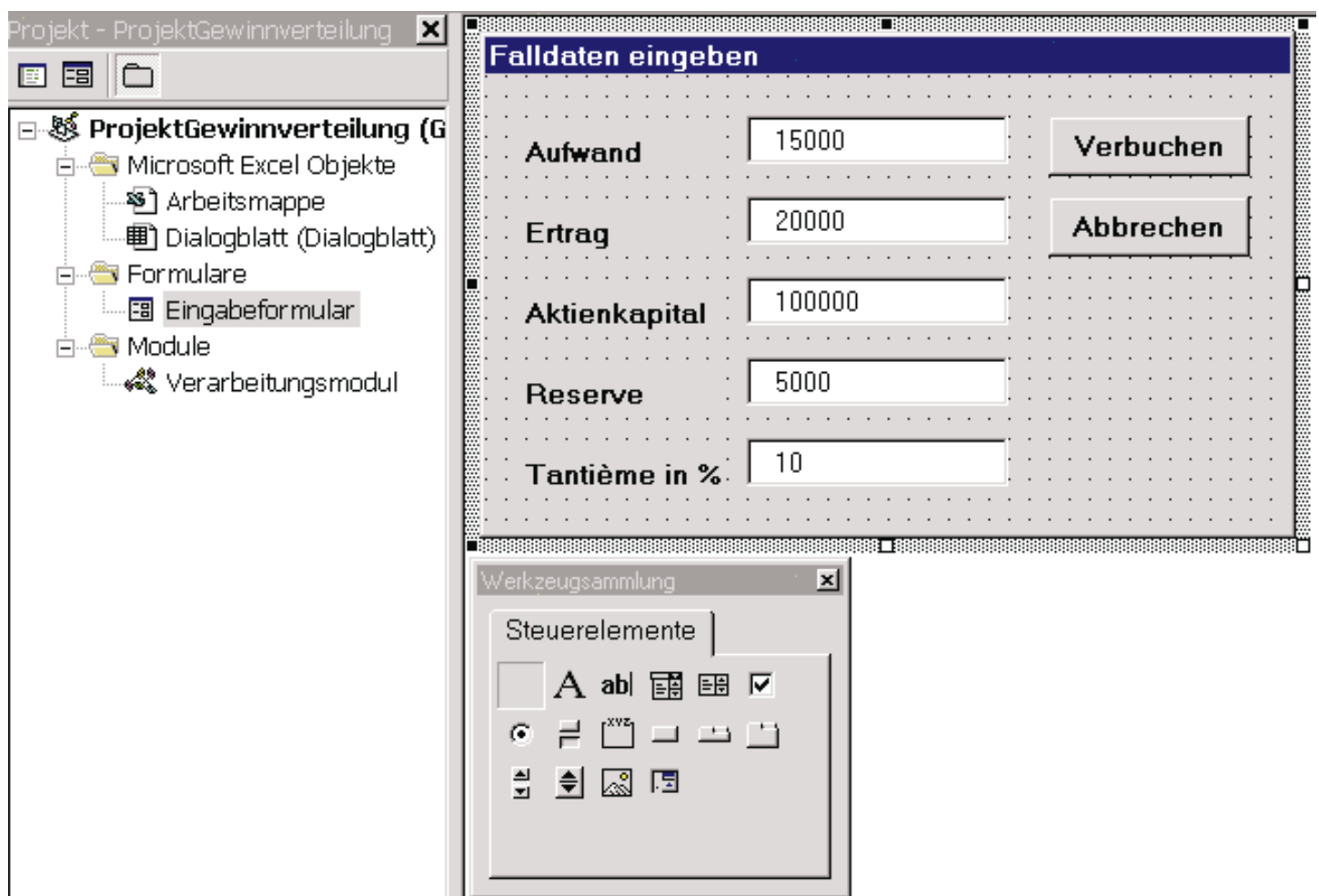
Nur numerische Eingaben möglich

OK

**Bezeichnungs- Textfelder Ausgabefeld felder**

Einem Formular sind Objekte,  
Eigenschaften und Methoden zugeordnet

# Eingabeformular entwerfen



1. **Entwicklersicht** aufrufen (Alt/F11)
2. **Grösse** bestimmen
3. **Steuerelemente** platzieren  
(Bezeichnungsfelder, Textfelder; Schaltflächen)
4. **Ereignisprozeduren** zuordnen  
(abbrechen\_Click, verbuchen\_Click)

Show für Zeigen und Hide für Ausblenden eines Formulars

## ④ Ereignisprozeduren entwurfssprachlich

- ✓ Dialogblatt entwerfen
- ✓ Eingabeformular entwerfen
- ⇒ Prozeduren entwerfen
- ⇒ Prozeduren codieren

Steueranweisung

Excel-Objekte

Ereignisprozeduren

‘Schaltfläche’

### Globale Variablen

--- Variablen, die in allen Prozeduren gelten

Dialogblatt

Eingabeformular

abgebrochen (boolsche Variable)

### Ereignisprozedur Start\_Click()

Ausgabebereich von Dialogblatt löschen

Eingabeformular mit ‘Abbrechen’ und ‘Verbuchen’ anzeigen

Falls nicht abgebrochen dann Buchungssätze

### Ereignisprozedur abbrechen\_Click()

Eingabeformular nach Klick auf ‘Abbrechen’ schliessen

abgebrochen = True

### Ereignisprozedur verbuchen\_Click()

--- Leere und nichtnumerische Eingaben verhindern

Für jedes Feld von Eingabeformular

Falls leer oder nichtnumerisch dann

nach Klick auf ‘Verbuchen’ Formular nicht schliessen

zeige Fehlermeldung

Kontrolle an den Benutzer zurückgeben

## ④ Prozeduren entwurfssprachlich

### Prozedur **Buchungssätze**

Felder vom **Eingabeformular** lesen

(Aufwand, Ertrag, Aktienkapital, Reserve, Tantième in %)

Buchungssätze gemäss OR 671 auf **Dialogblatt** schreiben

**Falls** Gewinn **dann**

**BS** "ER", "GV", Gewinn

**Falls** Reserve < AK **dann** **BS** "GV", "Reserve", Gewinn / 20

**BS** "GV", "Dividende", Gewinn / 20

**BS** "GV", "Reserve", Gewinn / 20 / 10

**BS** "GV", "Tantieme", Gewinn \* Tantiemensatz / 100

**BS** "GV", "Reserve", Gewinn \* Tantiemensatz / 100 / 10

**BS** "GV", "Dividende", AK \* Int(GV / (AK / 100)) / 100

**BS** "GV", "Reserve", AK \* Int(GV / (AK / 100)) / 100 / 10

**Falls** GV < 0 **dann** **BS** "GV", "Reserve", GV

**sonst** ' --- Verlust

**Falls** Verlust < Reserve **dann**

**BS** "Reserve", "ER", Verlust

**sonst**

**BS** "Reserve", "ER", Reserve

**BS** "GV", "ER", Verlust - Reserve

### Prozedur **BS**

Buchungssatz auf **Dialogblatt** schreiben

Konto Gewinnverteilung nachführen

# Dim, Private und Public

## Projekt

### Modul

Unterprogramm

### Modul

Unterprogramm

**Public** macht eine Vereinbarung für das ganze *Projekt* sichtbar

**Private** macht sie nur für das vereinbarende *Modul* sichtbar

**Dim** macht sie nur für das vereinbarende *Unterprogramm* sichtbar

# Ereignisprozedur **Start\_Click**

- ✓ Dialogblatt entwerfen
- ✓ Eingabeformular entwerfen
- ✓ Prozeduren entwerfen
- ⇒ Prozeduren codieren

Steueranweisung, Objekt, Eigenschaft

```
' --- Für Verarbeitungsmodul und Modul Eingabeformular  
Public abgebrochen As Boolean ' True, falls "Abbrechen"
```

```
' --- Nur für Verarbeitungsmodul  
Private Ausgabezeile As Integer  
Private GV As Currency1 ' Konto Gewinnverteilung
```

**Sub Start\_Click()**

```
' Dialogblatt und Eingabeformular initialisieren -----
```

```
GV = 0
```

```
abgebrochen = False
```

```
' Ausgabebereich des Dialogblatts löschen
```

```
Dialogblatt.Range("A4:F12").ClearContents
```

```
' Eingabeformular die Kontrolle übergeben
```

```
Eingabeformular.Show ' laden und anzeigen
```

```
' --- Gefülltes Eingabeformular verarbeiten
```

```
If abgebrochen = False Then
```

```
    Buchungssätze
```

```
    Unload Eingabeformular ' aus Speicher entfernen
```

```
End If
```

```
End Sub
```

<sup>1</sup> Eine Currency-Zahl ist sehr genau (15 Stellen vor und 4 nach dem Komma)



# Ereignisprozeduren des Eingabeformulars

```
' Modul Eingabeformular -----

Private Sub Abbrechen_Click()

' Eingabeformular nach Klick auf "Abbrechen"
entladenabgebrochen = True
Unload Eingabeformular

End Sub

Private Sub Verbuchen_Click()
' Eingabeformular nach Klick auf "Verbuchen" prüfen -----

' --- Aufwand, etc. sind Textfelder des Eingabeformulars
' --- Aufwand = "" bedeutet dasselbe wie Aufwand.Value = ""
If Aufwand = "" Or _
    Ertrag = "" Or _
    Aktienkapital = "" Or _
    Reserve = "" Or _
    TantièmeInProzent = ""
Then
    MsgBox "Keine Leereingaben möglich"
ElseIf1 Not (IsNumeric2(Aufwand) And _
    IsNumeric(Ertrag) And _
    IsNumeric(Aktienkapital) And _
    IsNumeric(Reserve) And _
    IsNumeric(TantièmeInProzent))
Then
    MsgBox "Nur numerische Eingaben möglich"
Else
    Eingabeformular.Hide      'vorübergehend ausblenden
End If

End Sub
```

1 Elself ist ein wahlfreier Bestandteil der If-Anweisung.

2 isNumeric(X) True, falls X eine Zahl

## ④ Verbuchungsprozeduren I

**Private Sub Buchungssätze ()**

```
' Hier Aufwand, Ertrag, Aktienkapital, Reserve, Zusatzdividende,
' TantièmeInProzent, Gewinn, Verlust "As Currency" vereinbaren

' --- Erster Buchungssatz auf der vierten Ausgabezeile
Ausgabezeile = 4

' --- Eingabeformular lesen (CCur heisst Convert to Currency)
Aufwand = CCur(Eingabeformular.Aufwand)
Ertrag = CCur(Eingabeformular.Ertrag)
Aktienkapital = CCur(Eingabeformular.Aktienkapital)
Reserve = CCur(Eingabeformular.Reserve)
TantièmeInProzent = CCur(Eingabeformular.TantièmeInProzent)

' --- Buchungssätze nach OR 671 ausgeben
If Aufwand < Ertrag Then '*** Gewinn
    Gewinn = Ertrag - Aufwand
    BS "ER", "GV", Gewinn, "Reingewinn"
    If Reserve < Aktienkapital/5 Then
        BS "GV", "Reserve", Gewinn/20, "5% vom Gewinn"
    End If
    BS "GV", "Dividende", Gewinn/20, "5% vom Gewinn"
    BS "GV", "Reserve", Gewinn/20/10, "10% von der Grunddiv..."
    BS "GV", "Tantième", Gewinn*TantièmeInProzent/100, ...
    BS "GV", "Reserve", Gewinn*TantièmeInProzent/100/10, ...
    Zusatzdividende=Aktienkapital*Int(GV/(Aktienkapital/100))/100
    BS "GV", "Dividende", Zusatzdividende, ...
    BS "GV", "Reserve", Zusatzdividende / 10, ...
    If GV > 0 Then
        BS "GV", "Reserve", GV, "Gewinnrest"
    End if
Else '*** Verlust
    Verlust = Aufwand - Ertrag
    If Verlust < Reserve Then
        BS "Reserve", "ER", Verlust, "Verlust"
    Else
        BS "Reserve", "ER", Reserve, ...
        BS "GV", "ER", Verlust - Reserve, "Ungedeckter Verlust"
    End If
End If
End Sub
```

## ④ Verbuchungsprozeduren II

```
Private Sub BS(Soll As String, Haben As String, Betrag As  
Currency, Bemerkung As String)
```

```
'--- Einen Buchungssatz auf das "Dialogblatt" schreiben
```

```
With Dialogblatt
```

```
    .Cells(Ausgabezeile, 1) = Soll
```

```
    .Cells(Ausgabezeile, 2) = Haben
```

```
    .Cells(Ausgabezeile, 3) = Betrag
```

```
    .Cells(Ausgabezeile, 4) = Bemerkung
```

```
End With
```

```
Ausgabezeile = Ausgabezeile + 1
```

```
' --- Konto GV (Gewinnverteilung) nachführen
```

```
If Soll = "GV" Then
```

```
    GV = GV - Betrag
```

```
Else
```

```
    If Haben = "GV" Then
```

```
        GV = GV + Betrag
```

```
    End If
```

```
End If
```

```
End Sub
```

# ° Gültigkeitsbereich

## Schlüsselwörter für den Gültigkeitsbereich

**Dim** Aufwand As Currency

**Private** GV As Currency

**Public** abgebrochen As Boolean

**Gültigkeitsbereich** := Bereich, in dem eine Variable, Konstante oder Prozedur für andere Programmelemente (Unterprogramme und Module) sichtbar ist, das heisst, Lese- und Schreibvorgänge ermöglicht

## Gültigkeitsbereich abhängig von ...

### *Vereinbarungsort*

- Vereinbarung zu Beginn des *Moduls*
- Vereinbarung zu Beginn der *Prozedur*

### *Schlüsselwort*

- Dim und Private
- Public

## ° Gültigkeitsbereich und Lebensdauer

Schlüsselwort	Vereinbarungsort	Gültigkeitsbereich
<b>Dim</b>	Prozedurbeginn	Prozedur
<b>Private</b>	Modulbeginn	Modul
<b>Public</b>	Modulbeginn	Projekt

Prozedurvariablen sterben (in der Regel) nach dem Verlassen der Prozedur. Andere Variablen leben während der ganzen Laufzeit

---

## °Wo vereinbare ich Programmelemente?

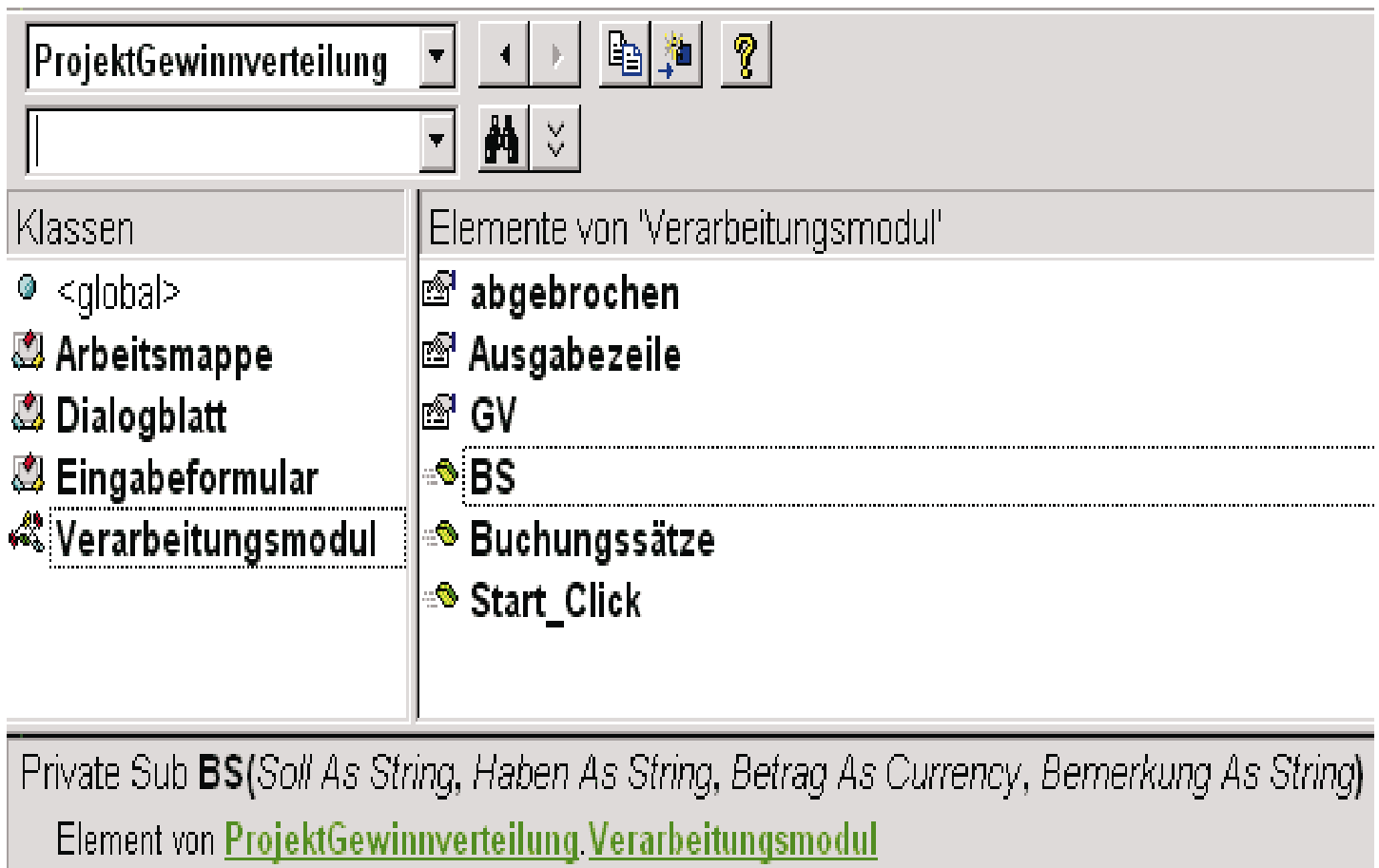
---

- ① Je *kleiner* der Gültigkeitsbereich, desto besser!
  - Kleine Gültigkeitsbereiche erleichtern das **Debugging**
  - Kleine Gültigkeitsbereiche sind **sparsamer**
- ② Übergebe *Variablen möglichst als Argumente*
  - Die explizite Übergabe verbessert die **Leserlichkeit**
  - Die explizite Übergabe verbessert die **Zusammenarbeit**
- ③ Vermeide *gleiche Namen* in verschiedenen Projektteilen
  - Unterscheidbare Namen verbessern die **Leserlichkeit**

### Aufgabe

- a) Vereinbaren Sie GV lokal zur Prozedur Buchungssätze.
- b) Diskutieren Sie die Vor- und Nachteile

## °④ GVModul - Objektkatalog (object browser)



### Aufruf

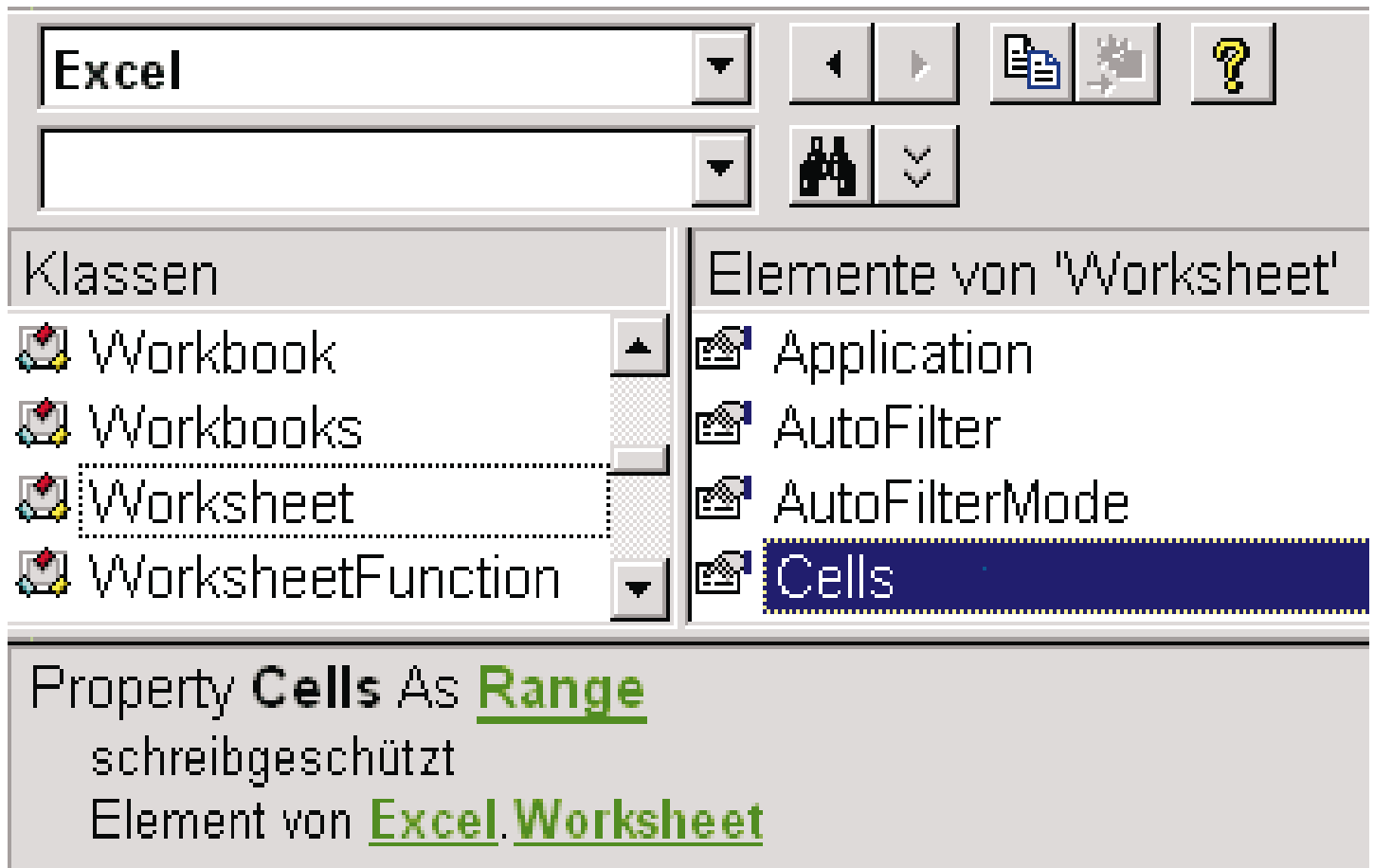
Ansicht/Objektkatalog ..

### Funktion

⇒ Zugriff auf das laufende **Projekt** (zum Beispiel Gewinnverteilung)

⇒ Zugriff auf vordefinierte **Objektmodelle** →

## °Objektkatalog - Bibliotheken



### *Funktion*

**Aufrufschablonen aus Bibliotheken** in den Code einfügen

### *Bibliotheken*

- **Excel-Objekte**
- VBA-Elemente
- übrige Objekte, zum Beispiel aus MS Forms



## Aufgabe 8.1 (🖱️ GEWINNVERTEILUNG)

### Lernziele

- ⇒ Eine komplexe Anwendung über ihre Datenstrukturen und Algorithmen erschliessen
- ⇒ Entwurf und Verarbeitung eines Eingabeformulars verstehen
- ⇒ Mit dem Objektkatalog Objekte und Prozeduren einfügen


### Wiederholungsfragen

1. Was ist die Funktion eines Debuggers?
  - a) Syntaxfehler finden
  - b) logische Fehler entdecken
  - c) Laufzeitfehler verhindern
2. Wie unterscheiden sich `Dim` und `Private`?
  - a) Mit `Dim` deklarierte Variablen sind der gesamten Arbeitsmappe bekannt, mit `Private` deklarierte Variablen sind für andere Prozeduren versteckt.
  - b) `Dim` und `Private` können synonym verwendet werden.
  - c) `Dim` wird innerhalb einer Prozedur verwendet und die Variablen sind nur dieser Prozedur bekannt. `Private` definiert Variablen für das gesamte Modulblatt, ausserhalb dieses Modulblattes sind die Variablen jedoch unsichtbar.
3. Betrachten Sie den [Algorithmus zur Gewinnverteilung](#) nach OR 671 und füllen Sie die folgende Gewinnverteilungs-Tabelle aus. Die Ausgangsdaten seien:

Aufwand = 14000  
 Ertrag = 20000  
 Aktienkapital (Grundkapital) = 100000  
 Reserve = 5000  
 Tantiemensatz = 10.5 %.

<i>Sollkonto</i>	<i>Habenkonto</i>	<i>Betrag</i>	<i>Bemerkung</i>
ER	GV	.....	Reingewinn
GV	Reserve	.....	5% vom Gewinn
GV	Dividende	.....	5% vom Gewinn
GV	Reserve	.....	10% von der Grunddividende
GV	Tantième	.....	10.5% vom Gewinn
GV	Reserve	.....	10% von der Tantième
GV	Dividende	.....	? vom Aktienkapital
GV	Reserve	.....	10% von der Zusatzdividende
GV	Reserve	.....	Gewinnrest

## Vertiefungsfragen

Laden Sie  [GewinnverteilungExperimentSkelett.xls](#). Die Bedeutung neuer VBA-Elemente und Excel-Objekte, -Methoden und -Eigenschaften erfahren Sie, wenn Sie den Cursor auf das entsprechende Wort setzen und F1 drücken.

a) Aus welchen Modulen, Subroutinen und Funktionen besteht das Programm?

.....

.....

.....

b) Welche globalen Variablen werden benutzt?

.....

c) Wann wird welche benutzerdefinierte Prozedur von wo aufgerufen. Gehen Sie dazu die folgenden Eingabeszenarien durch:

- “Start” und “Verbuchen” drücken, ohne die voreingestellten Werte des Eingabeformulars zu ändern

.....

- “Start” und “Verbuchen” drücken. Für Aufwand eine Zeichenkette statt eine Zahl eingeben.

.....

d) Vervollständigen Sie folgende Tabelle:

<i>Variable</i>	<i>wird in folgenden Prozeduren manipuliert</i>
Eingabeformular	..... .....
Dialogblatt	.....
abgebrochen	.....
Ausgabezeile	.....
GV	.....

In welchem Szenario von b) und in welcher Prozedur wird die Variable `abgebrochen` auf `False` gesetzt? Wann wird sie auf `True` gesetzt?

.....

.....

e) Beantworten Sie die folgenden Fragen:

- Wozu dient das Zeichen `_` am Ende einer Zeile?

.....

- Weshalb werden die Variablen `Ausgabezeile` und `abgebrochen` ausserhalb der Prozeduren vereinbart?

.....  
 .....  
 .....

- Was bedeutet `Private`?

.....

- Weshalb tragen zwei der vier Prozeduren das Anhängsel (Suffix) `_Click`?

.....

- Was bedeutet die Adresse `A4 : F12`?

.....

f) Wählen Sie in der Programmierumgebung den Menüpunkt “Ansicht/Symbolleisten/Testen”. Interpretieren Sie die Prozedur `BS`, indem Sie den Debugger zur Beantwortung der folgenden Fragen heranziehen:

- Klicken Sie die Zeile `Sub BS` und dann den Menüpunkt “Testen/Haltepunkt ein/aus” (F9).
- Markieren Sie die Variablen `Soll` und wählen Sie “Testen/Aktuellen Wert anzeigen” (Umschalt/F9) und im anschliessenden Eingabeformular “Hinzufügen”.
- Verfahren Sie mit `Haben`, `GV` und `Betrag` analog.
- Drücken Sie die Schaltfläche “Start” erneut (Benutzen Sie dabei die ursprünglichen Voreinstellungen)
- Klicken Sie im geöffneten Testfenster “Überwachen”.
- Gehen Sie die Prozedur im Einzelschrittmodus durch (F8).

g) Implementieren Sie die folgenden Ergänzungen:

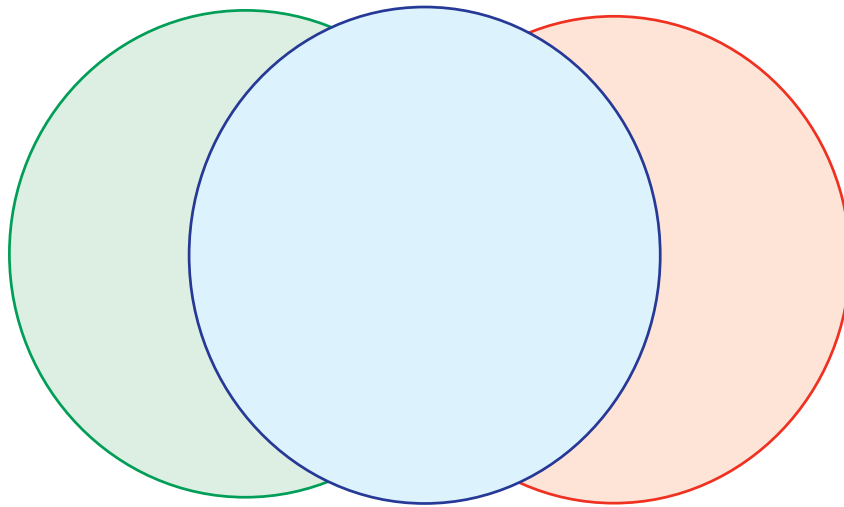
- Schreiben Sie den Buchungsbetrag fett.
- Verhindern Sie Formulareingaben von 0.

✓ [GewinnverteilungExperiment.xls](#)

---

## °Excel, VBA und VB überschneiden sich

---



### Excel

Applikationsspaket zum Lösen von **Tabellenkalkulations-** Problemen

### Visual Basic for Applications (“... für Anwendungspakete”)

Programmierungsumgebung zur Erstellung von Anwendungen mit Excel, Word, Powerpoint, Project und Access

### Visual Basic

zur Erstellung **beliebiger** Anwendungen

---

## °Excel aus Word aufrufen

---

**Sub** **StarteExcel**

Dim **Excel** as Excel.Application

Set **Excel** = CreateObject("Excel.Application")

**Excel**.Visible = True

**End Sub**

# °Excel- und VBA-Objekte mischen

VBA Excel-Objekt VBA wählen

Option Explicit

```
Private Zeile As Integer  
Private abgebrochen As Boolean  
  
Sub initialisieren_NachKlick()  
    ' --- Letzte Programmausgabe  
    Worksheets("GVTabelle")  
    ' --- Nach Klick auf "P  
    DialogSheets("GVDialog")  
    ' --- Nach ESC und Klick  
    DialogSheets("GVDialog")  
    abgebrochen = False  
    ' --- Nach Klick auf "V
```

Objektkatalog

Bibliotheken/Arbeitsmappen: VBA

Objekte/Module: Constants, Conversion, DateTime, FileSystem, Information, Interaction, Math, Strings

Methoden/Eigenschaften: AppActivate, Beep, CreateObject, DoEvents, GetObject, InputBox, MacScript, **MsgBox**, SendKeys, Shell

MsgBox(Prompt, Buttons, Title, HelpFile, Context)

Displays a message box and returns the user's selection value

Visual Basic-Sprachverzeichnis

Hilfethemen Zurück Optionen

### MsgBox-Funktion

[Siehe auch](#) [Beispiel](#) [Zusatzinfo](#)

Zeigt eine Meldung in einem Dialogfeld an und wartet auf die Auswahl einer Schaltfläche. **MsgBox** liefert einen Wert, der die ausgewählte Schaltfläche bestimmt.

**Syntax**

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

**Elemente**

Die Funktion **MsgBox** verwendet die folgenden **benannten Argumente**:

Argument	Beschreibung
<b>prompt</b>	<b>Zeichenfolgenausdruck</b> , der als Meldung im Dialogfeld erscheint. Die Maximallänge von <b>prompt</b> ist etwa 1024 Zeichen, abhängig von der Breite der verwendeten Zeichen. Wenn <b>prompt</b> aus mehreren Zeilen besteht, müssen Sie die Zeilen mit einem Wagenrücklaufzeichen (Zeichencode 13) oder einer Folge aus



## Verbreitete Anwendungen der Praxis sind ...

- Textverarbeitung an MS Word
- Tabellenkalkulation an MS Excel
- Datenbankverwaltung an MS Access → Informatik II und Oberstufe

## aber es gibt andere und neue kommen hinzu ...

- Entscheidungsunterstützende Systeme → Oberstufe
- Betriebliche Multimedia-Anwendungen → Oberstufe
- Groupware, Workflow Management
- Business Process Reengineering
- Internet- bzw. Intranet-spezifische Anwendungen
- ...

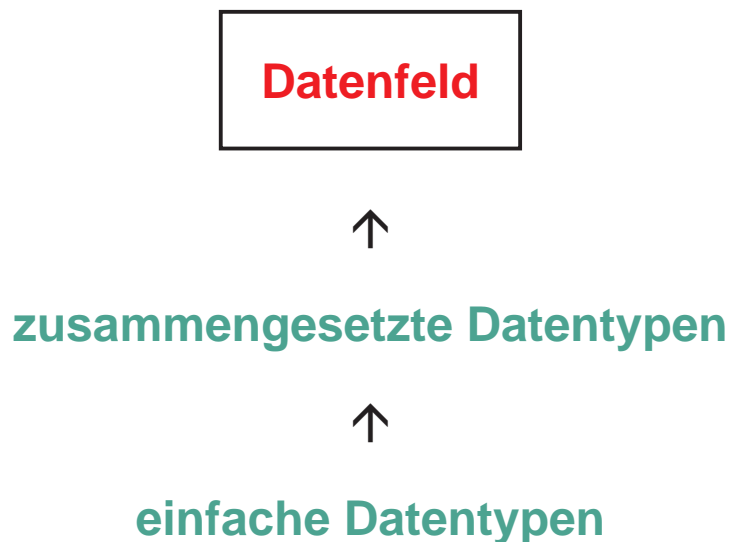


# Überblick Datenfelder

---

- ✓ 1 Einführung (Modularisierung, Subroutine)
- ✓ 2 Datentypen und Ablaufstrukturen
- ✓ 3 Objekte und Funktionen
- ✓ 4 Benutzerschnittstelle

 [Web Quiz](#) “Benutzerschnittstelle”



# Zusammengesetzte Datentypen

## ① **einfache** Datentypen

- Integer (z.B. 12)
- Double (z.B. 3.14)
- ...

## ② **zusammengesetzte** Datentypen

- **Zeichenfolge** (string)  
*"Paul"*
- **Datensatz** → Informatik II / Datenbanken  
*Fritz Meier*  
*Flaschenstrasse 41*  
*Mörschwil*
- **Datenfeld** (engl. array)  
*[ 3, 5, 1, 7, 11, 0 ]*

### Überblick über die VBA-Datentypen



*Datenfeld = Datenfolge mit darauf definierten Operationen*

- Datenfeld lesen →
- Datenfeld schreiben →

# Datenfeld

**Datenfeld** (engl. array) := benannte Folge von *Elementen* des *gleichen* Datentyps, auf die unter dem gleichen **Namen**, aber verschiedenen **Indizes** zugegriffen werden kann

Name	Läufer(1)	Läufer(2)	Läufer(3)	Läufer(4)	Läufer(5)
Inhalt	"Anna"	"Bruno"	"Dora"	"Franz"	"Karin"

Name: Läufer

Grösse (Dimension): 5

Untergrenze: 1

Obergrenze: 5

## Vorteile

- Ganzes *und* Teile verarbeitbar
- Teile über eine Schleife *einfach* zugreifbar

# Datenfeld vereinbaren

- ✓ Datenfeld vereinbaren
- ✓ Datenfeld lesen
- ⇒ Datenfeld schreiben
- ⇒ Datenfeld durchlaufen
- ⇒ Datenfeld suchen

Dim Läufer ( 1 To 5 ) As String

**Dim Name ( Untergrenze To Obergrenze ) As ...**

---

# Datenfeld lesen

---

## Datenfolge

Läufer(1)	Läufer(2)	Läufer(3)	Läufer(4)	Läufer(5)
“Anna”	“Bruno”	“Dora”	“Franz”	“Karin”

## Element lesen

`Tmp = Läufer(5)`

liest den Wert des fünften Elements von Läufer (“Karin”) und weist ihn der Variablen Tmp zu

---

# Datenfeld schreiben

---

## Datenfolge

Läufer(1)	Läufer(2)	Läufer(3)	Läufer(4)	Läufer(5)
"Anna"	"Bruno"	"Dora"	"Franz"	"Karin"

## Element schreiben

`Läufer(5) = "Pauline"`

ersetzt "Karin" durch "Pauline"

---

# Datenfeld mit einer Zählschleife durchlaufen

---

```
Dim Läufer(1 To 5) As String  
Dim i As Integer
```

```
For i = 1 To 5  
    Läufer(i) = "Fritz"  
Next i
```

## Syntax

```
Dim Index As Integer  
For Index = Untergrenze To Obergrenze  
    Anweisungen  
Next Index
```

## Aufgabe 9.1 (Zählschleife)

Transformieren Sie die folgende Zählschleife in eine gleichwertige While-Schleife:

```
Dim Läufer(1 To 5) As String  
Dim i As Integer
```

### ① *Zählschleife*

```
For i = 1 To 5  
    Läufer(i) = "Fritz"  
Next i
```



### ② *Wiederholung mit Ausführungsbedingung*

```
.....  
Do While .....  
    .....  
    .....  
Loop
```



## Aufgabe 9.2 (Geometrische Folge)

Speichern Sie ein Feld  $F$ , das eine geometrische Folge mit den folgenden Merkmalen enthält:

Anfangsglied            **2**  
Quotient                 $1/2$   
untere Grenze        **1**  
obere Grenze         **10**

F(1)	F(2)	F(3)	F(4)	F(5)	F(6)	F(7)	F(8)	F(9)	F(10)
2	1	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256

```
Sub Geometrische_Folge;
```

```
Dim F(1 To 10) As Double
```

```
Dim i As Integer
```

```
.....  
.....  
.....  
.....
```

```
Sub End
```

# Suche im Telefonbuch

Suche	Suchbeispiel	Bedingung	Effizienz
direkt	Seite 143	Seiten numeriert	sehr schnell
sequentiell	061 / 422 13 07	keine	langsam
binär	Bohnenblust Fritz	Einträge sortiert	schnell

## ① Direkt suchen

---

*Gesucht*

Wie heisst der 3. Läufer ?

*Lösung*

Name = Läufer(3)

**Direkt suchen** := Feldelement mit  
einem gegebenen **Index** bestimmen

---

## ② Sequentiell suchen

---

### *Gesucht*

Welches ist die **Nummer** der Läuferin "Dora" ?

### *Lösung*

Vergleiche *der Reihe nach* alle Elemente, bis die Nummer gefunden wird

```
For Nummer = 1 To 5
    If Läufer(Nummer) = "Dora" Then
        MsgBox "Nummer = " & Nummer
        Exit For
    End If
Next Nummer
```

**Sequentiell suchen** := jedes Element mit dem Suchwert vergleichen, bis die Suche Erfolg hat

For 167

---

## Aufgabe 9.4 (🖱️ MAXIMUM)

---

### *Lernziele*

- ⇒ sequentielle Suche auf einem Datenfeld
- ⇒ sequentielle Suche auf einem Tabellenbereich

### *Wiederholungsfragen*

1. Was ist ein Datenfeld?

- a) ein Bereich von Integer-Werten
- b) ein Bereich von Elementen verschiedener Datentypen, auf die unter dem gleichen Namen und verschiedenen Indizes zugegriffen werden kann
- c) eine benannte Folge von Elementen des gleichen Datentyps, auf die unter dem gleichen Namen und verschiedenen Indizes zugegriffen werden kann.

2. Was vereinbart `Dim Läufer(1 To 7) As String`?

- a) ein Datenfeld mit den String-Indizes 1 bis 7
- b) einen String mit einem bis maximal sieben Zeichen
- c) ein Datenfeld des Typs String mit den Indizes 1 bis 7

3. Die sequentielle Suche...

- a) durchsucht alle Datensätze nacheinander.
- b) ist schneller als die binäre Suche.
- c) funktioniert nur für Datenfelder vom Typ String.

## Vertiefungsfragen

a) Betrachten Sie das folgende Programmskelett:

```
Private Liste(1 To 10) As Integer


Sub sucheMaximum()

    Liste(1) = 30567
    Liste(2) = 30567
    Liste(3) = 8457
    Liste(4) = 23532
    Liste(5) = 30945
    Liste(6) = 968
    Liste(7) = 2485
    Liste(8) = 30564
    Liste(9) = 30755
    Liste(10) = 4375

    'Grösstes Element von Liste suchen
    MsgBox Prompt:=Maximum(), Title:="Maximum"
End Sub
```

b) Welchen Datentyp muss Maximum( ) zurückgeben?

c) Programmieren Sie die fehlende Funktion Maximum( ), indem Sie  [MaximumSkelett.xls](#) vervollständigen.

d) Programmieren Sie eine Prozedur, die mit einer geschachtelten Zählschleife den grössten Wert eines *Zellbereichs* ermittelt. Vervollständigen Sie dazu  [MaximumzelleSkelett.xls](#).

## Zusatzaufgabe

Entwickeln Sie ein analoges Programm Minimum.xls.

✓ [Maximum.xls](#)

✓ [Maximumzelle.xls](#).

---

# Programmbeispiele

---

## Quadrat.xls (Excel Spreadsheet)

Anweisung, Subroutine, Datentyp, Konstante, Variable, Grafikausgabe

## Wort.xls

Funktion, Objekt, Wiederholung, Entscheidung, Textausgabe

## QuadratDialog.xls

Dialog

## Gewinnverteilung.xls

Dialog, Objekt

## ⇒ sucheBinär.xls

Datenfeld, Suchalgorithmen

---

### ③ Binär suchen

---

Wer im Telefonbuch blättert, ...

- ✓ springt in die **Mitte**,
- ✓ schaut, ob der Suchwert in der **linken** oder **rechten** Hälfte liegt,
- ✓ setzt die Schritte 1 und 2 in der gefundenen Hälfte fort, bis er nach weiteren Halbierungen die Seite mit dem Suchwert gefunden hat.

**Binär suchen** := sortiertes Datenfeld solange halbieren, bis das gesuchte Element gefunden worden ist



## a) Binärsuche **entwerfen** I

### Suche “Paul” binär im folgenden Datenfeld!

① Aufrufbeispiel: **sucheBinär**(1, 7, “Paul”, 4)

② Datenfeldbeispiel “Feld” und Programmablauf

Nummer	1	2	3	4	5	6	7
Feld(Nr)	ANNA	BRUNO	DORA	FRANZ	KARIN	NORA	PAUL
1. Schritt	l			M			r
2. Schritt					l	M	r
3. Schritt							l,M,r

l links      Beginn des zu durchsuchenden Feldbereichs  
M Mitte      mittlere Zelle des zu durchsuchenden Feldbereichs  
r rechts      Ende zu durchsuchenden Feldbereichs

## a) Binärsuche **entwerfen II**

### ③ *Entwurf des Unterprogramms*

```
sucheBinär( links, rechts, Name, Mitte )  
  Mitte = (links + rechts) \ 2  
  Falls links <= Mitte  
    Falls Name = Feld (Mitte)  
      Name gefunden  
  sonst  
    Falls Name < Feld (Mitte)  
      suche_binär( links, Mitte-1, Name, Mitte )  
    sonst falls Name > Feld (Mitte)  
      suche_binär( Mitte+1, rechts, Name, Mitte )  
  sonst  
    Name nicht gefunden
```

*\ dividiert ganzzahlig*

Bsp.  $5 \setminus 2$  ergibt 2 (und nicht 2.5)

## b) Binärsuche als Funktion aufrufen

```
Private Läufer(1 To 7) As String
Sub sucheBinaer
    ` Datenfeldbeispiel vereinbaren und initialisieren
Dim Nr As Integer, Suchname As String
    Läufer(1) = "ANNA"
    Läufer(2) = "BRUNO"
    Läufer(3) = "DORA"
    Läufer(4) = "FRANZ"
    Läufer(5) = "KARIN"
    Läufer(6) = "NORA"
    Läufer(7) = "PAUL"
    Suchname = InputBox("Name in Grossbuchstaben: ")
If gefunden(1, 7, Suchname, Nr) = True Then
        MsgBox "Gesuchte Nummer = " & Nr
ElseIf Suchname <> ""    ` kein Klick auf "Abbrechen"
        MsgBox "Kein Element mit diesem Wert"
End If
End Sub
```

## c) Binärsuche implementieren

```
Function gefunden( _  
links As Integer, _  
rechts As Integer, _  
Läufername As String, _  
Mitte As Integer) _ ' Eingabe- und Ausgabewert  
As Boolean  
  
Mitte = (links + rechts) \ 2  
If links <= Mitte Then  
    If Läufername = Läufer(Mitte) Then  
        gefunden = True  
    Else  
        If Läufername < Läufer(Mitte) Then  
            gefunden = _  
            gefunden(links, Mitte-1, Läufername, Mitte)  
            ' . Argument gibt Ergebnis zurück (by ref)  
        Else  
            gefunden = _  
            gefunden(Mitte+1, rechts, Läufername, Mitte)  
    Else  
        gefunden = False  
End If  
  
End Function
```

Das vierte Argument der Funktion gefunden enthält nach dem letzten (erfolgreichen) Aufruf die gesuchte Nummer. Es ist wie alle Argumente ein Referenzargument (by ref); der Eingabewert ("Mitte") kann deshalb vom letzten (erfolgreichen) Aufruf abgeändert werden.

Jeder rekursive Aufruf setzt voraus, dass die Zwischenergebnisse des letzten Aufrufs von gefunden aufbewahrt werden.

---

## Aufgabe 9.5 (☞ SUCHEITERATIVBINÄR)

---

### *Lernziele*

- ⇒ Binäre Suche
- ⇒ Rekursion

### *Wiederholungsfragen*

#### 1. Die binäre Suche ...

- a) halbiert das Datenfeld solange, bis das gesuchte Element gefunden ist.
- b) betrachtet hintereinander Element für Element, bis das gesuchte Element gefunden ist.
- c) durchsucht ein Datenfeld mit einer For-Next-Schleife.

#### 2. Der Operator \ ...

- a) berechnet den Rest einer ganzzahligen Division (Bsp.  $5 \setminus 2 = 1$ ).
- b) dividiert ganzzahlig (Bsp.  $5 \setminus 2 = 2$ ).
- c) dividiert (Bsp.  $5 \setminus 2 = 2.4$ ).

#### 3. Wodurch unterscheiden sich sequentielles und binäres Suchen?

- a) Die sequentielle Suche ist immer langsamer als die binäre Suche.
- b) Für die binäre Suche muss das Datenfeld sortiert sein, für die sequentielle Suche nicht.
- c) Für die sequentielle Suche muss das Datenfeld sortiert sein, für die binäre Suche nicht.

### *Vertiefungsfragen*


- a) Suchen Sie in den folgenden *Tabellen* binär (vgl. Tabelle des Skriptabschnitts 9.2):

Gesucht sei EMIL:


Nummer	1	2	3	4	5	6	7
Feld(Nr)	ANNA	BERTA	EMIL	PETER	RICHARD	TITUS	VOLKER
1. Schritt							
2. Schritt							
3. Schritt							

Gesucht sei SAMUEL:

Nummer	1	2	3	4	5	6	7
Feld(Nr)	ANNA	BERTA	EMIL	PETER	RICHARD	TITUS	VOLKER
1. Schritt							
2. Schritt							
3. Schritt							
4. Schritt							

- b) Beschreiben Sie den *Algorithmus* der binären Suche in eigenen Worten und Schritt für Schritt. Vergleichen Sie Ihre Formulierungen mit dem Algorithmus von Skriptabschnitt 9.2.
- c) Die binäre Suche lässt sich nicht nur rekursiv, sondern auch nicht-rekursiv, und zwar in Form einer Schleife (Iteration) programmieren. Vervollständigen Sie  [SucheIterativBinärSkelett.xls](#) und beantworten Sie die folgende Fragen:
- Wie oft muss die Schleife durchlaufen werden?
  - Welche Schleife ist geeignet und warum (For-Next, For-Each, Do-While)?
  - Wann ist die Suche fertig, das heisst, wann kann die Schleife verlassen werden?
  - Welche Variablen benötigen Sie?

## Zusatzaufgabe

Nach einem Klick auf die Schaltfläche “Quadrat zeichnen” des Beispiels  [QuadratDialogEinfach.xls](#) soll eine Passwort-Prozedur garantieren, dass nur Berechtigte ein Quadrat zeichnen können. Die Berechtigung soll durch eine binäre Suche in einem Datenfeld von Passwörtern nachgeschlagen werden.

Implementieren Sie Ihre Lösung als Arbeitsmappe “QuadratDialog-Passwort.xls”.

✓ [SucheIterativBinär.xls](#)

 [Web Quiz](#) Datenfelder

 [Musterklausur, 1 . Teil](#)





# Stichwortverzeichnis

!

= 49

## A

Add 34, 37

And 122

Application 76, 158

array 163

As 48, 52 - 54, 164

Assemblersprachen 23

## B

Boolean 44, 90

Border 31 - 33

ByVal 93

## C

Calculate 74, 83

Cells 87 - 89

CInt 121

ColorIndex 83

Columns 74, 83

Const	48
Currency	44
<b>D</b>	
Datenfeld	162 - 165
Dim	48, 148 - 149
Do While	63, 68
Double	44
<b>E</b>	
Each	79
Else	63
Exit For	172
<b>F</b>	
Font	83
For	167
Formular	115
Formulare	115
<b>H</b>	
Hochsprachen	23
<b>I</b>	
If	63, 68
InputBox	67, 107, 110, 178

InStr	90
Integer	44
<b>L</b>	
Len	90
Lines	31 - 33
Loop	63
<b>M</b>	
MakroSkelett.xls	58
Maschinensprachen	23
Modul	115
MsgBox	77 - 78, 107, 172, 177
<b>N</b>	
Name	77
Not	122
<b>O</b>	
Option Explicit	46
Or	122
<b>P</b>	
Private	148 - 149
Programmiersprachen	23
Projekt	115

Projekt-Explorer	115
Public	148 - 149
<b>R</b>	
Range	76, 83
Right	90
<b>S</b>	
Selection	83
Softwareentwicklung	10 - 11
String	44
Sub	30, 34 - 36
Suchalgorithmen	170
sucheIterativBinärSkelett.xls	182
summe	83
SUMME	83
<b>T</b>	
Then	63
To	164
<b>V</b>	
Value	76
Visible	158

# W

<u>With</u>	<u>34, 37</u>
<u>Workbooks</u>	<u>76</u>
<u>Worksheet</u>	<u>74</u>
<u>Worksheets</u>	<u>31 - 32, 74, 76 - 77, 83, 88</u>