

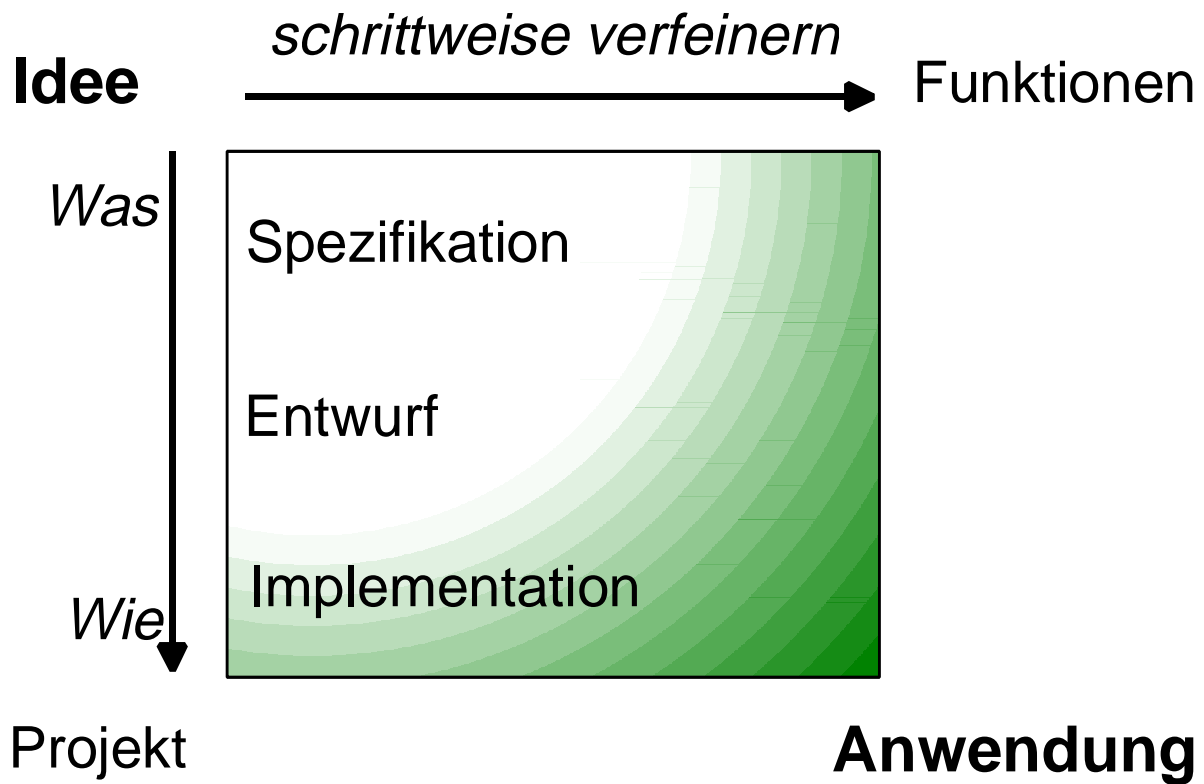
Andreas Born

Anwendungsentwicklung

2. Teil

Programmieren mit Objekten
unter MS Excel und VBA

Anwendungsentwicklung



andere Blickrichtungen

- Eingabe - Verarbeitung - Ausgabe
- Daten - Ablauf

Einordnung

Deklarativ Anwendungen entwickeln

- 📖 Tabellenkalkulation
- 📖 Elementare Was-Wenn-Analyse
- 📖 Lineare Optimierung

Prozedural Anwendungen entwickeln

- 📖 Datentypen und Ablaufstrukturen
- 📖 Benutzerschnittstelle
- 📖 Datenfeld
- ⇒ 2.4. Elemente von Programmiersprachen
- ⇒ 9.4. Objekthierarchie und Auflistung
- ⇒ 16.4. Objektvariablen
(*Vorlesung am Do, keine Übungen*)
- ⇒ 23.4. benutzerdefinierte Objektklassen
- ⇒ 30.4. (*keine Vorlesung, nur Übungen*)
- ⇒ 7.5. Algorithmen, Datenstrukturen
- ⇒ 14.5. Bäume

Datenbankanwendungen entwickeln

- 📖 Dateiverwaltung
- 📖 Datenentwurf
- 📖 Datenbankverwaltung
- 📖 Anwendungsentwicklung

Organisation

Vorlesungen und Übungen

- Vorlesung Montag (oder Donnerstag)
 - Sprechstunden
- Übungen
 - verteilt auf die ganze Woche
 - Tutoren
 - WISI
 - Vorbereitung

Veranstaltungsunterlagen

- Merkblätter → Web, Anschlagbrett 3.Stock
- Druckerkärtchen → Sekretariat
- Skript → Sekretariat
- Beispiele → Web
- Übungen → Web, Laborserver
- Foliensammlung → Web

WebSite

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

<http://www.wwz.unibas.ch/wi/lehre/i2/mbi2.html>

Vorlesung: Elemente von Programmiersprachen

Entwicklungswerkzeuge

Modularisierung

Daten

Konstante, Variable

Datentyp

ausführbare Anweisungen

Prozedur

Sequenz

Entscheidung

Wiederholung

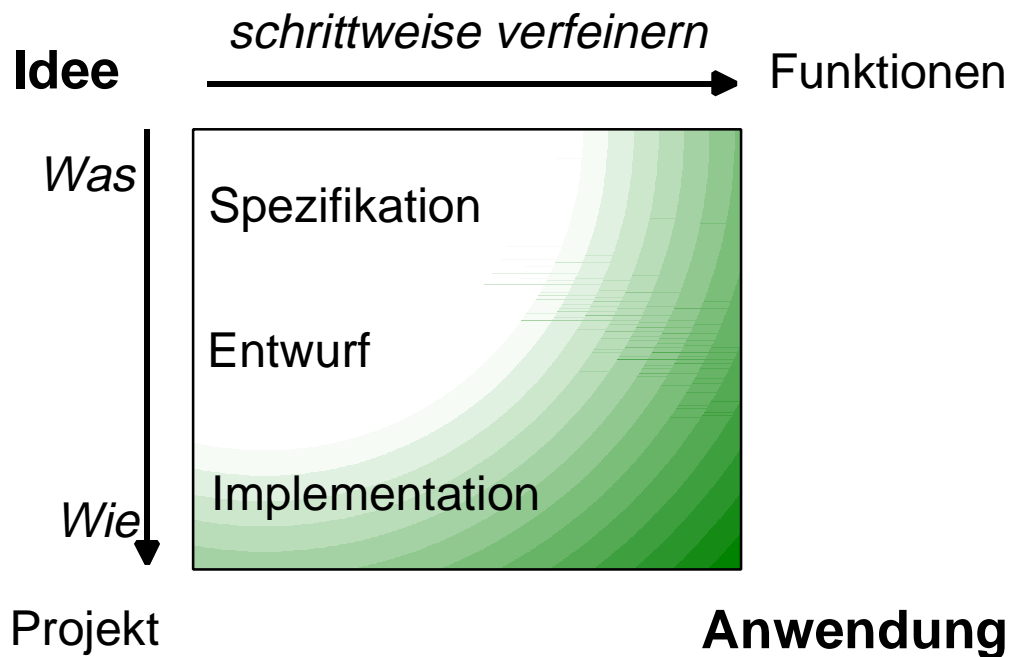
Verschachtelung

Testen

Übung: Turtle

Koordinatensystem

Entwicklungswerkzeuge



Werkzeuge eines Entwicklers

- a) Tabellenblatt und Formulare
- b) Programmeditor
- c) Projektextplorer
- d) Textverarbeitung
- e) Debugger
- f) Papier und Bleistift

Verfeinern durch Modularisieren

Verfeinerungsstufen

Projekt

Modul

Prozedur

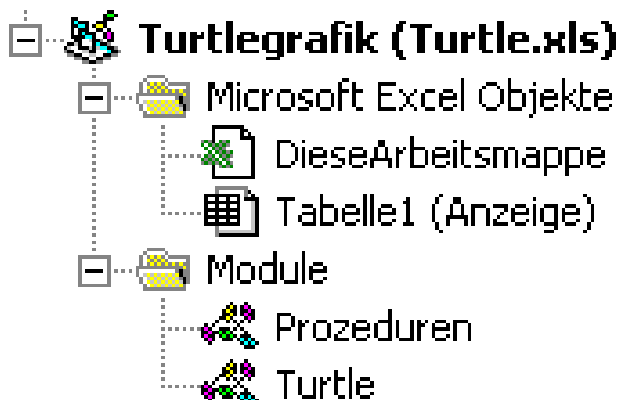
elementare Anweisung

Modul

Ein **Modul** ist ein entwurfs- oder programmiersprachlicher Baustein, der Aufgaben zusammenfasst, die eng verbunden sind

- Aufgabe
- Modularten

Beispiele



2 Klassen

2.1 Objektbeschreibungen.....

2.2 Instanzen.....

2.2.1 Instanzen erstellen.....

2.2.2 Beispiel Telefonverzeichnis

Aufbau eines Code-Moduls

1. Anweisungen an den Übersetzer

z.B.

Option Explicit

2. Vereinbarungsabschnitt

- Variablen und Konstanten
- öffentlich und privat

z.B.:

```
'Private Konstanten  
Private Const Pi2 As Single = 6.2831856  
  
'Private Variablen  
Private X As Integer
```

3. Prozeduren

- Sub-Prozeduren, Funktionen und Ereignisprozeduren
- öffentlich und privat

z.B.:

```
'Öffentliche Prozeduren  
Public Sub Start()  
  
    lösche  
    With Worksheets("Anzeige")  
        ...  
  
End Sub
```


Vereinbarungsanweisungen

Eine **Variable** (Veränderliche) ist ein Name eines Speicherbereiches mit änderbarem Inhalt.
Eine **Konstante** ist ein Name eines Speicherbereiches, dessen Inhalt sich während des Programmablaufes nicht ändert.

Eine Variable (Konstante) **vereinbaren** heisst einem Speicherplatz einen *Namen*, einen *Datentyp*, einen *Gültigkeitsbereich* und einen *Anfangswert* zuweisen

Ort

- Vereinbarungabschnitt eines Moduls
- Prozedurkopf oder Prozedurrumpf

Syntax

[**Public** | **Private** | **Dim**] [**Const**] Name **As** Typ [=Wert]

Begriffe

- Gültigkeitsbereich
- Lebensdauer
- Initialisieren
- Operationen
- Wertebereich

Datentypen

Ein **Datentyp** legt den *Speicherplatzbedarf*, den *Wertebereich* und die *Operationen* einer Variablen oder Konstanten fest.

Datentypen

Name	Speicher	Wertebereich	Operationen
einfach			
Integer	2 Bytes	-32'768 bis 32'767	+, -, *, /, Mod
Boolean	2 Bytes	True oder False	Not, And, Or
Single	4 Bytes	-10E38 bis 10E38	+, -, *, /, Mod
zusammengesetzt			
String	1 Byte pro Zeichen	&, +	
Datenfeld	(elementabhängig)		
Satz	(elementabhängig)		

Begriffe

- einfach - zusammengesetzt
- systemdefiniert - benutzerdefiniert

Zusammengesetzte Datentypen

Datenfeld

Ein **Datenfeld** ist eine Folge von Zellen des *gleichen* Datentyps, auf die unter dem gleichen Namen, aber verschiedenen *Indizes* zugegriffen werden kann.

Vereinbarung:

```
Private Läufer(1 To 5) As String
```

Zugriff:

```
Läufer(1) = "Anna"  
Läufer(2) = "Beat"  
Läufer(3) = "David"  
...
```

Satz

Ein **Satz** ist eine Struktur aus gleichen oder *verschiedenen* Datentypen.

Vereinbarung:

```
Type tPerson  
    Name As String  
    Rang As Integer  
End Type  
  
Private Läuferin As tPerson
```

Zugriff:

```
Läuferin.Name = "Anna"  
Läuferin.Rang = 2
```

Prozeduren

Eine **Prozedur** (Unterprogramm) ist ein Baustein, der aus Programmteilen aufgerufen werden kann.

Ort

- nach Vereinbarungsabschnitt eines Moduls

Syntax

```
[Public | Private] {Sub | Function} Name( _  
    [[ByVal | ByRef] Argument1 As Typ, ...] ) [As Typ]  
    <Vereinbarungsanweisungen>  
    <ausführbare Anweisungen>  
End {Sub | Function}
```

Begriffe

- Gültigkeitsbereich
- Sub-Prozedur, Funktion, Ereignisprozedur
- Argument
- Rückgabewert
- lokale Variable
- ausführbare Anweisung

Eine Prozedur ist ein Modul

Anweisung

Vereinbarungsanweisungen

ausführbare Anweisungen

vordefiniert

einfach	Calculate
parametrisiert	AddLine X1 , Y1 , X2 , Y2
mit Rückgabewert	Len(Text)

benutzerdefiniert

einfach	QuadratFesterLänge
parametrisiert	Quadrat X , Y , Seitenlänge
mit Rückgabewert	einWortWeniger(Text)

Zuweisungsanweisungen

schreiben (links von '=')	Länge = Len(Text)
lesen (rechts von '=')	neueLänge = Länge + 10

Entscheidungsanweisungen

If-Anweisung	If Länge = 0 Then
--------------	---------------------------------

Wiederholungsanweisungen

Zählschleife	For i = 1 To 5 ... Next
mit Ausführbedingung	While Len(Text) > 0 ... Loop

Ausführbare Anweisungen stehen immer innerhalb einer Prozedur.

andere

Kommentar	'x-Koordinate
Einstellungen	Option Explicit
Zeile fortsetzen	—

Operatoren

Zuweisung

<linke Seite> = <rechte Seite>

1. werte linke Seite aus

Operatorvorrang (arithmetisch)

^	Potenzierung
-	Negation
*, /	Multiplikation und Division
\	Ganzzahldivision
Mod	Restwert
+, -	Addition und Subtraktion
&	Zeichenverkettung

2. weise Resultat der rechten Seite zu

Datentypen müssen sich vertragen!

Vergleich

<linke Seite> {=, <>, <, >, <=, >=} <rechte Seite>

1. werte linke und rechte Seite aus

Operatorvorrang (logisch oder boolsch)

arithmetische Operatoren:

Not

And

Or

2. vergleiche Resultate

Datentypen müssen sich vertragen!

Runde **Klammern** ändern den Vorrang.

Argumente

Das **Übergeben** eines Arguments (als Wert) bei einem Prozeduraufruf entspricht einer **Zuweisung** an eine Variable.

Beispiel

```
Public Sub Quadrat(ByVal X As Integer, _  
                  ByVal Y As Integer)  
    ...
```

```
Public Sub Test()  
    ...  
    A = 100  
    Quadrat A, 200
```

Entspricht Zuweisungen: **X** = **A**, **Y** = **200**

Begriffe

- als Wert (ByVal) - als Adresse (ByRef)
- Positionsargumente - benannte Argumente
- voreingestellte Argumentwerte

Entscheidungsanweisungen

If-Anweisung

If Bedingung Then

<Anweisungsblock>

[Else]

<Anweisungsblock>

End If

Ein Gruppe zusammengehörender Anweisungen
heisst **Anweisungsblock**.

- einzelne Anweisung
- Prozedur
- Sequenz
- Entscheidungsanweisungen
- Wiederholungsanweisungen

Anweisungsblöcke werden in VBA meist mit **End** abgeschlossen.

Schleifenanweisungen

Zählschleife

```
For Zähler = Anfang To Ende  
    <Anweisungsblock>  
[Exit For]  
    <Anweisungsblock>  
Next [Zähler]
```

Schleife mit Ausführbedingung

```
<Initialisierung>  
Do While Bedingung  
    <Anweisungsblock>  
[Exit Do]  
    <Anweisungsblock>  
Loop
```

Schleife mit Abbruchbedingung

```
<Initialisierung>  
Do  
    <Anweisungsblock>  
[Exit Do]  
    <Anweisungsblock>  
Loop While Bedingung
```

Begriffe

- Initialisierung
- Schleifenbedingung
- Schleifenkörper

Schleifenanweisungen

Wann benutzt man welchen Schleifentyp?

Schleife	For ... To ... Next	Do While ... Loop	Do ... Loop While
Kriterium			
Abbruch- bedingung	vor Schleifen- körper	vor Schleifen- körper	nach Schlei- fenkörper
Abbruch- kriterium	Zählerstand	True	True
Anzahl Durchläufe	≥ 0	≥ 0	≥ 1
Anzahl vorher bekannt?	ja	nein	nein
Initialisierung	implizit	explizit	explizit
Beispiel	Datenfeld	Datei	Datei

Die **Rekursion** ist eine weitere Form der Wiederholung.

Verschachtelung ausführbarer Anweisungen

Durch rekursive Definition von Anweisungsblöcken führt zu **Verschachtelungen**.

- z.B.: Die If-Anweisung ist ein Anweisungsblock.

If Bedingung **Then**

<Anweisungsblock>

[Else]

<Anweisungsblock>

End If

- z.B.: Schleife und Entscheidungen

```
Do While Len(Text) > 0
  If erstesZeichen(Text) = "a" Then
    Anzahl = Anzahl + 1
  End If
  entferneErstesZeichen(Text)
Loop
```

Der Entscheidungs- oder Schleifenanweisung, die *zuletzt* geöffnet wurde, muss *zuerst* wieder geschlossen werden.

Verschachtelung von Datenstrukturen

Durch verschachteln benutzerdefinierte Datentypen entstehen komplexe **Datenstrukturen**.

- z.B.: Satz und Datenfeld (->Tabelle)

```
Type tPerson                                'Spalten  
    Name As String  
    Rang As Integer  
End Type  
Dim Läufer(1 To 5) As tPerson              'Zeilen
```

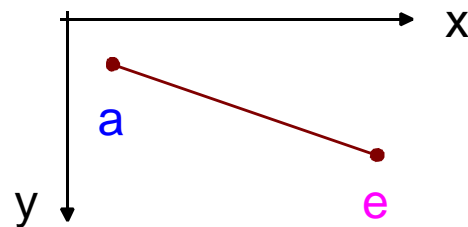
Läufer	Name	Rang
1	Anna	2
2	Beat	4
3	Carmen	1
4	David	3
5	Elisabeth	5

Koordinatensystem

Excel

In Excel zeichnet man Linien durch Angabe eines **Anfangs-** und eines **Endpunktes** bezüglich eines Koordinatensystems (**absolut**).

```
Ausgabeblatt.Shapes _  
    .AddLine(Xa, Ya, Xe, Ye).Visible = True
```



Turtle-Grafik

In der Turtle-Grafik zeichnet man Linien durch Angabe einer **Richtung** und einer **Länge relativ** zur aktuellen Position und Richtung.

- Start
- geheVorwärts <Schritte>
- dreheLinks <Winkel>
- dreheRechts <Winkel>



zusätzliche Prozeduren

- `istSalatVorne`
- `istSalatLinks`
- `istSalatRechts`
- `istSalatGefunden`

