



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN  
University of Applied Sciences

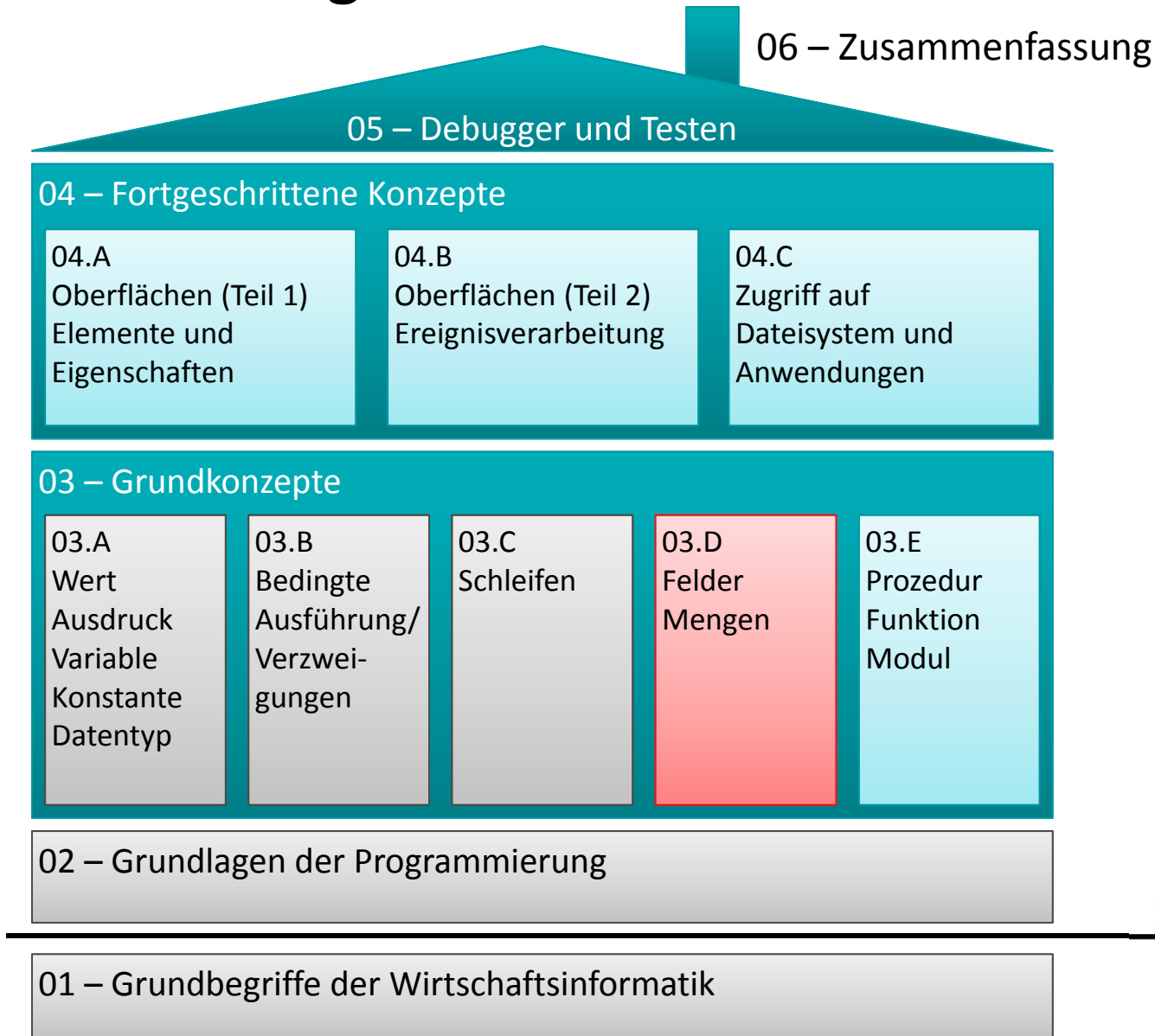
# **Wirtschaftsinformatik 1**

## **LE 06 – Felder, Mengen und zusammengesetzte Datentypen**

Prof. Dr. Thomas Off

<http://www.ThomasOff.de/lehre/beuth/wi1>

# Einordnung





# **Inhalt**

## **Rückblick**

## **Ausgangspunkt**

## **Zusammengesetzte Datentypen**

## **Felder**

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## **Maps**

## **Abschluss und Ausblick**



# Inhalt

## Rückblick

## Ausgangspunkt

## Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick

# Rückblick

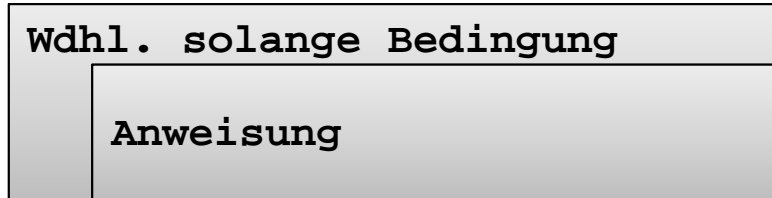


# Rückblick



## Vorprüfende/Kopfprüfende Schleife

- Einsatzzweck
  - erst Bedingung prüfen
  - dann ggf. Anweisung ausführen
  - anschließend Wiederholung der Prüfung usw.



### – Generelle Syntax

```
Do While <Bedingung>  
  <Anweisung(en)>  
Loop
```

```
Do Until <Bedingung>  
  <Anweisung(en)>  
Loop
```

### – Beispiel

```
Dim i As Integer  
Let i = 0  
Do While i < 5  
  Let i = i + 1  
  Debug.Print i  
Loop
```

```
Dim j As Byte  
Let j = 0  
Do Until j > 4  
  Let j = j + 1  
  Debug.Print j  
Loop
```

# Rückblick



## Nachprüfende/Fußprüfende Schleife

- Einsatzzweck
  - erst Anweisung ausgeführt
  - dann Bedingung prüfen
  - anschließend Wiederholung der Anweisungsausführung usw.



### – Generelle Syntax

```
Do  
  <Anweisung(en)>  
Loop While <Beding.>
```

```
Do  
  <Anweisung(en)>  
Loop Until <Beding.>
```

### – Beispiel

```
Dim i As Integer  
Let i = 0  
Do  
  Let i = i + 1  
  Debug.Print i  
Loop While i < 5
```

```
Dim j As Byte  
Let j = 0  
Do  
  Let j = j + 1  
  Debug.Print j  
Loop Until j > 4
```

# Rückblick



## Zählerschleifen

### – Einsatz

- Vorher bekannte Anzahl von Wiederholungen
- Anzahl gesteuert über Start und Ende
- Ausführung der Anweisung solange Anzahl Wiederholungen das Ende noch überschritten hat

Zähle von Start bis Ende

Anweisung

### – Generelle Syntax

```
For <Var> = <Wert/Ausdr> To <Ausdr> Step <Schrittw>  
  <Anweisung(en)>  
Next
```

### – Beispiel

```
Dim i As Integer  
For i = 1 To 10 Step 2  
  Debug.Print i  
Next
```



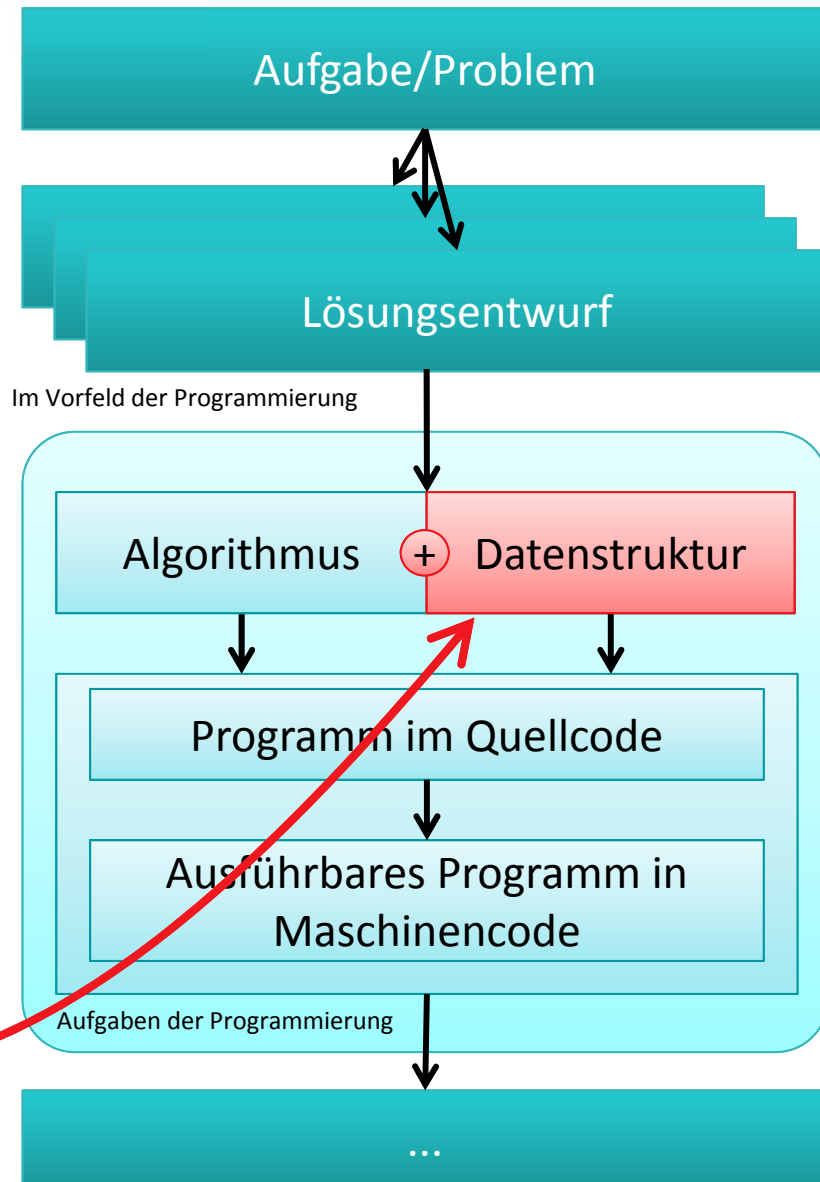
# Rückblick



## Programmierung als stufenweise Umsetzung

- von Algorithmus und
- Datenstruktur
- in Quellcode einer Programmiersprache und
- der Überführung in ein ausführbares Programm im Maschinencode

*Heute: Datenstrukturen für Felder, Mengen und zusammengesetzte Typen*

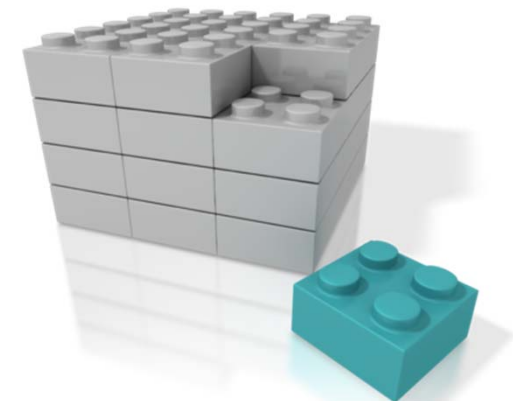
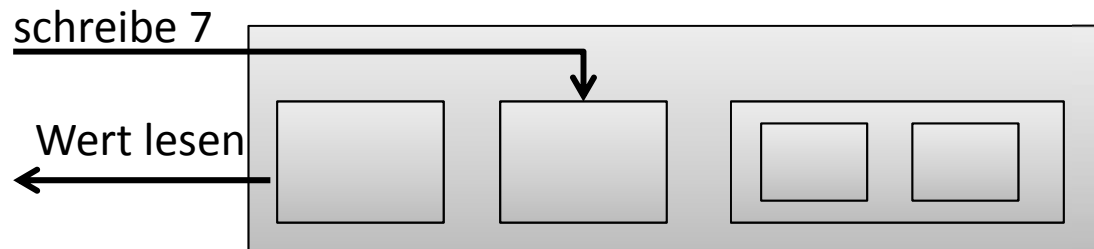


# Rückblick (LE02)



## Komplexe Datenelemente

- Aufbau
  - fassen mehrere einfache Datenelemente oder andere komplexe Datenelemente zusammen
  - Repräsentieren häufig Dinge und Konzepte der Realität
- Beispiel: komplexes Datenelement "Person" fasst einfache Datenelemente "Name", "Vorname" und komplexes Datenelement "Adresse" zusammen
- Zugriff auf einzelne Elemente des komplexen Datenelementes möglich (Lesen, Schreiben)

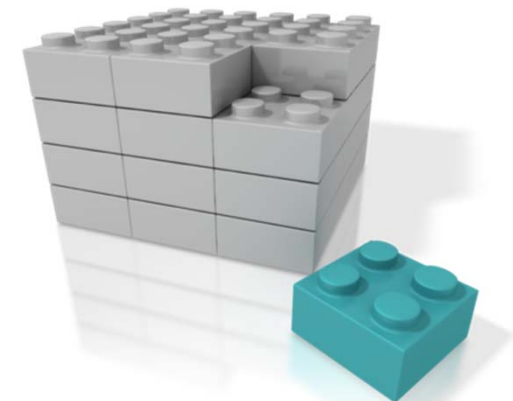


# Rückblick (LE02)



## Datenstrukturen

- Sequentielle Liste/Feld (Array)
- Verkettete Liste
- Map
- Stapel (Stack)
- Schlange (Queue)
- Graph
- Baum
- ...



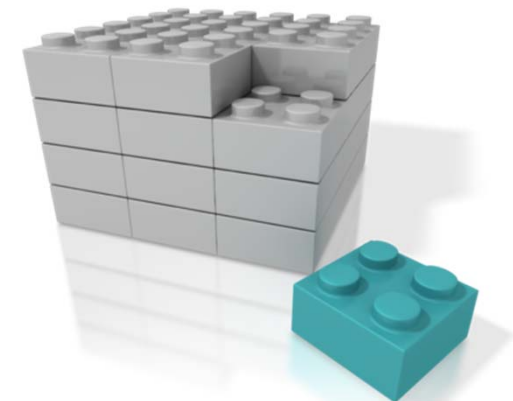
# Rückblick (LE02)



## Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

Liste/Feld	$i_0$	$i_1$	$i_2$	...	$i_{n-1}$	$i_n$
Index	0	1	2	...	n-1	n



# Rückblick (LE02)

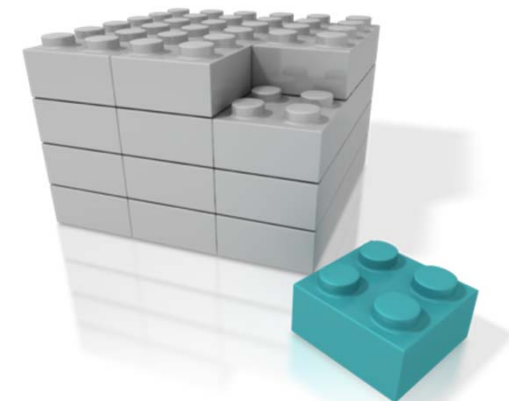


## Sequentielle Liste/Feld (Array)

- Lesender Zugriff über Index frei möglich

Liste/Feld	$i_0$	$i_1$	$i_2$	...	$i_{n-1}$	$i_n$
Index	0	1	2	...	n-1	n

- Einfügen

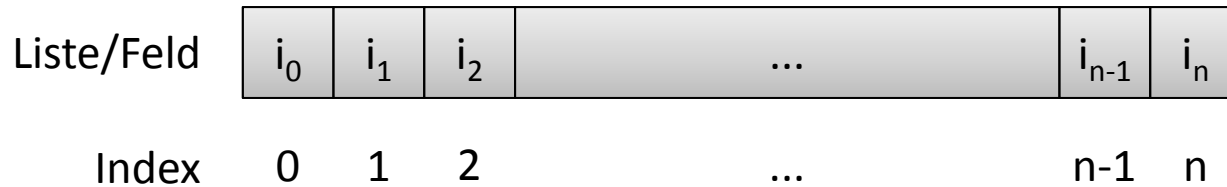


# Rückblick (LE02)

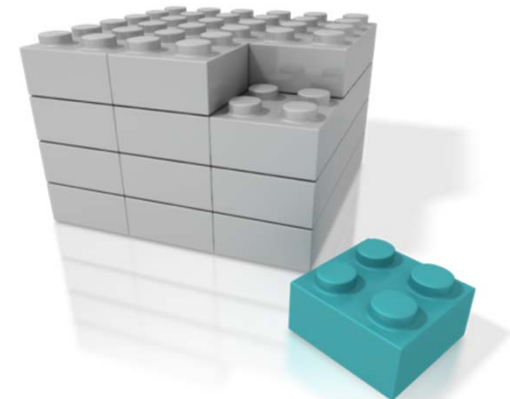
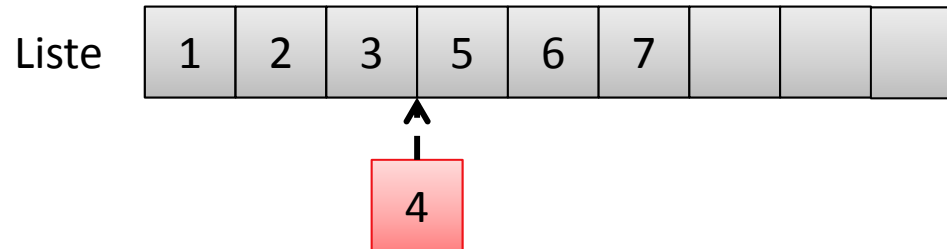


## Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich



– Einfügen

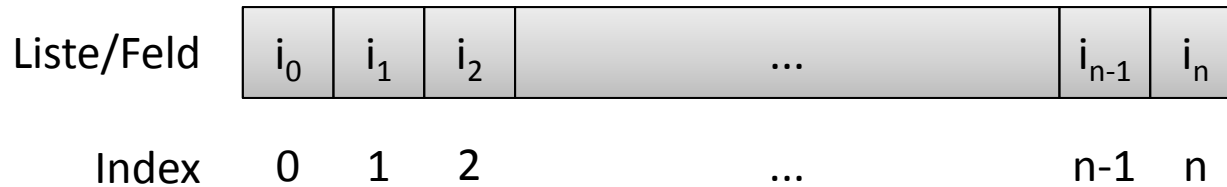


# Rückblick (LE02)

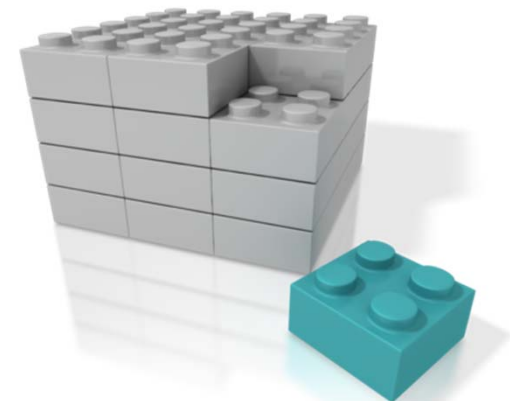
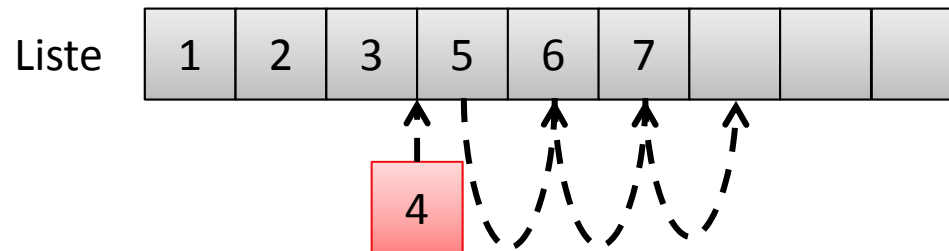


## Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich



– Einfügen

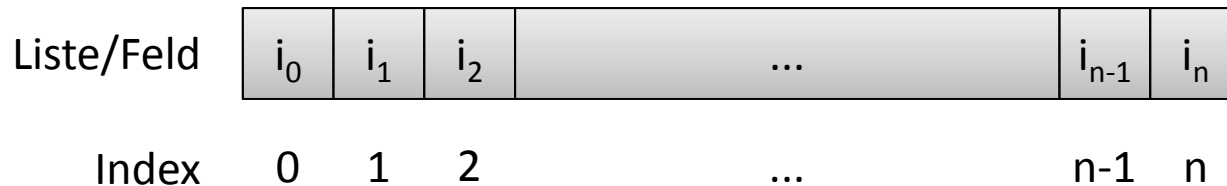


# Rückblick (LE02)

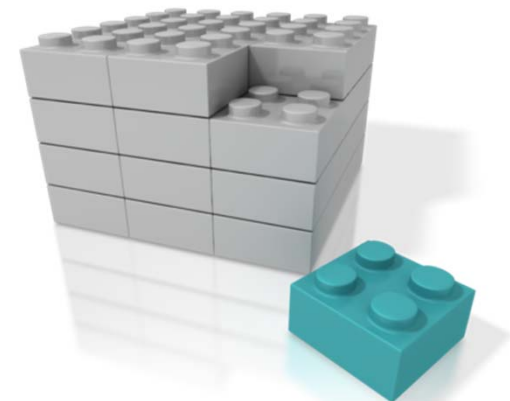
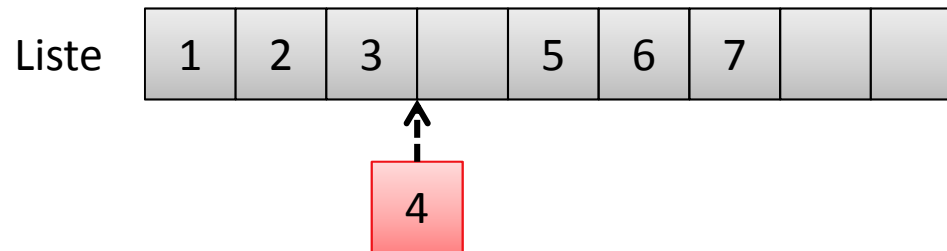


## Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich



– Einfügen





# Rückblick (LE02)



## Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

Liste/Feld	$i_0$	$i_1$	$i_2$	...	$i_{n-1}$	$i_n$
Index	0	1	2	...	n-1	n

– Einfügen

Liste	1	2	3	4	5	6	7		
-------	---	---	---	---	---	---	---	--	--

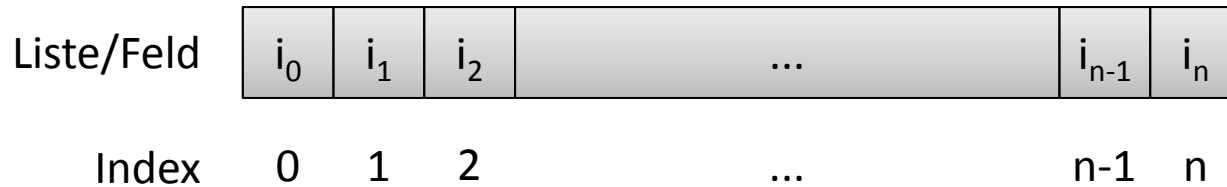


# Rückblick (LE02)

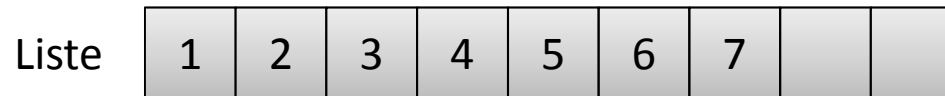


## Sequentielle Liste/Feld (Array)

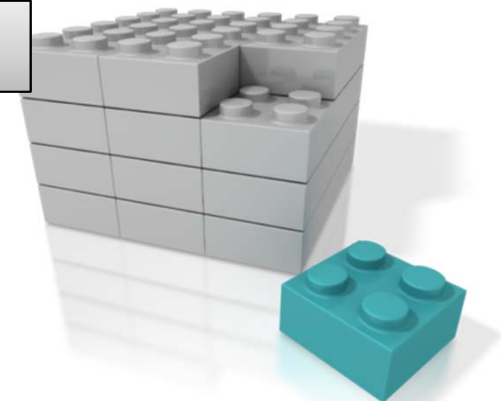
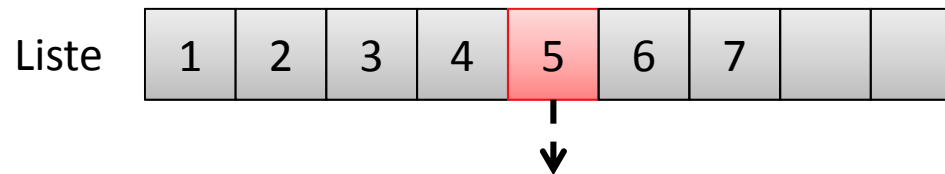
– Lesender Zugriff über Index frei möglich



– Einfügen



– Entfernen

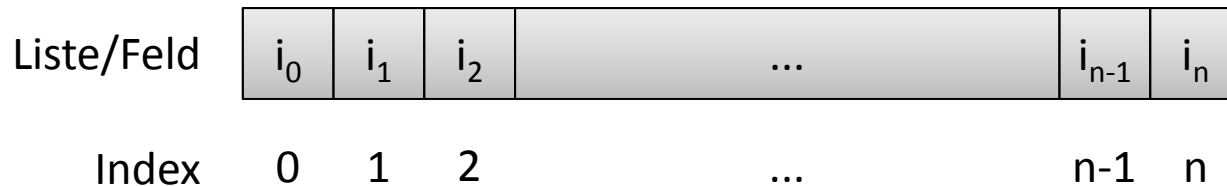


# Rückblick (LE02)



## Sequentielle Liste/Feld (Array)

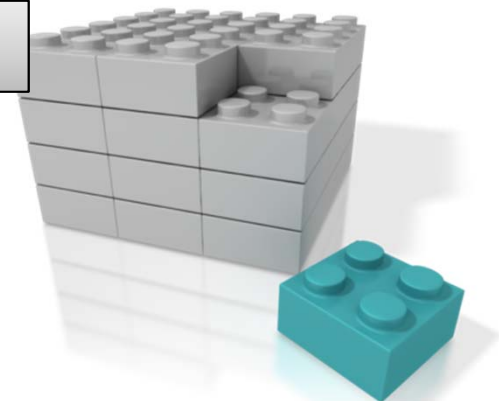
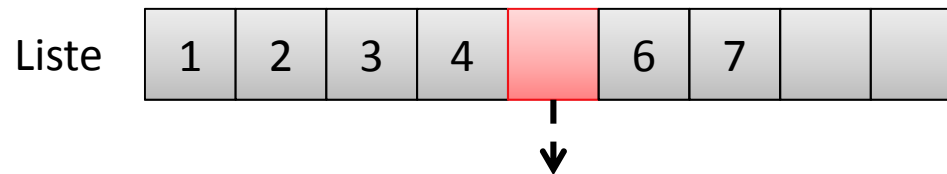
– Lesender Zugriff über Index frei möglich



– Einfügen



– Entfernen

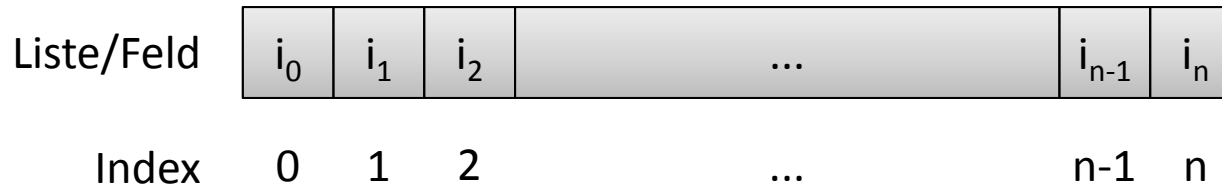


# Rückblick (LE02)

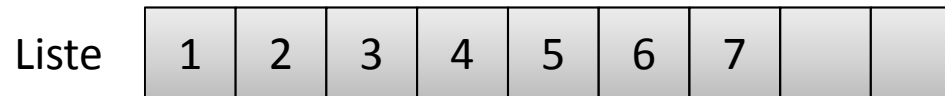


## Sequentielle Liste/Feld (Array)

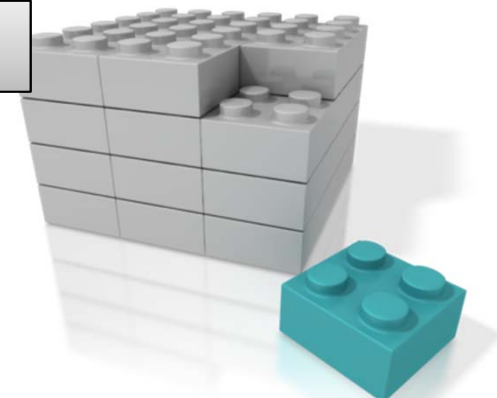
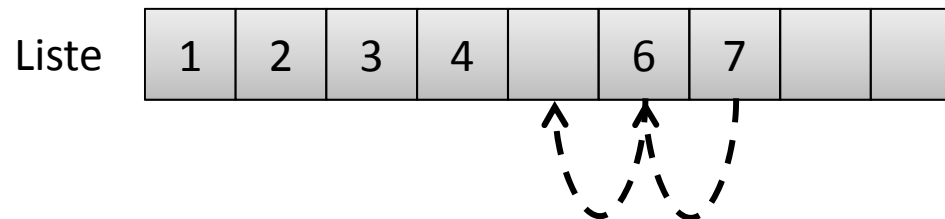
– Lesender Zugriff über Index frei möglich



– Einfügen



– Entfernen



# Rückblick (LE02)



## Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

Liste/Feld	$i_0$	$i_1$	$i_2$	...	$i_{n-1}$	$i_n$
------------	-------	-------	-------	-----	-----------	-------

– Einfügen

Index	0	1	2	...	n-1	n
-------	---	---	---	-----	-----	---

Liste	1	2	3	4	5	6	7		
-------	---	---	---	---	---	---	---	--	--

– Entfernen

Liste	1	2	3	4	6	7			
-------	---	---	---	---	---	---	--	--	--

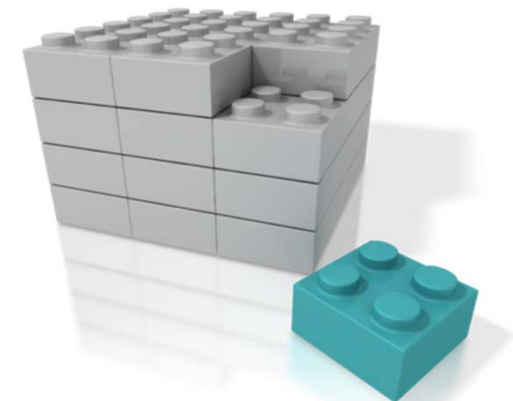


# Rückblick (LE02)



## Datenstrukturen

- Sequentielle Liste/Feld (Array)
- Verkettete Liste
- Map
- Stapel (Stack)
- Schlange (Queue)
- Graph
- Baum
- ...

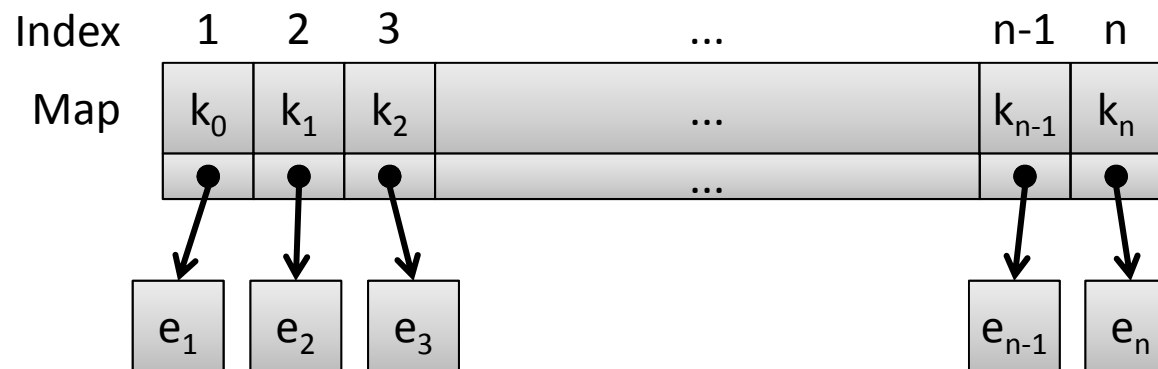


# Rückblick (LE02)

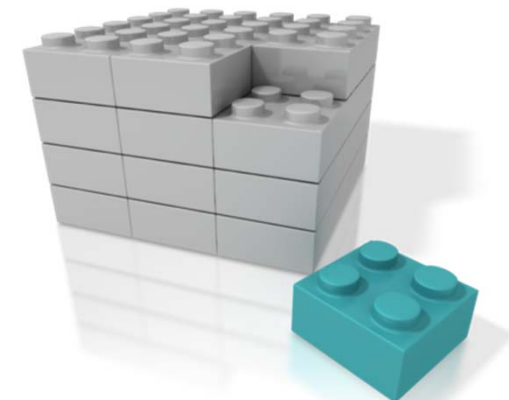


## Map

- Lesender Zugriff
  - über Schlüssel (Key) frei möglich,
  - teilweise zusätzlich Zugriff über einen Index möglich (z.B. in VBA)



- Einfügen
  - mit Angabe eines Schlüssels (der in VBA noch nicht vergeben sein darf)
  - Vergrößerung erfolgt bei Bedarf automatisch
- Entfernen über Schlüssel bzw. Index

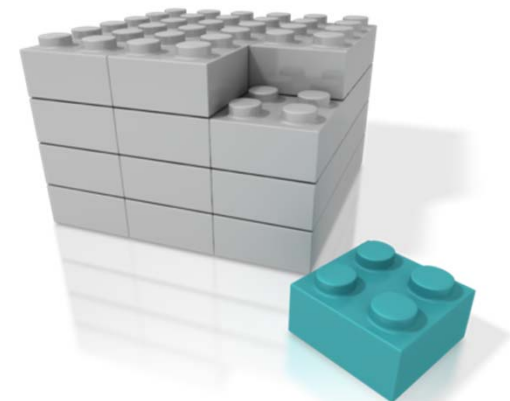


# Rückblick (LE02)



## Datenstrukturen

- Sequentielle Liste/Feld (Array)
- Verkettete Liste
- Map
- Stapel (Stack)
- Schlange (Queue)
- Graph
- Baum
- ...







# Inhalt

## Rückblick

## Ausgangspunkt

## Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick





# Inhalt

## Rückblick

## Ausgangspunkt

## Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick



# Inhalt

Rückblick

## Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick



# Ausgangspunkt

- Variable hat Ähnlichkeiten mit Schubladen in Möbeln
  - speichern Wert eines bestimmten Typs
  - Wert oder Ausdruck kann ihr zugewiesen werden
  - bietet lesenden, schreibenden und ändernden Zugriff
  - hat Bezeichner



IKEA FAKTUM  
Küchenunterschrank



IKEA SKÄR  
Schuhschrank



IKEA ASPVIK  
Aktenschrank



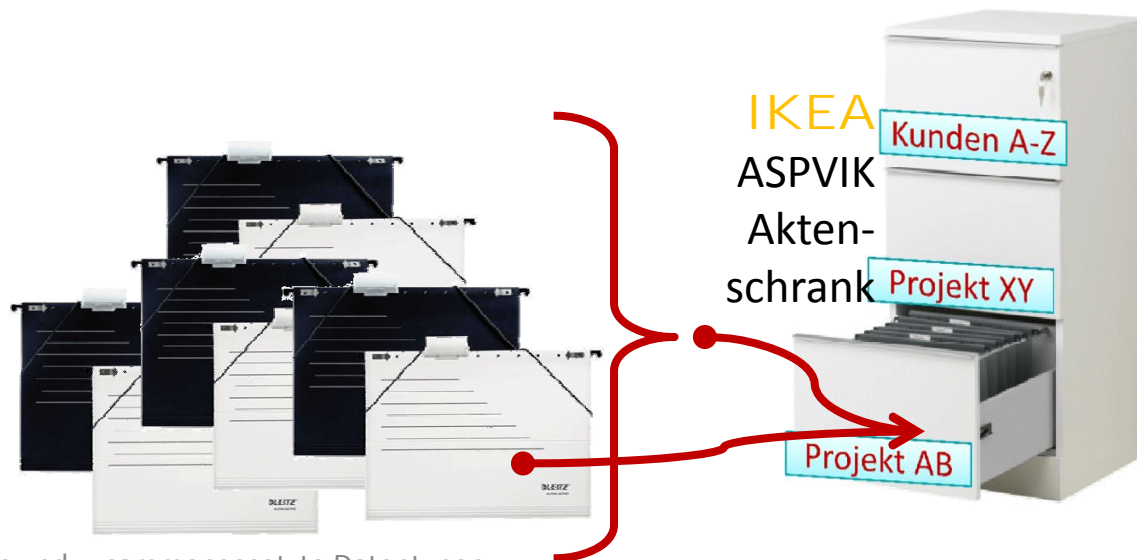
Abbildungen hier und im Folgenden: [1]



# Ausgangspunkt

... aber

- in eine Schublade kann ich mehrere Dinge des gleichen Typs legen
- in einer Variable kann ich jedoch nur einen Wert eines bestimmten Typs speichern

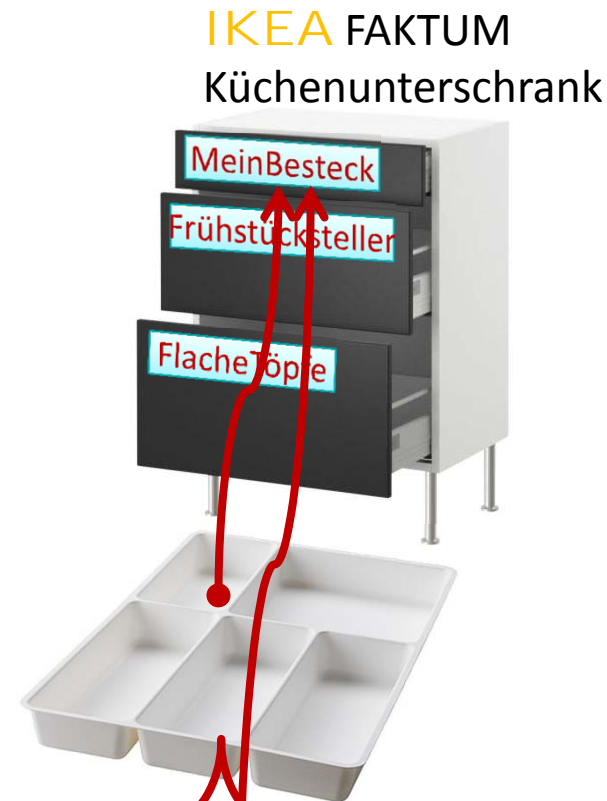




# Ausgangspunkt

## ... und außerdem

- hat die Besteckschublade einen Einsatz, damit z.B. 6 Löffel, 6 Messer, 6 Teelöffel und 6 Gabeln nicht durcheinander kommen
- eine Variable, selbst wenn sie mehrere Werte speichern könnte, würde mir aber keine Struktur anbieten



# Ausgangspunkt

... oder geht das doch?





# Ausgangspunkt

## Angenommen,

- wir haben drei Kunden
- wir wollen die Kunden mit
  - Name
  - Vorname
  - Straße, HausNr
  - Plz und Ort
- in Variablen speichern und im Direktbereich ausgeben.

## Wie ginge das?

```
Sub Bsp60()  
  
    Dim strKundeName1 As String  
    Dim strKundeVorname1 As String  
    Dim strKundeStrasseNr1 As String  
    Dim strKundePlzOrt1 As String  
  
    Dim strKundeName2 As String  
    Dim strKundeVorname2 As String  
    Dim strKundeStrasseNr2 As String  
    Dim strKundePlzOrt2 As String  
  
    Dim strKundeName3 As String  
    Dim strKundeVorname3 As String  
    Dim strKundeStrasseNr3 As String  
    Dim strKundePlzOrt3 As String  
  
    ' ...  
  
End Sub
```





# Ausgangspunkt

## Angenommen,

- wir haben drei Kunden
- wir wollen die Kunden mit
  - Name
  - Vorname
  - Straße, HausNr
  - Plz und Ort
- in Variablen speichern und im Direktbereich ausgeben.

## Wie ginge das?

```
Sub Bsp60()  
  
    ...  
  
    Let strKundeName1 = "Yilmaz"  
    Let strKundeVorname1 = "Samir"  
    Let strKundeStrasseNr1 = "Straße 2"  
    Let strKundePlzOrt1 = "12345 Stadt"  
  
    Let strKundeVorname2 = "Michael"  
    Let strKundeName2 = "Müller"  
    Let strKundePlzOrt2 = "12346 Stadt"  
    Let strKundeStrasseNr2 = "Weg 14"  
  
    Let strKundeVorname3 = "Barbara"  
    Let strKundeName3 = "Müller"  
    Let strKundePlzOrt3 = "12346 Stadt"  
    Let strKundeStrasseNr3 = "Weg 14"  
  
    ...  
  
End Sub
```



# Ausgangspunkt

## Angenommen,

- wir haben drei Kunden
- wir wollen die Kunden mit
  - Name
  - Vorname
  - Straße, HausNr
  - Plz und Ort
- in Variablen speichern und im Direktbereich ausgeben.

## Wie ginge das?

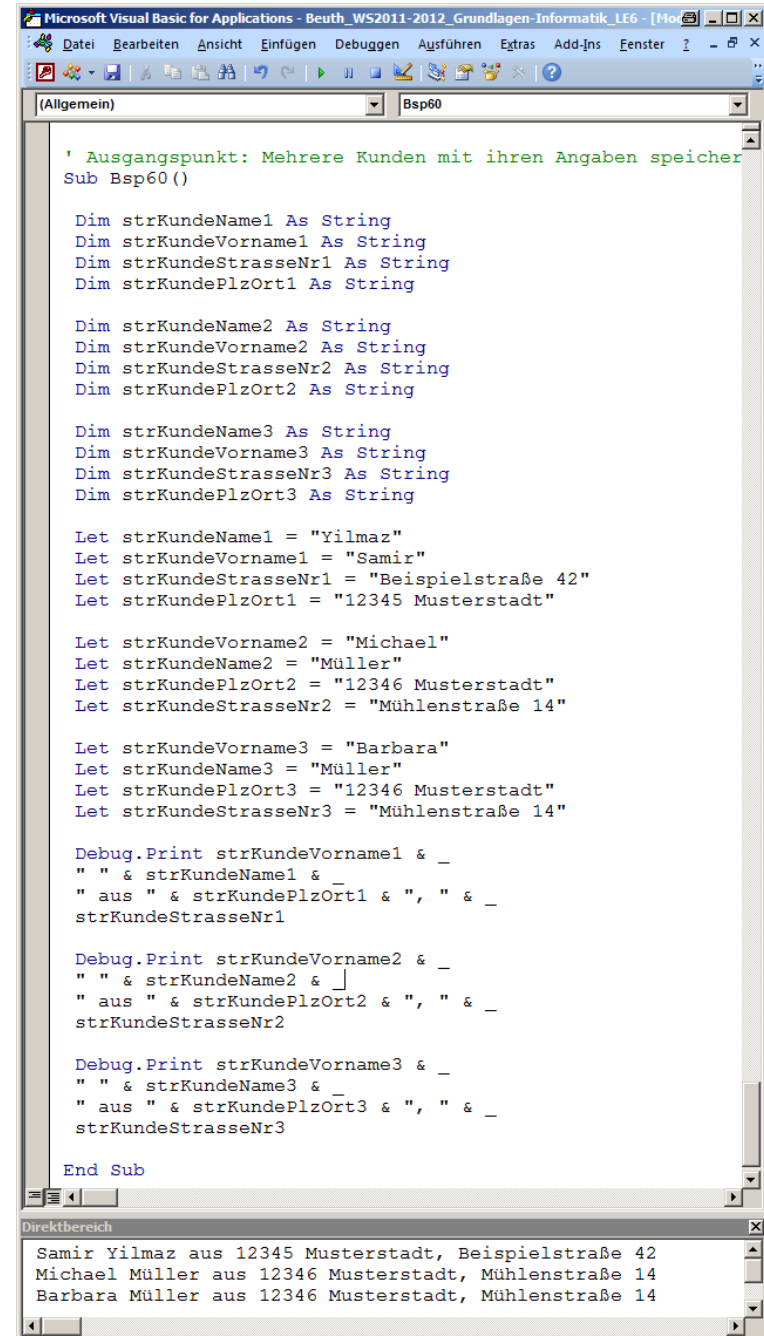
```
Sub Bsp60()  
  
    ...  
  
    Debug.Print strKundeVorname1 & _  
                " " & strKundeName1 & _  
                " " & strKundePlzOrt1 & _  
                " " & strKundeStrasseNr1  
  
    Debug.Print strKundeVorname2 & _  
                " " & strKundeName2 & _  
                " " & strKundePlzOrt2 & _  
                " " & strKundeStrasseNr2  
  
    Debug.Print strKundeVorname3 & _  
                " " & strKundeName3 & _  
                " " & strKundePlzOrt3 & _  
                " " & strKundeStrasseNr3  
  
End Sub
```

# Ausgangspunkt

## Angenommen,

- wir haben drei Kunden
- wir wollen die Kunden mit
  - Name
  - Vorname
  - Straße, HausNr
  - Plz und Ort
- in Variablen speichern und im Direktbereich ausgeben.

## Wie ginge das?



```
' Ausgangspunkt: Mehrere Kunden mit ihren Angaben speichern
Sub Bsp60 ()

    Dim strKundeName1 As String
    Dim strKundeVorname1 As String
    Dim strKundeStrasseNr1 As String
    Dim strKundePlzOrt1 As String

    Dim strKundeName2 As String
    Dim strKundeVorname2 As String
    Dim strKundeStrasseNr2 As String
    Dim strKundePlzOrt2 As String

    Dim strKundeName3 As String
    Dim strKundeVorname3 As String
    Dim strKundeStrasseNr3 As String
    Dim strKundePlzOrt3 As String

    Let strKundeName1 = "Yilmaz"
    Let strKundeVorname1 = "Samir"
    Let strKundeStrasseNr1 = "Beispielstraße 42"
    Let strKundePlzOrt1 = "12345 Musterstadt"

    Let strKundeVorname2 = "Michael"
    Let strKundeName2 = "Müller"
    Let strKundePlzOrt2 = "12346 Musterstadt"
    Let strKundeStrasseNr2 = "Mühlenstraße 14"

    Let strKundeVorname3 = "Barbara"
    Let strKundeName3 = "Müller"
    Let strKundePlzOrt3 = "12346 Musterstadt"
    Let strKundeStrasseNr3 = "Mühlenstraße 14"

    Debug.Print strKundeVorname1 & _
    " " & strKundeName1 & _
    " aus " & strKundePlzOrt1 & ", " & _
    strKundeStrasseNr1

    Debug.Print strKundeVorname2 & _
    " " & strKundeName2 & _
    " aus " & strKundePlzOrt2 & ", " & _
    strKundeStrasseNr2

    Debug.Print strKundeVorname3 & _
    " " & strKundeName3 & _
    " aus " & strKundePlzOrt3 & ", " & _
    strKundeStrasseNr3

End Sub
```

Direktbereich

```
Samir Yilmaz aus 12345 Musterstadt, Beispielstraße 42
Michael Müller aus 12346 Musterstadt, Mühlenstraße 14
Barbara Müller aus 12346 Musterstadt, Mühlenstraße 14
```



# Ausgangspunkt

## Angenommen,

- wir haben drei Kunden
- wir wollen die Kunden mit
  - Name
  - Vorname
  - Straße, HausNr
  - Plz und Ort
- in Variablen speichern und im Direktbereich ausgeben.

## Es ginge aber, ...

- nicht mit vielen Kunden und
- die zusammenhängende Struktur fehlt.





# Inhalt

Rückblick

## Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick





# Inhalt

## Rückblick

## Ausgangspunkt

## Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick



# Inhalt

Rückblick

Ausgangspunkt

**Zusammengesetzte Datentypen**

**Felder**

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

**Maps**

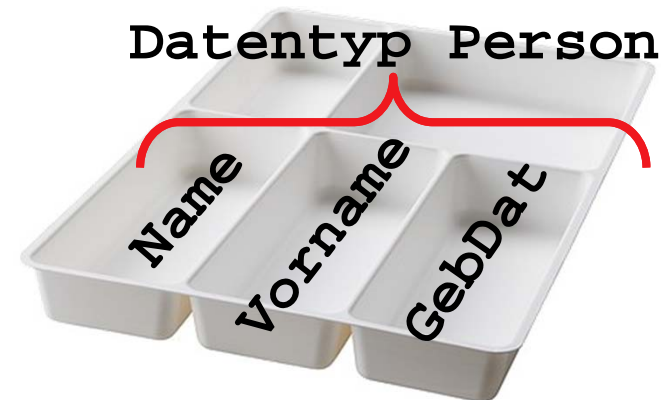
**Abschluss und Ausblick**

# Zusammengesetzte Datentypen



## Strukturdefinition

- Definition des Typs
- einzelne Eigenschaften des Typs basierend auf vorhanden Datentypen
- innerhalb des Moduls



```
' Generelle Syntax
Type <Typbezeichner>
  <Eigenschaft1> As <Datentyp>
  <Eigenschaft2> As <Datentyp>
  ...
End Type
```

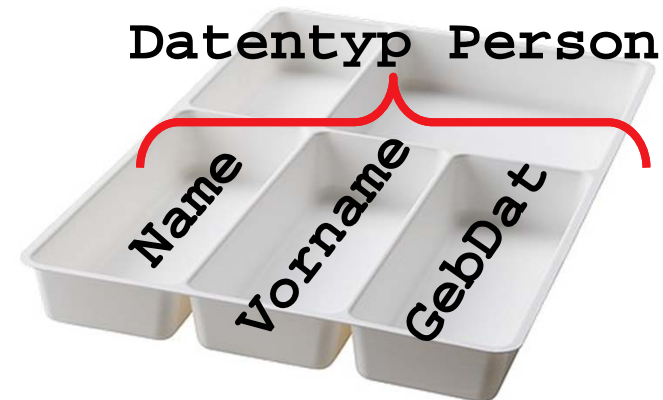


# Zusammengesetzte Datentypen



## Strukturdefinition

- Definition des Typs
  - Schlüsselwort `Type`
  - Bezeichner, per Konvention beginnend mit großem `T`
  - Schlüsselwort `End Type`
- einzelne Eigenschaften des Typs basierend auf vorhanden Datentypen
- innerhalb des Moduls



```
' Generelle Syntax
Type <Typbezeichner>
  <Eigenschaft1> As <Datentyp>
  <Eigenschaft2> As <Datentyp>
  ' ...
End Type
```

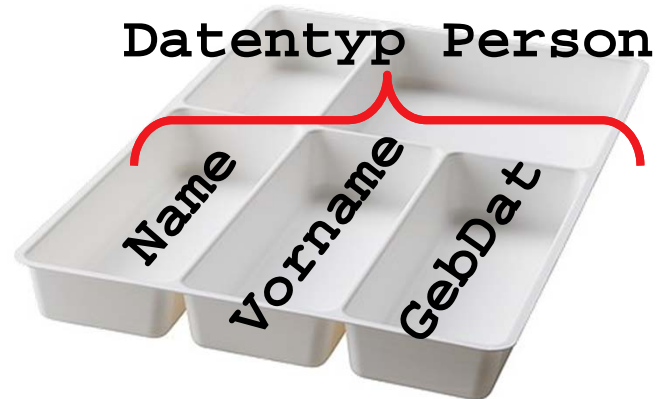
```
' Beispiel
Type TPerson
  strName As String
  strVorname As String
  datGebDat As Date
End Type
```

# Zusammengesetzte Datentypen



## Strukturdefinition

- Definition des Typs
- einzelne Eigenschaften des Typs basierend auf vorhanden Datentypen, z.B.
  - Vorname, Name als String
  - GebDat als Datum
  - Weitere als Typen (auch benutzerdefinierte)
- innerhalb des Moduls



```
' Generelle Syntax  
Type <Typbezeichner>  
  <Eigenschaft1> As <Datentyp>  
  <Eigenschaft2> As <Datentyp>  
  ...  
End Type
```

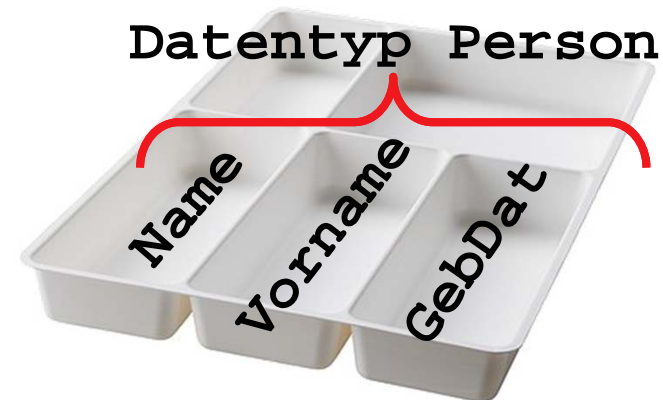
```
' Beispiel  
Type TPerson  
  strName As String  
  strVorname As String  
  datGebDat As Date  
End Type
```

# Zusammengesetzte Datentypen



## Strukturdefinition

- Definition des Typs
- einzelne Eigenschaften des Typs basierend auf vorhanden Datentypen
- innerhalb des Moduls
  - außerhalb unseres "Programms", d.h. der Prozedur



```
' Generelle Syntax  
Type <Typbezeichner>  
  <Eigenschaft1> As <Datentyp>  
  <Eigenschaft2> As <Datentyp>  
  ...  
End Type
```

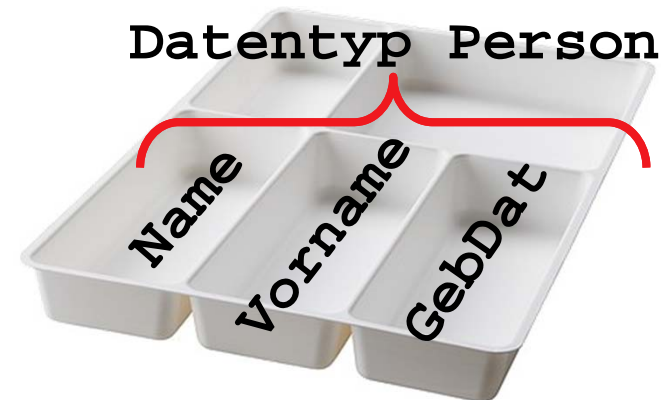
```
' Beispiel  
Type TPerson  
  strName As String  
  strVorname As String  
  datGebDat As Date  
End Type
```

# Zusammengesetzte Datentypen



## Strukturdefinition

- Definition des Typs
- einzelne Eigenschaften des Typs basierend auf vorhanden Datentypen
- innerhalb des Moduls



```
' Generelle Syntax
Type <Typbezeichner>
  <Eigenschaft1> As <Datentyp>
  <Eigenschaft2> As <Datentyp>
  ...
End Type
```

```
' Beispiel
Type TPerson
  strName As String
  strVorname As String
  datGebDat As Date
End Type
```



# Zusammengesetzte Datentypen

## Strukturdefinition

- Definition des Typs
- einzelne Eigenschaften des Typs basierend auf vorhanden Datentypen
- innerhalb des Moduls

## Nutzung

- zur Deklaration von Variablen
  - Konvention: Präfix wie Typ
- Zugriff auf die Eigenschaften des Typs mit der Punkt-Notation
- Abkürzung mit **With** und **End With** möglich

```
Option Compare Database
Option Explicit

Type TPerson
    strName As String
    strVorname As String
    datGebDat As Date
End Type

Sub xyz()
    Dim perKarl As TPerson
    Dim perKurt As TPerson

    Let perKarl.strName = "Meier"
    Let perKarl.strVorname="Karl"
    Let perKarl.datGebDat= #11/11/2011#

    With perKurt
        Let .strName = "Meier"
        Let .strVorname="Kurt"
        Let .datGebDat= #11/11/2011#
    End With

    ' ...
End Sub
```

# Zusammengesetzte Datentypen: Beispiel 06.01



## Ziel

- Definition und Nutzung eines zusammengesetzten Datentypen

## Aufgabe

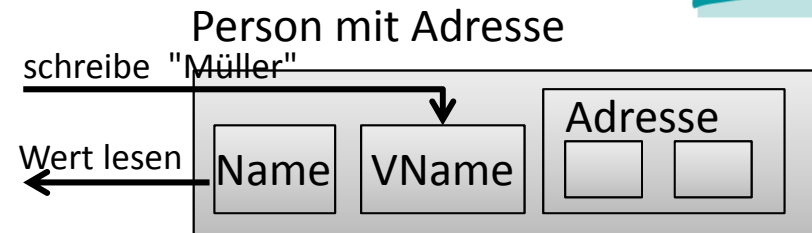
- Definieren Sie einen Datentyp für Kunden mit
  - Kundennummer
  - Name, Vorname
  - Straße und HausNr.
  - Plz und Ort
- Nutzen Sie den Datentyps zur Deklaration von drei Variablen des Typs Kunde
- Initialisieren Sie die Variablen mit Werten
- Geben Sie die Variablen im Direktbereich aus



# Zusammengesetzte Datentypen



- fassen mehrere Eigenschaften definierter Datentypen zusammen
- Repräsentieren häufig Dinge der Realität, z.B. "Person" mit Eigenschaften "Name", "Vorname" und "Adresse"
- Werden als Type definiert und zur Deklaration von Variablen benutzt
- Zugriff auf einzelne Elemente der Variable des zusammengesetzten Datentypen über Punkt-Notation möglich (Lesen, Schreiben)



## ' Generelle Syntax

```
Type <Typbezeichner>  
  <Eigenschaft1> As <Datentyp>  
  <Eigenschaft2> As <Datentyp>  
  ...  
End Type
```

## ' Definition

```
Type TPerson  
  strName As String  
  adrWohnanschrift As TAdresse  
End Type
```

## ' Deklaration und Nutzung

```
Dim perTom As TPerson  
Let perTom.strName = "Tom"  
Debug.Print perTom.strName
```



# Inhalt

Rückblick

Ausgangspunkt

**Zusammengesetzte Datentypen**

**Felder**

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

**Maps**

**Abschluss und Ausblick**







# Inhalt

## Rückblick

## Ausgangspunkt

## Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick



# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

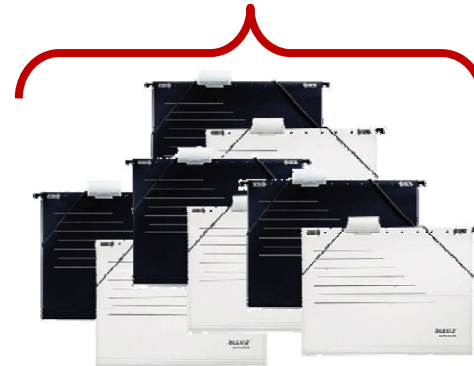
## Maps

## Abschluss und Ausblick

# Einfache Felder

Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze



# Einfache Felder

Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze



**Variable mit Datentyp  
und Bezeichner**

A diagram showing five empty, light blue rectangular boxes stacked vertically. These boxes represent an array of variables, each with its own index or position within the array.



# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

**Variable mit Datentyp  
und Bezeichner**

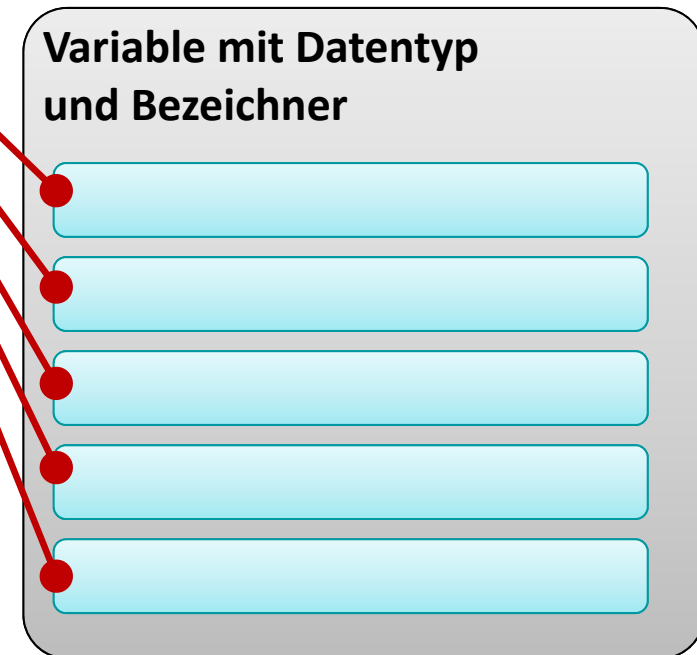
A diagram illustrating a variable container. It consists of a light gray rounded rectangle with a thin black border. Inside, at the top, is the text 'Variable mit Datentyp und Bezeichner'. Below this text are five horizontal, light blue rounded rectangles, each representing a slot for a value in an array. The slots are stacked vertically and are empty.



# Einfache Felder

**Felder (Array) können benutzt werden, um**

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

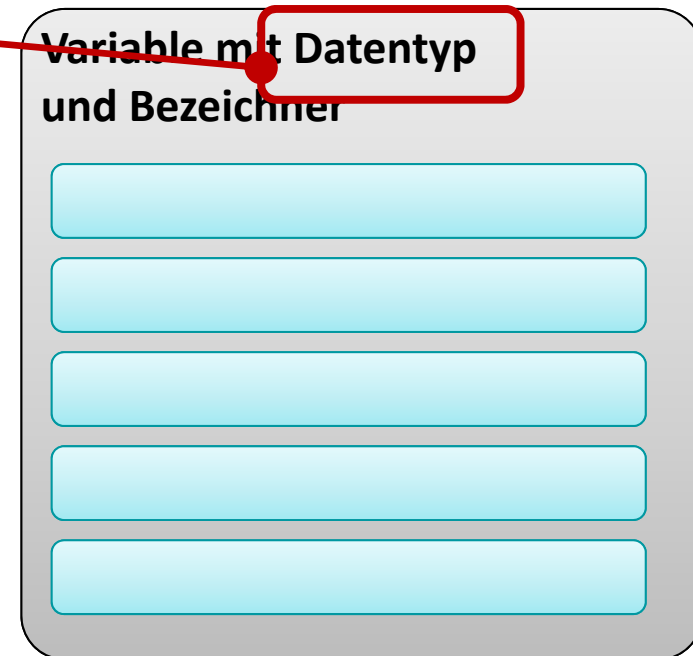




# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

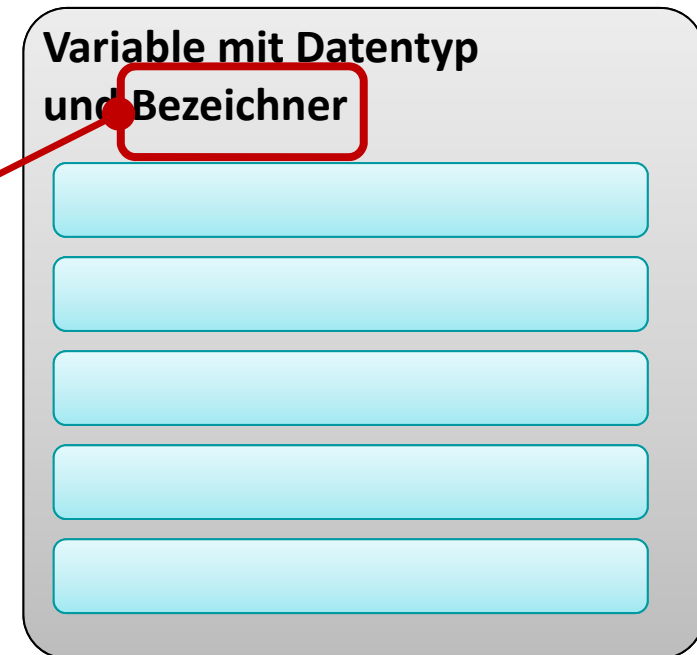




# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze



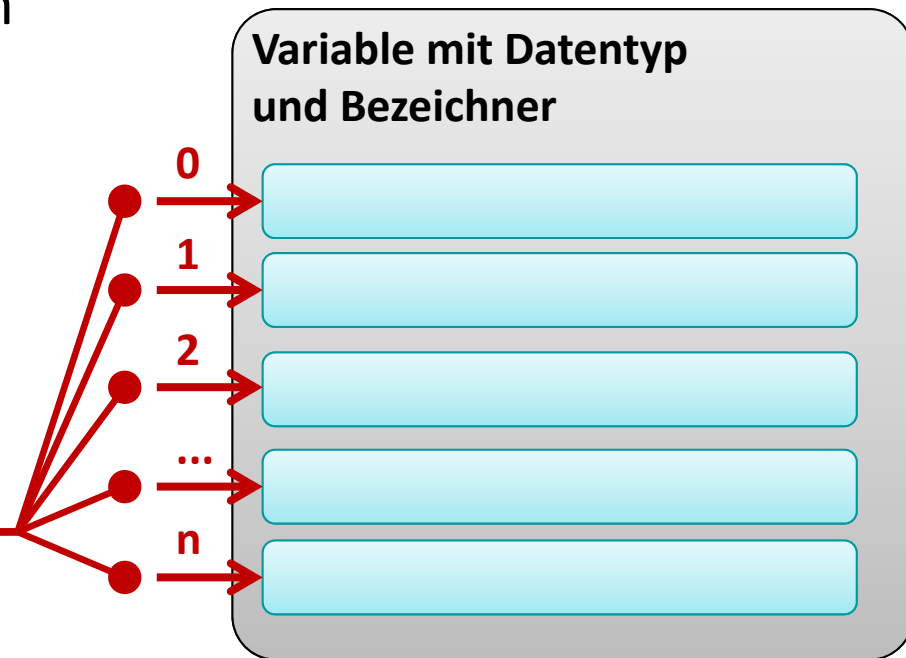




# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

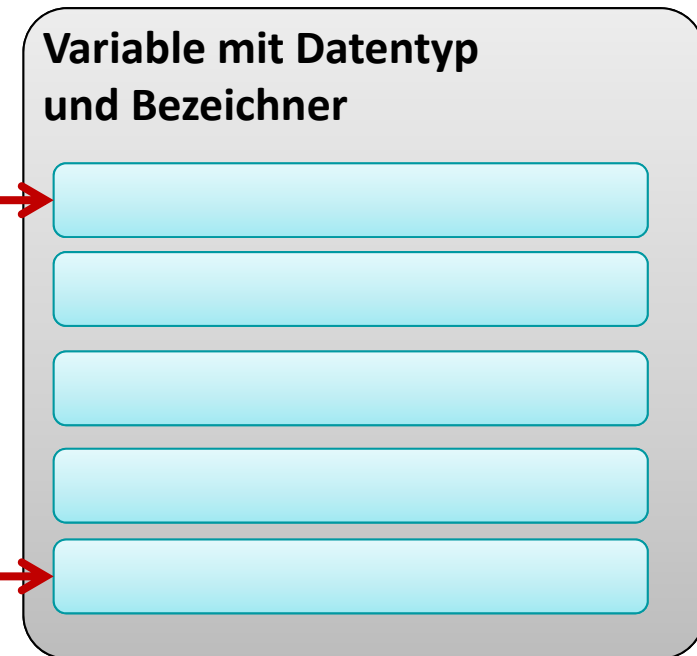




# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

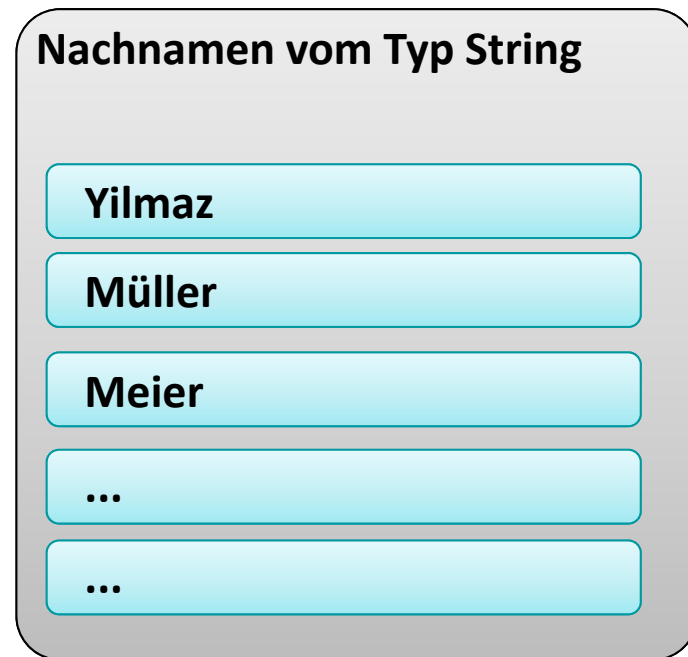




# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze





# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

```
Sub Bsp61a()
```

```
Dim strNamen(4) As String
```

```
Let strNamen(0) = "Yilmaz"
```

```
Let strNamen(1) = "Müller"
```

```
Let strNamen(2) = "Meier"
```

```
Let strNamen(3) = "Muster"
```

```
Let strNamen(4) = "Berg"
```

```
Debug.Print strNamen(0)
```

```
Debug.Print strNamen(1)
```

```
Debug.Print strNamen(2)
```

```
End Sub
```



# Einfache Felder

## Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

```
Sub Bsp61a()  
  
    Dim strNamen(4) As String  
  
    Let strNamen(0) = "Yilmaz"  
  
    Let strNamen(1) = "Müller"  
  
    Let strNamen(2) = "Meier"  
  
    '...  
  
    '...  
  
    Debug.Print strNamen(0)  
    Debug.Print strNamen(1)  
    Debug.Print strNamen(2)  
  
End Sub
```



# Einfache Felder

Felder (Array) können benutzt werden, um

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

```
Sub Bsp61a ()
    Dim strNamen(3) As String

    Let strNamen(0) = "Yilmaz"
    Let strNamen(1) = "Müller"
    Let strNamen(2) = "Meier"

    Debug.Print strNamen(0)
    Debug.Print strNamen(1)
    Debug.Print strNamen(2)
End Sub
```

Direktbereich

```
Yilmaz
Müller
Meier
```

# Einfache Felder: Beispiel 06.02.



## Ziel

- einfache Felder verwenden, um mehrere Werte zu speichern und zu lesen

## Aufgabe

- Schreiben Sie ein Programm in dem Sie die Tageshöchst- und Tiefstwerte der Temperaturen des heutigen und der folgenden 6 Tage verarbeiten
  - Deklarieren Sie hierfür zwei Felder mit geeigneten Datentypen
  - Weisen Sie den Felder selbst gewählte Werte zu
  - Geben Sie alle Werte der Felder aus, nutzen Sie hierfür eine Zählerschleife.
  - Erweitern Sie die Ausgabe um die Differenz zwischen Höchst- und Tiefsttemperatur



# Einfache Felder



**Felder (Array) können benutzt werden, um**

- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln über einen Index anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

Liste/Feld  
Index

$i_0$	$i_1$	$i_2$	...	$i_{n-1}$	$i_n$
0	1	2	...	n-1	n

## ' Generelle Syntax

```
Dim <Bez>(<n>) As <DTyp>
```

```
Let <Bez>(0) = <WertAusd>
```

```
Let <Bez>(1) = <WertAusd>
```

```
' ...
```

## ' Beispiel

```
Dim strFeld(2) As String
```

```
Let strFeld(0) = "Wert 1"
```

```
Let strFeld(1) = "Wert 2"
```

```
Let strFeld(2) = "Wert 3"
```

```
Debug.Print strFeld(1)
```





# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick





# Inhalt

## Rückblick

## Ausgangspunkt

## Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## Maps

## Abschluss und Ausblick



# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick



# Dynamische Felder

## Obergrenze und Untergrenze

- Feld verfügt stets über eine Ober- und eine Untergrenze
- in VBA bei Deklaration Angabe beider Grenzen möglich



- Syntax

```
' Generelle Syntax (wobei  $n \leq m$ )  
Dim <Bezeichner>(<n> To <m>) As <Datentyp>
```

- Beispiel

```
' Beispiel  
Dim strFeld(2 To 4) As String  
  
Let strFeld(3) = "Wert 2"  
Debug.Print strFeld(3)
```



# Dynamische Felder

## Obergrenze und Untergrenze

– können durch Hilfsfunktionen ermittelt werden

```
' Beispiel
Dim strFeld(2 To 4) As String

' Untergrenze (Lower Bound)
Debug.Print LBound(strFeld)

' Obergrenze (Upper Bound)
Debug.Print UBound(strFeld)
```

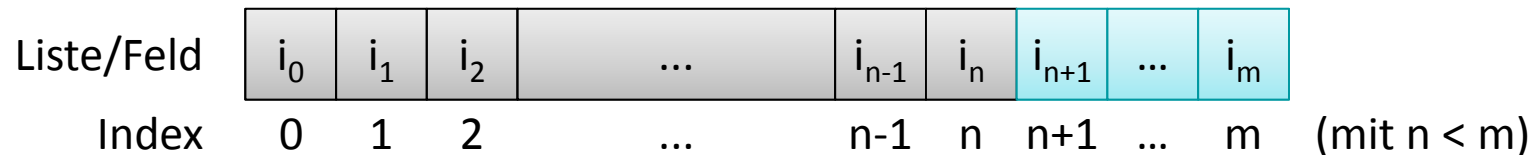
```
2
4
```



# Dynamische Felder

## Erweiterung des Feldes

- Ober- und Untergrenze legen mögliche Speicherplätze fest
- Erweiterung um zusätzliche Speicherplätze möglich



### – Syntax

- Vorhandene Inhalte werden bei Vergrößerung gelöscht

```
Dim <Bezeichner>() As <Datentyp>  
ReDim <Bezeichner>( <n> )
```

- Vorhandene Inhalte bleiben bei Vergrößerung mit Preserve erhalten

```
ReDim Preserve <Bezeichner>( <m> )
```

# Dynamische Felder: Beispiel 06.03



## Ziel

- Verwenden von Ober- und Untergrenzen eines Feldes
- Größenänderung des Feldes

## Aufgabe

- Schreiben Sie ein Programm mit dem Sie Kunden mit Ihren Kundennummern (im Bereich von 1000 bis 4999) verarbeiten können
- Deklarieren Sie dazu ein Feld
  - vom Typ String, in dem Sie die Namen der Kunden speichern können
  - mit Untergrenze 1000 und Obergrenze 4999
- Erfassen Sie Kundennamen für die Kundennummern 1000 bis 1002
- Erweitern Sie die Obergrenze des Feldes auf 9999. Erfassen Sie den Kunden Nr. 9999.
- Geben Sie erneut den Kunden mit der Kundennummer 1002 aus.

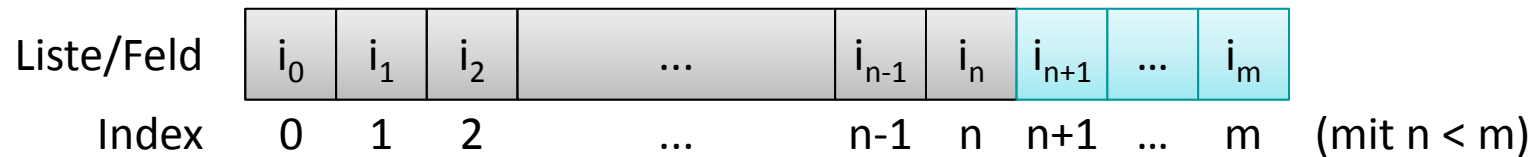




# Dynamische Felder

## Größenanpassung des Feldes

- Ober- und Untergrenze legen mögliche Speicherplätze fest
- Erweiterung um zusätzliche Speicherplätze möglich



### – Syntax

- Vorhandene Inhalte werden bei Vergrößerung gelöscht

```
Dim <Bezeichner>() As <Datentyp>  
ReDim <Bezeichner>( <n> )
```

- Vorhandene Inhalte bleiben bei Vergrößerung erhalten

```
ReDim Preserve <Bezeichner>( <m> )
```





# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick





# **Inhalt**

## **Rückblick**

## **Ausgangspunkt**

## **Zusammengesetzte Datentypen**

## **Felder**

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## **Maps**

## **Abschluss und Ausblick**



# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick



# Mehrdimensionale Felder

## Einfache Felder als Liste von einzelnen Werten

Liste/Feld	$i_0$	$i_1$	$i_2$	...	$i_{n-1}$	$i_n$
Index	0	1	2	...	n-1	n

## Mehrdimensionale Felder als Matrix

– Feld mit Zeilen und Spalten, d.h. zwei Dimensionen

	Index	0	1	2	...	n-1	n
Mehrdimensionales Feld	0	$i_{0,0}$	$i_{0,1}$	$i_{0,2}$	...	$i_{0,n-1}$	$i_{0,n}$
	1	$i_{1,0}$	$i_{1,1}$	$i_{1,2}$	...	$i_{1,n-1}$	$i_{1,n}$
	...	...	...	...	...	...	...
	m	$i_{m,0}$	$i_{m,1}$	$i_{m,2}$	...	$i_{m,n-1}$	$i_{m,n}$

– Felder mit mehr als zwei Dimensionen möglich (ab vier schwer vorstellbar)

# Mehrdimensionale Felder



## Syntax

### ' Zweidimensionales Feld

```
Dim <Bezeichner>(<n>, <m>) As <Datentyp>
```

### ' Mehrdimensionales Feld

```
Dim <Bezeichner>(<n>, <m>, ...) As <Datentyp>
```

## Beispiel

```
Dim strTicTacToe(2,2) As String
```

```
Let strTicTacToe(0,2) = "X"
```

```
Let strTicTacToe(0,0) = "O"
```

```
Let strTicTacToe(2,0) = "X"
```

```
Let strTicTacToe(1,1) = "O"
```

```
Let strTicTacToe(2,2) = "X"
```

```
' ...
```

```
' Wer gewinnt?
```

strTicTacToe(zeile, spalte):

O		X
	O	
X		X

# Mehrdimensionale Felder: Beispiel 06.04



## Ziel

- Einsatz mehrdimensionaler Felder

## Aufgabe

- Schreiben Sie eine Prozedur, die den Umsatz der ersten drei Geschäftsjahre für jeweils vier Quartale
- Deklarieren Sie ein zweidimensionales Feld eines geeigneten Datentypen
- Initialisieren Sie das Feld mit selbst gewählten Umsatzzahlen
- Geben Sie die Umsatzzahlen mit Hilfe von Zählerschleifen im Direktbereich aus
- Erweiterung: Bilden Sie die Jahressummen und geben Sie diese ebenfalls aus





# Mehrdimensionale Felder

## Speichern Daten als Matrix, z.B. mit Zeilen und Spalten

	Index	0	1	2	...	n-1	n
Mehrdimensionales Feld	0	$i_{0,0}$	$i_{0,1}$	$i_{0,2}$	...	$i_{0,n-1}$	$i_{0,n}$
	1	$i_{1,0}$	$i_{1,1}$	$i_{1,2}$	...	$i_{1,n-1}$	$i_{1,n}$
	...	...	...	...	...	...	...
	m	$i_{m,0}$	$i_{m,1}$	$i_{m,2}$	...	$i_{m,n-1}$	$i_{m,n}$

mehr als zwei Dimensionen möglich

## Syntax und Beispiel

```
' Mehrdimensionales Feld  
Dim <Bez>(<n>, <m>, ...) As <DTyp>
```

```
Dim strTtt(2,2) As String  
  
Let strTtt(0,2) = "X"  
Let strTtt(0,0) = "O"  
Let strTtt(2,0) = "X"  
Let strTtt(1,1) = "O"  
Let strTtt(2,2) = "X"  
' ...
```



# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

## Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick







# **Inhalt**

## **Rückblick**

## **Ausgangspunkt**

## **Zusammengesetzte Datentypen**

## **Felder**

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## **Maps**

## **Abschluss und Ausblick**



# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

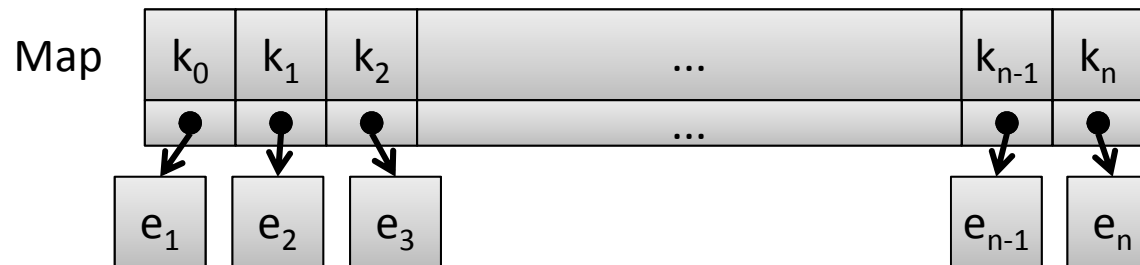
Abschluss und Ausblick



# Map

## Map in Form der VBA-Collection

- dient der Speicherung von Datenelementen auf die anhand eines eindeutigen Schlüssels zugegriffen werden kann



- Generelle Syntax für Deklaration und Initialisierung

```
' Deklaration  
Dim <Bezeichner> As Collection  
' Initialisierung (Schlüsselwort Set beachten!)  
Set <Bezeichner> = New Collection
```

- Beispiel

```
Dim colKunden As Collection  
Set colKunden = New Collection
```

# Map



## Map in Form der VBA-Collection

– Generelle Syntax für Zugriffe

```
' Hinzufügen
<CollectionBezeichner>.Add <WertAusdObj>, <KeyAlsString>
' Lesen
<CollectionBezeichner>.Item(<KeyAlsString>)
' Entfernen
<CollectionBezeichner>.Remove(<KeyAlsString>)
```

– Beispiel

```
' ...
colKunden.Add "Mike Müller", "KndNr0815"
colKunden.Add "Ali Yilmaz", "KndNr4711"
colKunden.Add "Beate Meier", "KndNr9021"

Debug.Print colKunden.Item("KndNr4711")

colKunden.Remove("KndNr9021")
```

# Maps: Beispiel 06.05



## Ziel

- Nutzung einer Map in Form einer VBA-Collection

## Aufgabe: Freunden anhand von E-Mailadresse ermitteln

- Deklaration einer Collection und Initialisierung mit einem neuen Objekt
- Hinzufügen der Namen von Freunden zur Collection, dabei deren E-Mailadresse als Schlüssel verwenden
- Vom Benutzer (mit einer InputBox) eine E-Mailadresse erfragen
- Ermitteln des zur eingegebenen E-Mailadressen gehörigen Namens
- Ausgabe des Namens im Meldungsfenster

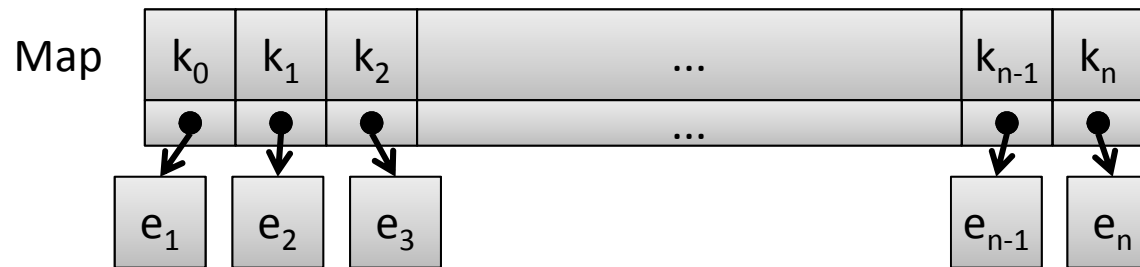




# Map

## Map in Form der VBA-Collection

- dient der Speicherung von Datenelementen auf die anhand eines eindeutigen Schlüssels zugegriffen werden kann



- Generelle Syntax für Deklaration und Initialisierung

' Deklaration

```
Dim <Bezeichner> As Collection
```

' Initialisierung

```
Set <Bezeichner> = New Collection
```

```
Dim colKnd As Collection
```

```
Set colKnd = New Collection
```

```
colKnd.Add "Müller", "K1"
```

```
colKnd.Add "Yilmaz", "K4"
```

```
colKnd.Add "Meier", "K2"
```

```
Debug.Print colKnd.Item("K4")
```

```
colKnd.Remove("K2")
```

- Generelle Syntax für Zugriffe

' Hinzufügen, Lesen und Entfernen

```
<CollectionBezeichner>.Add <WertAusdr>
```

```
<CollectionBezeichner>.Item(<KeyAlsAusdr>)
```

```
<CollectionBezeichner>.Remove(<KeyAlsAusdr>)
```



# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick





# **Inhalt**

## **Rückblick**

## **Ausgangspunkt**

## **Zusammengesetzte Datentypen**

## **Felder**

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

## **Maps**

## **Abschluss und Ausblick**





# Inhalt

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

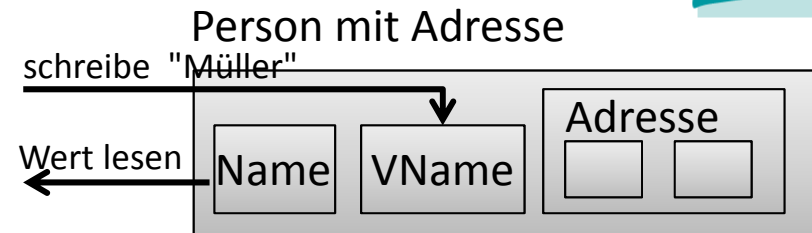
**Abschluss und Ausblick**



# Abschluss

## Zusammengesetzte Datentypen

- fassen mehrere Eigenschaften definierter Datentypen zusammen
- Repräsentieren häufig Dinge der Realität, z.B. "Person" mit Eigenschaften "Name", "Vorname" und "Adresse"
- Werden als Type definiert und zur Deklaration von Variablen benutzt
- Zugriff auf einzelne Elemente der Variable des zusammengesetzten Datentypen über Punkt-Notation möglich (Lesen, Schreiben)



### ' Generelle Syntax

```
Type <Typbezeichner>  
  <Eigenschaft1> As <Datentyp>  
  <Eigenschaft2> As <Datentyp>  
  ...  
End Type
```

### ' Definition

```
Type TPerson  
  strName As String  
  adrWohnanschrift As TAdresse  
End Type
```

### ' Deklaration und Nutzung

```
Dim perTom As TPerson  
Let perTom.strName = "Tom"  
Debug.Print perTom.strName
```

# Abschluss



## Einfache Felder (Array)

Liste/Feld

Index

$i_0$	$i_1$	$i_2$	...	$i_{n-1}$	$i_n$
0	1	2	...	n-1	n

- speichern mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln über einen Index anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

```
' Generelle Syntax  
Dim <Bez>( <n> ) As <DTyp>  
  
Let <Bez>(0) = <WertAusd>  
Let <Bez>(1) = <WertAusd>  
' ...
```

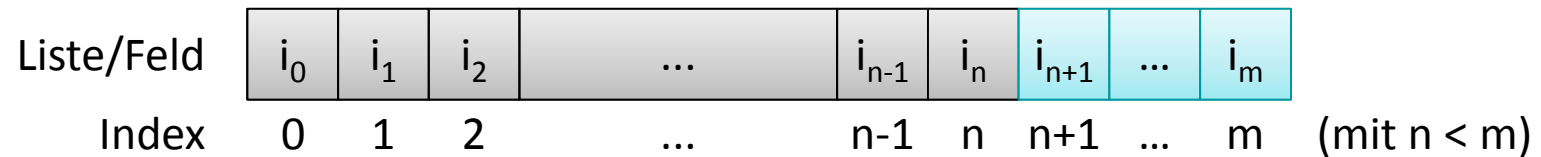
```
' Beispiel  
Dim strFeld(2) As String  
  
Let strFeld(0) = "Wert 1"  
Let strFeld(1) = "Wert 2"  
Let strFeld(2) = "Wert 3"  
  
Debug.Print strFeld(1)
```



# Abschluss

## Dynamische Erweiterung des Feldes

- Ober- und Untergrenze legen mögliche Speicherplätze fest
- Erweiterung um zusätzliche Speicherplätze möglich



### – Syntax

- Vorhandene Inhalte werden bei Vergrößerung gelöscht

```
Dim <Bezeichner>() As <Datentyp>  
ReDim <Bezeichner>(<n>)
```

- Vorhandene Inhalte bleiben bei Vergrößerung erhalten

```
ReDim Preserve <Bezeichner>(<m>)
```

# Abschluss



## Mehrdimensionale Felder

- speichern Daten als Matrix, z.B. mit Zeilen und Spalten

	Index	0	1	2	...	n-1	n
Mehrdimensionales Feld	0	$i_{0,0}$	$i_{0,1}$	$i_{0,2}$	...	$i_{0,n-1}$	$i_{0,n}$
	1	$i_{1,0}$	$i_{1,1}$	$i_{1,2}$	...	$i_{1,n-1}$	$i_{1,n}$
	...	...	...	...	...	...	...
	m	$i_{m,0}$	$i_{m,1}$	$i_{m,2}$	...	$i_{m,n-1}$	$i_{m,n}$

- mehr als zwei Dimensionen möglich
- Syntax

### ' Mehrdimensionales Feld

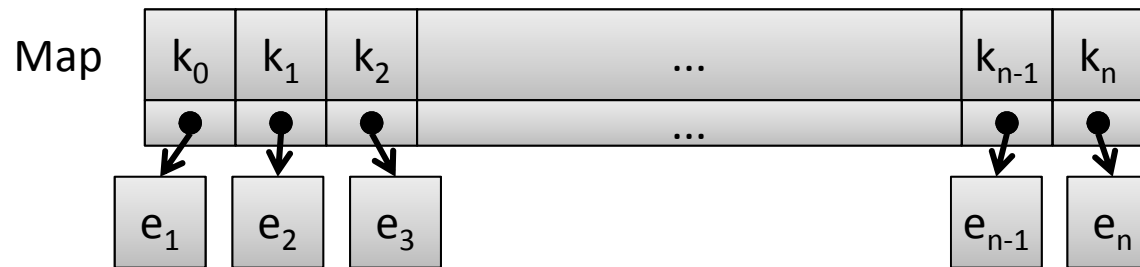
**Dim** <Bezeichner>( <n>, <m>, ... ) **As** <Datentyp>



# Abschluss

## Map in Form der VBA-Collection

- dient der Speicherung von Datenelementen auf die anhand eines eindeutigen Schlüssels zugegriffen werden kann



- Generelle Syntax für Deklaration und Initialisierung

```
' Deklaration  
Dim <Bezeichner> As Collection  
' Initialisierung  
Set <Bezeichner> = New Collection
```

```
Dim colKnd As Collection  
Set colKnd = New Collection
```

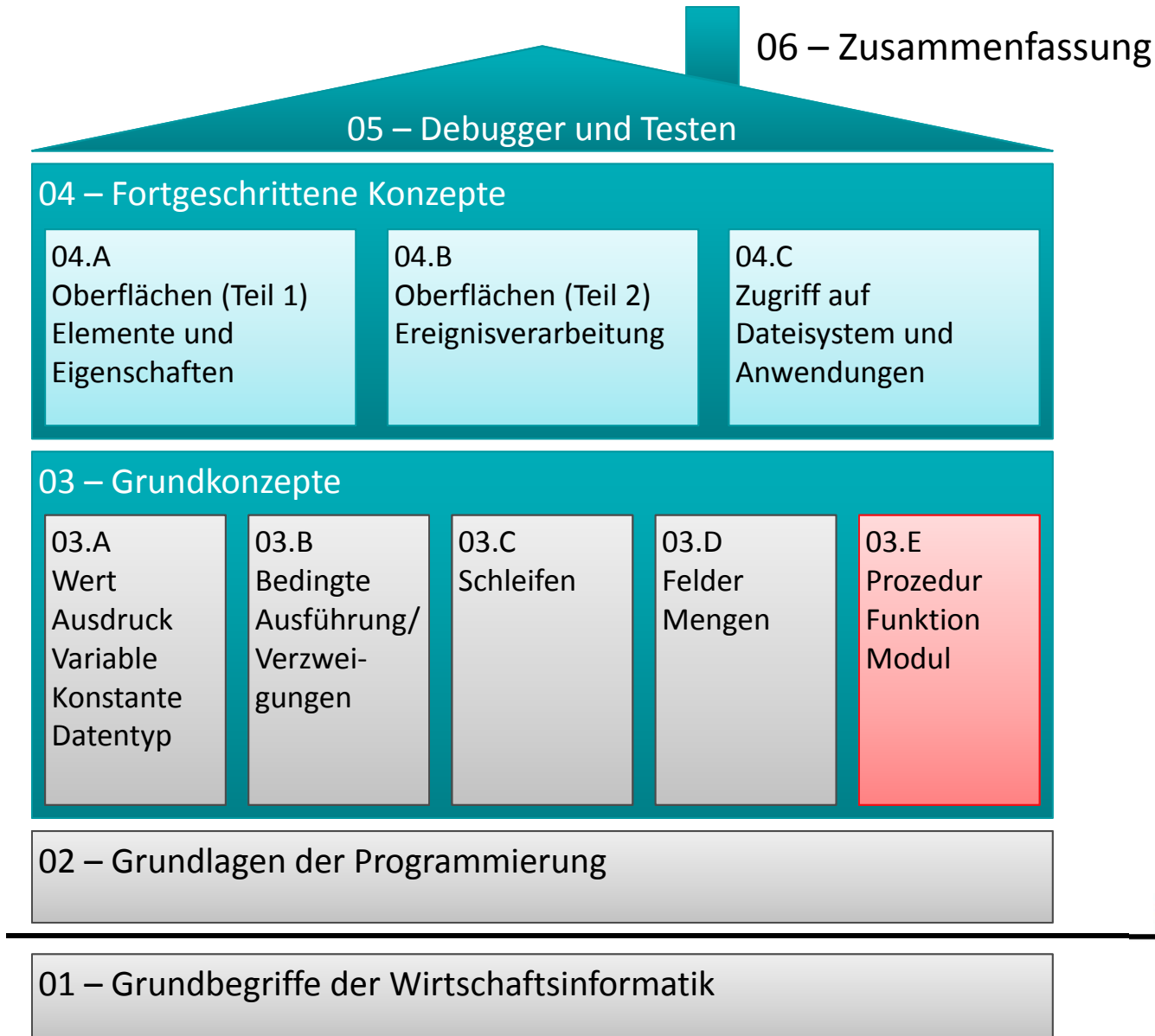
- Generelle Syntax für Zugriffe

```
' Hinzufügen, Lesen und Entfernen  
<CollectionBezeichner>.Add <WertAusdr>  
<CollectionBezeichner>.Item(<KeyAls>  
<CollectionBezeichner>.Remove(<KeyAls>
```

```
colKnd.Add "Müller", "K1"  
colKnd.Add "Yilmaz", "K4"  
colKnd.Add "Meier", "K2"
```

```
Debug.Print colKnd.Item("K4")  
colKnd.Remove("K2")
```

# Ausblick



# Quellen

[1] Produktfotos: <http://www.ikea.de>







BEUTH HOCHSCHULE FÜR TECHNIK BERLIN  
University of Applied Sciences

# **Wirtschaftsinformatik 1**

## **LE 06 – Felder, Mengen und zusammengesetzte Datentypen**

Prof. Dr. Thomas Off

<http://www.ThomasOff.de/lehre/beuth/wi1>