BHT Berliner Hochschule für Technik

Wirtschaftsinformatik 1
LE 06 – Felder, Mengen und zusammengesetzte Datentypen

Prof. Dr. Thomas Off
http://www.ThomasOff.de/lehre/beuth/wi1

1

Einordnung

2

Inhalt

3

Inhalt

4

Rückblick

BHT

5

Rückblick

BHT

Vorprüfende/Kopfgesteuerte *Schleife*

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

```
Wdhl. solange Bedingung
    Anweisung
```

- 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
```


6

Rückblick

BHT

Nachprüfende/Fußgesteuerte *Schleife*

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

```
    Anweisung
Wdhl. solange Bedingung wahr
```

- 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
```


7

Rückblick

BHT

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
```

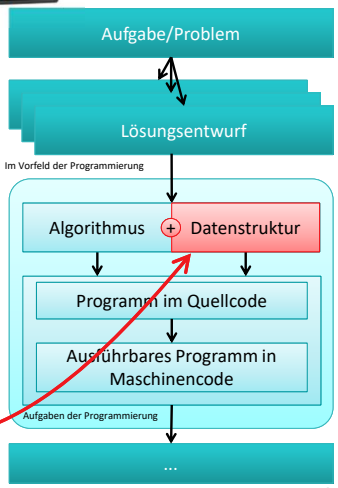

8

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

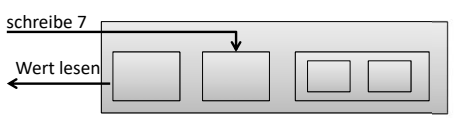


9

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)



10

Rückblick (LE02)

Datenstrukturen

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

11

Rückblick (LE02)

Sequentielle Liste/Feld (Array)

- Lesender Zugriff über Index frei möglich

12

Rückblick (LE02)

BHT

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen

13

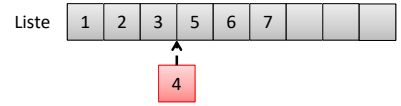
Rückblick (LE02)

BHT

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen



14

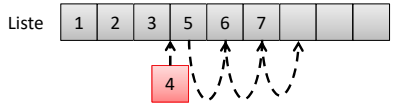
Rückblick (LE02)

BHT

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen



15

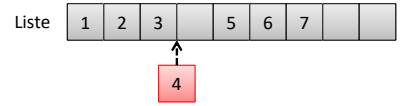
Rückblick (LE02)

BHT

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen



16

Rückblick (LE02)

BHT

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen

17

Rückblick (LE02)

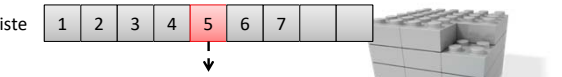
BHT

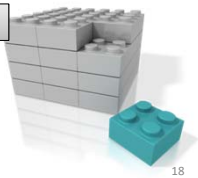
Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen

– Entfernen





18

Rückblick (LE02)

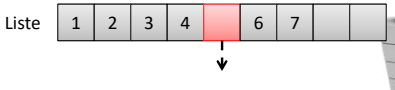
BHT

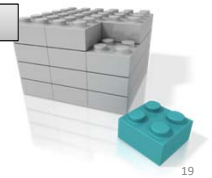
Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen

– Entfernen





19

Rückblick (LE02)

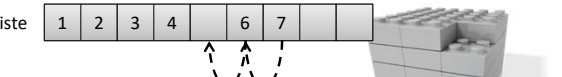
BHT

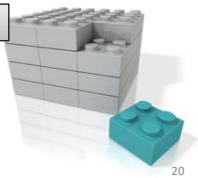
Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

– Einfügen

– Entfernen





20

Rückblick (LE02)

BHT

Sequentielle Liste/Feld (Array)

- Lesender Zugriff über Index frei möglich

- Einfügen

- Entfernen

21

Rückblick (LE02)

BHT

Datenstrukturen

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

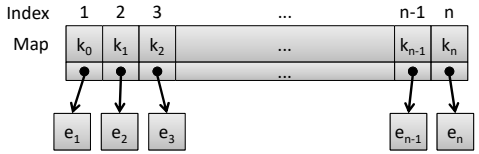
22

Rückblick (LE02)

BHT

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

23

Rückblick (LE02)

BHT

Datenstrukturen

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

24

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

25

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

26

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

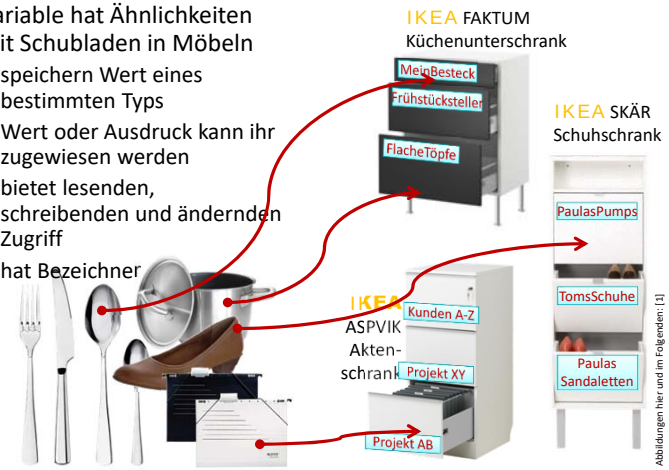
Abschluss und Ausblick

27

Ausgangspunkt

BHT

- 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



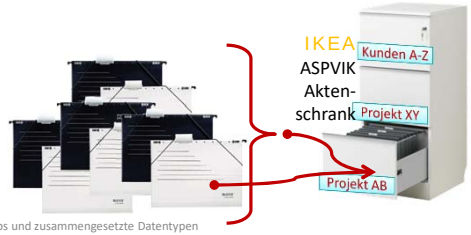
28

Ausgangspunkt

BHT

... 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



29

Ausgangspunkt

BHT

... und außerdem

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

30

Ausgangspunkt

BHT

... oder geht das doch?

31

Ausgangspunkt

BHT

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
```


32

Ausgangspunkt

BHT

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
```


33

Ausgangspunkt

BHT

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
```


34

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?

35

Ausgangspunkt

BHT

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.

36

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

37

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

38

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

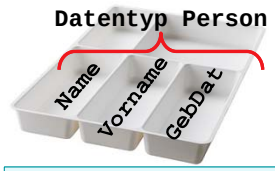
39

Zusammengesetzte Datentypen

BHT

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
```

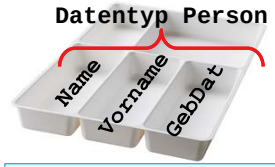

40

Zusammengesetzte Datentypen

BHT

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
```

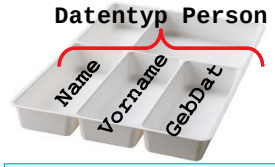

41

Zusammengesetzte Datentypen

BHT

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
```

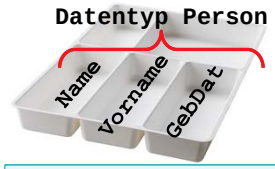

42

Zusammengesetzte Datentypen

BHT

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
```

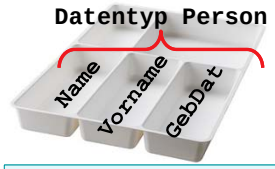

43

Zusammengesetzte Datentypen

BHT

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
```


44

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 perErste As TPerson
 Dim perZweite As TPerson

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

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

 ' ...
End Sub
```


45

Zusammengesetzte Datentypen: Beispiel 06.01

Ziel

- Definition und Nutzung eines zusammengesetzten Datentypen

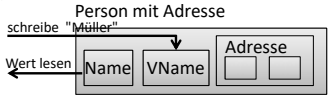
Aufgabe

- Definieren Sie einen Datentyp für Kunden mit
  - Kundenummer
  - 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

46

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 Variablenbenutzt
- 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
```


50

Inhalt

51

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

52

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

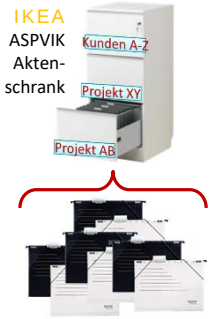
53

Einfache Felder

BHT

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



54

Einfache Felder

BHT

Felder (Array) können benutzt werden, um

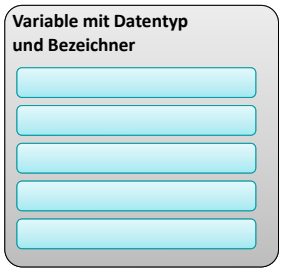
- mehrere Werte des gleichen Datentyps
- unter einem gemeinsamen Namen (Bezeichner) zu speichern
- jeden Wert einzeln anzusprechen
- innerhalb eines Bereichs zwischen Untergrenze und Obergrenze

55

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

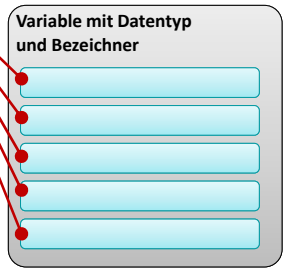


56

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

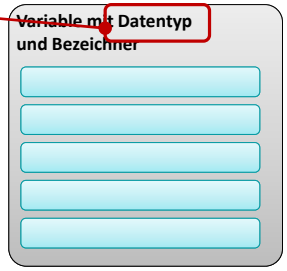


57

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

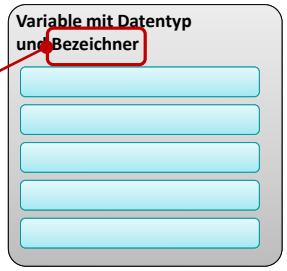


58

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

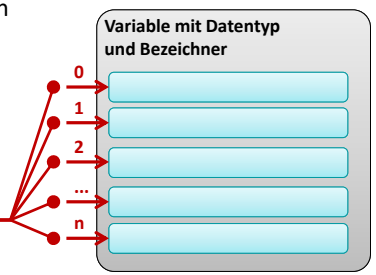


59

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

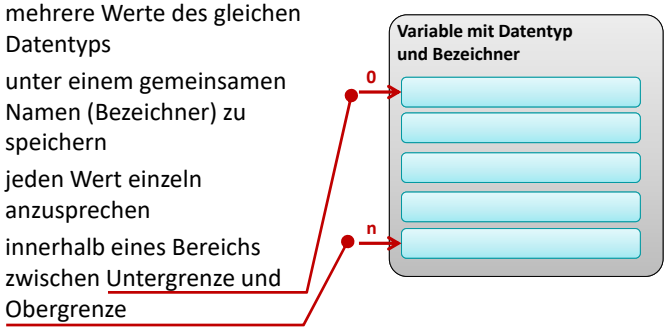


60

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

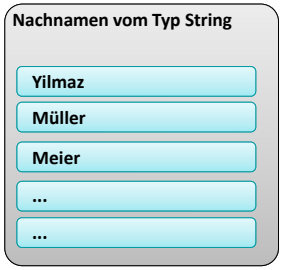


61

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



62

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
```


63

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

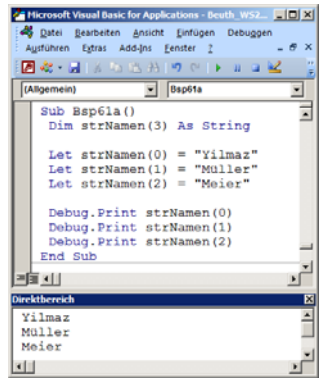
```
Sub Bsp61a()

 Dim strNamen(4) As String


 Let strNamen(0) = "Yilmaz"

 Let strNamen(1) = "Müller"

 Let strNamen(2) = "Meier"

 Let strNamen(3) = "Meier"

 Let strNamen(4) = "Meier"


 Debug.Print strNamen(0)
 Debug.Print strNamen(1)
 Debug.Print strNamen(2)

End Sub
```


64

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



65

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

66

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 Bei der Deklaration wird die Anzahl Elemente angegeben


```
' Generelle Syntax
Dim <Bez>(<n>) As <DTyp>
```

Obergrenze, nicht Anzahl der Elemente

```
Let <Bez>(0) = <WertAusd>
Let <Bez>(1) = <WertAusd>
'...
```

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

3 Elemente

```
Let strFeld(0) = "Wert 1"
Let strFeld(1) = "Wert 2"
Let strFeld(2) = "Wert 3"

Debug.Print strFeld(1)
```


67

Erweiterte Möglichkeiten bei Feldern

BHT

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 <= 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)
```


68

Erweiterte Möglichkeiten bei Feldern

BHT

Obergrenze und Untergrenze

- können durch Hilfsunktionen 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
```


69

Inhalt

BHT

70

Inhalt

BHT

71

Inhalt

BHT

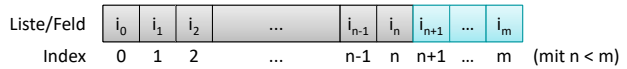
72

Dynamische Felder

BHT

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>)
```

Obergrenze des Feldes n

  • Vorhandene Inhalte bleiben bei Vergrößerung mit Preserve erhalten

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

Neue Obergrenze des Feldes m

73

Dynamische Felder: Beispiel 06.03

BHT

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 dynamisches 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 Kundenummer 1002 aus.

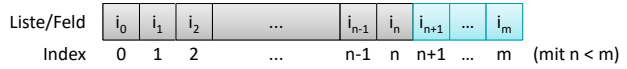
74

Dynamische Felder

BHT

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>)
```

Obergrenze des Feldes n

  • Vorhandene Inhalte bleiben bei Vergrößerung erhalten

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

Neue Obergrenze des Feldes m

75

Inhalt

BHT

Rückblick
Ausgangspunkt
Zusammengesetzte Datentypen
Felder
– Einfache Felder
– Dynamische Felder
– Mehrdimensionale Felder
Maps
Abschluss und Ausblick

76

Inhalt

BHT

Rückblick
Ausgangspunkt
Zusammengesetzte Datentypen
Felder
– Einfache Felder
– Dynamische Felder
– Mehrdimensionale Felder
Maps
Abschluss und Ausblick

77

Inhalt

BHT

Rückblick
Ausgangspunkt
Zusammengesetzte Datentypen
Felder
– Einfache Felder
– Dynamische Felder
– Mehrdimensionale Felder
Maps
Abschluss und Ausblick

78

Mehrdimensionale Felder

BHT

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)

79

Mehrdimensionale Felder

BHT

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

80

Mehrdimensionale Felder: Beispiel 06.04

BHT

Ziel

- Einsatz mehrdimensionaler Felder

Aufgabe

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

81

Mehrdimensionale Felder

BHT

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"
' ...
```


82

Inhalt

BHT

83

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

84

Inhalt

BHT

Rückblick

Ausgangspunkt

Zusammengesetzte Datentypen

Felder

- Einfache Felder
- Dynamische Felder
- Mehrdimensionale Felder

Maps

Abschluss und Ausblick

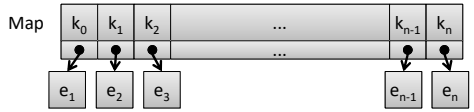
85

Map

BHT

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
```


86

Map

BHT

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")
```


87

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

88

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
```

– Generelle Syntax für Zugriffe

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

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

Debug.Print colKnd.Item("K4")

colKnd.Remove("K2")
```


89

Inhalt

90

Inhalt

91

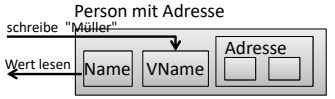
Inhalt

92

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 Variablenbenutzt
- 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
```


93

Abschluss

Einfache Felder (Array)

- 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
- Bei der Deklaration wird die Anzahl Elemente angegeben


```
' Generelle Syntax
Dim <Bez>(<n>) As <DTyp>
```

Obergrenze, nicht Anzahl der Elemente

```
Let <Bez>(0) = <WertAusd>
Let <Bez>(1) = <WertAusd>
'...
```

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

3 Elemente

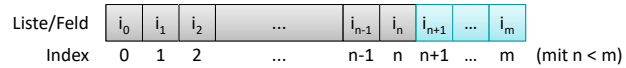
```
Let strFeld(0) = "Wert 1"
Let strFeld(1) = "Wert 2"
Let strFeld(2) = "Wert 3"

Debug.Print strFeld(1)
```


94

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>)
```

Obergrenze des Feldes n

  - Vorhandene Inhalte bleiben bei Vergrößerung erhalten

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

Neue Obergrenze des Feldes m

95

Abschluss

Mehrdimensionale Felder

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

Mehrdimensionales Feld	Index	0	1	2	...	n-1	n
	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>
```


96

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
```

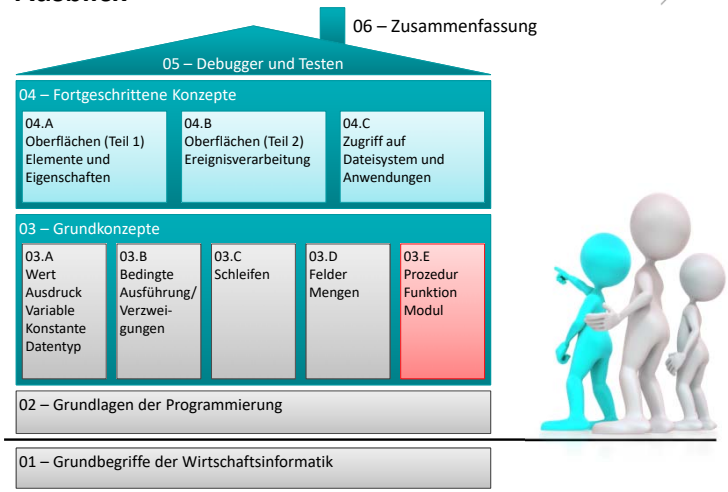
– Generelle Syntax für Zugriffe

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

```
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")
```


97

Ausblick



98

Quellen

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

99

BHT Berliner Hochschule für Technik

Wirtschaftsinformatik 1
LE 06 – Felder, Mengen und zusammengesetzte Datentypen

Prof. Dr. Thomas Off
http://www.ThomasOff.de/lehre/beuth/wi1

100