Programmieren 1

Wintersemester 2016/2017

Marcus Riemer, B.Sc.

Basierend auf den Unterlagen „Programmstrukturen 1“
von Prof. Dr. Andreas Häuslein

12 Benutzerdefinierte strukturierte Datentypen

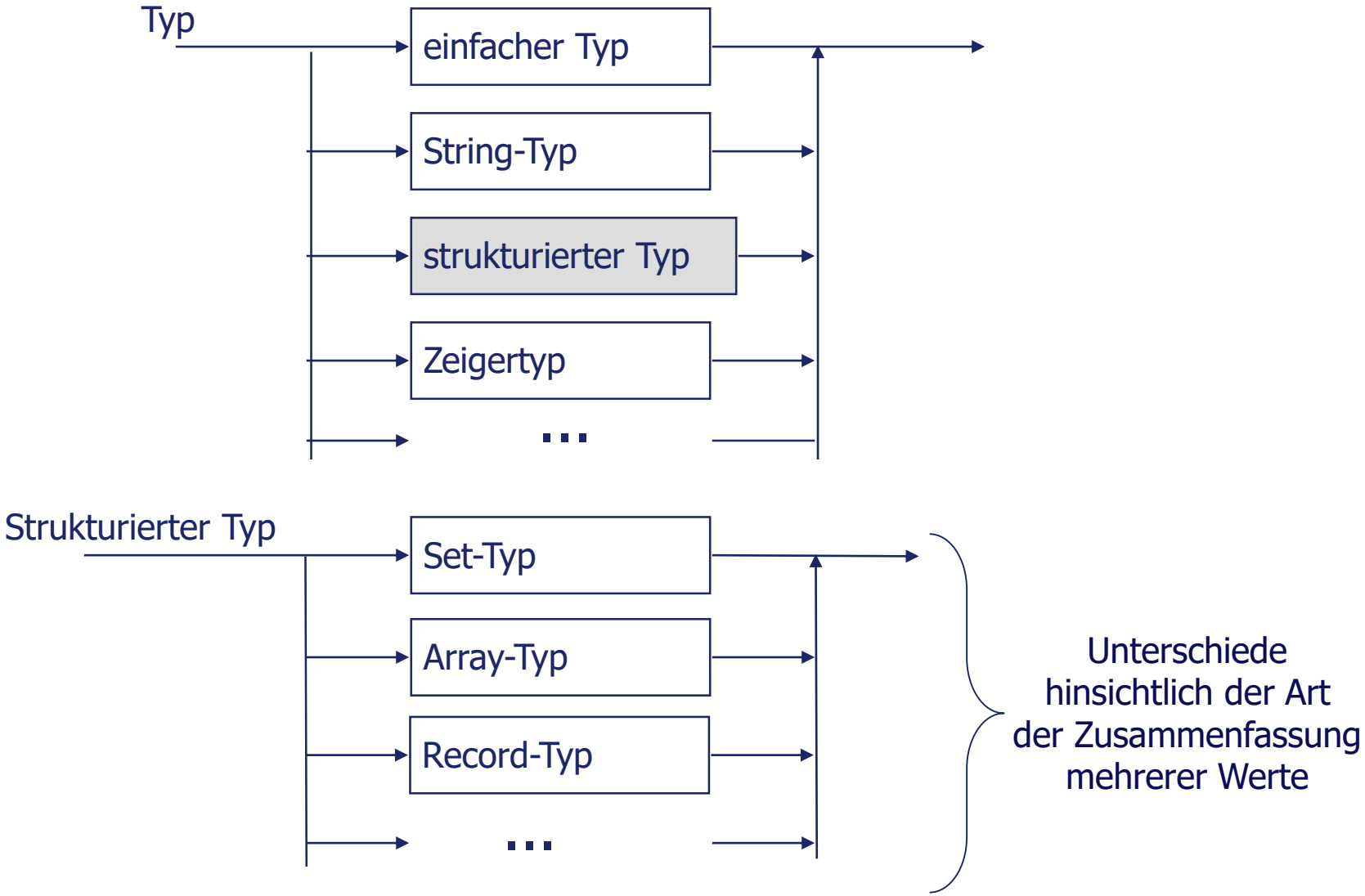
- Merkmale strukturierter Datentypen
  - Jeder Wert des Wertebereichs ist aus mehrere Werten eines anderen Datentyps/mehrerer anderen Datentypen zusammengesetzt
  - Strukturierte Datentypen erlauben die Speicherung/Verarbeitung mehrerer Werte als Einheit, da die mehreren Werte in einem Wert zusammengefasst sind
  - Strukturierte Datentypen sind immer benutzerdefiniert

12 Benutzerdefinierte strukturierte Datentypen

- Strukturierte Datentypen sind immer dann sinnvoll, wenn
  - statt einem Wert zu einem Sachverhalt mehrere Werte anzugeben/zu speichern/zu verarbeiten sind
  - zwischen mehreren Werten eine inhaltlich/fachlich geprägte Verbindung besteht
  - Beispiele:
    - Nicht nur ein Freund (ein Wert des Datentyps `TPersonen`), sondern mehrere Freunde (eine Menge von Freunden).
    - Nicht nur ein Messwert (Wert des Datentyps `Double`), sondern eine Liste von Messwerten
    - Nicht nur einzelne, separate Angaben zu Merkmalen eines Buchs, sondern alle relevanten Aspekte eines Buches verbunden zu *einem* Datensatz (z.B. Autor, Titel, Verlag, Erscheinungsjahr...)

12 Benutzerdefinierte strukturierte Datentypen

- Syntaktische Einordnung



12 Array-Datentyp

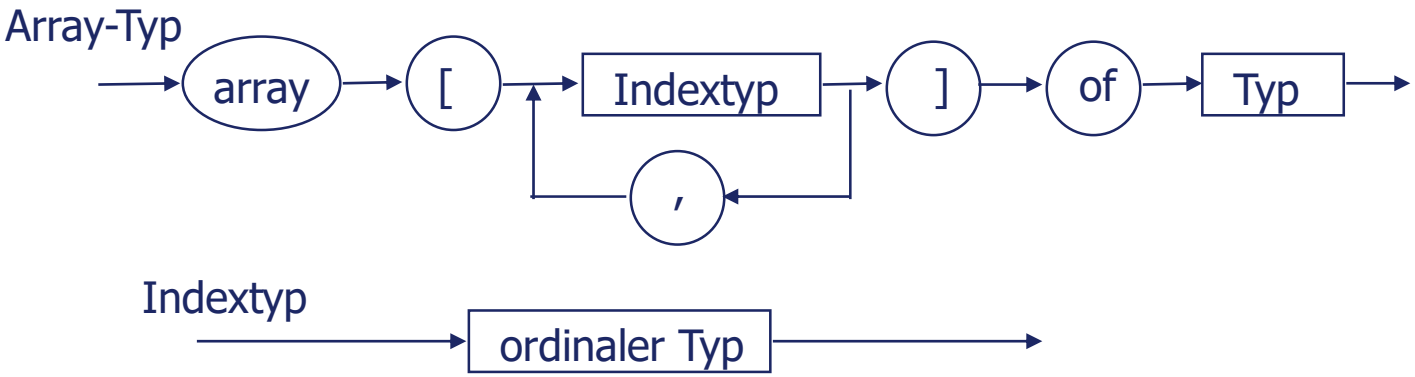
- Ein Array-Datentyp fasst eine feste Anzahl von Werten eines anderen Datentyps (Basistyp) zu einem neuen Datentyp zusammen
- Jeder einzelne Wert eines Array-Datentyps besteht aus der festgelegten Anzahl von Werten des Basistyps
- Jeder Wert des Basistyps in einem Array (Array-Element) ist einem Indexwert zugeordnet
- Jeder einzelne Wert in einem Array kann über seinen Indexwert gezielt angesprochen werden (lesend und schreibend)
- Beispiel für *einen* Array-Wert (für Array-Typ mit 6 Elementen, Basistyp `real`, Indexwerte von 0 bis 5) als bildliche Darstellung:

Indexwerte	0	1	2	3	4	5
Zusammengefasste Werte des Basistyps	3.2	27.9	-0.25	3.2	25.5E01	72.0

1 Wert des Array-Typs

12 Array-Datentyp

- Zur Definition eines Array-Typs müssen angegeben werden:
  - Wertebereich(e) der Indexwerte
  - Basistyp der Elemente (beliebiger Datentyp)
- Syntax:



- Anzahl der Indextypen legt die Dimension des Array-Typs fest

12 Array-Datentyp

- Beispiele für (eindimensionale) Array-Datentypen:

```
type
     TMesswerte = array [byte] of real;

     TZahlenvektor = array [1..10] of integer;

     TNamensliste = array [1..30] of string;

     TZeichenanzahl = array ['A' .. 'Z'] of word;

     TFarbe = (rot, gruen, blau, gelb, schwarz);
     TStoff = array [1 .. 3] of TFarbe;
     TMuster = array [TFarbe] of Boolean;

     TMonate = (Januar, Februar, Maerz, April, Mai, Juni,
                Juli, August, September, Oktober,
                November, Dezember);
     TUmsaetze = array [TMonate] of currency;
```

- Deklaration von Variablen mit Array-Typen:

```
var Zahlen : TZahlenvektor;
    Gaeste : TNamensliste;
    Zeichenhaeufigkeit : TZeichenanzahl;
    MeineUmsaetze : TUmsaetze;
```

12 Array-Datentyp

- Bildliche Vorstellung von eindimensionalen Arrays (z.B. Array-Variable `Zahlen`, siehe vorherige Seite):

Zahlen

1	2	3	4	5	6	7	8	9	10
<integer>	<integer>	<integer>	<integer>	<integer>	<integer>	<integer>	<integer>	<integer>	<integer>

- Zugriff auf einzelne Array-Elemente durch explizite Angabe der Array-Variablen in Verbindung mit einem Indexwert (in eckigen Klammern), z.B.:

```
Zahlen[3] := 25;
```

Zahlen

1	2	3	4	5	6	7	8	9	10
<integer>	<integer>	25	<integer>	<integer>	<integer>	<integer>	<integer>	<integer>	<integer>

12 Array-Datentyp

- Weitere Beispiele für Zugriffe auf einzelne Elemente durch explizite Angabe des Indexwertes:

```
Gaeste[5] := 'Tom';
Zeichenhaeufigkeit['E'] := 24;
MeineUmsaetze[Mai] := 2500.50;
```

- Indexwerte, die zum Zugriff auf Array-Elemente angegeben werden, müssen zum Wertebereich des Indextyps gehören (insbes. wichtig für *berechnete* Indexwerte)
- Einzelne Elemente wie Variablen des Basistyps verwendbar
  - Für die Elemente eines Arrays stehen alle Operationen des Basistyps zur Verfügung
  - Array-Elemente müssen vor lesendem Zugriff ebenfalls initialisiert werden
- Direkte Zuweisung vollständiger Array-Werte (alle Elemente) zwischen Variablen identischer Array-Typen möglich

12 Array-Datentyp

- Weitere Beispiele für Array-Zugriff/-nutzung:

```
var  Zahlen1, Zahlen2 : TZahlenvektor;
     Index : TIndex;

begin
  Zahlen1[3] := 63;
  Zahlen1[4] := 19;
  Zahlen1[7] := 3;
  Zahlen1[4] := Zahlen1[4] + Zahlen1[7];
  writeln (Zahlen1[4]);
...
  Zahlen2 := Zahlen1;
...
  Index := 3;
  Zahlen1[Index] := Zahlen1[Index] + Zahlen1[Index + 1];
...
  Zahlen1[Index] := Zahlen1[Index] + Zahlen2[Index];
...
  Index := 10;
  Zahlen1[Index] := Zahlen1[Index] + Zahlen1[Index + 1];
```

12 Array-Datentyp

- Adäquates Programmkonstrukt zum sequenziellen Zugriff auf (alle) Array-Elemente:

for-Schleife

- Beispiel: Initialisierung aller Array-Elemente

```
var index : TIndex;
    vektor : array [TIndex] of integer;

...
for index := low(index) to high(index) do
    vektor[index] := 0;
...
```

12 Aufgabe „Ziffern Arrays“

In dieser Aufgabe sollen verschiedene Operationen auf Ziffern einer Zahl durchgeführt werden. Im Gegensatz zu den bisherigen Implementierungen der geforderten Verfahren sollen diese aber alle auf einem Array von Dezimalziffern arbeiten.

- Deklarieren Sie einen geeigneten Datentyp `TZiffer` und ein Array `TZifferArray` um alle Ziffern einer 64 Bit langen Zahl abzubilden. Deklarieren sie den Typ `TZifferIndex` als einen Teilbereichstyp und nutzen Sie diesen als Index-Typ für das geforderte Array. Dabei soll die niedrigstwertige Ziffer (also die Einerstelle) mit dem Index 1 versehen werden.
- Lesen Sie vom Benutzer eine positive Ganzzahl ein und speichern Sie deren Ziffern in einem Array `ziffern` vom Typ `TZifferArray`.
- Berechnen Sie die Anzahl der genutzten Stellen für die jeweilige Zahl. Merken Sie sich dafür das erste Vorkommen der Ziffer 0 auf das nur weitere Nullen folgen. Alle weiteren Teilaufgaben beziehen sich nur auf diese signifikanten Ziffern.

12 Aufgabe „Ziffern Arrays“

- Alle der im folgenden genannten Teilaufgaben sollen anhand des Arrays aus Ziffern gelöst werden.
  - Geben Sie die Zahl rückwärts aus. Die Eingabe `1234` soll also mit der Ausgabe `4321` beantwortet werden.
  - Berechnen Sie die Quersumme und das Querprodukt der Zahl.
  - Berechnen Sie die größte und die kleinste vorkommende Ziffer.
  - Berechnen Sie, ob es sich bei dieser Zahl um eine Schnapszahl handelt. Schnapszahlen haben ausschließlich identische Ziffern.
  - Berechnen Sie, ob es sich bei der Zahl um eine Palindromzahl handelt. Palindromzahlen lesen sich von hinten und vorne identisch.
  - Berechnen Sie die Häufigkeit jeder vorkommenden Ziffer. Nutzen Sie zur Speicherung der Häufigkeit ein weiteres Array.

12 Arrays für Aufzählungen

- In Delphi können Werte einer Aufzählung nicht unmittelbar ausgegeben werden. Ein normaler Anwender des Programms kommt mit den Aufzählungswerten nicht in Berührung.
  - Entkoppelt Implementierungssprache von Benutzersprache, ohne diese Trennung müssten Aufzählungen übersetzt werden:
    ```
    TMonat = (January, February, March, April, …)
    TMonat = (Januar, Febuar, Maerz, April, …)
    TMonat = (Janvier, Février, Mars, Avril, …)
    ```
  - Um dennoch Aufzählungswerte auszugeben, kann ein passendes Array verwendet werden:
    ```
    TMonatName = array[TMonat] of string
    ```
  - Mit diesem Array können dann die passenden Strings für alle Benutzergruppen definiert werden.

12 Arrays für Aufzählungen

```
program array_months;
{$APPTYPE CONSOLE}
{$R+,Q+,X-}
type TMonat = (Januar, Februar, Maerz, April, Mai, Juni, Juli,
               August, September, Oktober, November, Dezember);
const
   MONATE_DEUTSCH : array[TMonat] of string =
      ('Januar', 'Februar', 'März', 'April', 'Mai', 'Juni', 'Juli',
       'August', 'September', 'Oktober', 'November', 'Dezember') ;
   MONATE_ENGLISCH: array[TMonat] of string =
      ('January', 'February', 'March', 'April', 'May', 'June', 'July',
       'August', 'September', 'October', 'November', 'December') ;
var i : TMonat;
begin
   writeln('Deutsche Monate');
   for i := low(TMonat) to high(TMonat) do
      writeln(ord(i):3, ' = ', MONATE_DEUTSCH[i]);
   writeln('Englische Monate');
   for i := low(TMonat) to high(TMonat) do
      writeln(ord(i):3, ' = ', MONATE_ENGLISCH[i]);
end.
```

12 Arrays für Aufzählungen

```
program array_digits;
{$APPTYPE CONSOLE}
{$R+,Q+,X-}
type  TZiffer = 0..9;
const
   ZIFFERN_DEUTSCH : array[TZiffer] of string =
      ('Null', 'Eins', 'Zwei', 'Drei', 'Vier', 'Fünf',
       'Sechs', 'Sieben', 'Acht', 'Neun');
   ZIFFERN_ENGLISCH : array[TZiffer] of string =
      ('Zero', 'One', 'Two', 'Three', 'Four', 'Five',
       'Six', 'Seven', 'Eight', 'Nine');
var  i : TZiffer;
begin
   writeln('Deutsche Ziffern');
   for i := low(TZiffer) to high(TZiffer) do
      writeln(ord(i):3, ' = ', ZIFFERN_DEUTSCH[i]);
   writeln('Englische Ziffern');
   for i := low(TZiffer) to high(TZiffer) do
      writeln(ord(i):3, ' = ', ZIFFERN_ENGLISCH[i]);
end.
```

12 Mehrdimensionale Arrays

- Deklaration mehrdimensionaler Arrays durch Angabe entsprechend vieler Indextypen:
  - Direkt:

```
TUmsaetze = array [(Berlin, Hamburg, Leipzig, Muenchen),
                   1..52] of currency;
```

  - Mit Typbezeichner für die Indextypen (meist sinnvoller):

```
TKalenderwoche = 1..52;
TFiliale = (Berlin, Hamburg, Koeln, Leipzig, Muenchen);

TUmsaetze = array [TFiliale, TKalenderwoche] of currency;
```

- Pro Indextyp entsteht eine Dimension im Array
- Nutzung der Arrays, Zugriff auf einzelne Elemente durch Angabe eines Indexwertes für jede Dimension:

```
var Umsaetze: TUmsaetze;
...
Umsaetze[Hamburg, 46] := 32645.50;
```

Datentypen für Schachbrett/-figuren

```
type
  TFigur = (w_Bauer, w_Turm, w_Springer, w_Laeufer, w_Dame,
            w_Koenig, s_Bauer, s_Turm, s_Springer,
            s_Laeufer, s_Dame, s_Koenig, leer);
  TSchachfeld = array [1..8, 1..8] of TFigur;

var
  Schachfeld : TSchachfeld;
begin
...
if (Schachfeld [x,y] = w_Springer) and
   (Schachfeld [x+2, y+1] = leer) then
   begin
       Schachfeld [x,y] := leer;
       Schachfeld [x+2, y+1] := w_Springer;
   end;
...
end.
```

12 Mehrdimensionale Arrays

Datentypen für Stundenplan der PTL Wedel (inkl. drei-dim. Array)

```
type
  TZeit = (Block1, Block2, Block3, Block4, Block5, Block6);

  THoersaele = 1..6;

  TWochentage = (Montag, Dienstag, Mittwoch, Donnerstag, Freitag);

  TVL_Titel = string;

  TPlan = array [TWochentage, TZeit, THoersaele] of TVL_Titel;

var Plan_WS16 : TPlan;

begin   {Hauptprogramm}
...
  Plan_WS15[Montag, Block4, 2] := 'Programmieren 1';
...
  if Plan_WS15[Donnerstag, Block4, 2] <> '' then
    Plan_WS15[Donnerstag, Block5, 2] := 'Programmieren 1';
...
```

12 Aufgabe „Kinofilme“

In dieser Aufgabe Erstellen ein Programm zur Anzeige von wochenweisen Kinofilmplänen für das Kino „Vier Jahreszeiten“.

- Deklarieren Sie (Aufzählungs)-Typen und String-Array-Konstanten (für sprechende Bezeichner) für die folgenden Sachverhalte:
  - Die sieben Tage der Woche, beginnend mit Montag.
  - Das Kino verfügt über vier Kinosäle. Diese sind nummeriert und tragen zusätzlich die folgenden Namen:
    1. Frühjahr
    2. Sommer
    3. Herbst
    4. Winter
  - Jeden Tag werden je Kino bis zu sechs Filme gezeigt. Diese laufen von 6:00 an in dreistündigen Intervallen. In jedem Intervall kann genau ein Film gezeigt werden.
  - Ein Film hat eine beliebige Bezeichnung. Ein String mit dem Inhalt „`-- Kein Film --`“ steht für einen leeren Zeitslot.
- Initialisieren Sie zunächst einen leeren `kinoZeitplan` und geben Sie Ihn aus.

12 Aufgabe „Kinofilme“

```
program kino_vorgabe;
{$APPTYPE CONSOLE}
{$R+,Q+,X-}
Type TKinoplan = array[TWochentag, TKinosaal, TZeitslot] of string;
const
   LEERER_FILM = '-- Kein Film --';
   WOCHENTAG : array[TWochentag] of string =
     ('Montag', 'Dienstag', 'Mittwoch',
      'Donnerstag', 'Freitag', 'Samstag', 'Sonntag');
   KINOSAAL : array[TKinosaal] of string =
      ('Fruhling', 'Sommer', 'Herbst', 'Winter');
   ZEITSLOT : array[TZeitslot] of string =
      ('9:00', '12:00', '15:00', '18:00', '21:00');
begin
end.
```

12 Aufgabe „Kinofilme“

- Bilden Sie die genannten Filme in dem von Ihnen angelegten `kinoZeitplan` ab. Einige der folgenden Angaben widersprechen sich, geben Sie daher am Ende des Programms den endgültigen Plan aus. Eine später definierte Filmangabe überschreibt dabei eine vorher genannte.
  - Montags bis Freitags wird in Kino 2 den ganzen Tag lang der Film „Sommer“ mit Jimi Blue Ochsenknecht gezeigt.
  - Der Film „High Noon“ wird jeden Tag um 12:00 Mittags in allen Kinos gezeigt.
  - „A Nightmare before Christmas“ wird Freitags bis Samstags als letzter Film des Tages in allen Kinos gezeigt.
  - Die Serie „Gilmore Girls – Ein neues Jahr“ wird auch im Vier Jahreszeiten gezeigt. Die Episoden tragen die Namen „Winter“, „Frühling“, „Sommer“ & „Herbst“ (es handelt sich also um 4 Filme) und werden die ganze Woche über um 18:00 in den jeweiligen Kinos abgespielt.