Programmieren 1

ptlwedel

PRIVATE BERUFSFACHSCHULE

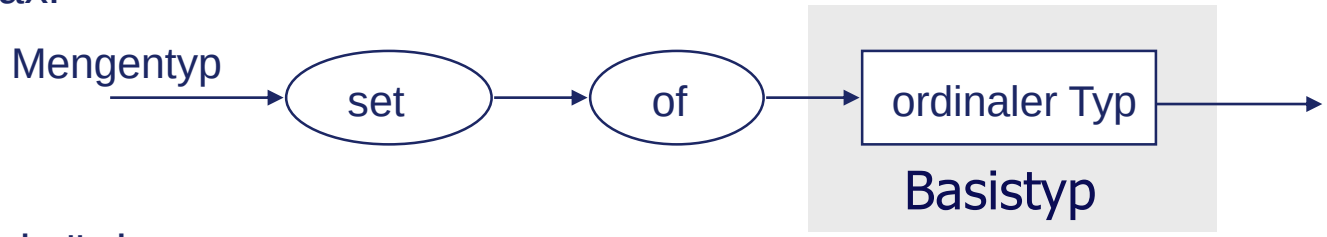
Wintersemester 2016/2017

Marcus Riemer, B.Sc.

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

14 Mengen

- In zahlreichen Anwendungssituationen mehrere Werte eines Datentyps zusammen festzuhalten
- Wenn es darum geht, welche Werte in der Zusammenfassung vorhanden (bzw. nicht vorhanden) sind, können die (mehreren) Werte zusammen als *Mengenwert* gespeichert werden
- **Set-Datentypen beschreiben Mengen von Werten eines Basistyps**
- Ein einzelner Wert eines Set-Datentyps ist eine beliebige Menge von Werten des Basistyps
- Syntax:



- Einschränkungen:
  - Der Basistyp von Set-Datentyps muss ein *Ordinaltyp* sein.
  - Basistypen von Set-Datentypen dürfen maximal 256 Werte umfassen

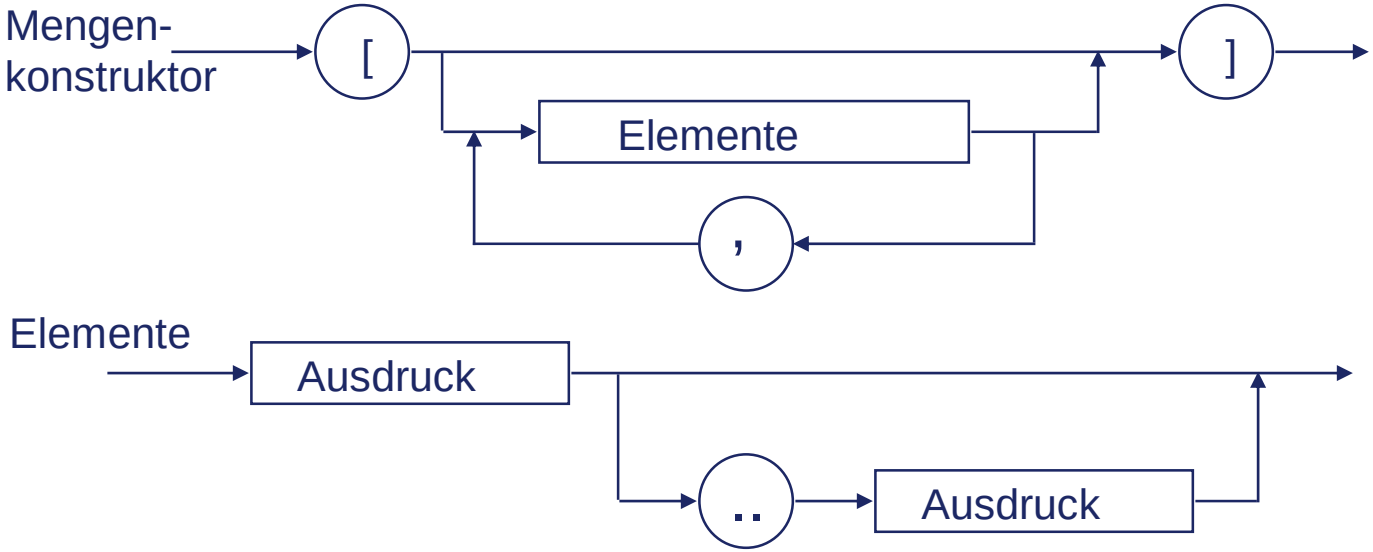
14 Mengen

- Beispiele für Set-Datentypen (Mengentypen)

```
type
        TZahlen = set of byte;
        TKleinbuchstaben = set of 'a'..'z';
        TZiffer = set of '0' .. '9';

        TKleineZahlen = 0 .. 9;
        TEinstellige = set of TKleineZahlen;

        TFarbe = (rot, gelb, gruen);
        TAmpelfarbe = set of TFarbe;

        TPersonen = (Andreas, Bernd, Dennis, Jan,
                     Kevin, Malte, Rainer, Tom, Willi);
        TPersonenmenge = set of TPersonen;
```

14 Mengenkonstruktoren

- Werden zur Angabe von Mengenwerten (Werte von Set-Datentypen) eingesetzt
- Syntax



- Beispiele

```
[]                leere Menge
[1, 2, 3]         Menge mit Elementen 1, 2 und 3
[1, 2, 2, 3]      entspricht [1, 2, 3]
[1+3, 4*5]        entspricht [4, 20]
[1..4]            entspricht [1, 2, 3, 4]
[1..3, 5..7]      entspricht [1, 2, 3, 5, 6, 7]
```

14 Mengenvariablen

- Jeder einzelne Wert einer Mengenvariablen ist eine Menge von Werten aus dem Basistypen
- Zuweisung von Werten an Mengenvariablen mit Hilfe von Mengenkonstruktoren oder Resultaten von Mengenoperationen (s.u.)
- Beispiel für Deklaration einer Mengenvariablen und Zuweisung von Mengenwerten:

```
type   TLottozahlen  = 1 .. 49;
       TGewinnzahlen = set of TLottozahlen;

var    Ziehung : TGewinnzahlen;

begin
  ...
  Ziehung := [5, 16..18, 25, 43];
  ...
  Ziehung := Ziehung + [9];
  ...
end.
```

14 Mengenoperatoren

- Mengenoperationen anwendbar auf Mengenwerten

Stufe Prior.	Operator mathem.	Operator Pascal	Bedeutung	Typ der Operanden	Ergebnistyp
1	$\cap$	`*`	Durchschnitt	Menge	Menge
2	$\cup$	`+`	Vereinigung	Menge	Menge
2	$\setminus$	`-`	Differenz	Menge	Menge
3	$=$	`=`	Gleichheit	Menge	Boolean
3	$\neq$	`<>`	Ungleichheit	Menge	Boolean
3	$\subseteq$	`<=`	Teilmenge	Menge	Boolean
3	$\supseteq$	`>=`	Teilmenge	Menge	Boolean
3	$\in$	`in`	Element von	Element/ Menge	Boolean

14 Mengenoperatoren

- Exemplarische Anwendung von Mengenoperationen:

```
type TPersonen = (Andreas, Bernd, Dennis, Jan, Kevin, Malte,
                  Rainer, Tom, Willi);
     TPersonenmenge = set of TPersonen;

const ALLE_PERSONEN = [low(TPersonen) .. High(TPersonen)];
var
  MeineFreunde, DeineFreunde, KeineFreunde,
  GemeinsameFreunde, NurFreunde, AlleFreunde: TPersonenmenge;
  MeineFeinde, DeineFeinde : TPersonenmenge;

MeineFreunde := [Bernd, Dennis, Jan];
DeineFreunde := [Andreas, Bernd, Dennis, Jan, Kevin];
MeineFeinde := [Kevin, Malte];
DeineFeinde := [Jan, Tom];

AlleFreunde := MeineFreunde + DeineFreunde;
GemeinsameFreunde := MeineFreunde * DeineFreunde;
```

14 Mengenoperatoren

- Exemplarische Anwendung von Mengenoperationen:

```
NurFreunde := MeineFreunde + DeineFreunde
              - MeineFeinde - DeineFeinde;
KeineFreunde := ALLE_PERSONEN - MeineFreunde - DeineFreunde;
MeineFreunde := MeineFreunde + [Kevin];

if MeineFreunde <= DeineFreunde then
  writeln ('Alle meine Freunde sind auch deine Freunde')
else
  writeln ('Ich habe Freunde, die du nicht hast');

if Bernd in DeineFreunde then
  writeln ('Bernd ist dein Freund');
```

14 Ausgabe von Mengen

- Es gilt (wie für alle strukturierten Datentypen): Mengen können nicht unmittelbar mit `writeln` ausgegeben werden.
- Daher (wie bei Arrays): Ausgabe mit einer `for` – Schleife

```
for zahl := low(zahl) to high(zahl) do
  if zahl in [1, 7, 2, 5] then
    write(' ', zahl);
```

- Alternativ: `for .. in` – Schleife

```
for zahl in [1, 7, 2, 5] do
  write(' ', zahl);
```

14 Ausgabe von Mengen

```
program mengen_arbeitstage;
{$APPTYPE CONSOLE}
{$R+,Q+,X-}

type
   TWochentag = (Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag);
   TWoche = set of TWochentag;

const
   ALLE_WOCHENTAGE = [low(TWochentag)..high(TWochentag)];
   WOCHENTAG_NAMEN : array[TWochentag] of string =
      ('Montag', 'Dienstag', 'Mittwoch', 'Donnerstag',
       'Freitag', 'Samstag', 'Sonntag');

var
   meineArbeitstage, deineArbeitstage : TWoche;
   wochentag: TWochentag;
```

14 Ausgabe von Mengen

```
begin
   meineArbeitstage := [Montag .. Freitag];
   deineArbeitstage := [Dienstag, Donnerstag, Samstag, Sonntag];

   write('Meine Arbeitstage: ');
   for wochentag in meineArbeitstage do
      write(' ', WOCHENTAG_NAMEN[wochentag]);

   writeln;
   write('Deine Arbeitstage: ');
   for wochentag in deineArbeitstage do
      write(' ', WOCHENTAG_NAMEN[wochentag]);

   writeln;
   write('Unsere Arbeitstage:');
   for wochentag in deineArbeitstage * meineArbeitstage do
      write(' ', WOCHENTAG_NAMEN[wochentag]);
end.
```

14 Aufgabe "Mengenoperationen"

Dieses Programm soll für zwei vom Benutzer einzugebende Mengen verschiedene Mengenoperationen durchführen.

- Legen Sie einen Typ `TZahlenmenge` an, dieser soll Werte zwischen 0 und 255 speichern können. Deklarieren Sie von diesem Typ zwei Variablen namens `mengeLinks` und `mengeRechts`.
- Fragen Sie vom Benutzer so lange ganze Zahlen ab, bis er die Eingabe mit einem `'x'` oder einem `'X'` beendet. Die Zahlen werden dabei zunächst in die linke Menge eingelesen, nach der Eingabe eines Trennzeichens `,:'` werden alle folgenden Werte in die rechte Menge eingelesen.
  - **Hinweis:** Für diese Anforderung muss die Benutzereingabe zunächst in einen String eingelesen werden.
  - **Hinweis:** Mit den Mengenkonstruktoren können einzelne Werte in Mengen umgewandelt werden.
  - **Hinweis:** Mengen werden mit dem '+'-Operator erweitert.
- Geben Sie die linke und die rechte Menge aus.

Beispiel:

```
5
2
7
22
:
5
22
X
Links = 2 5 7 22
Rechts = 5 22
a) false
b) true
c) false
d) 2 5 7 12 22
e) 5 22
f) 2 7
g)
```

14 Aufgabe "Mengenoperationen"

Geben Sie die Ergebnisse der folgenden Mengen-Operationen aus:

$a)\ links \subset rechts$

$b)\ rechts \subset links$

$c)\ links = rechts$

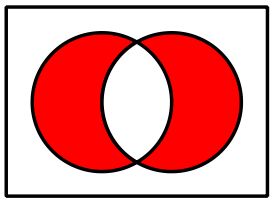
$d)\ links \cup rechts$

$e)\ links \cap rechts$

$f)\ links \setminus rechts$

$g)\ rechts \setminus links$

- Implementieren Sie auch Tests für echte Teilmengen. Für diese Eigenschaft muss die rechte Seite mindestens ein Element enthalten, dass in der linken Seite nicht enthalten ist.
- Implementieren Sie die symmetrische Differenz. In dieser Ergebnismenge sind alle Elemente enthalten, die nur in der linken oder der rechten Menge auftauchen.



14 Aufgabe "Mengenoperationen"

In diesem Aufgabenteil werden String-Operationen wiederholt. Ermöglichen Sie dem Benutzer die Eingabe einer einzigen Zeile, in welcher die einzelnen Elemente der Mengen und die Mengen selbst durch Trennzeichen kenntlich gemacht werden.

- Dabei trennt das Leerzeichen mehrere Werte der gleichen Menge.
- Der Doppelpunkt trennt die erste Menge von der zweiten Menge.

Beispielhafte Eingaben

- ```
  12 6 21 : 12 21
  mengeLinks  = 6 12 21
  mengeRechts = 12 21
  ```
- ```
  : 14 21
  mengeLinks  =
  mengeRechts = 14 21
  ```
- ```
  20 10
  mengeLinks  = 10 20
  mengeRechts =
  ```
- ```
  3 6 9 :
  mengeLinks  = 3 6 9
  mengeRechts =
  ```