Informatik I Übung, Woche 43

Giuseppe Accaputo

22. Oktober, 2015

Plan für heute

1. Fragen & Nachbesprechung Übung 5
2. Zusammenfassung der bisherigen Vorlesungsslides
3. Vorbesprechung Übung 6

Arrays

Syntax: `ARRAY [start_w..end_w] OF SOMETYPE`

Semantik: Speichert mehrere Variablen vom gleichen Typ `SOMETYPE` ab

- `start_w <= end_w`
- Index `i` $\in$ [`start_w, end_w`]
- Elemente im Array a sind $a_{\texttt{start\_w}}, a_{\texttt{start\_w+1}}, \ldots, a_{\texttt{end\_w}}$
- Auf das Element a_i wird mittels `a[i]` zugegriffen

Funktionen auf Arrays

- `Low(a)` $\Leftrightarrow$ kleinster Index des Arrays `a`
- `High(a)` $\Leftrightarrow$ grösster Index des Arrays `a`
- `Length(a)` $\Leftrightarrow$ Anzahl Elemente im Array `a`

Beispiel:

```
a : ARRAY [0..5] OF INTEGER;
```

- `Low(a)` gibt `0` zurück
- `High(a)` gibt `5` zurück
- `Length(a)` gibt `6` zurück (Elemente im Array: $a_0, a_1, \ldots, a_5$)

Arrays und die FOR-Schleife

Ziel: Iteriere durch alle Elemente des Arrays

1. Erste Möglichkeit:

```
FOR i := Low(a) TO High(a) DO
h := h + a[i];
```

   - Umgekehrte Reihenfolge auch möglich:

```
FOR i := High(a) DOWNTO Low(a) DO
```

2. Zweite, elegantere Möglichkeit:

```
FOR array_element IN a DO
    h := h + array_element;
```

Array Beispiel

Sei folgendes Array a gegeben:

a := | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Frage: Wie sieht der Inhalt des Arrays a nach der Ausführung des folgenden Code-Ausschnitts aus?

```
FOR i := Low(a) TO (High(a)-1) DO
a[i] := a[i] + a[i+1];
```

Array Beispiel

Sei folgendes Array a gegeben:

a :=

0	1	2	3	4	5	6	7	8	9

Frage: Wie sieht der Inhalt des Arrays a nach der Ausführung des folgenden Code-Ausschnitts aus?

```
FOR i := Low(a) TO (High(a)-1) DO
a[i] := a[i] + a[i+1];
```

Antwort: a :=

1	3	5	7	9	11	13	15	17	9

Dynamische Arrays

Syntax: `ARRAY OF SOMETYPE`

Semantik: Speichert mehrere Variablen vom gleichen Typ `SOMETYPE` ab, jedoch kann die Grösse des Arrays beliebig angepasst werden

- Verwende `SetLength(a, n)` um die Grösse von `a` auf `n` Elemente zu setzen
- Index `i` $\in [0, n-1]$
- Beim Ändern der Grösse bleiben die bestehenden Elemente erhalten

Arrays als Funktions- und Prozedur-Argumente

Open Array:

- Arrays können als Funktions- und Prozedur-Argumente verwendet werden
- Array *ohne Grössenangabe*
- Statisches und Dynamisches Array als Argument möglich
  - Wichtig: Auf Open Arrays kann `SETLENGTH` nicht angewendet werden!
- Indizes sind immer von 0 bis $n-1$
  - Wichtig: Vorallem zu beachten, wenn statisches Array übergeben wird da beliebige Indexangabe möglich ist, z.B. `ARRAY [1000...4582] OF ...`

Arrays als Funktions- und Prozedur-Argumente

Syntax: `FUNCTION M (wer : ARRAY OF INTEGER)...`

- Statisches oder dynamisches Array übergeben:

```
M(arr)
```

- Array bestehend aus den Elementen $\text{arr}_3, \text{arr}_4, \text{arr}_5$ übergeben (Slice):

```
M(arr[3..5])
```

Vor- und Nachbedingungen

Es muss klar sein, was eine Funktion / Prozedur erreicht. Dies können wir mit Kontrakte erreichen, welche durch eine Vor- und Nachbedingung definiert werden.

- **Vorbedingung** (engl. *Precondition*): Was muss erfüllt sein
  - Ist Vorbedingung nicht erfüllt, können wir auf unerwartetes Verhalten stossen
  - Vorbedingungen können mit `ASSERT` überprüft werden
- **Nachbedingung** (engl. *Postcondition*): Was wird erreicht