Informatik I Übung, Woche 42

Giuseppe Accaputo

16. Oktober, 2014

Plan für heute

1. Fragen & Nachbesprechung Übung 4
2. Zusammenfassung der bisherigen Vorlesungsslides
3. Vorbesprechung Übung 5

Aufgabe 4.2: Definition der harmonischen Reihe

Definition der harmonischen Reihe:

$$\sum_{i=1}^{\infty} \frac{1}{i}$$

Da unsere Rechner nur endliche Rechnungen ausführen können, berechnen wir die (endliche) harmonische Reihe h_n:

$$h_n = \sum_{i=1}^{n} \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

Aufgabe 4.2: Fragen zur Berechnung von h_n

1. Wieviele Summanden benötige ich für die Berechnung von h_n?
2. Welcher Schleifentyp kann ich verwenden, um h_n zu berechnen?

Aufgabe 4.2: Fragen zur Berechnung von h_n

1. Wieviele Summanden benötige ich für die Berechnung von h_n?
   Antwort: n
2. Welcher Schleifentyp kann ich verwenden, um h_n zu berechnen?
   Antwort: Da die Anzahl Summanden (und somit die Anzahl Iterationen) bekannt
   ist kann man eine `FOR`-Schleife verwenden um h_n zu berechnend

Nächster Schritt: Fragen zur Implementation.

Aufgabe 4.2: Repetition FOR-Schleife

Syntax:

```
FOR lauf_v := start_w TO end_w DO
    { zu wiederholende Anweisung; lauf_v
       ↪ kann verwendet werden hier }
```

- Die Laufvariable `lauf_v` ändert sich in der FOR-Schleife und nimmt folgende Werte an:
  $\texttt{lauf\_v} = \texttt{start\_w}, \texttt{start\_w}+1, \ldots, \texttt{end\_w}-1, \texttt{end\_w}$

Aufgabe 4.2: Fragen zur Implementation

$$h_n = \sum_{i=1}^{n} \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

1. In welchem Teil der Berechnung von h_n wird eine (sich ändernde) Laufvariable benötigt?
2. Welchen Startwert (`start_w`) hat die Laufvariable?
3. Welchen Endwert (`end_w`) hat die Laufvariable?

Aufgabe 4.2: Fragen zur Implementation

1. In welchem Teil der Berechnung von h_n wird eine (sich ändernde) Laufvariable benötigt?
   Antwort: Im Nenner des Summanden, da sich nur dieser von Summand zu Summand ändert
2. Welches ist der Startwert der Laufvariable?
   Antwort: Wie man anhand der Definition von h_n sieht $(h_n = \frac{1}{\mathbf{1}} + \frac{1}{2} + \cdots + \frac{1}{n})$ ist der Startwert unserer Laufvariable 1
3. Welches ist der Endwert der Laufvariable?
   Antwort: Wie man anhand der Definition von h_n sieht $(h_n = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{\boldsymbol{n}})$ ist der Endwert unserer Laufvariable n

Aufgabe 4.2: Bisherige Informationen zur Implementation

Unsere Laufvariable beginnt bei 1 und endet bei n, wobei n vom Benutzer angegeben wird (`Readln(n)`).

Unsere `FOR`-Schleife sieht daher wie folgt aus:

```
FOR lauf_v := 1 TO n DO
{ Berechne h_n }
```

Frage: Wie implementiere ich nun die Berechnung von h_n in der `FOR`-Schleife?

Aufgabe 4.2: Berechnung von h_n in der FOR-Schleife

- Die Laufvariable nimmt konkret folgende Werte an:
  `lauf_v := 1, 2, 3, ..., n`
- Die Laufvariable kommt im Nenner eines Summanden vor
- Die Hauptaufgabe der FOR-Schleife ist das Aufsummieren aller Summanden

```
h1 = 1;
h2 = h1 + 1/2;
h3 = h2 + 1/3 = h1 + 1/2 + 1/3;
...
```

Aufgabe 4.2: Berechnung von h_n in der FOR-Schleife

- Die Laufvariable nimmt konkret folgende Werte an:
  `lauf_v := 1, 2, 3, ..., n`
- Die Laufvariable kommt im Nenner eines Summanden vor
- Die Hauptaufgabe der FOR-Schleife ist das Aufsummieren aller Summanden

```
VAR
    h, summand : REAL;
...
FOR lauf_v := 1 TO n DO
BEGIN
    summand := 1/lauf_v;
    h := h + summand;
END;
```

Aufgabe 4.3: Collatz-Folge

Erste Implementation der Collatz-Folge:

```
...
a : INTEGER
...
WHILE a > 1 DO
BEGIN
    Write(a, ' ');
    IF a mod 2 = 0 THEN a := a div 2
    ELSE a := 3*a+1;
END;
```

Aufgabe 4.3: Collatz-Folge

Problem: Bei der Eingabe von 23001 wird ein Überlauf generiert

Grund: ?

Aufgabe 4.3: Collatz-Folge

Problem: Bei der Eingabe von 23001 wird ein Überlauf generiert

Grund: Free Pascal verwendet für `INTEGER` standardmässig `SMALLINT` (Wertebereich $[-32768, 32767]$). Da 23001 ungerade ist erhalten wir $a = 3 * 23001 + 1 = 69004$, jedoch kann `SMALLINT` diese Zahl nicht darstellen und wir erhalten $a = 3468$

Aufgabe 4.3: Collatz-Folge

Lösungsvorschlag: Überprüfe ob $a > 32767$ ist. Wird diese Bedingung erfüllt, so brich das Programm ab.

```
WHILE a > 1 DO
BEGIN
    Write(a, ' ');
    IF a > 32767 THEN BREAK; { Ueberlauf }
    IF a mod 2 = 0 THEN a := a div 2
    ELSE a := 3*a+1;
END;
```

Aufgabe 4.3: Collatz-Folge

Lösungsvorschlag: Überprüfe ob $a > 32767$ ist. Wird diese Bedingung erfüllt, so wird das Programm abgebrochen.

```
WHILE a > 1 DO
BEGIN
    Write(a, ' ');
    IF a > 32767 THEN BREAK; { Ueberlauf }
    IF a mod 2 = 0 THEN a := a div 2
    ELSE a := 3*a+1;
END;
```

Problem: Da wir mit `SMALLINT` arbeiten und der Maximalwert 32767 ist, können wir so nie herausfinden, ob a tatsächlich grösser als 32767 ist

Aufgabe 4.3: Collatz-Folge

- Ein Überlauf kann im Programm nur dann geschehen, wenn a ungerade ist (ansonsten wird a einfach halbiert)
- Ist a ungerade, so wird a neu wie folgt berechnet:

$$a = 3 \cdot a + 1$$

- a darf maximal den Wert 32767 annehmen (`SMALLINT`)
  - $a > 32767$ zu überprüfen funktioniert nicht mit `SMALLINT`

Lösung: $32767 = 3 \cdot a + 1 \Leftrightarrow a = 10922$

Ist $a \leq 10922$, so folgt sofort $3 \cdot a + 1 \leq 32767$, d.h. mit der Bedingung $a \leq 10922$ versichern wir, dass wir keinen Überlauf generieren falls a ungerade ist

Aufgabe 4.3: Collatz-Folge

```
WHILE a > 1 DO
BEGIN
    Write(a, ' ');
    IF a mod 2 = 0 THEN a := a div 2
    ELSE IF a <= 10922 THEN a := 3*a+1
    ELSE
    BEGIN
        Writeln('Ueberlauf detektiert!');
        BREAK;
    END;
END;
```

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

Erste Möglichkeit:

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

Zweite, elegantere Möglichkeit:

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

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