

### 3. Aufgabenblatt

(Besprechung in den Tutorien 06.11.2023–10.11.2023)

#### Aufgabe 1. LOOP-Programme

1. Seien  $P_1$  und  $P_2$  zwei LOOP-Programme. Simulieren sie das Konstrukt

**IF**  $x_1 > x_2$  **THEN**  $P_1$  **ELSE**  $P_2$  **END**

durch ein LOOP-Programm.

2. Geben Sie ein LOOP-Programm an, das die folgende Funktion berechnet:

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, \quad f(x_1, x_2) := \begin{cases} x_1 \bmod x_2, & x_2 > 0 \\ x_1, & x_2 = 0 \end{cases}.$$

---

—Lösungsskizze—

1.  $x_3 := x_1 + 0$ ;

**LOOP**  $x_2$  **DO**  $x_3 := x_3 - 1$  **END**;

$x_4 := x_3 - 1$ ;

**LOOP**  $x_4$  **DO**  $x_3 := x_3 - 1$  **END**;

$x_5 := x_5 + 1$ ;

**LOOP**  $x_3$  **DO**  $x_5 := x_5 - 1$  **END**;

**LOOP**  $x_3$  **DO**  $P_1$  **END**;

**LOOP**  $x_5$  **DO**  $P_2$  **END**

2. Wir verwenden  $x_i := x_i - x_j$  als Abkürzung für **LOOP**  $x_j$  **DO**  $x_i := x_i - 1$  **END** und das Programm aus dem ersten Aufgabenteil.

$x_0 := x_1 + 0$ ;

**LOOP**  $x_1$  **DO**

**IF**  $x_2 > x_1$

**THEN**  $x_0 := x_1 + 0$

**ELSE**  $x_1 := x_1 - x_2$

**END**

**END**;

$x_0 := x_1 + 0$

---

#### Aufgabe 2. LOOP- und WHILE-Programme

1. Warum stoppt jedes LOOP-Programm nach endlicher Zeit?

---

—Lösungsskizze—

Weil sich in LOOP-Programmen keine Endlosschleifen implementieren lassen. (Und es keine weiteren terminierungskritischen Sprachkonstrukte wie z.B. Rekursion gibt.)

Genauer, per Induktion über den Aufbau von LOOP-Programmen:

Jede Zuweisung terminiert offensichtlich.

Terminieren die LOOP-Programme  $P_1$  und  $P_2$ , dann zweifellos auch  $P_1; P_2$ .

Terminiert  $P$ , dann auch **LOOP**  $x_i$  **DO**  $P$  **END**, denn  $P$  wird nur endlich oft ausgeführt (bestimmt durch den Wert von  $x_i$  vorher, der aber in jedem Kontext (d.h. äußeren/vorgeschalteten LOOP-Programm) eine natürliche Zahl ist).

Somit terminieren alle LOOP-Programme.

---

2. Geben Sie ein WHILE-Programm an, welches niemals stoppt.

—————Lösungsskizze—————

```
 $x_0 := x_0 + 1;$ 
WHILE  $x_0 \neq 0$  DO
   $x_1 := x_0 + 0$ 
END
```

Initial wird  $x_0$  auf 1 gesetzt. Da dieser Wert im Schleifenrumpf nicht verändert wird, ist die Schleifenbedingung niemals verletzt und die Schleife läuft endlos.

---

### Aufgabe 3. Eingeschränkte LOOP-Syntax

Wir betrachten mehrere abgeänderte Formen der Syntax von LOOP. Zeigen oder widerlegen Sie jeweils, dass diese Versionen von LOOP die gleiche Mächtigkeit wie LOOP haben.

1. Anstatt der Zuweisungen  $x_i := x_j + c$  und  $x_i := x_j - c$  dürfen nur  $x_i := x_j + 1$  und  $x_i := x_j - 1$  verwendet werden.
2. Anstatt der Zuweisungen  $x_i := x_j + c$  und  $x_i := x_j - c$  dürfen nur  $x_i := x_j + 2$  und  $x_i := x_j - 2$  verwendet werden.
3. Anstatt  $x_i := x_j + c$  und  $x_i := x_j - c$  dürfen nur  $x_i := x_j \cdot c$  und  $x_i := x_j / c$  verwendet werden. Die Semantik der ersten Anweisung ist dann die Multiplikation und die der zweiten die ganzzahlige Division (sprich: wenn  $x_j$  nicht durch  $c$  teilbar ist, dann wird das Ergebnis abgerundet).

—————Lösungsskizze—————

1. Diese Variante hat die gleiche Mächtigkeit wie gewöhnliches LOOP. Die Anweisung  $x_i := x_j + c$  kann durch  $c$ -malige Wiederholung der Anweisung  $x_i := x_j + 1$  ersetzt werden. Ähnlich kann auch  $x_i := x_j - c$  ersetzt werden.
2. Diese Variante ist weniger mächtig als gewöhnliches LOOP. Die Funktion  $f: \mathbb{N} \rightarrow \mathbb{N}$  mit  $f(n) := n + 1$  ist in gewöhnlichem LOOP definierbar, z.B. durch folgendes Programm:

```
 $x_0 := x_1 + 1.$ 
```

Doch in dieser eingeschränkten Variante von LOOP kann  $f$  nicht definiert werden. Sei  $P$  ein beliebiges Programm in der LOOP-Variante. Falls die Eingabe in  $x_1$  gerade ist, dann kann niemals eine ungerade Zahl in irgendeine Variable geschrieben werden. Doch die Ausgabe bei einer geraden Eingabe muss ungerade sein.

3. Diese Variante ist weniger mächtig als gewöhnliches LOOP.

Wenn z.B. alle Eingaben zu Beginn der Berechnung 0 sind, dann lassen sich durch die gegebenen Anweisungen nur 0en in Variablen schreiben. Damit ist die Funktion  $n \mapsto n + 1$  wieder nicht berechenbar.

---

#### Aufgabe 4. WHILE- und GOTO-Programme

1. Geben Sie ein GOTO-Programm für die Funktion an, die von folgendem WHILE-Programm (mit Eingaben  $x_1, x_2$ ) berechnet wird.

```
 $x_3 := x_1 + 1;$   
WHILE  $x_3 \neq 0$  DO  
     $x_2 := x_2 - 1;$   
     $x_3 := x_3 - 1$   
END;  
 $x_0 := x_2 + 0$ 
```

2. Geben Sie ein WHILE-Programm für die Funktion an, die von folgendem GOTO-Programm (mit Eingabe  $x_1$ ) berechnet wird.

```
 $M_1 :$    $x_2 := x_2 + 1;$   
         $x_0 := x_2 + 0;$   
        IF  $x_1 = 0$  THEN GOTO  $M_3;$   
         $x_1 := x_1 - 1;$   
        GOTO  $M_2;$   
 $M_2 :$    $x_0 := x_0 + 1;$   
        GOTO  $M_1;$   
 $M_3 :$    $x_0 := x_0 + 0$ 
```

---

#### Lösungsskizze

1. Das WHILE-Programm berechnet  $(x_1, x_2) \mapsto x_2 - (x_1 + 1)$ .

```
 $M_1 :$    $x_3 := x_1 + 1;$   
 $M_2 :$    $x_2 := x_2 - 1;$   
         $x_3 := x_3 - 1;$   
        IF  $x_3 = 0$  THEN GOTO  $M_3;$   
        GOTO  $M_2;$   
 $M_3 :$    $x_0 := x_2 + 0;$ 
```

2. Das GOTO-Programm berechnet  $x_1 \mapsto x_1 + 1$ .

```
 $x_0 := x_0 + 1;$   
WHILE  $x_1 \neq 0$  DO  
     $x_1 := x_1 - 1;$   
     $x_0 := x_0 + 1$   
END
```

Alternativ:

```
 $x_0 := x_1 + 1$ 
```

---