Universität Siegen
Lehrstuhl Theoretische Informatik
Markus Lohrey

Übungsblatt 2

Aufgabe 1
Wahr oder falsch?

(a) Das folgende LOOP-Programm terminiert nicht.

$$x_1 := 5;\ \text{LOOP}\ x_1\ \text{DO}\ x_1 := x_1 + 1;\ \text{END}$$

(b) Das folgende WHILE-Programm berechnet die Funktion $f(x) = 0$.

$$\text{WHILE}\ x \neq 0\ \text{DO}\ x := x - 2; x := x + 1;\ \text{END}$$

Lösung

(a) falsch

Der LOOP wird $x_1 = 5$ mal ausgeführt, das veränderte x_1 wird nicht erneut gelesen. Grundsätzlich gilt: LOOP-Programme terminieren immer.

(b) falsch

Das Programm berechnet $f(x) = \begin{cases} 0 & \text{falls } x = 0 \\ \text{undef} & \text{falls } x \geq 1 \end{cases}$

Fall 1, $x = 0$

Das Programm terminiert mit einem Rückgabewert von 0.

Fall 2, $x \geq 1$

Das Programm terminiert nicht. Falls $x \geq 2$, dekrementiert ein Durchlauf der Schleife den Wert um 1, bis $x = 1$ erreicht ist.

Für $x = 1$ ist der Wert nach einem Durchlauf der Schleife aber wieder 1, da wir beim Subtrahieren nicht ins Negative, sondern nur bis zur 0 gehen (also $a - b = 0$ für $b \geq a$). Somit ergibt $x := x - 2$ zuerst $x = 0$ und anschließend erhalten wir durch $x := x + 1$ wiederum $x = 1$, was zu einer Endlosschleife führt.

Aufgabe 2

(a) Schreiben Sie ein LOOP-Programm, das für eine Zahl n die n-te *Fibonacci-Zahl* berechnet.

(b) Schreiben Sie ein LOOP-Programm, das die Funktion $f(x, y) = x^y$ für $x \neq 0$ berechnet.

(c) Schreiben Sie ein LOOP-Programm, das die Funktion $f(x, y) = \max(x, y)$ berechnet.

(d) Schreiben Sie ein LOOP-Programm, das die Funktion $f(x, y, z) = \min(x, y, z)$ berechnet.

(e) Schreiben Sie ein WHILE-Programm, das für eine gegebene Zahl $n \geq 2$ den kleinsten Teiler p von n mit $p \geq 2$ ausgibt.

(f) Schreiben Sie ein WHILE-Programm, das die Funktion $f(n) = \lceil \sqrt{n} \rceil$ berechnet.

(g) Schreiben Sie ein GOTO-Programm für Aufgabenteil (c).

Lösung

(a) Wir verwenden die folgende Definition der Fibonacci-Zahlen:

$$F(n) = \begin{cases} 0 & \text{falls } n = 0 \\ 1 & \text{falls } n = 1 \\ F(n-2) + F(n-1) & \text{falls } n \geq 2. \end{cases} \tag{1}$$

Zu Beginn ist der Eingabewert n in der Variablen x_1 gespeichert.

In x_1 und x_2 speichern wir $F(i)$ und $F(i+1)$, beginnend mit $i = 0$. Mit jeder Iteration setzen wir $x_1 := x_2$ und $x_2 := x_1 + x_2$.

$x_4 := x_1;$
$x_1 := 0;$
$x_2 := 1;$
LOOP x_4 DO
  $x_3 := x_1 + x_2;$
  $x_1 := x_2;$
  $x_2 := x_3$
END

Beachte: Das Ergebnis steht bei Termination des Programms in der Variablen x_1 (siehe Folie 28).

(b) $f(x, y) = x^y$. Zu Beginn steht x in x_1 und y in x_2.

$x_3 := x_1;$
$x_1 := 1;$
LOOP x_2 DO
  $x_1 := x_1 \cdot x_3$
END

(c) Wir simulieren zunächst die (abgeschnittene) Subtraktion von Variablen ($x_i := x_j - x_k$ für $i \neq k$) als LOOP-Programm:

$x_i := x_j;$
LOOP x_k DO $x_i := x_i - 1$ END

$f(x, y) = \max(x, y)$. Zu Beginn steht x in x_1 und y in x_2.

$x_3 := x_2 - x_1;$
LOOP x_3 DO $x_1 := x_2$ END

Falls $y > x$ wird die LOOP-Schleife (mindestens einmal) ausgeführt und der „Rückgabewert“ mit y (x_2) überschrieben.

(d) $f(x, y, z) = \min(x, y, z)$. Zu Beginn steht x in x_1, y in x_2 und z in x_3.

$x_4 := x_2 - x_3;$
LOOP x_4 DO $x_2 := x_3$ END;
$x_4 := x_1 - x_2;$
LOOP x_4 DO $x_1 := x_2$ END;

Wir verwenden eine ähnliche Technik wie bei Aufgabenteil (c).

Mit den ersten zwei Zeilen prüfen wir, ob $x_2 > x_3$. Falls ja, ist $x_4 > 0$ und wir überschreiben x_2 mit x_3 ($x_2 := \min(y, z)$).

Dann wiederholen wir den Vorgang mit x_1 und x_2. Am Ende steht in x_1 der Wert $\min(x, \min(y, z)) = \min(x, y, z)$.

(e) Definiere zunächst $x_i := x_j \bmod x_k$ (für $i \neq j$ und $i \neq k$) durch

```
x_i := 0;
Loop x_j Do
  x_i := x_i + 1;
  x_h := x_k − x_i;      // h verschieden von i, j, k
  If x_h = 0 Then x_i := 0 End;
End
```

Idee: Wir inkrementieren x_i x_j-mal um 1. Nach jedem Durchlauf der Schleife prüfen wir mit Hilfe der Hilfsvariable x_h, ob $x_i = x_k$ gilt (also $x_k - x_i = 0$). Falls ja, setzen wir x_i auf 0 zurück.

Zu Beginn steht n in x_1. Falls p ein Teiler von n ist, gilt $n \bmod p = 0$.

```
x_2 := 1;                    // Speicher für x_1 mod x_3
x_3 := 1;                    // Zähler für den Teiler p
While x_2 ≠ 0 Do
  x_3 := x_3 + 1;
  x_2 := x_1 mod x_3
End;
x_1 := x_3                   // p in x_1 kopieren
```

(f) $f(n) = \lceil \sqrt{n} \rceil$, n steht in x_1.

```
x_2 := 0;                    // Counter c
x_3 := 0;                    // Speicher für c^2
x_4 := x_1;                  // Speicher für n − c^2
While x_4 ≠ 0 Do
  x_2 := x_2 + 1;
  x_3 := x_2 · x_2;
  x_4 := x_1 − x_3
End;
x_1 := x_2
```

Idee: Wir zählen c hoch und berechnen nach jedem Schritt $n - c^2$. Falls c die kleinste Zahl ist, sodass $c^2 \geq n$ (also $c = \lceil \sqrt{n} \rceil$) ist x_4 dann 0, wir beenden die WHILE-Schleife und speichern c in x_1 (dem Rückgabewert).

(g)

$x_3 := x_1;$
$x_4 := x_2;$

$M_1 :$ IF $x_3 = 0$ THEN GOTO M_2 END;
IF $x_4 = 0$ THEN HALT END;
$x_3 := x_3 - 1;$
$x_4 := x_4 - 1;$
GOTO $M_1;$

$M_2 : x_1 := x_2;$
HALT

Idee: Zu Beginn steht x in x_1 und y in x_2. Wir zählen parallel beide Zahlen (x in x_3 und y in x_4) runter. Falls x_3 zuerst 0 wird, gilt $\max(x, y) = y$ und wir springen zu M_2, um x_1 (den Rückgabewert) mit x_2 zu überschreiben, und beenden das Programm. Wird x_4 zuerst 0, ist $\max(x, y) = x$. Da x schon in x_1 steht, können wir das Programm direkt beenden.