Prof. Andreas Goerdt — TU Chemnitz
M. Pippig, M. Riedel, U. Schönke — Wintersemester 2003/2004

Algorithmen und Programmierung

2. Übung – Lösungsvorschläge

1. Aufgabe:

   a) $\sum_{0\le n\le 5} \frac{1}{2n+1} = \frac{1}{1}+\frac{1}{3}+\frac{1}{5}+\frac{1}{7}+\frac{1}{9}+\frac{1}{11} = \frac{6508}{3465} = 1,87821$

   b) $\sum_{0\le n^2\le 5} \frac{1}{2n^2+1} = \frac{1}{1}+\frac{1}{3}+\frac{1}{9} = \frac{13}{9}$

   c) $\sum_{0\le n^2\le 5} \frac{1}{2n+1} = \frac{1}{1}+\frac{1}{3}+\frac{1}{5} = \frac{23}{15}$

2. Aufgabe:
   Wir wissen, dass $\sum_{1\le j\le n} j = \frac{n(n+1)}{2}$ (Summenformel). $\sum_{m\le j\le n} j$ kann man wie folgt aufspalten:

$$\sum_{m\le j\le n} j = \sum_{1\le j\le n} j - \sum_{1\le j\le m-1} j = \frac{n(n+1)}{2} - \frac{(m-1)(m)}{2} = \frac{n^2+n-m^2+m}{2}$$

3. Aufgabe:
   Invariante: $c_l \le 1$ für $l \ge 0$

   Induktionsanfang:
   $c_0 = 0$ gilt offensichtlich, da kein Übertrag in die 0. Stelle vorhanden ist.

   Für c_1 gibt es 4 Fälle:

x_0	y_0	Summe	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

   Daraus folgt, dass $c_1 \le 1$.

   Induktionsschritt:
   Sei $2 \le l \le n$ fest gesetzt und $c_{l-1} \le 1$. Für c_l gibt es nun folgende Möglichkeiten:

x_{l-1}	y_{l-1}	c_{l-1}	Summe	c_l
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Daraus folgt, dass $c_l \leq 1$ für $2 \leq l \leq n$ (1. Teil der Behauptung).

Die Berechnung von c_l erfolgt durch

$$c_l = DIV\left((x_{l-1} + y_{l-1} + c_{l-1}), 2\right)$$

Wir wissen, dass $x_l = y_l = 0$ für $l \geq n$ gilt. Deshalb gilt

$$x_n + y_n + c_n \leq 1$$

weil $c_n \leq 1$ (oben bewiesen). Es kann folglich keinen Übertrag in die $n+1$-te Stelle geben. Dies gilt analog für alle Stellen $> n$. Also ist $c_l = 0$ für $l > n$ (2. Teil der Behauptung).

4. Aufgabe:

   a)

```
import Prog1Tools.IOTools;
public class Fakultaet {
  public static void main(String[] args) {
    int k, fak, zaehler;
    k=IOTools.readInteger("Bitte geben Sie k ein: ");
    fak=1;
    zaehler=1;  // Verbesserung: Zaehler=2
    while (zaehler <= k) {
      fak=fak*zaehler;
      zaehler++;
    }
    System.out.println("Fakultaet von " + k + " ist " + fak);
    return;
  }
}
```

   b) Sei $zaehler_l$ der Wert der Variable *zaehler* (Zähl-Variable der Schleife) nach dem l-ten Schleifendurchlauf. Es gilt folgende Invariante für $l \geq 1$:

$$zaehler_l = l + 1 \text{ und } fak_l = \prod_{j=1}^{l} j.$$

   Induktionsanfang

   Vor dem ersten Durchlauf der Schleife sind die Werte von *zaehler* und fak gleich Eins, also

$$zaehler_0 = 1 \text{ und } fak_0 = 1.$$

Nach dem ersten Durchlauf ist *zaehler* um Eins erhöht, also

$$zaehler_1 = 2$$

und fak hat immer noch den Wert Eins, da in der Schleife mit Eins multipliziert wurde

$$fak_1 = 1.$$

Induktionsschritt
Für $1 < l \leq k$ ist laut Voraussetzung (Invariante)

$$zaehler_{l-1} = l \text{ und } fak_{l-1} = \prod_{j=1}^{l-1} j.$$

Im Schleifenrumpf wird nun multipliziert, d. h. es gilt

$$\begin{aligned} fak_l &= fak_{l-1} * zaehler_{l-1} \\ &= \left(\prod_{j=1}^{l-1} j \right) * l \\ &= \prod_{j=1}^{l} j \end{aligned}$$

Aufgrund der Inkrementierung im Schleifenrumpf ist

$$zaehler_l = l + 1.$$

Gilt also die Invariante nach dem $l-1$-ten Schleifendurchlauf, so gilt sie auch nach dem l-ten Durchlauf. Somit ist für $k = l$

$$zaehler_l = k + 1 \text{ und } fak_k = \prod_{j=1}^{k} j = k!.$$

Nach dem k-ten Schleifendurchlauf ist also die Bedingung der While-Schleife $zaehler \geq k$ nicht mehr erfüllt und die Schleife wird verlassen. Am Ende hat fak_k somit den Wert $k!$.

5. Aufgabe:

a) Unabhängig von den gezogenen Farben werden bei jedem Zug zwei Bohnen entnommen und eine wieder in die Kanne gelegt. Die Anzahl der Bohnen in der Kanne wird also in jedem Schritt um Eins verringert. Folglich sind genau $m-1$ Züge möglich, da sich nach dem $m-1$. Zug nur noch eine Bohne in der Kanne befindet.

b) Es gibt drei Möglichkeiten für die Farbkombination der zwei gezogenen Bohnen:

   1) eine weiße und eine schwarze

2) zwei schwarze
3) zwei weiße

Im ersten und zweiten Fall wird jeweils die Anzahl der schwarzen Bohnen in der Kanne um eins verringert, während die Anzahl der weißen Bohnen gleich bleibt. Im dritten Fall wird die Anzahl der weißen Bohnen um zwei verringert, während die Anzahl der schwarzen um eins erhöht wird.

Sei w_i die Anzahl der weißen und s_i die Anzahl der schwarzen Bohnen nach dem i-ten Zug, wobei $0 \leq i \leq m-1$. Im ersten und zweiten Fall ist folglich $w_i = w_{i-1}$ und $s_i = s_{i-1} - 1$, im dritten Fall jedoch $w_i = w_{i-1} - 2$ und $s_i = s_{i-1} + 1$. Daraus ist nun leicht erkennbar, dass sich die Anzahl der weißen Bohnen nur in Zweier-Schritten verringern kann, während sich die Anzahl der schwarzen in Einer-Schritten erhöhen oder verringern kann.

Nun können wir folgende Invariante für $i > 0$ formulieren:

$$\textbf{Parität}(w_i) = \textbf{Parität}(w_0).$$

Laut Aufgabenstellung sollen jetzt $w_0 = l$ und $s_0 = k$ sein. Wir wissen, dass am Ende nach $m-1$ Schritten genau eine Bohne übrigbleibt. Ist l gerade, dann muss die letzte Bohne schwarz sein, da sich die Anzahl der weißen in Zweier-Schritten auf 0 verringert. Ist l jedoch ungerade, dann bleibt am Ende auf jeden Fall eine weiße Bohne übrig. Das Endergebnis ist folglich unabhängig davon, ob k gerade oder ungerade ist.