Prof. Andreas Goerdt
M. Pippig, M. Riedel, U. Schönke

TU Chemnitz
Wintersemester 2003/2004

Algorithmen und Programmierung

3. Übung – Lösungsvorschläge

1. Aufgabe:
   Sei k_l der Wert der Variable k (Zähl-Variable der Schleife) und m_l der Wert der Variable m (Ergebnisvariable) nach dem l-ten Schleifendurchlauf. Das ursprünglich eingelesene k wird im Folgenden mit k_{start} bezeichnet. Es gilt folgende Invariante für $l \geq 1$:

$$k_l = k_{start} - l \text{ und } m_l = \sum_{j=1}^{l+1} h = (l+1) \cdot h.$$

Induktionsanfang

Vor dem ersten Durchlauf der Schleife ist

$$k_0 = k_{start} \text{ und } m_0 = h.$$

Nach dem ersten Durchlauf ist k um Eins verringert, also

$$k_1 = k_0 - 1 = k_{start} - 1$$

und m um h erhöht, also

$$m_1 = m_0 + h = h + h = 2 \cdot h.$$

Induktionsschritt
Für $1 < l < k_{start}$ ist laut Voraussetzung (Invariante)

$$k_{l-1} = k_{start} - (l-1) = k_{start} - l + 1$$

und

$$m_{l-1} = \sum_{j=1}^{l} h = l \cdot h.$$

Im Schleifenrumpf wird nun k veringert, d. h. es gilt

$$k_l = k_{start} - l + 1 - 1 = k_{start} - l.$$

und m um h erhöht

$$\begin{aligned} m_l &= m_{l-1} + h \\ &= \sum_{j=1}^{l} h + h = l \cdot h + h \\ &= \sum_{j=1}^{l+1} h \quad = (l+1) \cdot h \end{aligned}$$

Gilt also die Invariante nach dem $(l-1)$-ten Schleifendurchlauf, so gilt sie auch nach dem l-ten Durchlauf. Somit ist für $l = k_{start} - 1$

$$\begin{aligned} k_l &= k_{start} - l \\ &= k_{start} - (k_{start} - 1) \\ &= 1 \end{aligned}$$

und

$$\begin{aligned} m_{k_{start}-1} &= \sum_{j=1}^{k_{start}-1+1} h \\ &= \sum_{j=1}^{k_{start}} h \\ &= k_{start} \cdot h \end{aligned}$$

Nach dem $(k_{start}-1)$-ten Schleifendurchlauf ist also die Bedingung der While-Schleife $k > 1$ nicht mehr erfüllt und die Schleife wird verlassen. Am Ende hat $m_{k_{start}-1}$ somit den Wert $k_{start} \cdot h$.

2. Aufgabe:

   a) Wir dividieren im Binärsystem:

$$\begin{array}{cccccccccccccl}
 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & : & 1\ 0\ 1\ 0 = 11001010 \quad \text{Rest } 1001 \\
- & 1 & 0 & 1 & 0 & & & & & & & & \rightarrow 1 & \\ \cline{1-5}
 & 0 & 1 & 0 & 1 & 1 & & & & & & & & \\
- & & 1 & 0 & 1 & 0 & & & & & & & \rightarrow 1 & \\ \cline{1-6}
 & & 0 & 0 & 0 & 1 & 1 & & & & & & \rightarrow 0 & \\
 & & & & & 1 & 1 & 0 & & & & & \rightarrow 0 & \\
 & & & & & 1 & 1 & 0 & 1 & & & & & \\
- & & & & & 1 & 0 & 1 & 0 & & & & \rightarrow 1 & \\ \cline{1-9}
 & & & & & 0 & 0 & 1 & 1 & 1 & & & \rightarrow 0 & \\
 & & & & & & & 1 & 1 & 1 & 0 & & & \\
- & & & & & & & 1 & 0 & 1 & 0 & & \rightarrow 1 & \\ \cline{1-11}
 & & & & & & & 0 & 1 & 0 & 0 & 1 & \rightarrow 0 &
\end{array}$$

$$\begin{array}{ccccccccccl}
 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & : & 1\ 0\ 1\ 0 = 10100 \quad \text{Rest } 10 \\
- & 1 & 0 & 1 & 0 & & & & & \rightarrow 1 & \\ \cline{1-5}
 & 0 & 0 & 1 & 0 & 1 & & & & \rightarrow 0 & \\
 & & & 1 & 0 & 1 & 0 & & & & \\
- & & & 1 & 0 & 1 & 0 & & & \rightarrow 1 & \\ \cline{1-7}
 & & & 0 & 0 & 0 & 0 & 1 & & \rightarrow 0 & \\
 & & & & & & & 1 & 0 & \rightarrow 0 &
\end{array}$$

$$\begin{array}{ccccccccl}
 & 1 & 0 & 1 & 0 & 0 & : & 1\ 0\ 1\ 0 = 10 \quad \text{Rest } 0 \\
- & 1 & 0 & 1 & 0 & & \rightarrow 1 & \\ \cline{1-5}
 & 0 & 0 & 0 & 0 & 0 & \rightarrow 0 &
\end{array}$$

$$1\ 0 : 1\ 0\ 1\ 0 = 0 \quad \text{Rest } 10$$

Schauen wir uns nun die bei der Division enstandenen Reste an:

$$\begin{aligned}1001_2 &= 9_{10}\\ 10_2 &= 2_{10}\\ 0_2 &= 0_{10}\\ 10_2 &= 2_{10}\end{aligned}$$

Der Dualzahl 11111101101_2 entspricht folglich die Dezimalzahl 2029_{10}.

b) Wir dividieren systematisch durch 2:

DIV(1000,2)	=	500	MOD(1000,2)	=	0
DIV(500,2)	=	250	MOD(500,2)	=	0
DIV(250,2)	=	125	MOD(250,2)	=	0
DIV(125,2)	=	62	MOD(125,2)	=	1
DIV(62,2)	=	31	MOD(62,2)	=	0
DIV(31,2)	=	15	MOD(31,2)	=	1
DIV(15,2)	=	7	MOD(15,2)	=	1
DIV(7,2)	=	3	MOD(7,2)	=	1
DIV(3,2)	=	1	MOD(3,2)	=	1
DIV(1,2)	=	0	MOD(1,2)	=	1

Der Dezimalzahl 1000_{10} entspricht folglich die Dualzahl 1111101000_2.

c)

```
import Prog1Tools.IOTools;

public class DezimalBinaer{

  public static void main (String[] args){

    short d;
    long b;
    long position;
    short rest;
    boolean negativ;

    d = IOTools.readShort("Dezimalzahl:␣");
    b = 0;
    position = 1;

    if (d<0) {
      d = (short) -d;
      negativ = true;
```

```
        }
        else {
          negativ = false;
        }

        while (d>0){
          rest = (short) (d%2);
          b = b + (rest * position);
          position = position * 10;
          d = (short) (d/2);
        }

        if (negativ) {
          b = -b;
        }

        System.out.println("entspricht der Binaerzahl " + b);
        return;
      }
    }
```

3. Aufgabe:

   a) Die größte positive darstellbare Zahl ist $2^{n-1}-1$, also

   $$2^9-1 = 511_{10} = 0111111111_{2Kpl}.$$

   Die kleinste negative darstellbare Zahl ist -2^{n-1}, also

   $$-2^9 = -512_{10} = 1000000000_{2Kpl}.$$

   b) $-333_{10} = 1010110011_{2Kpl}$
   $+285_{10} = 0100011101_{2Kpl}$

4. Aufgabe:

   Wir definieren das 10er-Komplement analog zum 2er-Komplement:

   Zahl a im Zweier-Komplement: $\mathrm{MOD}\left(a,2^n\right)$
   Zahl a im Zehner-Komplement: $\mathrm{MOD}\left(a,10^n\right)$

   Gegenüberstellung für $n=4$ Stellen:

	2er-Komplement	10er-Kompl
Zahlenbereich insges.	$2^4 = 16$	$10^4 = 10000$
Aufteilung pos./neg. Zahlen	$\frac{2^4}{2} = 2^3 = 8$	$\frac{10^4}{2} = 5000$
größte pos. darst.-bare Z.	$2^3 - 1 = 7_{10} = 0111_2$ $= 0111_{2Kpl}$	$-\frac{10^4}{2} - 1 = 4999_{10}$ $= 4999_{10Kpl}$
kleinste neg. darst.-bare Z.	$-2^3 = -8_{10}$ $= 1000_{2Kpl}$	$-\frac{-10^4}{2} = -5000_{10}$ $= 5000_{10Kpl}$

Das 10er-Komplement einer Zahl wird analog wie beim 2er-Komplement gebildet:

- Das 10er-Komplement einer positiven Zahl $a \geq 0$ ist a selbst.
- Das 10er-Komplement einer negativen Zahl $a' < 0$ wird folgendermaßen ermittelt (3 Möglichkeiten):
  1) formal nach Def. mit $\text{MOD}\left(a, \text{Basis}^n\right)$
  2) durch Subtraktion jeder Ziffer der entsprechenden positiven Zahl a von der um 1 verringerten Basis (also 9 im 10er-Komplement) und abschließendem Addieren einer 1 zum Ergebnis
  3) Addition der (gesamten) negativen Zahl zu der nächsthöheren Potenz der Basis (bei vorgegebener Stellenanzahl n zu $Basis^n$).

Beispiel (10er-Komplement mit 4 Stellen): -1835_{10}

- nach Methode 1:

$$\begin{aligned} -1835_{10} &= \text{MOD}\left(-1835_{10}, 10^4\right) \\ &= \text{MOD}\left(-1835_{10}, 10000\right) \\ &= 8165_{10Kpl} \end{aligned}$$

- nach Methode 2:

$$\begin{aligned} 9 - 5 &= 4 \\ 9 - 3 &= 6 \\ 9 - 8 &= 1 \\ 9 - 1 &= 8 \end{aligned}$$

  haben also 8164
  Addition von Eins $\Rightarrow 8164 + 1 = 8165_{10Kpl}$

- nach Methode 3:
  $10000 - 1835 = 8165_{10Kpl} = -1835_{10}$
  8165 entspricht offensichtlich $\text{MOD}\left(-1835, 10000\right)$.

Wie kann nun die Subtraktion $a-b$ durch die Addition simuliert werden? Wir formen einfach um $(a+(-b))$ und addieren das 10er-Komplement von $-b$ zu a.

Beispiel:

121 - 9 = 121 + (-9) = 112
Darstellung der 121 mit vier Stellen ist 0121_{10Kpl}
Darstellung der -9 (Zehnerkomplement) ist $\text{MOD}(-9, 10000) = 10000 - 9 = 9991_{10Kpl}$
Addition

$$\begin{array}{ccccc} & 0 & 1 & 2 & 1 \\ + & 9 & 9 & 9 & 1 \\ \hline 1 & 0 & 1 & 1 & 2 \end{array}$$

Durch Abschneiden der vordersten Stelle (mod 10^4) erhalten wir 0112_{10Kpl}, also 112.