Dr. Ekkart Kindler
FG Softwaretechnik
Institut für Informatik

Technische Informatik für Ingenieure
Winter 2006/2007 – Übungsblatt Nr. 5

21. November 2006

Übungsgruppenleiter: K. Hojenski, S. Pook, M. Meyer, A. Znamenshchykov, E. Münch, D. Travkin

Aufgabe 1: (Schleifen)

Gegeben sind die folgenden Programmteile:

I. *for*-Schleife:

```
for(int i = x; i < 3; i = i
+ 3) {
     Out.println(i);
}
```

x:	Ausgabe:
-1	-1, 2
1	1
3	keine Ausgabe

II. *while*-Schleife:

```
int i = x;
while(i < 3) {
     Out.println(i);
     i = i + 3;
}
```

x:	Ausgabe:
-1	-1, 2
1	1
3	keine Ausgabe

III. *do-while*-Schleife:

```
int i = x;
do {
     Out.println(i);
     i = i + 3;
} while(i < 3);
```

x:	Ausgabe:
-1	-1, 2
1	1
3	3

Welche Ausgabe liefern diese Programme wenn für x die Zahlen -1, 1 und 3 eingesetzt werden?

Aufgabe 2: (Schleifen)

Die Schleifen in den folgenden Programmteilen wurden nicht optimal gewählt. Schreiben sie die Programme so um, dass sich eine kompaktere Schreibweise ergibt.

Umständlich:

I.

```
if(y < a) {
    do {
         a = a*2;
         y = a*a - 16;
    } while(y < a);
}
```

II.

```
k = 1;
while( true ) {
    k = k + 1;
    if( k > 13)
         break;
    y = y * 2;
}
```

Besser:

I.

```
while(y < a) {
    a = a*2;
    y = a*a - 16;
}
```

II.

```
for( k = 2; k <= 13; k++) {
    y = y * 2;
}
```

Aufgabe 3: **(Intervallschachtelung)**

Implementieren Sie ein Programm, das die Quadratwurzel einer Zahl näherungsweise berechnet. Dazu soll das Prinzip der Intervallschachtelung benutzt werden.

Intervallschachtelung: Die Intervallschachtelung startet mit einem Intervall, welches so groß gewählt werden muss, dass sich das Ergebnis in diesem Intervall befindet. Dann teilt man dieses Intervall in der Mitte in zwei Teilintervalle und überlegt sich, in welchem der beiden Teilintervalle das Ergebnis liegen muss. Dann fährt man mit diesem Teilintervall fort und teilt es erneut. Dies wiederholt man nun solange, bis die Obergrenze und die Untergrenze des Intervalls nahe genug beieinander liegen (d.h. die gewünschte Genauigkeit des Ergebnisses erreicht ist). Die Mitte dieses letzten Intervalls gibt man dann als Ergebnis aus.

a) Formulieren Sie den Algorithmus für die Berechnung der Quadratwurzel in natürlicher Sprache. Wie muss man für eine Zahl, die größer als 1.0 ist, das Startintervall wählen, damit sich die Quadratwurzel dieser Zahl auf alle Fälle in dem Startintervall befindet? Wie entscheiden Sie nach der Teilung eines Intervalls, in welchem der beiden Teilintervalle die Quadratwurzel liegt?

b) Formulieren Sie eine Abbruchbedingung für Ihren Algorithmus, die eine Genauigkeit des Ergebnisses von zehn Stellen garantiert (relative Genauigkeit).

Absolute Genauigkeit sagt: (obergrenze - untergrenze) < genauigkeit

Relative Genauigkeit sagt:
(obergrenze - untergrenze) / mitte < genauigkeit
(anstelle von mitte kann ersatzweise auch durch die obergrenze oder untergrenze geteilt werden)
Im Programm ist: genauigkeit = 1.0E-10

c) Implementieren Sie Ihren Algorithmus in Java so, dass das Programm die Quadratwurzel einer Zahl, die größer als 1.0 ist, mit einer Genauigkeit von zehn Dezimalstellen berechnet.

```
public class Wurzel {
   public static void main(String[] args) {
      // 10 Stellen Genauigkeit
      double genauigkeit = 1.0e-10;

      Out.print("Geben Sie eine Zahl ein : ");
      double zahl = In.readDouble();

      // Das Ausgangsintervall
      double untergrenze = 1.0;
      double obergrenze = zahl;
      double mitte;

      do {
         // Ermittlung der Mitte des Intervalls
         mitte = (obergrenze + untergrenze) / 2.0;

         // Testen, in welcher Hälfte die Wurzel liegt
         if (zahl < mitte * mitte) {
            obergrenze = mitte;
         }
         else {
            untergrenze = mitte;
         }
      }
      while ((obergrenze - untergrenze) / mitte > genauigkeit);

      Out.println("Die Wurzel ist : " + mitte);
   }

}
```

d) Was passiert, wenn Sie diesen Algorithmus mit einer Zahl zwischen 0.0 und 1.0 starten?
Das Programm springt beim ersten Durchlauf aus der Schleife (Obergrenze muss im Programm auf 1.0, Untergrenze auf 0.0 gesetzt werden).

e) Ergänzen Sie das Programm aus c) so, dass es auch die Quadratwurzel einer Zahl zwischen 0.0 und 1.0 korrekt berechnet!

```
public class Wurzel {
   public static void main(String[] args) {
      //10 Stellen Genauigkeit
      double genauigkeit = 1.0e-10;

      Out.print("Geben Sie eine Zahl ein : ");
      double zahl = In.readDouble();

      // Das Ausgangsintervall
      double untergrenze = 0.0;
      double obergrenze = zahl;
      double mitte;

      if(zahl < 1.0) {
         obergrenze  = 1.0;
      }

      do {
         // Ermittlung der Mitte des Intervalls
         mitte = (obergrenze + untergrenze) / 2.0;

         // Testen, in welcher Hälfte die Wurzel liegt
         if (zahl < mitte * mitte) {
            obergrenze = mitte;
         }
         else {
            untergrenze = mitte;
         }
      } while ((obergrenze - untergrenze) / mitte > genauigkeit);

      Out.println("Die Wurzel ist : " + mitte);
   }
}
```