RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

Lehrstuhl für Hochleistungsrechnen
RWTH Aachen · D-52074 Aachen · GERMANY
http://programmierung.informatik.rwth-aachen.de

Prof. Christian Bischof, Ph.D.
Andre Vehreschild, Jakob T. Valvoda

Übung *Programmierung WS 06/07* – Blatt 4

Lösungen müssen bis zum **20. November 2006, 17:00 Uhr** in den Kasten Ihrer Übungsgruppe eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Bitte vergessen Sie nicht, die **Nummer Ihrer Übungsgruppe**, sowie die **Namen** und **Matrikelnummern** der Mitglieder Ihrer 2er-Gruppe auf jedes Lösungsblatt Ihrer Abgabe zu schreiben. Bitte heften Sie ihre Blätter zusammen.

Alle erstellten Java-Programme sind sowohl in gedruckter Form abzugeben, als auch per E-Mail an den jeweiligen Tutor zu senden. Vergessen Sie auch bei der elektronischen Abgabe per E-Mail nicht die Angabe der **Nummer Ihrer Übungsgruppe**, sowie die jeweiligen **Namen** und **Matrikelnummern**.

Aufgabe 1 (1 + 1 + 1 = 3 Punkte)

Die drei in Java zur Verfügung stehenden Schleifen `while`, `do` und `for` erlauben eine effiziente Notation für unterschiedliche Problemfälle. Sie sind jedoch in sich redundant, können also ineinander umgewandelt werden. Realisieren Sie die nachfolgenden Code-Sequenzen mit den jeweils anderen beiden Schleifenarten. Vergessen Sie hierbei nicht evtl. notwendige Deklarationen.

a)
```
double eingabe = AdvancedIO.readDouble();
double epsilon = 1.0e-10;
while( eingabe > epsilon ) {
  eingabe = eingabe / 2.0;
}
```

b)
```
int laenge = AdvancedIO.readInt();
double summe = 0.0;
for( int i = 0; i < laenge; i++ ) {
  summe += i * i / ( 2.0 * ( i + 1 ) );
}
```

c)
```
int zahl = 0;
do {
  System.out.print( "Bitte Zahl >= 0 eingeben: " );
  zahl = AdvancedIO.readInt();
} while( zahl < 0 );
```

Aufgabe 2 (5 Punkte + 1 Bonuspunkt)

In der Vorlesung wird ein Sortieralgorithmus behandelt, welcher das gesamte zu sortierende Array $a_0, \ldots, a_n$ einmal durchläuft und jeweils das aktuelle Element a_i mit allen Nachfolgern $a_{i+1}, \ldots, a_n$ vergleicht. Wird ein kleineres Element a_j gefunden, d.h. $a_i > a_j$, dann wird es mit dem aktuellen Element getauscht. Der Nachteil dieses Algorithmus besteht darin, dass viele Vertauschungen stattfinden müssen.

In dieser Aufgabe sollen Sie eine Variante implementieren, welche das Sortierproblem im Allgemeinen mit weniger Vertauschungen löst. Hierbei wählt man die folgende Strategie: Das Array $a_0, \ldots, a_n$ wird wiederum einmal durchlaufen, diesmal jedoch in umgekehrter Reihenfolge, und zwar für $i = n, \ldots, 1$. In einer inneren Schleife wird bis zum jeweils aktuellen Element a_i der Anfang des Arrays durchlaufen, wobei nacheinander die Nachbarn a_j und a_{j+1} für $j = 0, \ldots, i-1$ verglichen werden. Ist der linke Nachbar größer ($a_j > a_{j+1}$), dann werden die benachbarten Elemente vertauscht und es wird das nächste Paar verglichen. Ist der linke Nachbar nicht größer, dann wird direkt das nächste Paar verglichen. Nach jeder Iteration sind alle großen Elemente etwas weiter nach rechts gerückt und es ist garantiert, dass das größte Element der aktuell betrachteten Sequenz $a_0, \ldots, a_i$ auf der Position a_i steht.

Nachfolgendes Beispiel veranschaulicht die Arbeitsweise des Algorithmus. Das aktuell betrachtete Paar a_j, a_{j+1} ist in runde Klammern eingefasst. Findet eine Vertauschung mit dem rechten Nachbarn statt, so sind beide fett hervorgehoben. Der jeweilige Index i ist nach dem Element a_i mit einer schließenden eckigen Klammer] gekennzeichnet.

1. Durchlauf, $i = 4$	(**48**	**03**)	83	18	31]
	03	(48	83)	18	31]
	03	48	(**83**	**18**)	31]
	03	48	18	(**83**	**31**)]
2. Durchlauf, $i = 3$	(03	48)	18	31]	83
	03	(**48**	**18**)	31]	83
	03	18	(**48**	**31**)]	83
3. Durchlauf, $i = 2$	(03	18)	31]	48	83
	03	(18	31)]	48	83
4. Durchlauf, $i = 1$	(03	18)]	31	48	83

Implementieren Sie das beschriebene Sortierverfahren. Ihr Programm sollte den Benutzer auffordern, die Größe des Arrays $m = n + 1$ festzulegen und danach die einzelnen Elemente (vom Typ `int`) abfragen. Überprüfen Sie, ob $m > 0$ und fordern Sie andernfalls den Benutzer zu einer Neueingabe auf. Zum Schluß soll das sortierte Array ausgegeben werden.

In dieser Aufgabe können Sie einen Bonuspunkt erhalten, wenn Sie das Sortierverfahren dahingehend optimieren, dass Sie, statt dem kompletten Durchlaufen der Elemente für $i = n, \ldots, 1$ eine effiziente Abbruchbedingung angeben und implementieren.

Aufgabe 3 (5 Punkte)

Zur Berechnung der Liste von Primzahlen bis zu einem Schwellenwert n kann das Verfahren „Sieb des Eraosthenes“ verwendet werden. Man schreibt hierbei die Zahlen $2, \ldots, n$ auf. In jeder Iteration wählt man die nächste kleinste unmarkierte Zahl p_i mit $p_i > p_{i-1}$ und markiert bzw. streicht alle Vielfachen dieser Zahl $k \cdot p_i$, $k \in \mathbb{N}$ für die gilt $p_i^2 \leq k \cdot p_i \leq n$. Diesen Prozess beginnt man mit $p_0 = 2$. Alle am Ende nicht markierten Zahlen sind Primzahlen.

Beispiel: es werden zunächst die entsprechenden Vielfachen von $p_0 = 2$ markiert bzw. gestrichen: $4, 6, 8, \ldots$, danach die von der nächsten unmarkierten Zahl $p_1 = 3$, also $9, 12, 15, \ldots$, danach die entsprechenden Vielfachen von $p_2 = 5$, usw.

Schreiben Sie ein Programm, welches alle Primzahlen bis zu einem Schwellenwert n (abhängig von der Benutzereingabe, Wert überprüfen!) mit dem oben beschriebenen Verfahren berechnet. Das Programm soll im Anschluss die Primzahlen auf der Konsole ausgeben. Sie sollten in Ihrem Programm auf die optimale Abbruchbedingung der Schleife achten.

Aufgabe 4 (2 + 3 + 3 = 8 Punkte)

Mit einem zweidimensionalen `char`-Array können im Textmodus auf der Konsole einfache Zeichnungen realisiert werden, indem zuerst das Array beschrieben und am Ende einmal ausgegeben wird.

a) Realisieren Sie eine solche Zeichenfläche als zweidimensionales `char`-Array. Verwenden Sie zwei Konstanten für die Breite b und die Höhe h. Initialisieren Sie alle Elemente des Arrays mit einem Leerzeichen. Erzeugen Sie einen Rahmen der Breite 1 und stellen Sie diesen mit dem Zeichen `*` dar. Der Rahmen ist hierbei ein Bestandteil der Zeichenfläche und kann ggf. übermalt werden. Gemalt wird, indem für eine gewünschte Position (x, y) an entsprechender Stelle im Array der Wert `#` gesetzt wird. Der Einfachheit halber soll der Ursprung $(0, 0)$ der Zeichenfläche oben links dargestellt werden. Die positive x-Achse zeigt nach rechts, die positive y-Achse nach unten.

   Malen Sie zwei unterschiedliche Punkte (Benutzereingabe) auf Ihre Zeichenfläche und geben Sie die Zeichenfläche aus.

   Beispiel: $b = 20$, $h = 5$; linker Punkt $p_0 = (3, 2)$, unterer rechter Punkt $p_1 = (17, 4)$ (befindet sich im Rahmen)

```
**********************
*                    *
*  #                 *
*                    *
******************#***
```

b) Geraden zwischen zwei Punkten $p_0 = (x_0, y_0)$ und $p_1 = (x_1, y_1)$ können mit der allgemeinen Geradengleichung berechnet werden. Diese lautet:

$$y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0$$

Bis auf vertikale Geraden, d.h. $x_0 = x_1$ kann mit dieser Gleichung zu jedem x-Wert der entsprechende y-Wert berechnet werden.

Implementieren Sie ein Programm, welches als Eingabe jeweils die x und die y-Koordinate (als `int`) zweier Punkte anfordert und dann die Verbindungsgerade in die im Aufgabenteil a) implementierte Zeichenfläche einzeichnet. Setzen Sie hierfür ganzzahlige x-Werte in die allgemeine Geradengleichung ein. Geben Sie anschliessend die Zeichenfläche aus. Vertikale Geraden können mit diesem Ansatz nicht gezeichnet werden. Ihr Programm sollte entsprechende Eingaben abfangen.

Hinweis: bei Steigungen ab $\pm 45°$ kann die Gerade Lücken aufweisen.

Beispiel: Zeichenfläche $b = 20$, $h = 10$; Punkte $p_0 = (24, 13)$, $p_1 = (2, 3)$

```
**********************
*                    *
*                    *
* ###                *
*    ##              *
*      ##            *
*        ##          *
*          ##        *
*            ###     *
******************##**
```

c) Für Linien mit einer Steigung kleiner oder gleich 45° wird in der Computergraphik ebenfalls der folgende Algorithmus verwendet.

Gegeben seien zwei ganzzahlige Punkte, der Startpunkt $p_s = (x_s, y_s)$ und der Endpunkt $p_e = (x_e, y_e)$. Es soll die Gerade zwischen diesen Punkten gezeichnet werden. Sei o.B.d.A. $x_s < x_e$ (ansonsten werden beide Punkte vertauscht). Da nur Geraden mit einer Steigung im Intervall $[0°; 45°]$ betrachtet werden, gilt somit $y_s \leq y_e$. Ferner gilt, dass die Gerade in $n = x_e - x_s$ Schritten gezeichnet werden kann. Ausgegangen wird vom (ganzzahligen) Startpunkt $(x_0, y_0) = (x_s, y_s)$. Anstatt in jeder Iteration $i = 1, \ldots, n$ den y-Wert mit der allgemeinen Geradengleichung zu berechnen, wird der jeweils letzte Wert y_{i-1} für die Berechnung verwendet. Anhand einer Fehlerabschätzung wird entschieden, ob der Punkt (x_i, y_i) an der Position $(x_{i-1}+1, y_{i-1})$ oder $(x_{i-1}+1, y_{i-1}+1)$ gesetzt wird. Hierfür addiert man die Steigung der Geraden $\frac{y_e - y_s}{x_e - x_s}$ in jeder Iteration zu einer Fehlervariablen. Diese gibt an, wie weit sich die ganzzahlige Koordinate von der tatsächlichen Geraden entfernt. Solange der Wert der Fehlervariablen kleiner als 0.5 ist, wählt man $(x_{i-1} + 1, y_{i-1})$, sonst inkrementiert man den y-Wert und wählt $(x_{i_1} + 1, y_{i-1} + 1)$ als Koordinate (x_i, y_i). Im letzteren Fall wird die Fehlervariable entsprechend ausgeglichen, d.h. um den Wert 1.0 dekrementiert. Die Berechnung wird nach n Schritten beendet, also sobald $x_i = x_e$ gilt.

Durch diese Strategie tauscht man pro Iteration eine Multiplikation und eventuelles Runden, welche bei der Berechnung mit der allgemeinen Geradengleichung anfallen, gegen eine Addition, welche rechnerintern im Allgemeinen wesentlich effizienter durchgeführt werden kann.

Implementieren Sie den oben beschriebenen Algorithmus für Geraden mit einer Steigung kleiner oder gleich 45°, d.h. $0 \leq \frac{y_e - y_s}{x_e - x_s} \leq 1$. Fordern Sie den Benutzer zur Eingabe von zwei Punkten auf. Überprüfen Sie die Eingabe hinsichtlich der Steigung und tauschen Sie ggf. den Start- und den Enpunkt. Fordern Sie den Benutzer zur Neueingabe auf, falls keine Gerade mit einer Steigung kleiner oder gleich 45° gezeichnet werden kann. Zeichnen

Sie die mit dem beschriebenen Verfahren berechnete Gerade in die im Aufgabenteil a) implementierte Zeichenfläche ein. Geben Sie die Zeichenfläche am Ende aus.