Übungsblatt 08

zu bearbeiten bis 01.07.2010

Geschachtelte Schleifen und Arrays

Mit der WHILE-Schleife haben wir uns seit Beginn unserer Vorlesung angefreundet, und seit neuestem kennen wir ja auch den sehr nahen Verwandten: die FOR-Schleife. Schleifen können auch ineinander geschachtelt werden, und dieses Prinzip erlaubt eine sehr übersichtliche Programmierung mit weitreichenden Möglichkeiten. Deshalb widmen wir den Schwerpunkt dieses Übungsblattes den verschachtelten Schleifen.

Ein zweites sehr wichtiges Themengebiet steht dann in der letzten Aufgabe im Fokus: die Arrays.

Aufgabe 1 – FOR-Schleifen und WHILE-Schleifen (16 Punkte)

Es ist für unsere Zwecke im Wesentlichen eine Frage des persönlichen Geschmacks, ob man lieber mit WHILE- oder lieber mit FOR-Schleifen arbeiten möchte. Selbstverständlich sind auch Mischformen möglich.

Auf der Homepage findet Ihr das Programm „`JustWhile.java`“, das zwei ineinander geschachtelte WHILE-Schleifen enthält. Nehmt dieses, um sukzessive die nachstehenden Aufgaben zu lösen, für die es jeweils 4 Punkte gibt.

Teilaufgabe a)

Übersetzt das Programm und lasst es laufen. Vollzieht nach (d.h.: erklärt Eurem Tutor), warum die Ausgabe so ist, wie sie ist. Was passiert genau an den Stellen, an denen jeweils die Zeile mit den Bindestrichen ausgegeben werden? (Konkret: macht Euch die Funktionsweise der inneren und der äußeren WHILE-Schleife klar!)

Teilaufgabe b)

Schreibt das Programm so um, dass es anstelle der *inneren* WHILE-Schleife eine FOR-Schleife verwendet (die *äußere* WHILE-Schleife soll unverändert bleiben).

Teilaufgabe c)

Schreibt das ursprüngliche Programm (d.h. nicht Eure Lösung aus Teil b) so um, dass es anstelle der *äußeren* WHILE-Schleife eine FOR-Schleife verwendet. (Diesmal soll also die *innere* WHILE-Schleife unverändert bleiben.)

Teilaufgabe d)

Schreibt das Programm nun so um, dass es *keinerlei* WHILE-Schleifen, sondern nur noch (zwei) FOR-Schleifen verwendet.

Aufgabe 2 – Eine selbstgestrickte „formatierte“ Ausgabe (8 Punkte)

Schreibt ein Programm, das von der Standardeingabe eine Zahl zwischen 1 und 9 einliest. Abhängig von dieser eingegebenen Zahl soll es dann eine Ausgabe produzieren, die wir Euch auf der Homepage in der Datei „`output_A2.txt`“ zur Verfügung stellen.

Selbstverständlich sollte Euer Programm so einfach wie möglich geschrieben sein – und dass bedeutet: verwendet eine geschachtelte Schleife!

Aufgabe 3 – Optimierung beim Würger King (16 Punkte)

Bei Eurem ersten Tag im neuen Job beim Würger King werdet Ihr mit einer Optimierungsaufgabe empfangen. Euer Chef jammert, dass man die Zubereitung der leckeren Soße auf dem momentan so gefragten WM-Burger noch nicht perfektioniert hat und bittet Euch als frischgebackene WiWis um Rat. Dazu erklärt er Euch den Produktionsablauf für ein Fass Geheimsoße:

Zeit: Die Soße muß mindestens 3 Stunden garen, darf aber höchstens 6 Stunden garen. Einstellbar ist der Garvorgang in Schritten von jeweils einer Viertelstunde.

Geheimzutat: Die Soße enthält eine geheime Zutat, durch welche sich die Soße beim Würger King von jener bei McDoof unterscheidet. Was genau es ist, wissen nur die Putzhilfe und der Hausmeister, aber Ihr erfahrt, dass man von dieser Zutat mindestens 1 kg zugeben muss, jedoch höchstens 5 kg zugeben darf. Stückeln lässt sie sich in halben Kilos.

Wasser: Für jedes Fass Soße benötigt man mindestens 4, jedoch höchstens 8 Liter Wasser. Die Dosierung erfolgt in auf 0,2 Liter genauen Schritten.

Während Ihr beginnt, Euch nach dem Lecker-Und-Fein-Essen aus der Mensa zurückzusehnen, wird Euch der Chef-Mathematiker vorgestellt, der im Rahmen seiner Doktorarbeit eine bahnbrechende Formel zum Dick-Mach-Faktor des WM-Burgers entdeckt hat (siehe Kasten unten). Ihr findet die Formel zwar unglaublich obskur, aber am ersten Tag widerspricht man halt nicht...

Nun gibt Euch Euer Chef den Auftrag, diejenige (zulässige) Kombination aus Zeit, Menge an Geheimzutat und Wasser herauszufinden, bei der sich der *kleinste* Wert für den Dick-Mach-Faktor ergibt.

Um diese Aufgabe zu lösen, steht Euch ein leistungsfähiger Rechner mit der Programmiersprache Java zur Verfügung. Weil dieser Rechner leistungsstark ist, spricht auch nichts dagegen, einfach mal alle möglichen Kombinationen durchzuprobieren und auf diese Weise diejenige zu ermitteln, die zum geringsten Dickmach-Faktor führt.

Schreibt also ein Java-Programm, das durch bloßes Ausprobieren aller Möglichkeiten das gewünschte Ergebnis berechnet und ausgibt (d.h. den kleinstmöglichen Dick-Mach-Faktor sowie die dazu notwendige Zutatenkombination).

Aufgabe 4 – Spaß mit Arrays (10 Punkte)

Zum Abschluß für diese Woche noch eine kleine Aufgabe, die den Umgang mit Arrays einüben soll:

Teilaufgabe a) – Lückentext (6 Punkte)

Auf unserer Homepage findet Ihr die Datei „`Array.frag`“, die ein Fragment eines Java-Programms enthält. Ergänzt die fehlenden Stellen derart, dass dabei ein Programm entsteht, das genau das macht, was in den Kommentaren in dieser Datei beschrieben steht. (Als Ergebnis müsste übrigens 941 herauskommen.)

Teilaufgabe b) – Verallgemeinerung (4 Punkte)

Die 10 Zahlen, mit denen das Array aus Teilaufgabe a) gefüllt ist, stehen fest im Programm drin. Ändert Eure Lösung aus Teilaufgabe a) so ab, dass das Array seinen Inhalt von der Standardeingabe erhält (die Dimension 10 soll dabei erhalten bleiben).

Die Formel zum Dick-Mach-Faktor:

$$\frac{\frac{1}{\text{Zeit in Minuten}} \cdot (\text{Gewicht in kg} + 0,3 \cdot \text{Wasser in Liter})}{(\text{Wasser in Liter} - 3 \cdot \text{Gewicht in kg}^{-\text{Zeit in Stunden}})^2}$$

(Bitte auf Punkt- vor Strichrechnung sowie die Einheiten bei der Zeit achten!)