Programmierstarthilfe WS 2008/09
Fakultät für Ingenieurwissenschaften und Informatik

5. Blatt
Für die Woche vom 17.11. bis zum 21.11.2008 (KW 47)

Organisatorisches

Die Webseiten zur Veranstaltung sind unter `http://www.uni-ulm.de/in/mi/lehre/2008ws/programmierstarthilfe.html` zu finden.

Bevor du die Aufgaben bearbeitest sollst du im Skript den entsprechenden Abschnitt durchlesen. Falls du Fragen zum Stoff im Skript hast, stehen wir dir gerne zur Verfügung.
Achtet auch darauf, eure Programme sauber zu formatieren, das heißt Einrücktiefen zu beachten, Variablen sinnvoll zu benennen und Kommentare hinzuzufügen. Ein so formatiertes Programm ist einfacher zu lesen und zu verstehen. Dies ist wichtig, denn oft findet man Probleme sehr schnell, sobald die Anweisungen richtig eingerückt sind, und man so die nötige Übersicht erlangt.

1 Aufgaben

1.1 Ungerade Zahlen

Schreibe ein Programm, das ein Array mit den ungeraden Zahlen von 1 bis 99 füllt. Anschließend sollen die Zahlen in umgekehrter Reihenfolge wieder aus dem Array ausgelesen und ausgegeben werden.

Tipps: Man kann dafür zwei unabhängige `for`-Schleifen nutzen, eine zum Füllen, eine zum Ausgeben. Mach dir auch Gedanken, wie groß das Array sein sollte.

1.2 Suchen in Arrays

Schreibe ein Programm, das zu Anfang ein Array von Zeichen deklariert und ein Zeichen, das in dem Array gefunden werden soll, also etwa so:

```
1 char[] buchstaben = {'a','u','b','d','f','s','k'};
2 char zu_finden = 'd';
```

Danach soll das Programm die Position ausgeben, an der das zu findende Zeichen zum ersten Mal in dem Array vorkommt. In unserem Beispiel wäre das 3, denn das erste Zeichen im Array hat die Position 0, und an Position 3 steht das zu findende Zeichen `d`.

Dieses Programm soll natürlich auch funktionieren, wenn man das Array oder das zu findende Zeichen ändert, sogar, wenn sich die Länge des Array ändert.

Tipp: Die Anführungsstriche im Quellcode sind die gewöhnlichen einfachen Anführungszeichen `'`, das ist das Zeichen über dem #.

1.3 Sieb des Eratosthenes

Ein berühmtes und sehr altes Verfahren zur Primzahlbestimmung ist das Sieb des Eratosthenes. Mit Papier und Bleistift geht das so:

Man schreibe sich alle Zahlen von 2 bis zur gewünschten Größe, z.B. 10000, auf. Die 2 ist eine Primzahl. Um das kenntlich zu machen, male man einen Kringel um die 2. Alle Vielfachen von 2 sind aber dann keine Primzahlen, also streiche man die Zahlen 4, 6, 8, ... durch. Die erste Zahl die nun weder umkringelt noch durchgestrichen ist, ist dann die 3. Das ist wieder eine Primzahl, also wird sie umkringelt und alle Vielfachen werden getrichen. Die 4 überspringt man nun, da sie schon durchgestrichen ist. So fährt man nun schrittweise fort: Man wählt die nächste noch nicht markierte Zahl, umkringelt sie als Primzahl und streicht ihre Vielfachen. Am Ende sind alle Zahlen entweder gestrichen oder als Prinzahlen umkringelt.

Aufgabe: Schreibe ein Programm, das die Überprüfung vornimmt, ob eine Zahl p ($p \leq 10000$) eine Primzahl ist. Das Programm soll nach dem Prinzip des Sieb des Eratosthenes vorgehen. Zu Beginn deines Programmes, solltest du also ein Array von Zahlen anlegen und darauf das Sieb-Verfahren anwenden.

Tipps zur Implementierung: Es ist eleganter, anstatt im Array die Zahlen selbst zu speichern, lediglich für jede Position zu speichern, ob sie bereits gestrichen wurde. Auf jeden Fall sollte man darauf achten, dass beim Streichen der Vielfachen die Arraylänge nicht überschritten wird.

2 Bonusaufgaben

Durch die Aufgaben oben hast du ein Verständnis für das neue Konzept dieses Blatts bekommen. Durch Bonusaufgaben kannst du nun deine Kenntnisse vertiefen.

2.1 Sortieren in Arrays

Viele Algorithmen, die auf und mit Arrays arbeiten, müssen oft den Inhalt zweier Arrayfelder tauschen oder Arrays sortieren.

a) Schreibe ein Programm, das in einem eindimensionalen Array von Zahlen zwei Felder vertauscht.

b) Erweitere dein Programm und schreibe den sogennanten Bubblesort Algorithmus: Ein Array von Zahlen wird von vorne nach hinten durchlaufen. Wird eine Zahl gefunden, die größer ist als ihr Nachfolger im Array, so werden diese zwei Zahlen vertauscht. Wurde das Array durchlaufen, fängt man wieder von vorne an, bis das Array schließlich komplett sortiert ist.

2.2 Matrix-Welle (alte Klausuraufgabe)

Schreibe ein Programm in Java, das eine $n \times n$-Matrix wie folgt initialisiert und gib an, wie eine 5×5-Matrix erzeugt wird.

$$A = \begin{pmatrix} 1 & 2 & 3 & \cdots & n-1 & n \\ 2 & 2 & 3 & \cdots & n-1 & n \\ 3 & 3 & 3 & \cdots & n-1 & n \\ \vdots & \vdots & \vdots & \ddots & n-1 & n \\ n-1 & n-1 & n-1 & n-1 & n-1 & n \\ n & n & n & n & n & n \end{pmatrix}$$

2.3 Nicht-rechteckige Arrays

Ein mehrdimensionales Array ist bekanntlich nichts anderes als ein Array, in dem Array gespeichert sind. Mehrdimensionale Arrays müssen nicht rechteckig sein, das heißt die Arrays, die im Inneren gespeichert sind, dürfen verschiedene Längen haben.

Schreibe ein Programm, das ein dreieckiges Char-Array, das mit `'*'` gefüllt ist, anlegt und ausgibt. Probiere auch andere Formen aus, z.B. Formen mit mehreren Zacken.

2.4 Polynome

Entwickle eine Anwendung zur Berechnung des Funktionswerts von Polynomen $p(x) = a_n x^n + ... + a_1 x^1 + a_0$. Das Programm soll zuerst den Grad n des Polynoms als `int` und dann die Koeffizienten a_i (als `doubles`) in ein entsprechend zu dimensionierendes Array einlesen. Danach sollen so lange Argumente x eingelesen und die Funktionswerte $p(x)$ ausgegeben werden, bis für x die 0 eingegeben wird.

3 Für das nächste Blatt

Nächste Woche werden wir Methoden und Gültigkeit behandeln. Lies dazu die Kapitel *Methoden* und *Gültigkeit* im Skript.