Samuel Ferraz-Leite, Dirk Praetorius
Christoph Überhuber

Sommersemester 2008
23. April 2008

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Die Aufgaben mit Stern (*) sind bis zur nächsten Übung vorzubereiten und werden dort abgeprüft. Die übrigen Aufgaben dienen nur Ihrer Übung und mir als zusätzliche Grundlage für den Prüfungsstoff in den schriftlichen Tests. Kopieren Sie bitte den Source-Code in ein Unterverzeichnis `serie05` *Ihres Home-Verzeichnisses. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können. In den folgenden Aufgaben sollen* ***dynamische Matrizen*** *und* ***Strukturen*** *geübt werden.*

Aufgabe 41*. Für eine Matrix $A \in \mathbb{R}^{m \times n}$ ist die Spaltensummennorm durch

$$\|A\| = \max_{k=1,\ldots,n} \sum_{j=1}^{m} |A_{jk}|$$

gegeben. Schreiben Sie eine Funktion `spaltensummennorm`, die die Spaltensummennorm einer Matrix A berechnet. Dabei soll die Matrix spaltenweise als Vektor gespeichert werden. Schreiben Sie ein aufrufendes Hauptprogramm, in dem die Zeilen- und Spaltendimensionen $m, n \in \mathbb{N}$ und A eingelesen werden und $\|A\|$ ausgegeben wird.

Aufgabe 42*. Gegeben sei eine obere Dreiecksmatrix $U \in \mathbb{R}^{n \times n}$ mit $u_{jj} \neq 0$ für alle $j = 1, \ldots, n$. Zu gegebenem $y \in \mathbb{R}^n$ existiert dann ein eindeutiges $x \in \mathbb{R}^n$ mit $Ux = y$. Man schreibe mittels geeigneter Schleifen eine Funktion `solveU`, die x berechnet. Dabei soll auf die offensichtlichen Nulleinträge von U nicht zugegriffen werden, um den Rechenaufwand gering zu halten. Schreiben Sie ferner ein Hauptprogramm, in dem die Dimension $n \in \mathbb{N}$ sowie die Einträge der Matrix U und der rechten Seite y eingelesen und die Lösung x ausgegeben werden.
— Um den Algorithmus herzuleiten, schreibe man das Matrix-Vektor-Produkt $y = Ux$ komponentenweise für y_j mit $j = 1, \ldots, n$ als Summe hin. Man überlege, wie die spezielle Gestalt von U die Laufindizes der Summe vereinfacht und löse diese Gleichung nach x_j auf.

Aufgabe 43*. Man schreibe einen Strukturdatentyp `vector` zur Speicherung von `double`-Vektoren der Länge n. Die Struktur enthalte neben der Dimension n den dynamischen Datenvektor. Ferner schreibe man die zugehörigen Funktionen `mallocVector`, `freeVector`, `getVectorLength`, `getVectorEntry`, `setVectorEntry`. Dieser Strukturdatentyp sowie die Zugriffs- und Speicherverwaltungsfunktionen sollen in den Aufgaben 44-47 dieser Übungsserie verwendet werden.

Aufgabe 44*. Man schreibe eine Funktion `minmaxmean`, die von einem gegebenem Vektor $x \in \mathbb{R}^n$ das Minimum $\min_{j=1}^n x_j$, das Maximum $\max_{j=1}^n x_j$ sowie den Mittelwert $\frac{1}{n}\sum_{j=1}^n x_j$ berechnet und zurückgibt. Verwenden Sie den Strukturdatentyp und die Zugriffsfunktionen aus Aufgabe 43. Ferner schreibe man ein aufrufendes Hauptprogramm, das $n \in \mathbb{N}$ sowie den Vektor $x \in \mathbb{R}^n$ einliest und Minimum, Maximum und Mittelwert von x ausgibt. Da eine C-Funktion maximal einen elementaren Datentyp als Return-Wert zurückliefern kann, realisiere man die Rückgabe mittels Call by Reference (Pointer!).

Aufgabe 45. *Bubble-Sort* ist ein ineffizienter, aber kurzer Sortier-Algorithmus: Man vergleicht aufsteigend jedes Element x_j eines Vektors $x \in \mathbb{R}^n$ mit seinem Nachfolger x_{j+1} und — falls notwendig —

vertauscht die beiden. Nach dem ersten Durchlauf muss zumindest das letzte Element bereits am richtigen Platz sein, d.h. es gilt dann $x_n \geq x_j$ für alle $j = 1, \ldots, n-1$. Der nächste Durchlauf muss also nur noch bis zur vorletzten Stelle gehen, usw. Wie viele geschachtelte Schleifen braucht dieses Vorgehen? Man schreibe eine Funktion `bubblesort`, die einen gegebenen Vektor $x \in \mathbb{R}^n$ mittels Bubble-Sort aufsteigend sortiert, d.h. $x_1 \leq x_2 \leq \cdots \leq x_n$. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Vektor x einliest und in sortierter Reihenfolge ausgibt. — Man mache sich Bubble-Sort zunächst am Beispiel $x = (1, 3, 2, 5, 4)$ klar. Was ist der Vorteil von Bubble-Sort gegenüber Min-Sort aus der Vorlesung?

Aufgabe 46. Man schreibe eine verbesserte Variante von Bubble-Sort, bei der man sich die Stelle des letzten Tausches merkt. Ab dieser Stelle müssen die Daten ja bereits sortiert sein. Der nächste Durchlauf braucht also nur noch bis zu dieser Vektor-Position zu gehen.

Aufgabe 47. Schreiben Sie eine Funktion `unique`, die einen Vektor $x \in \mathbb{R}^n$ aufsteigend sortiert, doppelte Einträge streicht und den Vektor in gekürzter Form zurückgibt. Die Funktion soll also beispielsweise den Vektor $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ durch den Vektor $x = (1, 3, 4, 5) \in \mathbb{R}^4$ überschreiben. Um diese Aufgabe zu realisieren, müssen Sie eine Funktion `reallocVector` implementieren. Schreiben Sie ein aufrufendes Hauptprogramm, in dem n und $x \in \mathbb{R}^n$ eingelesen werden und das Ergebnis der Funktion `unique` ausgegeben wird.

Aufgabe 48. Man implementiere eine Funktion `eratosthenes`, die das *Sieb des Eratosthenes* realisiert. Dies ist ein Algorithmus, mit dem alle Primzahlen bis zu einer bestimmten Zahl `nmax` errechnet werden können (benannt nach dem griechischen Mathematiker Eratosthenes). Der Algorithmus sieht folgendermaßen aus:

- Man legt eine Liste (Vektor) $\texttt{prim} = (2, \ldots, \texttt{nmax}) \in \mathbb{N}^{\texttt{nmax}-1}$ an.
- Man streicht aus der Liste alle Vielfachen der ersten Zahl (also der Zahl 2).
- Wähle, solange es noch höhere Zahlen gibt, die nächsthöhere nicht durchgestrichene Zahl und streiche alle ihre Vielfachen.

Der Rückgabevektor `prim` soll am Ende (gekürzt auf minimale Länge) alle Primzahlen $\leq$ `nmax` enthalten (Sie müssen zusätzlich die Länge des Rückgabevektors zurückgeben!). Realisieren Sie das Streichen geeignet, z.B. indem Sie die entsprechenden Einträge auf 0 setzen.

Aufgabe 49. Schreiben Sie eine Funktion `primfaktoren`, die für eine natürliche Zahl $n \in \mathbb{N}$ deren Primfaktoren bestimmt und als Array $p \in \mathbb{N}^k$ zurückgibt. Die Koeffizienten p_j des Arrays $p \in \mathbb{N}^k$ sind also Primzahlen, und es gilt $n = \prod_{j=1}^{k} p_k$. Um eine Liste aller möglichen Primfaktoren zu erhalten, verwende man das Sieb des Eratosthenes aus Aufgabe 48.

Aufgabe 50. Das Produkt $U = AB$ zweier oberer Dreiecksmatrizen $A, B \in \mathbb{R}^{n \times n}$ ist wieder eine obere Dreiecksmatrix. Man beweise diese Aussage zunächst mathematisch, indem man sich die Formel für das Matrix-Matrix-Produkt hinschreibe und mittels der Voraussetzung an A und B die Indizes vereinfache. Danach schreibe man eine Funktion `matrixmatrixU`, die die Produktmatrix berechnet und zurückgibt. Dabei sollen natürlich nur die nicht-trivialen Einträge von U, d.h. U_{jk} für $j \leq k$, berechnet werden. Ferner soll auf die trivialen Einträge von A und B nicht zugegriffen werden, d.h. man verwende die anfangs hergeleitete Formel. Man schreibe ferner ein Hauptprogramm, in dem die Dimension $n \in \mathbb{N}$ sowie die Einträge der Matrizen A, B eingelesen und die Matrix U ausgegeben werden.