Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2008/09
05. November 2008

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Die Aufgaben mit Stern (*) sind bis zur Übung in der kommenden Woche vorzubereiten und werden dort abgeprüft. Die übrigen Aufgaben dienen nur Ihrer Übung und uns als zusätzliche Grundlage für den Prüfungsstoff in den schriftlichen Tests. Kopieren Sie bitte den Source-Code auf Ihren Account auf der `lva.student.tuwien.ac.at` *in ein Unterverzeichnis* `serie04`*. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können. In den folgenden Übungsaufgaben sollen* ***statische sowie dynamische Vektoren und Matrizen*** *geübt werden.*

Aufgabe 31*. Schreiben Sie eine Funktion `mvmultiplication`, die die Matrix-Vektor-Multiplikation einer Matrix $A \in \mathbb{R}^{m\times n}$ mit einem Vektor $x \in \mathbb{R}^n$ realisiert und den Ergebnisvektor $b = Ax \in \mathbb{R}^m$ mit $b_j = \sum_{k=1}^{n} A_{jk}x_k$ zurückgibt. Die Matrix A soll spaltenweise in Form eines Vektors gespeichert werden. Das heißt man speichert $A \in \mathbb{R}^{m\times n}$ in Form eines Vektors $a \in \mathbb{R}^{mn}$, wobei für Indizes $j = 1, \dots, m$ und $k = 1, \dots, n$ der Wert von A_{jk} an der Stelle `a[(j - 1) + (k - 1) * m]` gespeichert wird (mit Rücksicht auf C-Indizierung $\ell = 0, \dots, mn - 1$). Die Dimensionen $m, n \in \mathbb{N}$ können Konstanten im Hauptprogramm sein, müssen aber als Parameter an die Funktion übergeben werden. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem A und x eingelesen werden und $b = Ax$ ausgegeben wird. Speichern Sie den Source-Code unter `mvmultiplication.c` in das Verzeichnis `serie04`.

Aufgabe 32*. Man schreibe eine Funktion `transpose`, die zu einer spaltenweise gespeicherten Matrix $A \in \mathbb{R}^{n\times n}$ die transponierte Matrix $A^T \in \mathbb{R}^{n\times n}$ berechnet. Dabei sind die Einträge von A^T gerade durch $(A^T)_{jk} = A_{kj}$ definiert. Die Einträge der Matrix A sollen mit den Einträgen der Matrix A^T überschrieben werden. Die Dimension $n \in \mathbb{N}$ darf eine Konstante im Hauptprogramm sein. Die Funktion `transpose` ist aber für beliebige Dimension zu programmieren. Im Hauptprogramm soll A eingelesen und A sowie A^T ausgegeben werden. Speichern Sie den Source-Code unter `transpose.c` in das Verzeichnis `serie04`.

Aufgabe 33*. Die Frobeniusnorm einer Matrix $A \in \mathbb{R}^{m\times n}$ ist durch

$$\|A\|_F := \Big(\sum_{j=1}^{m}\sum_{k=1}^{n} A_{jk}^2\Big)^{1/2}$$

definiert. Schreiben Sie eine Funktion `frobeniusnorm`, die für gegebene Matrix A und gegebene Dimensionen $m, n \in \mathbb{N}$ die Frobeniusnorm berechnet. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Zeilen- und Spaltendimensionen $m, n \in \mathbb{N}$ und A eingelesen werden und $\|A\|_F$ ausgegeben wird. Die Matrix A soll dabei als dynamische Matrix vom Typ `double**` realisiert werden. Speichern Sie den Source-Code unter `frobeniusnorm.c` in das Verzeichnis `serie04`.

Aufgabe 34*. *Bubble-Sort* ist ein ineffizienter, aber einfacher 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 $j = 1, \dots, n - 1$. Der nächste Durchlauf muss also nur noch bis zur vorletzten Stelle gehen, usw. 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 und die Länge $n \in \mathbb{N}$ einliest und x 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? Speichern Sie den Source-Code unter `bubblesort.c` in das Verzeichnis `serie04`.

Aufgabe 35. 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 36. 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. Schreiben Sie ein aufrufendes Hauptprogramm, in dem $n \in \mathbb{N}$ und $x \in \mathbb{R}^n$ eingelesen werden und das Ergebnis der Funktion `unique` ausgegeben wird.

Aufgabe 37. Für $p \in [1, \infty)$ ist die ℓ_p-Norm auf $\mathbb{R}^n$ definiert durch

$$\|x\|_p := \Big(\sum_{j=1}^{n} |x_j|^p \Big)^{1/p}.$$

Schreiben Sie eine Funktion `pnorm`, die einen Vektor $x \in \mathbb{R}^n$, dessen Länge n sowie $p \in [1, \infty)$ übernimmt und $\|x\|_p$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Länge $n \in \mathbb{N}$, der Vektor x und p eingelesen werden und $\|x\|_p$ ausgegeben wird.

Aufgabe 38. 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).
- Fahre stets mit der nächsthöheren nicht gestrichenen Zahl fort, d.h. streiche deren Vielfache.

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 39. Gegeben seien Polynome $p(x) = \sum_{j=0}^{m} a_j x^j$ und $q(x) = \sum_{j=0}^{n} b_j x^j$ in Form ihrer Koeffizientenvektoren $a \in \mathbb{R}^{m+1}$ und $b \in \mathbb{R}^{n+1}$. Dann ist die Summe $r = p + q$ ein Polynom vom Grad $\max\{m, n\}$. Schreiben Sie eine Funktion, die für gegebene Koeffizientenvektoren a und b den Koeffizientenvektor c von r berechnet und zurückgibt. Im aufrufenden Hauptprogramm sollen $m, n \in \mathbb{N}_0$ sowie die dynamischen Vektoren $a \in \mathbb{R}^m$ und $b \in \mathbb{R}^n$ eingelesen werden und c ausgegeben werden.

Aufgabe 40. Man schreibe eine Funktion `matrixvectorU`, die das Matrix-Vektor-Produkt $y = Ux$ mit einer oberen Dreiecksmatrix $U \in \mathbb{R}^{n \times n}$ mittels geeigneter Schleifen berechnet. Dabei bezeichnet man eine Matrix

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1n} \\ & u_{22} & u_{23} & \ldots & u_{2n} \\ & & \ddots & & \vdots \\ \mathbf{0} & & & & u_{nn} \end{pmatrix}$$

als obere Dreiecksmatrix. Mathematisch formuliert gilt also $U_{jk} = 0$ für $j > k$. Bei der Berechnung 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 die Koeffizienten des Vektors x eingelesen und Ux ausgegeben werden.