Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2009/10
05. November 2009

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Die Aufgaben mit Stern (*) sind bis zur Übung in der kommenden Woche vorzubereiten. Kopieren Sie die Source-Codes vor der Übung auf Ihren Account auf der `lva.student.tuwien.ac.at` *und überprüfen Sie, ob diese mit dem* `gcc` *compiliert werden können. In den folgenden Übungsaufgaben sollen* ***dynamische Arrays*** *und* ***Zählschleifen*** *geübt werden.*

Aufgabe 4.1*. Die Gleitpunkt-Arithmetik ist nicht assoziativ. Das numerische Ergebnis der Summe $\sum_{j=0}^{n} x^j/j!$ hängt daher von der Summationsreihenfolge ab. Schreiben Sie eine Funktionen `forward` und `backward`, die diese Summe auf zwei Weisen berechnen: `forward` entspricht Vorwärtssummation $j = 0, \dots, n$, `backward` entspricht Rückwärtssummation $j = n, \dots, 0$. Welche Variante wird besser sein – zumindest für positives x und großes n? Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Schranke $n \in \mathbb{N}$ eingelesen und die Ergebnisse von `forward` bzw. `backward` ausgegeben werden. Speichern Sie den Source-Code im Verzeichnis `seie04` unter dem Namen `forwardbackward.c`.

Aufgabe 4.2*. 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 im Verzeichnis `serie04` unter dem Namen `frobeniusnorm.c`.

Aufgabe 4.3*. *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. Schreiben Sie eine Funktion `bubblesort`, die einen gegebenen Vektor $x \in \mathbb{R}^n$ mittels Bubble-Sort aufsteigend sortiert, d.h. $x_1 \leq x_2 \leq \cdots \leq x_n$. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor x und die Länge $n \in \mathbb{N}$ einliest und x in sortierter Reihenfolge ausgibt. — Machen Sie sich Bubble-Sort zunächst am Beispiel $x = (1, 3, 2, 5, 4)$ klar. Speichern Sie den Source-Code im Verzeichnis `serie04` unter dem Namen `bubblesort.c`.

Aufgabe 4.4*. 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. Speichern Sie den Source-Code im Verzeichnis `serie04` unter dem Namen `unique.c`.

Aufgabe 4.5. 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 `addpolynomials`, die für gegebene Koeffizientenvektoren a und b den Koeffizientenvektor c von r anlegt, berechnet und zurückgibt. Im aufrufenden Hauptprogramm sollen $m, n \in \mathbb{N}$ sowie die dynamischen Vektoren $a \in \mathbb{R}^m$ und $b \in \mathbb{R}^n$ eingelesen werden und c ausgegeben werden.

Aufgabe 4.6. Schreiben Sie eine Funktion `transpose`, die zu einer dynamisch gespeicherten Matrix $A \in \mathbb{R}^{m \times n}$ (vom Typ `double**`) die transponierte Matrix $A^T \in \mathbb{R}^{n \times m}$ berechnet. Dabei sind die Einträge von A^T gerade durch $(A^T)_{jk} = A_{kj}$ definiert. Im Hauptprogramm sollen die Dimensionen $m, n \in \mathbb{N}$ sowie die Einträge der Matrix A eingelesen und A^T ausgegeben werden.

Aufgabe 4.7. Um die Summe $\sum_{j=1}^{n} (-1)^j / j$ zu berechnen, ist es numerisch günstig, zunächst die negativen und die positiven Beiträge getrennt zu summieren und erst abschließend beide Teilsummen zu addieren. Warum? Schreiben Sie eine Funktion `sum`, die dieses Vorgehen realisiert. Schreiben Sie ferner ein Hauptprogramm, das n einliest und $\sum_{j=1}^{n} (-1)^j / j$ ausgibt.

Aufgabe 4.8. In vielen mathematischen Bibliotheken werden Matrizen $A \in \mathbb{R}^{m \times n}$ spaltenweise gespeichert, d.h. in Form eines Vektors $a \in \mathbb{R}^{mn}$, wobei $a_{j+km} = A_{jk}$ gilt, wenn die Indizierung (wie in C üblich) bei 0 beginnt. Schreiben Sie eine Funktion `mvmultiplication`, die die Matrix-Vektor-Multiplikation einer spaltenweise gespeicherten Matrix $A \in \mathbb{R}^{m \times n}$ mit einem Vektor $x \in \mathbb{R}^n$ realisiert. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem A und x eingelesen werden und $b = Ax$ ausgegeben wird.

Aufgabe 4.9. Programmieren Sie die Funktion `transpose` aus Aufgabe 4.6 erneut. Diesmal soll die Funktion jedoch mit spaltenweise gespeicherten Matrizen (vom Typ `double*`) wie in Aufgabe 4.8 beschrieben arbeiten. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Dimensionen $m, n \in \mathbb{N}$ sowie die Einträge A_{jk} eingelesen und die transponierte Matrix A^T ausgegeben werden.

Aufgabe 4.10. 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

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

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 aufrufendes 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.