Dirk Praetorius
Christoph Überhuber

Wintersemester 2007/08
24. Oktober 2007

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

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 `serie03` *Ihres Home-Verzeichnisses. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können. In den folgenden Aufgaben sollen (einfache und geschachtelte)* ***Zählschleifen*** *geübt werden.*

Aufgabe 21*. Gegeben sei ein Polynom $p(x) = \sum_{j=0}^{n} a_j x^j$ in Form seines Koeffizientenvektors $a = (a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$. Schreiben Sie eine Funktion `evalpolynomial`, die für gegebenen Koeffizientenvektor a und Auswertungspunkt x den Funktionswert $p(x)$ berechnet. Die Funktion `pow` zur Berechnung von x^j soll *nicht* verwendet werden. Schreiben Sie eine Funktion, die möglichst nur *eine* Schleife verwendet. Der Grad $n \in \mathbb{N}$ des Polynoms soll eine Konstante im Hauptprogramm sein, die Funktion `evalpolynomial` soll aber beliebigen Grad zulassen. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Koeffizienten a_j sowie der Auswertungspunkt x eingelesen werden und $p(x)$ ausgegeben wird. Speichern Sie den Source-Code unter `evalpolynomial.c` ins Verzeichnis `serie03`.

Aufgabe 22*. 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 A eingelesen und $\|A\|_F$ ausgegeben wird. Die Dimensionen der Matrix A sollen Konstanten im Hauptprogramm sein, die Funktion `frobeniusnorm` ist aber für beliebige Dimensionen $m, n \in \mathbb{N}$ zu programmieren. Die Matrix soll spaltenweise in Form eines Vektors gespeichert werden, d.h. man speichert $A \in \mathbb{R}^{m \times n}$ in Form eines Vektors $a \in \mathbb{R}^{mn}$, wobei für Indizes $j = 1, \ldots, M$ und $k = 1, \ldots, N$ der Wert von A_{jk} an der Stelle $\mathtt{a[(j-1)+(k-1)*m]}$ gespeichert wird (mit Rücksicht auf C-Indizierung $\ell = 0, \ldots, mn-1$). Speichern Sie den Source-Code unter `frobeniusnorm.c` ins Verzeichnis `serie03`.

Aufgabe 23*. Ein Tripel $(x, y, z) \in \mathbb{N}^3$ natürlicher Zahlen heißt *pythagoräisches Zahlentripel*, falls $x^2 + y^2 = z^2$ gilt. Das wohl bekannteste Beispiel ist $(3, 4, 5)$. Offensichtlich gelten $z > \max\{x, y\}$ sowie $x \neq y$ und ohne Beschränkung der Allgemeinheit ferner $x < y$. Man schreibe eine Prozedur `pythagoras`, die zu gegebener Schranke $n \in \mathbb{N}$ alle pythagoräischen Zahlentripel mit $x < y < z \leq n$ bestimmt und ausgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, in dem die Schranke n eingelesen wird. Den Source-Code speichere man unter `pythagoras.c` ins Verzeichnis `serie03`.

Aufgabe 24. Man schreibe eine Funktion `prim`, die überprüft ob eine natürliche Zahl $n \in \mathbb{N}$ eine Primzahl ist (Rückgabewert 1) oder nicht (Rückgabewert 0). Ferner schreibe man ein aufrufendes Hauptprogramm, das den Wert n von der Tastatur einliest und am Bildschirm ausgibt, ob n eine Primzahl ist.

Aufgabe 25. Die Fibonacci-Folge ist definiert durch $x_0 := 0$, $x_1 := 1$ und $x_{n+1} := x_n + x_{n-1}$. Man schreibe eine Funktion `fibonacci`, die zu gegebenem Index n das Folgenglied x_n zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Index n einliest und x_n ausgibt.

Aufgabe 26. Man schreibe eine Funktion `max`, die von einem gegebenem Vektor $x \in \mathbb{R}^n$ das Maximum $\max_{j=1}^n x_j$ berechnet und zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Vektor x einliest und das Maximum ausgibt. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `max` ist für beliebige Länge n zu programmieren.

Aufgabe 27. 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 x und p eingelesen werden und $\|x\|_p$ ausgegeben wird. Die Dimension n soll eine Konstante im Hauptprogramm sein.

Aufgabe 28. Eine Matrix $A \in \mathbb{R}^{n \times n}$ ist symmetrisch, falls $A_{jk} = A_{kj}$ für alle $j, k = 1, \ldots, n$ gilt. Schreiben Sie eine Funktion `issymmetric`, die eine Matrix A auf Symmetrie überprüft (Rückgabewert `1` bei Symmetrie und `0` bei Nicht-Symmetrie). Schreiben Sie ein aufrufendes Hauptprogramm, in dem A eingelesen wird und ausgegeben wird, ob A symmetrisch ist oder nicht. Speichern Sie die Matrix spaltenweise, vgl. Aufgabe 22. Die Dimension $n \in \mathbb{N}$ der Matrix soll eine Konstante im Hauptprogramm, aber ein Parameter der Funktion `issymmetric` sein.

Aufgabe 29. Für eine Matrix $A \in \mathbb{R}^{m \times n}$ ist die Zeilensummennorm durch

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

gegeben. Schreiben Sie eine Funktion `zeilensummennorm`, die die Zeilensummennorm einer Matrix A berechnet. Speichern Sie die Matrix spaltenweise, vgl. Aufgabe 22. Schreiben Sie ein aufrufendes Hauptprogramm, in dem A eingelesen und $\|A\|$ ausgegeben wird. Die Dimensionen $m, n \in \mathbb{N}$ der Matrix sollen Konstanten im Hauptprogramm, aber ein Parameter der Funktion `zeilensummenorm` sein.

Aufgabe 30. *Bubble-Sort* ist ein ineffizienter, aber kurzer Sortier-Algorithmus: Man vergleicht aufsteigend jedes Element x_j eines Arrays $x \in \mathbb{R}^n$ 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 `x = bubblesort(x)`, die einen gegebenen Vektor $x \in \mathbb{R}^n$ mittels Bubble-Sort aufsteigend sortiert, d.h. $x_1 \leq x_2 \leq \cdots \leq x_n$, und zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Vektor x einliest und in sortierter Reihenfolge ausgibt. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `bubblesort` ist für beliebige Länge n zu programmieren. Man mache sich Bubble-Sort zunächst am Beispiel $x = (1, 3, 2, 5, 4)$ klar.