Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Sommersemester 2009
01. April 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 und werden dort abgeprüft. Kopieren Sie bitte die Source-Codes auf Ihren Account auf der `lva.student.tuwien.ac.at` *in ein Unterverzeichnis* `serie04`*. Überprüfen Sie vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können. In den folgenden Übungsaufgaben sollen* ***Bedingungsschleifen*** *sowie* ***dynamische Vektoren*** *und* ***Matrizen*** *geübt werden.*

Aufgabe 31*. 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. Speichern Sie den Source-Code unter `addpolynomials.c` in das Verzeichnis `serie04`.

Aufgabe 32*. Die Cosinus-Funktion hat die Darstellung $\cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$. Wir betrachten die Partialsummen

$$C_n(x) = \sum_{k=0}^{n} (-1)^k \frac{x^{2k}}{(2k)!}.$$

Man schreibe eine Funktion `cos_`, die für gegebene $x \in \mathbb{R}$ und $\tau > 0$ den Wert $C_n(x)$ zurückliefert, sobald

$$|C_n(x) - C_{n-1}(x)|/|C_n(x)| \leq \tau \quad \text{oder} \quad |C_n(x)| \leq \tau$$

gilt. Man schreibe ferner ein aufrufendes Hauptprogramm, in dem $x \in \mathbb{R}$ und $\tau > 0$ eingelesen werden. Neben dem berechneten Wert $C_n(x)$ sollen auch der korrekte Wert $\cos(x)$ und der absolute Fehler $|C_n(x) - \cos(x)|$ ausgegeben werden sowie der relative Fehler $|C_n(x) - \cos(x)|/|\cos(x)|$ im Fall $\cos(x) \neq 0$. Man schreibe die Funktion möglichst so, dass diese mit einer Schleife auskommt und dass x^{2k} und $(2k)!$ möglichst kostensparend realisiert werden. Man vermeide also insbesondere (vor- oder selbst implementierte) Funktionen zur Berechnung der Potenz oder der Faktoriellen. Speichern Sie den Source-Code unter `cos.c` ins Verzeichnis `serie04`.

Aufgabe 33*. Man schreibe 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. Speichern Sie den Source-Code unter `transpose.c` in das Verzeichnis `serie04`.

Aufgabe 34*. 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 35. Man schreibe eine Funktion `wurzelschranke`, die zu einer gegebenen Zahl $x \geq 0$ die natürliche Zahl $k \in \mathbb{N}_0$ mit $k \leq \sqrt{x} < k + 1$ zurückgibt. Dabei dürfen weder die Wurzel-Funktion `sqrt`, noch Rundungsoperationen (z.B. `floor` oder `ceil` etc.) verwendet werden.

Aufgabe 36. 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 37. Programmieren Sie die Funktion `transpose` aus Aufgabe 33 erneut. Diesmal soll die Funktion jedoch mit spaltenweise gespeicherten Matrizen (vom Typ `double*`) wie in Aufgabe 36 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 38. Gegeben sei eine stetige Funktion $f : [a, b] \to \mathbb{R}$. Es gelte

$$f(a) \cdot f(b) \leq 0.$$

Dann hat f eine Nullstelle z_0, die im Folgenden mittels Bisektion (= Intervallhalbierung) approximiert werden soll: Der Bisektionsalgorithmus arbeitet wie folgt: In jedem Bisektionsschritt definiert man $c := (a + b)/2$ als Intervallmittelpunkt. Aufgrund der Voraussetzung gilt

$$f(a) \cdot f(c) \leq 0 \quad \text{oder} \quad f(c) \cdot f(b) \leq 0.$$

Im Fall $f(a) \cdot f(c) \leq 0$ liegt eine Nullstelle im Intervall $[a, c]$, und man ersetzt daher b durch c. Im Fall $f(c) \cdot f(b) \leq 0$ liegt eine Nullstelle im Intervall $[c, b]$, und man ersetzt daher a durch c. Schreiben Sie eine Funktion `bisection`, die als Parameter a, b und eine Toleranz $\tau > 0$ übernimmt und den Bisektionsschritt wiederholt, bis man vom Startintervall $[a, b]$ zu einem Intervall $[a, b]$ mit Länge $|b - a| \leq \tau$ übergegangen ist. In diesem Fall gebe man a zurück. Es gilt dann $|a - z_0| \leq |a - b| \leq \tau$, d.h. a ist eine Approximation einer Nullstelle z_0 bis auf eine Genauigkeit von τ. Als Testfunktion verwende man $f(x) = x^2 + \exp(x) - 2$ auf $[0, 2]$, die man als eigene Funktion realisiere. Man schreibe ferner ein Hauptprogramm, das $b, \tau > 0$ einliest und die Approximation von z_0 ausgibt.

Aufgabe 39. Die Quotientenfolge $(a_{n+1}/a_n)_{n \in \mathbb{N}}$ zur Fibonacci-Folge $(a_n)_{n \in \mathbb{N}}$,

$$a_0 := 1, \quad a_1 := 1, \quad a_n := a_{n-1} + a_{n-2} \quad \text{für } n \geq 2,$$

konvergiert gegen den goldenen Schnitt $(1 + \sqrt{5})/2$. Insbesondere konvergiert die Differenz

$$b_n := \frac{a_{n+1}}{a_n} - \frac{a_n}{a_{n-1}}$$

gegen Null. Man schreibe eine Funktion `cauchy`, die zu gegebenem $k \in \mathbb{N}$ die kleinste Zahl $n \in \mathbb{N}$ mit $|b_n| \leq 1/k$ zurückgibt.

Aufgabe 40. Man schreibe eine Funktion `matrixvectorU`, die das Matrix-Vektor-Produkt $y = Ux$ mit einer oberen Dreiecksmatrix U, d.h. $U \in \mathbb{R}^{n \times n}$ mit $U_{jk} = 0$ für $j > k$, mittels geeigneter Schleifen berechnet. 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.