Dirk Praetorius
Christoph Überhuber

Wintersemester 2006/07
17. November 2006

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 im wesentlichen* ***Strukturen*** *und* ***Umgang mit dynamischem Speicher*** *geübt werden. Außer Zählschleifen, Verzweigungen und elementarer Arithmetik sind keine weiteren Elemente von C nötig.*

Aufgabe 39*. Man schreibe eine Struktur `polynomial` zur Speicherung von Polynomen, die bezüglich der Monombasis dargestellt sind, d.h. $p(x) = \sum_{j=0}^{n} a_j x^j$. Es ist also der Grad $n \in \mathbb{N}_0$ sowie der Koeffizientenvektor $a_0, \ldots, a_n \in \mathbb{R}$ zu speichern. Schreiben Sie alle nötigen Funktionen, um mit dieser Struktur arbeiten zu können (`newPoly`, `delPoly`, `getPolyDegree`, `getPolyCoefficient`, `setPolyCoefficient`). Der Source-Code der Aufgaben 39–42 soll als Bibliothek `polynomial.c` (+ Header-File `polynomial.h`) im Verzeichnis `serie05` gespeichert werden.

Aufgabe 40*. Die Summe $r = p+q$ zweier Polynome p, q ist wieder ein Polynom. Man schreibe eine Funktion `addPolynomials`, die die Summe r berechnet. Zur Speicherung verwende man die Struktur aus Aufgabe 39. Zum Test schreibe man eine Funktion, die zwei Polynome einliest und deren Summe ausgibt.

Aufgabe 41*. Das Produkt $r = pq$ zweier Polynome p, q ist wieder ein Polynom. Man schreibe eine Funktion `multiplyPolynomials`, die das Produkt r berechnet. Zur Speicherung verwende man die Struktur aus Aufgabe 39. Zum Test schreibe man eine Funktion, die zwei Polynome einliest und deren Produkt ausgibt.

Aufgabe 42*. Die k-te Ableitung $p^{(k)}$ eines Polynoms p ist wieder ein Polynom. Man schreibe eine Funktion `differentiatePolynomial`, die zu gegebenem p und $k \in \mathbb{N}$ die Ableitung $p^{(k)}$ berechnet. Zur Speicherung verwende man die Struktur aus Aufgabe 39. Zum Test schreibe man eine Funktion, die p und k einliest und $p^{(k)}$ ausgibt.

Aufgabe 43. Man schreibe eine Funktion `evalPolynomial`, die den Funktionswert $p(x)$ zurückgibt. Naive Realisierung führt auf die Schachtelung zweier Schleifen und damit $\mathcal{O}(n^2)$ viele Rechenoperationen. Man überlege sich einen weniger naiven Algorithmus, der die Auswertung mittels *einer* Schleife durchführt, d.h. in $\mathcal{O}(n)$ arithmetischen Operationen. Dazu hebe man x in der Darstellung $p(x) = \sum_{k=0}^{n} a_k x^k$ geeignet heraus (sog. *Horner-Schema*).

Aufgabe 44. Man schreibe eine Struktur `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 `newVector`, `delVector`, `getVectorLength`, `getVectorEntry`, `setVectorEntry`.

Aufgabe 45. Man schreibe eine Struktur `matrix` zur Speicherung von quadratischen $n \times n$ `double` Matrizen, in der neben vollbesetzten Matrizen (Typ `0`) auch untere (Typ `'L'`) und obere (Typ `'U'`) Dreiecksmatrizen gespeichert werden können. Dabei werde die vollbesetzte Matrix im Fortran-Format spaltenweise als dynamischer Vektor der Länge $n \cdot n$ gespeichert. Dreiecksmatrizen sollen in einem Vektor der Länge $\sum_{j=1}^{n} j = n \cdot (n+1)/2$ gespeichert werden. Man schreibe die Funktionen, um mit dieser Struktur arbeiten zu können (`newMatrix`, `delMatrix`, `getMatrixDimension`, `getMatrixType`, `getMatrixEntry`, `setMatrixEntry`).

Aufgabe 46. Man schreibe eine Funktion `solve`, die die Lösung x des linearen Gleichungssystem $Ax = b$ berechnet. Für die Vektoren und die Matrix verwende man die Formate aus Aufgabe 44–45. Dabei soll die Struktur des Gleichungssystems (Dreieckssystem!) ausgenutzt werden. Ist A keine Dreiecksmatrix, berechne man die LU-Zerlegung und löse die Faktorisierung $LUx = b$ in zwei Schritten: (1) $Ly = b$, (2) $Ux = y$.

Aufgabe 47. Die transponierte Matrix $A^T \in \mathbb{R}^{n \times m}$ einer Matrix $A \in \mathbb{R}^{m \times n}$ ist durch $(A^T)_{jk} = A_{jk}$ definiert. Man schreibe eine Funktion `transpose`, die zu gegebenem A die transponierte A^T zurückgibt. Die Matrizen A, A^T sind dabei in geeigneten Strukturen zu realisieren.

Aufgabe 48. Die Zeilensummennorm einer Matrix $A \in \mathbb{R}^{m \times n}$ ist durch

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

definiert. Man schreibe eine Funktion `norm`, die die Zeilensummennorm einer Matrix A berechnet. Die Matrix A werde dabei in einer geeigneten Struktur realisiert.