Dirk Praetorius
Christoph Überhuber

Wintersemester 2007/08
28. November 2007

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

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 `serie08` *Ihres Home-Verzeichnisses. In den folgenden Aufgaben sollen im wesentlichen* ***Strukturen*** *geübt werden.*

Aufgabe 71*. Schreiben Sie einen Strukturdatentyp `cdouble`, in dem Realteil und Imaginärteil einer Zahl $a + bi \in \mathbb{C}$ jeweils als `double` gespeichert werden. Schreiben Sie Funktionen `newCDouble`, `delCDouble` sowie die vier Zugriffsfunktionen `setCDoubleReal`, `getCDoubleReal`, `setCDoubleImag` sowie `getCDoubleImag`. Speichern Sie den Source-Code, aufgeteilt in Header-Datei `cdouble.h` und `cdouble.c`, ins Verzeichnis `serie08`.

Aufgabe 72*. Schreiben Sie Funktionen, die die Addition, die Subtraktion, die Multiplikation und die Division für komplexe Zahlen $a + bi \in \mathbb{C}$ realisieren. Verwenden Sie zur Speicherung die Struktur aus Aufgabe 71, und benutzen Sie beim Strukturzugriff nur die entsprechenden Zugriffsfunktionen. Schreiben Sie ein aufrufendes Hauptprogramm, in dem zwei komplexe Zahlen $w, z \in \mathbb{C}$ eingelesen werden und $w + z$, $w - z$, $w \cdot z$ sowie w/z ausgegeben werden. Binden Sie die Code aus Aufgabe 71 mittels `#include ''cdouble.c''` ein. Speichern Sie den Source-Code unter `carithmetik.c` ins Verzeichnis `serie08`.

Aufgabe 73*. 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`). Speichern Sie den Source-Code, aufgeteilt in Header-File `polynomial.h` und `polynomial.c`, ins Verzeichnis `serie08`.

Aufgabe 74*. Das Produkt $r = pq$ zweier Polynome p, q ist wieder ein Polynom. Man schreibe eine Funktion `multiplyPolynomials`, die das Produkt r berechnet. Das Ergebnispolynom r soll minimalen Grad $n \in \mathbb{N}$ haben, d.h. es gilt $a_n \neq 0$ für den höchsten Koeffizienten von r. Zur Speicherung verwende man die Struktur aus Aufgabe 73. Man schreibe ein aufrufendes Hauptprogramm, das zwei Polynome einliest und deren Produkt ausgibt. Binden Sie den Code aus Aufgabe 73 mittels `#include` ein. Speichern Sie die Source-Code unter `multiplyPolynomial.c` ins Verzeichnis `serie08`.

Aufgabe 75. Die Summe $r = p+q$ zweier Polynome ist wieder ein Polynom. Man schreibe eine Funktion `addPolynomials`, die die Summe r berechnet. Das Ergebnispolynom r soll minimalen

Grad $n \in \mathbb{N}$ haben, d.h. es gilt $a_n \neq 0$ für den höchsten Koeffizienten von r. Zur Speicherung verwende man die Struktur aus Aufgabe 73. Schreibe Sie ein aufrufendes Hauptprogramm, das zwei Polynome einliest und deren Summe ausgibt.

Aufgabe 76. 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. Das Ergebnispolynom $p^{(k)}$ soll minimalen Grad $n \in \mathbb{N}$ haben, d.h. es gilt $a_n \neq 0$ für den höchsten Koeffizienten von $p^{(k)}$. Zur Speicherung verwende man die Struktur aus Aufgabe 73. Zum Test schreibe man eine Funktion, die p und k einliest und $p^{(k)}$ ausgibt.

Aufgabe 77. 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 78. 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 bezeichnet man Matrizen

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ & u_{22} & u_{23} & \dots & u_{2n} \\ & & u_{33} & \dots & u_{3n} \\ & & & \ddots & \vdots \\ \mathbf{0} & & & & u_{nn} \end{pmatrix} \qquad L = \begin{pmatrix} \ell_{11} & & & & \mathbf{0} \\ \ell_{21} & \ell_{22} & & & \\ \ell_{31} & \ell_{32} & \ell_{33} & & \\ \vdots & \dots & \dots & \ddots & \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & \ell_{nn} \end{pmatrix}$$

als obere bzw. untere Dreiecksmatrix. Mathematisch formuliert, gilt also $u_{jk} = 0$ für $j > k$ bzw. $\ell_{jk} = 0$ für $j < k$. Eine vollbesetzte Matrix werde 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 79. Man schreibe eine Funktion `matrixvektor` zur Berechnung des Matrix-Vektor-Produkts $Ax \in \mathbb{R}^n$, wobei die Matrix $A \in \mathbb{R}^{n \times n}$ und der Vektor $x \in \mathbb{R}^n$ in den Datenstrukturen aus Aufgabe 77–78 gespeichert seien. Schreiben Sie die Funktion möglichst effizient, d.h. eventuelle Struktur (Dreiecksmatrix!) von A soll ausgenutzt werden.

Aufgabe 80. Man schreibe eine Funktion `loadmatrix`, die eine Matrix $A \in \mathbb{R}^{n \times n}$ aus einer Datei einliest. Zur Speicherung verwende man die Struktur aus Aufgabe 78. Die Funktion soll während des Lesens selbständig erkennen, ob die Matrix obere oder untere Dreiecksstruktur hat. Nach dem Einlesen soll die Matrix dann gegebenenfalls komprimiert als Dreiecksmatrix (um-) gespeichert werden.