Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2009/10
18. November 2009

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 6

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 erneut* ***Strukturen*** *und* ***Zählschleifen*** *geübt werden.*

Aufgabe 6.1*. Man schreibe eine Struktur `CPoly` zur Speicherung von Polynomen mit komplexwertigen Koeffizienten, die bezüglich der Monombasis dargestellt sind, d.h. $p(x) = \sum_{j=0}^{n} a_j x^j$. Es sind also der Grad $n \in \mathbb{N}_0$ sowie der Koeffizientenvektor $(a_0, \ldots, a_n) \in \mathbb{C}^{n+1}$ zu speichern. Verwenden Sie für die Darstellung der komplexwertigen Koeffizienten den Strukturdatentyp aus Aufgabe 5.1. Schreiben Sie die ferner die nötigen Zugriffsfunktionen `newCPoly`, `delCPoly`, `getCPolyDegree`, `getCPolyCoefficient` und `setCPolyCoefficient`. Speichern Sie den Source-Code unter `cpoly.c` in das Verzeichnis `serie06`.

Aufgabe 6.2*. Schreiben Sie eine Funktion `addCpolynomials`, die die Summe $r = p + q$ zweier komplexer Polynome p und q (auch unterschiedlichen Grades) berechnet und zurückgibt. Verwenden Sie zur Speicherung die Struktur aus Aufgabe 6.1. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem zwei Polynome p, q eingelesen und die Summe $r = p + q$ ausgegeben werden. Speichern Sie den Source-Code unter `addcpoly.c` in das Verzeichnis `serie06`.

Aufgabe 6.3*. 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

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

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 $\sum_{j=1}^{n} j = n(n+1)/2$ gespeichert werden. Man schreibe die Funktionen, um mit dieser Struktur arbeiten zu können (`newMatrix`, `delMatrix`, `getMatrixDimension`, `getMatrixType`, `getMatrixEntry`, `setMatrixEntry`). Speichern Sie den Source-Code unter `matrix.c` in das Verzeichnis `serie06`. Dabei hängen insbesondere die Funktionen `getMatrixEntry` und `setMatrixEntry` von dem Matrixtyp (Dreiecksmatrix!) ab.

Aufgabe 6.4*. Man schreibe eine Funktion `matrixvektor` zur Berechnung des Matrix-Vektor-Produkts $Ax \in \mathbb{R}^n$, wobei die Matrix $A \in \mathbb{R}^{n \times n}$ in der Datenstruktur aus Aufgabe 6.3 gespeichert werde. Der Vektor $x \in \mathbb{R}^n$ sein in der Datenstruktur aus der Vorlesung gespeichert. Schreiben Sie die Funktion möglichst effizient, d.h. eventuelle Struktur (Dreiecksmatrix!) von A soll ausgenutzt werden. Speichern Sie den Source-Code unter `matrixvector.c` in das Verzeichnis `serie06`.

Aufgabe 6.5. Schreiben Sie eine Funktion `spaltensummennorm`, die die Spaltensummennorm

$$\|A\|_S = \max_{j=1,\dots,n} \sum_{i=1}^{n} |A_{ij}|$$

einer Matrix $A \in \mathbb{R}^{n\times n}$ zurückgibt. Die Matrix A sei dabei in der Datenstruktur aus Aufgabe 6.3 gespeichert, und etwaige Struktur (Dreiecksmatrix!) von A soll ausgenutzt werden.

Aufgabe 6.6. Nicht jede Matrix $A \in \mathbb{R}^{n\times n}$ hat eine normalisierte LU-Zerlegung $A = LU$, d.h.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \ell_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n1} & \dots & \ell_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \dots & 0 & u_{nn} \end{pmatrix}.$$

Wenn aber A eine normalisierte LU-Zerlegung besitzt, so gilt

$$u_{ik} = a_{ik} - \sum_{j=1}^{i-1} \ell_{ij} u_{jk} \quad \text{für } i = 1,\dots,n, \quad k = i,\dots,n,$$

$$\ell_{ki} = \frac{1}{u_{ii}} \Big(a_{ki} - \sum_{j=1}^{i-1} \ell_{kj} u_{ji} \Big) \quad \text{für } i = 1,\dots,n, \quad k = i+1,\dots,n,$$

$$\ell_{ii} = 1 \quad \text{für } i = 1,\dots,n,$$

wie man leicht über die Formel für die Matrix-Matrix-Multiplikation zeigen kann. Alle übrigen Einträge von $L, U \in \mathbb{R}^{n\times n}$ sind Null. Man schreibe eine Funktion `computeLU`, die die LU-Zerlegung von A berechnet und zurückgibt. Dazu überlege man, in welcher Reihenfolge man die Einträge von L und U berechnen muss, damit die angegebenen Formeln wohldefiniert sind (d.h. alles was benötigt wird, ist bereits zuvor berechnet worden). Dabei sollen alle Matrizen in der Struktur aus Aufgabe 6.3 gespeichert werden.

Aufgabe 6.7. Man schreibe eine Struktur `CVector`, zur Speicherung von Vektoren mit komplexwertigen Koeffizienten. Benutzen Sie hierzu den Strukturdatentyp `cdouble` aus Aufgabe 5.1. Ferner schreibe man die zugehörigen Funktionen `newCVector`, `delCVector`, `getCVectorLength`, `getCVectorEntry`, `setCVectorEntry`.

Aufgabe 6.8. Schreiben Sie eine Funktion `cvectorvector`, die das Skalarprodukt $x \cdot y := \sum_{j=1}^{n} x_j \overline{y_j}$ zweier komplexwertiger Vektoren $x, y \in \mathbb{C}^n$ berechnet. Schreiben Sie ferner ein aufrufendes Hauptprogramm, indem die Vektoren x, y eingelesen und der Wert $x \cdot y \in \mathbb{C}$ ausgegeben werden. Speichern Sie den Source-Code unter `cvectorvector.c` in das Verzeichnis `serie06`.

Aufgabe 6.9. Man schreibe eine Struktur `CMatrix`, zur Speicherung von $(m \times n)$-Matrizen $A \in \mathbb{C}^{m\times n}$ mit komplexwertigen Koeffizienten. Benutzen Sie hierzu den Strukturdatentyp `cdouble` aus Aufgabe 5.1. Ferner schreibe man die zugehörigen Funktionen `newCMatrix`, `delCMatrix`, `getCMatrixM`, `getCMatrixN`, `getCMatrixCoeff`, `setCMatrixCoeff`.

Aufgabe 6.10. Schreiben Sie eine Funktion `cmatrixvector`, die die Matrix-Vektor-Multiplication $Ax \in \mathbb{C}^m$ mit einer Matrix $A \in \mathbb{C}^{m\times n}$ und einem Vektor $x \in \mathbb{C}^n$ realisiert. Verwenden Sie für die Koeffizienten die komplexe Arithmetik aus Aufgabe 5.2.