Dirk Praetorius,
Marcus Page

Sommersemester 2012
16.04.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 6

Aufgabe 6.1*. (dynamische Matrizen, effiziente Speicherung) Eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ hat höchstens $\frac{n(n+1)}{2} = \sum_{j=1}^{n} j$ nicht-triviale Einträge. Schreiben Sie eine Struktur `matrixU`, in der neben der Dimension $n \in \mathbb{N}$ die Koeffizienten U_{ij} in einem dynamischen Vektor der Länge $\frac{n(n+1)}{2}$ gespeichert werden. Schreiben Sie die entsprechenden Zugriffsfunktionen (`newMatrixU`, `delMatrixU`, `getMatrixUN`, `getMatrixUij`, `setMatrixUij`), und überlegen Sie sich zuvor, an welcher Stelle u_ℓ im dynamischen Vektor ein Eintrag U_{ij} gespeichert werden soll.

Aufgabe 6.2*. (dynamische Matrizen, effizientes Multiplizieren) Schreiben Sie eine Funktion `matrixvectorU`, die das Matrix-Vektor-Produkt $y = Ux$ mit einer oberen Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ mittels geeigneter Schleifen berechnet. Dabei bezeichnet man eine Matrix

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ & u_{22} & u_{23} & \dots & u_{2n} \\ & & \ddots & & \vdots \\ \mathbf{0} & & & & u_{nn} \end{pmatrix}$$

als obere Dreiecksmatrix. Mathematisch formuliert gilt also $U_{jk} = 0$ für $j > k$. Bei der Berechnung soll auf die offensichtlichen Nulleinträge von U nicht zugegriffen werden, um den Rechenaufwand gering zu halten. Schreiben Sie ferner ein aufrufendes 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. Speichern Sie den Source-Code unter `matrixvectorU.c` in das Verzeichnis `serie06`.

Aufgabe 6.3*. (dynamische Matrizen, effizientes Lösen) Es sei $U \in \mathbb{R}^{n\times n}$ eine obere Dreiecksmatrix mit $U_{jj} \neq 0$ für alle $j = 1, \dots, n$. Zu gegebener rechter Seite $b \in \mathbb{R}^n$ existiert dann ein eindeutiger Vektor $x \in \mathbb{R}^n$ mit $Ux = b$. Leiten Sie, ausgehend von der Formel für die Matrix-Vektor-Multiplikation, eine Formel für x her. Schreiben Sie sich dazu das Matrix-Vektor-Produkt $b = Ux$ komponentenweise für b_j mit $j = 1, \dots, n$ als Summe, und überlegen Sie, wie die spezielle Gestalt von U die Laufindizes der Summe vereinfacht. Schreiben Sie eine Funktion `solveU`, die für gegebenes U und b den Vektor x berechnet und zurückgibt. Die Matrix U soll dabei in der Struktur aus Aufgabe 6.1 gespeichert werden, die Vektoren $b, x \in \mathbb{R}^n$ in der Struktur aus der Vorlesung.

Aufgabe 6.4*. (Aufwand eines Algorithmus) Was versteht man und 'Aufwand' eines Algorithmus. Erklären Sie diesen Begriff formal. Welchen Aufwand hat das Lösen eines Systems mit oberer Dreiecksmatrix aus Aufgabe 6.3?

Aufgabe 6.5. (Strukturen für Polynome) Schreiben Sie einen Strukturdatentyp `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 $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ zu speichern. Schreiben Sie alle nötigen Funktionen, um mit dieser Struktur arbeiten zu können (`newPoly`, `delPoly`, `getPolyDegree`, `getPolyCoefficient`, `setPolyCoefficient`). Wo ist der Vorteil der Speicherung als Struktur gegenüber dem Vorgehen vom letzten Übungsblatt? Speichern Sie den Source-Code unter `polynomial.c` in das Verzeichnis `serie06`.

Aufgabe 6.6. (Stammfunktion von Polynomen) Schreiben Sie eine Funktion `antiderivative`, die zu einem gegebenen Polynom p die Stammfunktion q berechnet und zurückgibt. Sowohl p, als auch q sollen hierbei in der Struktur aus der letzten Aufgabe gespeichert werden. Schreiben Sie zudem ein aufrufendes Hauptprogramm in dem der Grad $n \in \mathbb{N}$ von p, sowie dessen Koeffizienten eingelesen, und die Stammfunktion ausgegeben wird. Speichern Sie den Source-Code unter `antiderivative.c` in das Verzeichnis `serie06`.

Aufgabe 6.7. (komplexwertige Polynome) Schreiben Sie 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. Schreiben Sie sich für die Verwendung der komplexwertigen Koeffizienten einen Strukturdatentyp zur Speicherung von komplexen Zahlen. 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 6.8. (komplexwertige Polynome) 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 6.7. 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 6.9. (dynamische Matrizen, effizientes Multiplizieren) Das Produkt $U = AB$ zweier oberer Dreiecksmatrizen $A, B \in \mathbb{R}^{n \times n}$ ist wieder eine obere Dreiecksmatrix. Man beweise diese Aussage zunächst mathematisch, indem man sich die Formel für das Matrix-Matrix-Produkt hinschreibe und mittels der Voraussetzung an A und B die Indizes vereinfache. Danach schreibe man eine Funktion `matrixmatrixU`, die die Produktmatrix berechnet und zurückgibt. Dabei sollen natürlich nur die nicht-trivialen Einträge von U, d.h. U_{jk} für $j \leq k$, berechnet werden. Ferner soll auf die trivialen Einträge von A und B nicht zugegriffen werden, d.h. man verwende die anfangs hergeleitete Formel. Alle Matrizen sollen in der Struktur aus Aufgabe 6.1 gespeichert werden.

Aufgabe 6.10. (dynamische Matrizen, Matrixnorm) Schreiben Sie eine Funktion `spaltensummennorm`, die die Spaltensummennorm

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

einer (oberen Dreiecks-) Matrix $A \in \mathbb{R}^{n \times n}$ zurückgibt. Die Matrix A sei dabei in der Datenstruktur aus Aufgabe 6.1 gespeichert, und die Struktur (Dreiecksmatrix!) von A soll ausgenutzt werden.