Dirk Praetorius,
Marcus Page,
Thomas Führer

Wintersemester 2012/13
20.11.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

Aufgabe 8.1*. (dynamische Matrizen, effiziente Speicherung) Eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$

$$L = \begin{pmatrix} \ell_{11} & & & & \mathbf{0} \\ \ell_{21} & \ell_{22} & & & \\ \ell_{31} & \ell_{32} & \ell_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \dots & \ell_{nn} \end{pmatrix}$$

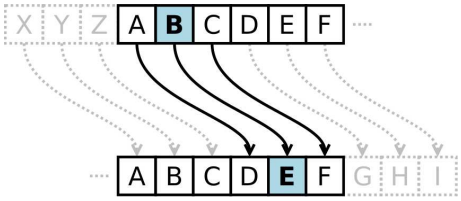
hat höchstens $\frac{n(n+1)}{2} = \sum_{j=1}^{n} j$ nicht-triviale Einträge. Erklären Sie wie man auf diese Formel kommt. Schreiben Sie eine Struktur `matrixL`, in der neben der Dimension $n \in \mathbb{N}$ die Koeffizienten L_{jk} in einem dynamischen Vektor ℓ der Länge $\frac{n(n+1)}{2}$ gespeichert werden. Überlegen Sie sich, an welcher Stelle ℓ_m im dynamischen Vektor ein Eintrag L_{jk} gespeichert werden soll. Schreiben Sie die entsprechenden Funktionen zum Allokieren und Freigeben des Speichers einer unteren Dreiecksmatrix (`newMatrixL`, `delMatrixL`) und die Funktion `getMatrixLN`, welche die Größe $n \in \mathbb{N}$ zurückgibt. Implementieren Sie ferner die Zugriffsfunktionen `getMatrixLjk` bzw. `setMatrixLjk`, um auf einzelne Koeffizienten der Matrix zuzugreifen.

Aufgabe 8.2*. (effizientes Multiplizieren) Schreiben Sie eine Funktion `matrixvectorL`, die das Matrix-Vektor-Produkt $y = Lx$ mit einer unteren Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mittels geeigneter Schleifen berechnet. Bei der Berechnung soll auf die offensichtlichen Nulleinträge von L 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 L und die Koeffizienten des Vektors x eingelesen und Lx ausgegeben werden. Speichern Sie den Source-Code unter `matrixvectorL.c` in das Verzeichnis `serie08`.

Aufgabe 8.3*. Schreiben Sie eine Funktion `matmult` welche das Produkt zweier Matrizen $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{p\times q}$ bestimmt (vgl. Aufgabe 4.4). Die Funktion soll zwei Matrizen übernehmen und das Produkt zurückgeben. Zur Speicherung der Matrizen verwende man die in der VO vorgestellte Struktur `Matrix`. Überprüfen Sie auch, ob die Dimensionen der Matrizen zusammenpassen! Die Funktion soll für die Produktmatrix $C = AB$ den nötigen speicher allokieren. Benutzen Sie die Struktur `Matrix` auch für die Produktmatrix. Verwenden Sie in Ihrer Funktion ausschließlich die `set` und `get` Funktionen der Struktur um auf Matrix-Elemente zuzugreifen. Speichern Sie den Source-Code unter `matmult.c` in das Verzeichnis `serie08`.

Aufgabe 8.4*. Die Cäsar-Chiffre ist ein primitiver Algorithmus zur Verschlüsselung von Texten. Als Schlüssel wählt man hierbei eine Zahl $z \in [1, \dots, 25]$ die angibt, um wieviel der Klartext für die Chiffrierung 'verschoben' wird. Ausgehend vom Klartext verschlüsselt man nun jeden Buchstaben einzeln, indem einfach der Buchstabe eingesetzt wird, der im Alphabet z Einträge weiter hinten steht. Für $z = 3$, wird

der Buchstabe B also beispielsweise mit E verschlüsselt. Ist man am Ende des Alphabets angekommen, wird einfach wieder von vorn begonnen, so dass Z mit C verschlüsselt wird. Insgesamt ergibt sich für $z = 3$ beispielsweise das folgende Schema:



Schreiben Sie eine Funktion `encipher` die ein Wort (nur Großbuchstaben), sowie den Schlüssel z von der Tastatur einliest und die verschlüsselte Version ausgibt. Schreiben Sie außerdem eine Funktion `decipher`, die ein verschlüsseltes Wort mittels Schlüssel z wieder dechiffriert. Einzelne Zeichen können hierbei mit Zahlen $n \in \mathbb{N}$ addiert werden. Den Integer Wert (ASCII) eines Zeichens können Sie in `printf` mittels `%i` ausgeben lassen. Achten Sie darauf, dass das große Alphabet die Werte $65 - 90$ (ASCII) hat. Speichern Sie den Source-Code unter `caesar.c` in das Verzeichnis `serie08`.
Tipp: Zur besseren Übersicht ist es sinnvoll eine eigene Funktion zur Chiffrierung eines einzelnen Buchstabens zu schreiben.

Aufgabe 8.5. Die Verschlüsselung aus Aufgabe 8.4 kann relativ leicht "geknackt" werden. Durch die Buchstabenhäufigkeit — in deutschsprachigen Texten kommt am häufigsten der Buchstabe 'e' vor, am zweithäufigsten der Buchstabe 'n' — kann man auf den Schlüssel $z \in [0, \ldots, 25]$ schließen. Die Verschlüsselung aus Aufgabe 8.4 kann verfeinert werden, indem man ein Passwort der Länge $n \in \mathbb{N}$ als Schlüssel benutzt. Dabei wird der erste Buchstabe des Passworts benutzt um den ersten Buchstaben des Textes zu verschlüsseln. Der zweite Buchstabe des Passworts dient zur Verschlüsselung des zweiten Buchstaben des Textes usw. Nach dem letzten Buchstaben des Passworts startet man wieder von vorne, d.h. der $n + 1$-te Buchstabe im Text wird mit dem ersten Zeichen im Passwort verschlüsselt. Als Verschlüsselungsmethode nimmt man dabei für jeden einzelnen Buchstaben wiederum die Cäsar-Chiffre. Wählt man etwa als Passwort `acf`, so soll der erste Buchstabe des zu verschlüsselnden Textes um den ASCII-Wert von `'a'` verschoben werden, der zweite um den ASCII-Wert von `'c'` und der dritte um den ASCII-Wert von `'f'`. Danach fängt man wieder beim ersten Buchstaben des Passworts an, d.h. der vierte Buchstabe des zu verschlüsselnden Textes wird um den ASCII-Wert von `'a'` verschoben usw. Implementieren Sie das oben beschriebene Verfahren. Speichern Sie den Source-Code unter `caesarPWD.c` in das Verzeichnis `serie08`.

Aufgabe 8.6. (dynamische Matrizen, effizientes Lösen) Es sei $L \in \mathbb{R}^{n \times n}$ eine untere Dreiecksmatrix mit $L_{jj} \neq 0$ für alle $j = 1, \ldots, n$. Zu gegebener rechter Seite $b \in \mathbb{R}^n$ existiert dann ein eindeutiger Vektor $x \in \mathbb{R}^n$ mit $Lx = b$. Leiten Sie, ausgehend von der Formel für die Matrix-Vektor-Multiplikation, eine Formel für x her. Schreiben Sie sich dazu das Matrix-Vektor-Produkt $b = Lx$ komponentenweise für b_j mit $j = 1, \ldots, n$ als Summe, und überlegen Sie, wie die spezielle Gestalt von L die Laufindizes der Summe vereinfacht. Schreiben Sie eine Funktion `solveL`, die für gegebenes L und b den Vektor x berechnet und zurückgibt. Die Matrix L soll dabei in der Struktur aus Aufgabe 8.1 gespeichert werden, die Vektoren $b, x \in \mathbb{R}^n$ in der Struktur aus der Vorlesung. Speichern Sie den Source-Code unter `solveL.c` in das Verzeichnis `serie08`.

Aufgabe 8.7. 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 8.8. Schreiben Sie eine kleine Bibliothek für die Arithmetik mit komplexen Zahlen und verwenden Sie den Datentyp aus Aufgabe 8.7. Schreiben Sie Funktionen für Addition, Subtraktion,

Multiplikation und Division von zwei komplexen Zahlen. Schreiben Sie außerdem eine Funktion `myabs` die den Betrag

$$|z| = \sqrt{a^2 + b^2}$$

einer komplexen Zahl $z = a + ib \in \mathbb{C}$ berechnet und zurückgibt.