Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Sommersemester 2009
09. April 2009

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Die Aufgaben mit Stern (*) sind bis zur Übung in der kommenden Woche vorzubereiten und werden dort abgeprüft. Kopieren Sie bitte die Source-Codes auf Ihren Account auf der `lva.student.tuwien.ac.at` *in ein Unterverzeichnis* `serie05`*. Überprüfen Sie vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können. In den folgenden Übungsaufgaben sollen* ***dynamische Arrays*** *und* ***Strukturen*** *geübt werden.*

Aufgabe 41*. Gegeben sei eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ (vgl. Aufgabe 40) mit $u_{jj} \neq 0$ für alle $j = 1, \dots, n$. Zu gegebenem $y \in \mathbb{R}^n$ existiert dann ein eindeutiges $x \in \mathbb{R}^n$ mit $Ux = y$. Man schreibe eine Funktion `solveU`, die x berechnet. Dabei 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 der rechten Seite y eingelesen und die Lösung x ausgegeben werden. Speichern Sie den Source-Code unter `solveU.c` in das Verzeichnis `serie05`.
— Um den Algorithmus herzuleiten, schreibe man das Matrix-Vektor-Produkt $y = Ux$ komponentenweise für y_j mit $j = 1, \dots, n$ als Summe (vgl. Aufgabe 31). Man überlege, wie die spezielle Gestalt von U die Laufindizes der Summe vereinfacht und löse diese Gleichung nach x_j auf.

Aufgabe 42*. Das Produkt $U = AB$ zweier oberer Dreiecksmatrizen $A, B \in R^{n\times n}$ 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. Man schreibe ferner ein Hauptprogramm, in dem die Dimension $n \in \mathbb{N}$ sowie die Einträge der Matrizen A, B eingelesen und die Matrix U ausgegeben werden. Speichern Sie den Source-Code unter `matrixmatrixU.c` in das Verzeichnis `serie05`.

Aufgabe 43*. 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 unter `cdouble.c` in das Verzeichnis `serie05`.

Aufgabe 44*. 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 43, 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 den Code aus Aufgabe 43 mittels `#include ''cdouble.c''` ein. Speichern Sie den Source-Code unter `carithmetik.c` ins Verzeichnis `serie05`.

Aufgabe 45. 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 $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$ zu speichern. Schreiben Sie alle nötigen Funktionen, um mit dieser Struktur arbeiten zu können (`newPoly`, `delPoly`, `getPolyDegree`, `getPolyCoefficient`, `setPolyCoefficient`).

Aufgabe 46. 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 45. Ferner schreibe man ein Hauptprogramm in dem p und k eingelesen und $p^{(k)}$ ausgegeben werden.

Aufgabe 47. Man implementiere eine Funktion `eratosthenes`, die das *Sieb des Eratosthenes* realisiert. Dies ist ein Algorithmus, mit dem alle Primzahlen bis zu einer bestimmten Zahl `nmax` errechnet werden können (benannt nach dem griechischen Mathematiker Eratosthenes). Der Algorithmus sieht folgendermaßen aus:

- Man legt eine Liste (Vektor) $\texttt{prim} = (2, \ldots, \texttt{nmax}) \in \mathbb{N}^{\texttt{nmax}-1}$ an.
- Man streicht aus der Liste alle Vielfachen der ersten Zahl (also der Zahl 2).
- Wähle, solange es noch höhere Zahlen gibt, die nächsthöhere nicht durchgestrichene Zahl und streiche alle ihre Vielfachen.

Der Rückgabevektor `prim` soll am Ende (gekürzt auf minimale Länge) alle Primzahlen $\leq$ `nmax` enthalten (Sie müssen zusätzlich die Länge des Rückgabevektors zurückgeben!). Realisieren Sie das Streichen geeignet, z.B. indem Sie die entsprechenden Einträge auf 0 setzen.

Aufgabe 48. Schreiben Sie eine Funktion `primfaktoren`, die für eine natürliche Zahl $n \in \mathbb{N}$ deren Primfaktoren bestimmt und als Vektor $p \in \mathbb{N}^k$ zurückgibt. Die Koeffizienten p_j des Vektors $p \in \mathbb{N}^k$ sind also Primzahlen, und es gilt $n = \prod_{j=1}^{k} p_k$. Um eine Liste aller möglichen Primfaktoren zu erhalten, verwende man das Sieb des Eratosthenes aus Aufgabe 47.

Aufgabe 49. Schreiben Sie eine Funktion `unique`, die einen Vektor $x \in \mathbb{R}^n$ aufsteigend sortiert, doppelte Einträge streicht und den Vektor in gekürzter Form zurückgibt. Die Funktion soll also beispielsweise den Vektor $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ durch den Vektor $x = (1, 3, 4, 5) \in \mathbb{R}^4$ überschreiben. Schreiben Sie ein aufrufendes Hauptprogramm, in dem n und $x \in \mathbb{R}^n$ eingelesen werden und das Ergebnis der Funktion `unique` ausgegeben wird.

Aufgabe 50. Man schreibe eine Struktur `Cpolynomial` 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 ist 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 43. Schreiben Sie neben den nötigen Funktionen, um mit dieser Struktur arbeiten zu können (`newPoly`, `delPoly`, `getPolyDegree`, `getPolyCoefficient`, `setPolyCoefficient`) auch eine Funktion `addCpolynomials`, die die Summe zweier komplexer Polynome berechnet. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem zwei Polynome p, q eingelesen und die Summe $r = p + q$ ausgegeben werden.