Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2008/09
12. November 2008

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. Die übrigen Aufgaben dienen nur Ihrer Übung und uns als zusätzliche Grundlage für den Prüfungsstoff in den schriftlichen Tests. Kopieren Sie bitte den Source-Code auf Ihren Account auf der `lva.student.tuwien.ac.at` *in ein Unterverzeichnis* `serie05`*. Überprüfen Sie bitte 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 $j = 1, \ldots, n$. 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 $j = 1, \ldots, n$ 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*. Gegeben seien Polynome $p(x) = \sum_{j=0}^{m} a_j x^j$ und $q(x) = \sum_{j=0}^{n} b_j x^j$ in Form ihrer Koeffizientenvektoren $a \in \mathbb{R}^{m+1}$ und $b \in \mathbb{R}^{n+1}$. Dann ist das Produkt $r = pq$ ein Polynom vom Grad $m+n$. Schreiben Sie eine Funktion `multiplyPolynomials`, die das Produkt r berechnet und zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem $m, n \in \mathbb{N}_0$ sowie die dynamischen Vektoren $a \in \mathbb{R}^{m+1}$ und $b \in \mathbb{R}^{n+1}$ eingelesen werden und r ausgegeben wird. Speichern Sie den Source-Code unter `multiplyPolynomials.c` in das Verzecihnis `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. Schreiben Sie eine Funktion `float2dec`, die für eine gegebene Mantissenlänge $M \in \mathbb{N}$, Ziffern $a_1, \ldots, a_M \in \{0, 1\}$ und einen Exponenten $e \in \mathbb{Z}$ den Dezimalwert $x = \left(\sum_{k=1}^{M} a_k 2^{-k}\right) 2^e$ berechnet und zurückgibt. Schreiben Sie ein aufrufendes Hauptprogramm, in dem M, a_j und e eingelesen und der Wert x ausgegeben wird. Realisieren die Potenzen 2^{-k} möglichst rechenökonomisch.

Aufgabe 46. Schreiben Sie eine Funktion `dec2float`, die für eine gegebene Dezimalzahl $x \in \mathbb{R}_{>0}$ und eine Mantissenlänge $M \in \mathbb{N}$ die Ziffern $a_1, \ldots, a_M \in \{0, 1\}$ und den Exponenten $e \in \mathbb{Z}$ der normalisierten Gleitkommadarstellung (d.h. $a_1 = 1$) berechnet und zurückgibt. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x eingelesen und die Gleitkommadarstellung von x ausgegeben wird. Um Ihre Funktion zu verifizieren, können Sie Aufgabe 45 verwenden.

Aufgabe 47. 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 48. 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 47. Ferner schreibe man ein Hauptprogramm in dem p und k eingelesen und $p^{(k)}$ ausgegeben werden.

Aufgabe 49. Man schreibe eine Funktion `savepolynomial`, dass ein Polynom in Form seines Koeffizientenvektors in eine Datei speichert. Ferner schreibe man ein aufrufendes Hauptprogramm, in dem ein Polynom sowie ein Dateiname von der Tastatur eingelesen werden und das Polynom in die angegebene Datei gespeichert wird. Als Datenformat für das Polynom verwende man die Struktur aus Aufgabe 47.

Aufgabe 50. Man schreibe eine Funktion `loadpolynomial`, dass ein Polynom aus einer Datei einliest. Zur Speicherung verwende man die Struktur aus Aufgabe 47. Die Funktion soll selbständig erkennen, ob das Polynom kleineren Grad hat, als angegeben – d.h. es soll überprüft werden, ob für die Koeffizienten $a_k \neq 0$ und $a_{k+1} = \ldots = a_n = 0$ gelten. In diesem Fall sollen nur die notwendigen Einträge des Koeffizientenvektors gespeichert werden.