Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2009/10
12. November 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. 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* ***Strukturen*** *und* ***Zählschleifen*** *geübt werden.*

Aufgabe 5.1*. 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 5.2*. 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 5.1, 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 5.1 mittels `#include "cdouble.c"` ein. Speichern Sie den Source-Code unter `carithmetik.c` ins Verzeichnis `serie05`.

Aufgabe 5.3*. 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`). Speichern Sie den Source-Code unter `polynomial.c` ins Verzeichnis `serie05`.

Aufgabe 5.4*. Die k-te Ableitung $p^{(k)}$ eines Polynoms p ist wieder ein Polynom. Man schreibe eine Funktion `diffPoly`, die zu gegebenem p und $k \in \mathbb{N}$ die Ableitung $p^{(k)}$ berechnet. Berücksichtigen Sie auch den Fall, dass k größer als der Grad von p ist. Zur Speicherung verwende man die Struktur aus Aufgabe 5.3. Binden Sie den entsprechenden Code mittels `#include "polynomial.c"` ein. Ferner schreibe man ein Hauptprogramm in dem p und k eingelesen und $p^{(k)}$ ausgegeben werden. Speichern Sie den Source-Code unter `diffpoly.c` ins Verzeichnis `serie05`.

Aufgabe 5.5. Schreiben Sie eine Funktion `evalDiffPoly`, die für ein gegebenes Polynom p, eine Ableitungsordnung $k \in \mathbb{N}_0$ und einen Punkt $x \in \mathbb{R}$ den Funktionswert $p^{(k)}(x)$ zurückgibt. Dabei soll das Polynom $p^{(k)}$ im Gegensatz zu Aufgabe 5.4 nicht explizit gebildet und gespeichert werden. Verwenden Sie für das Polynom p die Struktur aus Aufgabe 5.3. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem der Grad n, das Polynom p, die Ableitungsordnung k und der Punkt x eingelesen werden und $p^{(k)}(x)$ ausgegeben wird.

Aufgabe 5.6. Das Produkt $r = pq$ zweier Polynome $p(x) = \sum_{j=0}^{m} a_j x^j$ und $q(x) = \sum_{j=0}^{n} b_j x^j$ ist wieder in Polynom. Schreiben Sie eine Funktion `prodPoly`, die das Produktpolynom r berechnet und in der Struktur aus Aufgabe 5.3 speichert. Überlegen Sie sich zunächst, welchen Grad das Polynom r hat und wie sich die Koeffizienten berechnen lassen. Schreiben Sie ein aufrufendes Hauptprogramm, in dem p und q eingelesen und $r = pq$ ausgegeben wird.

Aufgabe 5.7. Eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ (vgl. Aufgabe 4.10) hat potentiell $\frac{n(n+1)}{2} = \sum_{j=1}^{n} j$ 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 5.8. Schreiben Sie eine Funktion `mvmU`, die die Matrix-Vektor-Multiplikation mit einer oberen Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ realisiert. Die Matrix U sei dabei in der Struktur aus Aufgabe 5.7 gespeichert, der Vektor in der Struktur aus der Vorlesung. Überflüssige Multiplikationen mit Nulleinträgen der Matrix U sollen aus Effizienzgründen vermieden werden.

Aufgabe 5.9. 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 5.7 gespeichert werden, die Vektoren $b, x \in \mathbb{R}^n$ in der Struktur aus der Vorlesung.

Aufgabe 5.10. Das Produkt $C = AB$ zweier Matrizen $A \in \mathbb{R}^{\ell\times m}$ und $B \in \mathbb{R}^{m\times n}$ ist eine Matrix $C \in \mathbb{R}^{\ell\times n}$, deren Einträge durch $C_{ik} = \sum_{j=1}^{m} A_{ij}B_{jk}$ gegeben ist. Das Produkt $U = AB$ zweier oberer Dreiecksmatrizen $A, B \in \mathbb{R}^{n\times n}$ ist wieder eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$. 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 $j \le k$, 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 5.7 gespeichert werden.