Dirk Praetorius,
Thomas Führer

Sommersemester 2014
29.04.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 7

Aufgabe 7.1. Schreiben Sie eine Funktion `doubleData`, die zu einer gegebenen Zahl $x \in \mathbb{R}$ und einer Mantissenlänge $M \in \mathbb{N}$ folgende Dinge berechnet:

- Vorzeichen $\sigma \in \{-1, +1\}$
- Ziffern $a_j \in \{0, 1\}$ für $j = 1, \dots, M$
- Exponenten $e \in \mathbb{Z}$

Sodass $x \approx (\sum_{j=1}^{M} a_j 2^{-j}) 2^e$. (Dass dies also unter den gegebenen Voraussetzungen die bestmögliche Approximation von x darstellt.) Sie können dazu ähnlich wie im Beweis der Vorlesung vorgehen. Geben Sie anschließend σ, e und den Vektor $(a_j)_{j=1}^{M}$ mittels der *call-by-reference*-Simulation durch Pointer zurück. Speichern Sie den Source-Code unter `doubleData.c` in das Verzeichnis `serie07`.

Aufgabe 7.2. Was ist ein Gleitkommazahlsystem? Aus welchen Bestandteilen setzt sich eine Gleitkommazahl zusammen? Wie bestimmt man daraus ihren Wert? Was verbirgt sich hinter den Symbolen `Inf`, `-Inf` und `NaN`? Was ist die Maschinengenauigkeit `eps`? Was ist eine normalisierte Gleitkommazahl? Was ist ein implizites erstes Bit?

Aufgabe 7.3. Schreiben Sie ein Programm, welches ein Wort einliest und überprüft ob es sich bei dem eingegeben Wort um ein Palindrom handelt. Ein Palindrom ist ein Wort, welches von vorne und hinten gelesen gleich lautet, z.B.: Anna, Otto, Reliefpfeiler. Speichern Sie den Source-Code unter `palindrom.c` in das Verzeichnis `serie07`.

Aufgabe 7.4. 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. Die Länge n der Vektoren ist dynamisch zu realisieren. Schreiben Sie ein aufrufendes Hauptprogramm, in dem n und $x \in \mathbb{R}^n$ eingelesen werden und das Ergebnis der Funktion `unique` ausgegeben wird. Speichern Sie den Source-Code unter `unique.c` in das Verzeichnis `serie07`.

Aufgabe 7.5. Schreiben Sie eine Bibliothek zur Verwaltung von *spaltenweise* gespeicherten $m \times n$-Matrizen. Implementieren Sie die folgenden Funktionen

- `double* mallocmatrix(int m, int n)`
  Allokieren von Speicher für eine spaltenweise gespeicherte $m \times n$- Matrix.
- `double* freematrix(double* matrix)`
  Freigeben des allokierten Speichers einer Matrix.
- `double* reallocmatrix(double* matrix, int m, int n, int mNew, int nNew)`
  Reallokieren und initialisieren von neuen Einträgen.

Speichern Sie die Funktionssignaturen in das Header-File `dynamicmatrix.h`. Schreiben Sie auch entsprechende Kommentare zu den Funktionen in das Header-File. In die Datei `dynamicmatrix.c` kommt dann die Implementierung der Funktionen.

Aufgabe 7.6. 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 $(a_0, \dots, a_n) \in \mathbb{R}^{n+1}$ zu speichern. Schreiben Sie alle nötigen Funktionen, um mit dieser Struktur arbeiten zu können (`newPoly`, `delPoly`, `getPolyDegree`, `getPolyCoefficient`, `setPolyCoefficient`). Schreiben Sie außerdem eine `double` Funktion `evalPoly`, die ein Polynom an einer gegebenen Stelle auswertet und den Wert zurückgibt. Speichern Sie den Source-Code unter `polynomial.c` in das Verzeichnis `serie07`.

Aufgabe 7.7. Die Summe $r = p + q$ zweier Polynome p, q ist wieder ein Polynom. Schreiben Sie eine Funktion `addPolynomials`, die die Summe r berechnet. Zur Speicherung verwende man die Struktur aus Aufgabe 7.6. Zum Test schreibe man eine Funktion, die zwei Polynome einliest und deren Summe ausgibt. Speichern Sie den Source-Code unter `addPolynomials.c` in das Verzeichnis `serie07`.

Aufgabe 7.8. Die k-te Ableitung $p^{(k)}$ eines Polynoms p ist wieder ein Polynom. Schreiben Sie 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 7.6. Zum Test schreibe man ein Hauptprogramm, das p und k einliest und $p^{(k)}$ ausgibt. Speichern Sie den Source-Code unter `differentiatePolynomial.c` in das Verzeichnis `serie07`.