Dirk Praetorius, Sommersemester 2014
Thomas Führer 29.04.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 7

Aufgabe 7.1. Write a function `doubleData` that computes for a given number $x \in \mathbb{R}$ and a mantissa with length $M \in \mathbb{N}$, the

- sign $\sigma \in \{-1, +1\}$,
- digits $a_j \in \{0, 1\}$ for $j = 1, \dots, M$,
- exponent $e \in \mathbb{Z}$,

such that $x \approx (\sum_{j=1}^M a_j 2^{-j}) 2^e$. Have a look at the proof in the lecture notes. The function should return σ, e, and the vector $(a_j)_{j=1}^M$ with *call-by-reference* (pointer).

Aufgabe 7.2. What is the system of floating-point numbers? Of which parts does a floating-point number consist? How can you determine its value thereof? What is the meaning of `Inf`, `-Inf`, and `NaN`? What is the machine accuracy `eps`? What is a normalized floating-point number? What is a first implicit bit?

Aufgabe 7.3. Write a program that reads in a word (string) und checks if this word is a *palindrome*. A palindrome is a word whose meaning is the same either in forward or backward direction, e.g., radar, level, madam.

Aufgabe 7.4. Write a function `void unique(double * x, int n)` which reads in a vector $x \in \mathbb{R}^n$, sorts this vector in ascending order, eliminates entries that appear more than once, and returns the shortened vector. For instance, the vector $x = (4, 3, 5, 1, 4, 3, 4) \in \mathbb{R}^7$ should be replaced by the vector $x = (1, 3, 4, 5) \in \mathbb{R}^4$. Write a main program that reads in the length $n \in \mathbb{N}$ and the vector $x \in \mathbb{R}^n$, and prints out the shortened vector. Work with dynamically allocated memory.

Aufgabe 7.5. Write a library for *columnwise*(!) stored $m \times n$-matrices. Implement the following functions

- `double* mallocmatrix(int m, int n)`
  Allocates memory for a columnwise stored $m \times n$ matrix.
- `double* freematrix(double* matrix)`
  Frees memory of a matrix.
- `double* reallocmatrix(double* matrix, int m, int n, int mNew, int nNew)`
  Reallocates memory and initializes new entries.

Store the signatures of the functions in the header file `dynamicmatrix.h`. Write also appropriate comments to this functions in the header file. The file `dynamicmatrix.c` should contain the implementations of the above functions.

Aufgabe 7.6. Write a structure (data-type) `polynomial` for the storage of polynomials that are represented as $p(x) = \sum_{j=0}^n a_j x^j$. Note that you have to store the degree $n \in \mathbb{N}_0$ as well as the coefficient vector $(a_0, \dots, a_n) \in \mathbb{R}^{n+1}$. Write all necessary functions to work with this structure (`newPoly`, `delPoly`, `getPolyDegree`, `getPolyCoefficient`, `setPolyCoefficient`). Moreover, write a function `evalPoly` that evaluates a polynomal at a given point $x \in \mathbb{R}$.

Aufgabe 7.7. The sum $r = p + q$ of two polynomials p, q is again a polynomial. Write a function `addPolynomials` that computes the sum r. For the storage of polynomials use the structure from Exersice 7.6. Additionally, write a main program that reads in two polynomials and computes the sum thereof.

Aufgabe 7.8. The k-th derivative $p^{(k)}$ of a polynomial p is again a polynomial. Write a function `differentiatePolynomial` that computes the k-th derivative of a polynomial. For the storage of polynomials use the structure from Exersice 7.6. Additionally, write a main program that reads in p and k, and prints out $p^{(k)}$.