Dirk Praetorius
Michele Ruggeri

Wintersemester 2014/15
25.11.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 9

Aufgabe 9.1. Write a function `checkoccurrence`, which, given a string s and a character b, returns how many times b occurs in s. Both the lowercase and the uppercase versions of b contribute to the number of occurrences. Then, write a main program which reads s and b from the keyboard and calls the function. Save your source code as `checkoccurrence.c` into the directory `serie09`.

Aufgabe 9.2. Given a convergent sequence $(x_n)_{n\in\mathbb{N}}$ with limit x, if there exist $p \geq 1$ and a constant $c > 0$ such that $|x_n - x| \leq c\,|x_{n-1} - x|^p$ for all $n \in \mathbb{N}$, then we say that p is the convergence rate of $(x_n)_{n\in\mathbb{N}}$ towards x. With the ansatz

$$|x_{n+2} - x| = c|x_{n+1} - x|^p \quad \text{and} \quad |x_{n+1} - x| = c|x_n - x|^p \quad \text{for } n \in \mathbb{N},$$

we can determine the values of p and c for any n. An easy computation shows

$$p = \frac{\log(|x_{n+2} - x|/|x_{n+1} - x|)}{\log(|x_{n+1} - x|/|x_n - x|)} \quad \text{and} \quad c = \frac{|x_{n+2} - x|}{|x_{n+1} - x|^p}.$$

To start with, derive the above formulas. Then, write a function `convorder`, which, given a sequence $(x_n)_{n=1}^N$ and a limit x, computes and returns the empirical convergence rate $p, c \in \mathbb{R}^{N-2}$. In concrete situations, the limit x is usually unknown and only the sequence $(x_n)_{n=1}^N$ is available. In this case, the function `convorder` should apply the above formulas to the subsequence $(x_n)_{n=1}^{N-1}$ and $x := x_N$. Save your source code as `convorder.c` into the directory `serie09`.

Aufgabe 9.3. 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 $x \in \mathbb{R}$. Save your source code as `polynomial.c` into the directory `serie09`.

Aufgabe 9.4. 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 Exercise 9.3. Additionally, write a main program that reads in two polynomials and computes the sum thereof. Save your source code as `addPolynomials.c` into the directory `serie09`.

Aufgabe 9.5. 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 Exercise 9.3. Additionally, write a main program that reads in p and k, and prints out $p^{(k)}$. Save your source code as `differentiatePolynomial.c` into the directory `serie09`.

Aufgabe 9.6. Write a structure data-type `squareMatrix` for the storage of quadratic matrices $A \in \mathbb{R}^{n\times n}$. The structure should contain the dimension $n \in \mathbb{N}$ and the entries given as `double*`, i.e., the entries of the matrix should be stored columnwise. Implement the functions `newSquareMatrix, delSquareMatrix, getSquareMatrixDimension, getSquareMatrixEntry` and `setSquareMatrixEntry`. Save your source code as `squareMatrix.c` into the directory `serie09`.

Aufgabe 9.7. What does the following function `func`, when it is called with the matrix

$$A = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 4 & 0 & 3 \\ 1 & 2 & 0 & 2 \\ 17 & 4 & 4 & 1 \end{pmatrix}?$$

A is stored in the structure from Aufgabe 9.6. Create a table, where you put in the values of all variables at the given time (the comment line in the following code). What does the function `func` checks? Is the code efficient? If not, how can you implement it efficiently?

```
int func(squareMatrix* mat) {
   double foo = 0;
   int mp, dp, tf;
   mp = 1;
   for (dp = 0; dp < getMatrixDim(mat); ++dp) {
      for (tf = dp+1; tf < getMatrixDim(mat); ++tf) {
         foo = getMatrixEntry(mat,dp,tf);
         if ( foo != 0 ) {
            mp = 0;
         }
      /* VALUE OF VARIABLES AT THIS POINT */
      }
   }
   return mp;
}
```

Aufgabe 9.8. A matrix $A \in \mathbb{R}^{n\times n}$ is symmetric if $A_{jk} = A_{kj}$ for all $j, k = 1, \dots, n$. Write a function `issymmetric` that returns 1 if the matrix A is symmetric and 0 if it is not. Use the structure of Exercise 9.6. Then, write a main program, which reads A and displays its nature (symmetric or nonsymmetric). Test your function appropriately. Save your source code as `issymmetric.c` into the directory `serie09`.