Dirk Praetorius,
Thomas Führer

Sommersemester 2014
05.05.2014

Übungen zur Vorlesung Einführung in das Programmieren für TM

Serie 8

Aufgabe 8.1. 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`.

Aufgabe 8.2. 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 Exercise 8.1. 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 8.3. 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 8.1. Test your function appropriately.

Aufgabe 8.4. Write a structure data-type `cdouble`, which stores the real part a and the imaginary part b of a complex number $a + bi \in \mathbb{C}$ as `double` variables. Implement the functions `cdouble* newCDouble(double a, double b)`, `cdouble* delCDouble(cdouble* c)` as well as `setCDoubleReal`, `getCDoubleReal`, `setCDoubleImag` sowie `getCDoubleImag`. Divide your source code into the header file `cdouble.h` and the file `cdouble.c`.

Aufgabe 8.5. Write functions `cadd`, `csub`, `cmul`, `cdiv`, which realize the addition, subtraction, multiplication, and division of complex numbers $a + bi \in \mathbb{C}$. Furthermore, implement the function `double cnorm(cdouble* c)`, which should compute the squared norm $|a + ib|^2 := a^2 + b^2$. Use the structure from Exercise 8.4 to store complex numbers and use the `get` and `set` functions thereof. Additionally, write a main program that reads in two complex numbers and tests the arithmetic operations.
Optional work: Why should you avoid the nested usage of the above functions? Correct the following code (it is an issue with dynamic memory):

```
cdouble* c1 = newCDouble(1,2);
cdouble* c2 = newCDouble(3,4);
cdouble* c3 = cadd(cmul(c1,c1),c2);
c1 = delCDouble(c1);
c2 = delCDouble(c2);
c3 = delCDouble(c3);
```

Aufgabe 8.6. Write a structure `CPoly` for the storage of polynomials, where the coefficients are complex numbers, i.e., $p(x) = \sum_{j=0}^{n} a_j x^j$ with $a_j \in \mathbb{C}$. The structure should contain the degree $n \in \mathbb{N}$ and the coefficients $(a_0, \dots, a_n) \in \mathbb{C}^{n+1}$. Use the structure from Exercies 8.4. Moreover, implement the functions `newCPoly`, `delCPoly`, `getCPolyDegree`, `getCPolyCoefficient`, and `setCPolyCoefficient`.

Aufgabe 8.7. Write a function `addCpolynomials` that computes the sum $r = p + q$ of two complex polynomials p, q and returns r. Use the structure from Exercise 8.6. Moreover, write a main program that reads in two polynomials p, q and prints out the sum $r = p + q$.

Aufgabe 8.8. Write a structure `CMatrix` for the storage of $m \times n$-matrices $A \in \mathbb{C}^{m \times n}$ with complex entries. Use the structure `cdouble` from Exercise 8.4. Furthermore, write the functions `newCMatrix`, `delCMatrix`, `getCMatrixM`, `getCMatrixN`, `getCMatrixCoeff`, `setCMatrixCoeff`.