Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
08.05.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

Aufgabe 8.1. 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`). Save your source code as `polynomial.c` into the directory `serie08`.

Aufgabe 8.2. 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 8.1. 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 `serie08`.

Aufgabe 8.3. The product $r = pq$ of two polynomials $p(x) = \sum_{j=0}^{m} a_j x^j$ and $q(x) = \sum_{j=0}^{n} b_j x^j$ is again a polynomial. Write a function `prodPoly` that computes the product r and stores it in the structure from Exercise 8.1. At first, think about the degree of the polynomial r. Additionally, write a main program that reads in two polynomials and computes the product thereof. Test your code on a suitable example. Save your source code as `prodPoly.c` into the directory `serie08`.

Aufgabe 8.4. Write a function `evalPoly` which computes for a given polynomial p and point $x \in \mathbb{R}$, the point value $p(x)$. For the storage of polynomials use the structure from Exercise 8.1. Test your code on a suitable example. Save your source code as `evalPoly.c` into the directory `serie08`.

Aufgabe 8.5. 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 8.1. Additionally, write a main program that reads in p and k, and prints out $p^{(k)}$. Test your code on a suitable example. Save your source code as `differentiatePolynomial.c` into the directory `serie08`.

Aufgabe 8.6. Write a structure `cdouble` to store the real part $a \in \mathbb{R}$ and the imaginary part $b \in \mathbb{R}$ of a complex number $a + bi \in \mathbb{C}$ as `double` variables. The imaginary unit i satisfies the identity $i^2 = -1$; see

`https://en.wikipedia.org/wiki/Complex_number`.

Implement the functions

- `cDouble* newCDouble(double a, double b)`,
- `cDouble* delCDouble(cDouble* z)`

as well as the mutator functions

- `void setCDoubleReal(cDouble* z, double a)`,
- `double getCDoubleReal(cDouble* z)`,
- `void setCDoubleImag(cDouble* z, double b)`,
- sowie `double getCDoubleImag(cDouble* z)`.

How did you test your implementation? Save the source code, split into a header file `cdouble.h` and `cdouble.c`, into the directory `serie08`.

Aufgabe 8.7. 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 Exercise 8.6. Moreover, implement the functions `newCPoly`, `delCPoly`, `getCPolyDegree`, `getCPolyCoefficient`, and `setCPolyCoefficient`. How did you test your implementation? Save your source code as `cpoly.c` into the directory `serie08`.

Aufgabe 8.8. 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 8.7. Moreover, write a main program that reads in two polynomials p, q and prints out the sum $r = p + q$. How did you test your implementation? Save your source code as `addcpoly.c` into the directory `serie08`.