Valentin Helml
Dirk Praetorius
Julian Streitberger

Summer term 2022
13 May 2022

Exercise
Scientific programming in mathematics

Series 8

Exercise 8.1. Write a class `Fraction` for the representation of fractions of the form $x = c/d$, where $c \in \mathbb{Z}$ and $d \in \mathbb{N}$. The enumerator and the (positive) denominator are stored as `int`. Implement the following features:

- The standard constructor (without parameters), which sets $c = 0$ and $d = 1$.
- A constructor, which takes $c, d \in \mathbb{Z}$ as input parameters and stores the fraction p/q (use `assert` to ensure valid input, i.e., $d \neq 0$ and, for the case $d < 0$, store the fraction $(-c)/|d|$).
- The destructor.
- The copy constructor.
- The assignment operator.
- Appropriate `get` and `set` functions.

Test your implementation appropriately.

Exercise 8.2. Extend the class `Fraction` from Exercise 8.1 by the method `void reduce()`, which reduces the fraction c/d to lowest terms, i.e., in the form c_0/d_0, where $c_0 \in \mathbb{Z}$ and $d_0 \in \mathbb{N}$ are coprime and satisfy $c_0/d_0 = c/d$. Use an algorithm of your choice to compute the greatest common divisor of enumerator and denominator, e.g., the Euclidean algorithm. Moreover, implement the type cast from `Fraction` to `double` and vice versa. For the second case, consider only the first 9 decimals and note that the return value should be in reduced form. Test your implementation appropriately.

Exercise 8.3. Overload the operators $+$, $-$, $*$ and $/$, in order to be able to calculate the sum, the difference, the product and the quotient of two fractions stored as objects of type `Fraction` from Exercise 8.1. For the division, use `assert` to avoid dividing by 0. For all cases, the return value should be in reduced form. Furthermore, overload the operator `<<` in order to be able to print a fraction stored in an object `x` of type `Fraction` by typing `cout << x`. Test your implementation appropriately.

Exercise 8.4. Write a class `Polynomial` to store polynomials p of degree $n \in \mathbb{N}_0$ represented with respect to the monomial basis, i.e.,

$$p(x) = \sum_{j=0}^{n} a_j x^j.$$

The class contains the polynomial degree $n \in \mathbb{N}_0$ (type `int`), and the dynamic vector $a = (a_0, \dots, a_n) \in \mathbb{R}^{n+1}$ of the coefficients (of type `double*`). Implement the following features:

- The constructor, which, given $n \in \mathbb{N}_0$, allocates a polynomial of degree n and initializes all entries of its coefficient vector with zero. Use `assert` to ensure that $n \geq 0$.
- The destructor.
- The copy constructor (note that you have to allocate new dynamic memory for the coefficient vector of the output, why?).
- The assignment operator.
- A method `int degree() const` to get the polynomial degree $n \in \mathbb{N}_0$.
- The possibility to print a polynomial `p` to the screen via the syntax `cout << p`.

Test your implementation appropriately.

Exercise 8.5. Extend the class `Polynomial` from the previous Exercise 8.4 by capability of accessing the coefficients of the polynomial via `[ ]`, i.e., for $0 \leq j \leq n$, the j-th coefficient a_j of a polynomial stored in an object `p` can be obtained by typing `p[j]`. Implement this feature for both `const` and 'normal' object, i.e., in the class definition using signatures

```
const double& operator [](int j) const;
double& operator [](int j);
```

Use `assert` to ensure that $0 \leq j \leq n$. Test your implementation appropriately.

Exercise 8.6. The sum of two polynomials is still a polynomial. What is the degree of the polynomial sum $p + q$ in terms of the degrees n and m of p and q, respectively? What are the coefficients of $p + q$? Extend the class `Polynomial` of Exercise 8.4 by the feature of adding two polynomials `p` and `q` by typing `r=p+q`. Note that the input polynomials might have different degrees. Moreover, implement the opportunity to add a number $c \in \mathbb{R}$ stored as `double` or `int` to a polynomial `p` in an appropriate way via `r = c+p` and `r=p+c`. Test your implementation appropriately.

Exercise 8.7. The product of two polynomials is also still a polynomial. What is the degree of the polynomial sum pq in terms of the degrees n and m of p and q respectively? What are the coefficients of pq? Extend the class `Polynomial` of Exercise 8.4 by the feature of multiplying two polynomials `p` and `q` by typing `r=p*q`. Note that the input polynomials might have different degrees. Moreover, implement the scalar multiplication with a number $c \in \mathbb{R}$ stored as `double` or `int` to a polynomial `p` in an appropriate way via `r = c*p` and `r=p*c`. Test your implementation appropriately. What is the computational complexity of your implementations for two polynomials of the same degree n?

Exercise 8.8. Explain in your own words the 'rule of three'. What is the actual reason why always destructor, copy constructor, and assignment operator must be implemented when dynamic memory is used?