Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
05.06.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Aufgabe 10.1. Explain the differences between references and pointers. Write some code which swaps the values of two variables. Implement a version which uses pointers and then implement a second version which uses references. What are the advantages of using references? What are the disadvantages?

Aufgabe 10.2. Write a class `Alcohol` for the storage of different alcoholic drinks. The class should contain the following members: name, alcoholic strength percent, price in €. Moreover, implement an appropriate constructor and overload `operator<`, that compares two objects of the class with respect to the ratio $\frac{\text{Vol.\%}}{€}$. Additionally, implement the methods `getName()`, `getPrice()`, and `getVolPercent()`.
Hint: In general, the operator `<` is overloaded by the syntax

```
bool operator<(const type& lhs, const type& rhs);
```

Here, `type` is an arbitrary datatype. In our case it is `Alcohol`. Moreover, overload the `<<`-operator in order to be able to print the data of your object on screen. Test your code appropriately. Save your source code as `Alcohol.{hpp/cpp}` into the directory `serie10`.

Aufgabe 10.3. Write a function `sortAlcohol` which sorts an array with length $n > 0$ and entries of the type `Alcohol` from Exercise 10.2 ascending with respect to the ratio $\frac{\text{Vol.\%}}{€}$. You may use every sorting algorithm you know. Moreover, write a `main`-programme in which you read in the length n and the entries of the type `alcohol` and call `sortAlcohol`. Test your implementation appropriately. Save your source code as `sortAlcohol.cpp` into the directory `serie10`.

Aufgabe 10.4. Extend the class `Fraction` from slide 229 by

- the standard constructor (without parameters), which sets $p = 0$ and $q = 1$,
- a constructor, which takes $p, q \in \mathbb{Z}$ and $q \neq 0$ as input, and stores the corresponding `Fraction`,
- the copy constructor,
- the assignment operator, and
- the destructor.

Use `assert` to ensure a valid input, i.e, $q \neq 0$. Note that, for the case $q < 0$, you have to store the fraction $(-p)/|q|$. Test your implementations appropriately! Save your source code as `Fraction.{hpp/cpp}` into the directory `serie10`.

Aufgabe 10.5. Implement a method `reduce` for the class `Fraction` from Exercise 10.4, which reduces the fraction to lowest terms, i.e., in the form p/q with $p \in \mathbb{Z}$ and $q \in \mathbb{N}$ coprime. Use an algorithm of your choice in order to compute the greatest common divisor of numerator and denominator, e.g., Euclid's algorithm (see slides 96–98 and slide 106 from the lecture) or an algorithm which computes the prime factorization of integers. Test your implementation appropriately!

Aufgabe 10.6. Overload the $+$, $-$, $*$ and $/$ operator in order to be able to calculate the sum, the difference, the product and the quotient of two fractions stored in the format `Fraction` from Exercise 10.4. For $/$, use `assert` to avoid dividing by 0. In all cases, the result is returned in reduced form. Test your implementations appropriately!

Aufgabe 10.7. Overload the comparison operators `==`, `!=`, `<`, `<=`, `>`, `>=` for the class `Fraction` from Exercise 10.4. Compare directly the numerator and the denominator of the fractions (*Hint:* At same point, you might need to determine a common denominator for the fractions). Test your implementation accurately!

Aufgabe 10.8. What is the output of the following C++ program? Explain why! What are the differences between the different variable types used?

```
#include <iostream>
using std::cout;
using std::endl;

const int proc(int & input){input = input*2; return input;}
int proc(const int & input){ int output = input; return output;}

void swap(int& x, int& y){
int tmp;
tmp = x;
x = y;
y = tmp;
}

void swap(const int& x,const int& y){;}

int main() {

int var1 = 1;
int var2 = 2;
int var3 = proc(var1);
int var4 = proc(var2);
const int var5 = proc(var1);
const int var6 = proc(var2);
int var7 = proc(proc(var1));
int var8 = proc(proc(var2));
int& var9 = var1;
int& var10 = var2;
const int& var11 = proc(var1);
const int& var12 = proc(var2);

swap(var3,var4);
swap(var5,var6);
swap(var7,var8);
swap(var9,var10);
swap(var11,var12);

cout << "var1 = " << var1 << " var2 = " << var2 << endl;
cout << "var3 = " << var3 << " var4 = " << var4 << endl;
cout << "var5 = " << var5 << " var6 = " << var6 << endl;
cout << "var7 = " << var7 << " var8 = " << var8 << endl;
cout << "var9 = " << var9 << " var10 = " << var10 << endl;
cout << "var11 = " << var11 << " var12 = " << var12 << endl;

return 0;
}
```