Dirk Praetorius
Michele Ruggeri

Sommersemester 2015
09.06.2015

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 11

Aufgabe 11.1. Adapt the Code from the class `Matrix` from Exercise 10.5 and Exercise 10.6, so that `new` resp. `delete` is used instead of `malloc` resp. `free`. What are the differences between `new` resp. `delete` and `malloc` resp. `free`? Last week you unknowingly used the "Rule of three". What is this rule saying? Why is this rule important in that context?

Aufgabe 11.2. Overload the operator `*` and `/` for the class `Matrix` from Exercise 10.5 and Exercise 10.6 so that the pointwise multiplication and pointwise division is performed. Overload the operator `*` again, to be able to perform `Matrix * double` and `double * Matrix`. Apply security checks, when necessary. Moreover, test your implementation.

Aufgabe 11.3. Overload the operator `<<` for the class `Matrix` from Exercise 10.5 and Exercise 10.6 to be able to run `cout << A` for a matrix A. Moreover, test your implementation.

Aufgabe 11.4. 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`.

Aufgabe 11.5. Implement the simple *Tic Tac Toe* game. The rules can be found at

http://en.wikipedia.org/wiki/Tic-tac-toe

Your program should fulfill the following criteria:

1. Two players who could play against each other.
2. An automatic check if one of the player has won.
3. Print out the playfield after each step.
4. If none of the player can win, print out `'tied game'`

Use the calss `Matrix` for the playfield. Each of the player should use its own character symbol, e.g., `'1'` for player 1 and `'0'` for player 2. The playfield should be an $n \times n$-array with $n \geq 3$!

Aufgabe 11.6. Implement a class `Person` which contains the members `name` and `address`. Derive a class `Student` from `Person`, that contains the additional data-fields `matriculationNumber` and `study`. Derive another class `Worker` that contains the additional data-fields `salary` and `work`. Write `set/get` functions, constructors and destructors for these. Moreover, write a main progam to test your implementation!

Aufgabe 11.7. Implement all the necessary constructors and destructors for the class fraction from Exercise 10.2. so that the "rule of three" is fulfilled. Moreover, implement a method `reduce()`. (Hint: "Recycle" the code from Exercise 9.4.) Then, overload the operators +,-,* and /. Apply security checks, when necessary. The result should be another fraction. Use the method `reduce()` in a suitable way. Test your implementation on suitable examples.

Aufgabe 11.8. Overload the operators `<,<=,>,>=` for the class `Fraction` from Exercise 10.2. Test your implementation on suitable examples.