Dirk Praetorius,
Thomas Führer

Sommersemester 2014
06.06.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 12

For the first two exercises inform yourself in the net about standard libraries. Use one of the following sites:

`http://www.cppreference.com` or `http://www.cplusplus.com`.

The last exercise deals with templates, which will be discussed in the last lecture. You can also have a look at the slides from the summer term 2013.

Aufgabe 12.1. A `pair` is a C++-datatype that contains two (possible) different types. A pair of `double`-values is denoted as `pair<double,double>`. The code `pair<double,double>(5.,3.)` creates a pair with the two `double`-values `5.`, `3.`. You have to include the header file `map` to use pairs.
Write a function `minmax` that takes a list of floating-point numbers as input, and returns the minimum and maximum of this list as a pair, i.e. the return data-type of the function should be a pair. Test your implementation properly.
Hint: You can access the first value of a pair with `.first` and the second one with `.second`. For example:

```
pair<int, double> x(5, 17.4);
cout << x.first << ", " << x.second << endl; // Output: 5, 17.4
```

Aufgabe 12.2. Write a function `sortfile` that reads in a file (row-wise) into a vector. This vector should be sorted in lexicographical order and then be printed out. Create a proper text file to test your program!
Hint: The following code reads in a file (row-wise):

```
fstream file("file.txt");
while (file.good()) {
   string row;
   getline(file, row);
}
```

Solve the exercise by adapting this code. You have to include the header file `fstream`!
Hint: The relation operator `<` for `string` corresponds to the lexicographical order. If you want, you can use the function `sort` from the standard library.

Aufgabe 12.3. Implement a class `Person` which contains the members `name` and `address`. Write `set/get` functions for these. 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`. Think about which members are `private`, `public`, or `protected`.

Aufgabe 12.4. Implement the method `void print()` in the basis class `Person` from exercise 12.3. The method should print out the name and address of a person. Redefine this function in the derived classes `Student` and `Worker` (the additional data-fields should also be printed out). Moreover, write a main programm for testing the `print`-methods of the different classes.

Aufgabe 12.5. Consider the class `Matrix` and the derived class `SquareMatrix` from the lecture. Implement the method `computeLU`, that computes the LU-factorization, for the class `SquareMatrix`. The return value (a matrix $R \in \mathbb{R}^{n\times n}$ is again of the type `SquareMatrix`, where the triangular matrices L and U should be stored in R. The diagonal of L does not need to be stored. Why?

Not every matrix $A \in \mathbb{R}^{n\times n}$ has a normalized LU-factorization $A = LU$, i.e.,

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \ell_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n1} & \dots & \ell_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \dots & 0 & u_{nn} \end{pmatrix}.$$

In the case there exists such a factorization, it holds

$$\begin{aligned} u_{ik} &= a_{ik} - \sum_{j=1}^{i-1} \ell_{ij} u_{jk} \quad \text{for } i = 1, \dots, n, \quad k = i, \dots, n, \\ \ell_{ki} &= \frac{1}{u_{ii}} \left(a_{ki} - \sum_{j=1}^{i-1} \ell_{kj} u_{ji} \right) \quad \text{for } i = 1, \dots, n, \quad k = i+1, \dots, n, \\ \ell_{ii} &= 1 \quad \text{for } i = 1, \dots, n, \end{aligned}$$

which can be verified by using the formula for the matrix-matrix multiplication.

Aufgabe 12.6. What is the computational cost of the LU-factorization from exercise 12.5? Write down your results in the $\mathcal{O}$-notation.

Aufgabe 12.7. The determinant of a matrix $A \in \mathbb{R}^{n\times n}$ can be computed with the normalized LU-factorization from exercise 12.5. It holds $\det(A) = \det(L)\det(U) = \det(U) = \prod_{j=1}^{n} u_{jj}$. Extend the class `SquareMatrix` by the method `detLU`, that computes and returns the determinant. The matrix A should not be overwritten.

Aufgabe 12.8. Write a template function `minsort(std::vector<T> v)` that takes a vector of `T`-objects as input, sorts the vector in ascending order and returns it. You can assume that the operator `<` for the data-type (class) `T` is defined. The sorting method should work as in the `minsort` function defined in the lecture notes, see slide 75. Test your implementation with different(!) data-types.