Laboration 4

The STL vector container, function objects
DT079G - Örogrammeringsmetodik

In this laboratory, it is necessary to develop a database as a part of some e-mail system, which could store and display the required data. First of all, class `Email` must be developed to keep information about a received letter: sender's name ("who"), date of receiving ("date"), and a "subject". These data members should be of type string. Class `Email` should contain a combined constructor, overloaded friend operator <<, as well as three friend function-like classes (structures): `CompWhoDateSubject`, `CompDateWhoSubject`, `CompSubjectWheDate`, objects of which will be used in the STL `sort` algorithm. Each of these classes has only one member function - overloaded function call `operator ()` returning a `bool` value. Such an operator takes and compares two parameters of type `Email`, say, lhs and rhs, i. e., it's a binary predicate. The comparing algorithm should be a multi-level one (cascaded). For example, for the class CompWhoDateSubject, we need first to compare lexicographically "who" data members of the objects lhs and rhs. If we choose an ascending order, the operation

```
lhs.who < rhs.who;
```

must return true. However, we should also consider a situation when

```
lhs.who == rhs.who;
```

and compare the data members "date". If "date"s are equal too, we should compare the data members "subject". That is, if we have two `Email` objects:

```
a("Anders", "2002-02-28", "lab 1"),
b("Anders", "2002-02-28", "lab 2"),
```

the call of `CompWhoDateSubject` returns true if a is lhs and b is rhs, i.e., a is less than b.

A nested if/else structure can be used for the multi-level comparing, or logical operators || and && may be used.

Secondly, a class `MailBox` must be defined, it should include as a private data member the STL `vector` container of type `Email`. The `MailBox`'s combined constructor must initialize its member, calling vector's parameterized constructor of 1st type that produces a sequence of n elements. An access function enabling both vector's reading and writing should also be defined.

The class `MailBox` must also contain three member functions - `SortWho`, `SortDate`, and `SortSubject`, each of them sorts the vector by calling the STL `sort` algorithm with a corresponding binary predicate - a function object of class `CompWhoDateSubject`, etc.

In order to display the `MailBox` 's vector in the main, a global function template `Show` should be created that takes a reference to a const vector's object.

In the main, create a `MailBox` 's object with 3 vector's elements by default, fill in it with 5 `Email` 's messages and test it.