Program specialization and metaprogramming

Laboratory work No. 4

Practical programming using Standard Template Library (STL)

1. Aims

To get acquainted with programming using Standard Template Library (STL).

2. Tasks

Learn how to use the classes and functions from STL.

3. Work

The Standard Template Library (STL) is a collection of C++ template classes that makes everyday programming tasks easier to complete, and greatly improves the quality of the code that we create. In this lab we take a close look at how we can use the vector in our own programs, and will take a glimpse into one very useful container sorting routine provided by STL. In addition, we will use a new object called an iterator, and we will see how the idea of a pointer can be used to allow for a powerful and useful extension, especially in the context of the STL.

To get started get the following source file from the course Moodle page: VectorDriver.cpp.

Tasks:

1. The function fill_vector assumes that its argument is a vector that already has allocated some space which needs to be filled. Implement the void add_numbers(vector<short> &data) method that adds 10 numbers to the data vector (feel free to use a for loop). Test your code by creating an empty vector in main(), call the add_number function, and then print the vector from main(). Try calling add_number twice in a row with the same argument - what happens?
2. Take a look at the sample print function provided in the VectorDriver, and then create a new function void print_even(const vector<short>& data) that prints all of the elements in the data vector that are stored at an even index (i.e. data.at(0), data.at(2), ...) .
3. In the tasks above, we used the at() method to access elements of a vector. There is another way to do it - by using an object called an iterator. Iterators are smart pointers that can be dereferenced using the * operator, and can be incremented via ++ (plus, plus) and decremented via -- (minus, minus). What we mean by "incrementing" an iterator is that it moves "forward" in a sequence of items stored in a container. For example, if you start with an iterator that points to the first

element in a vector and then increment the iterator, the iterator will now point to the second element in the vector. Look at the compute_sum function in the VectorDriver for an example of how one might use an iterator. Implement the bool is_present(const vector<short> &data, short value) routine that returns true if the value is present in the data vector, and false otherwise.

4. provides utilities in addition to data structures - namely useful algorithms that have been implemented. One of these algorithms is the sort algorithm that can efficiently sort a container for us in ascending order. All we need to do is tell the sort routine where the start and end of the container are. Now in main() function, manually create a small vector of about 5 numbers in random order, and call the sort routine on the begin() and end() iterators of your vector.

   What is the result?

4. Report

Prepare a report of your work in this lab using a standard word processing application. The report is a single chapter of your final semester report. Final semester report will have to be uploaded to course Moodle page.

The content of report:

1. Title
2. Aims and tasks
3. Work done, described in steps and illustrated by screenshots and written or modified source code. Provide comments what you have done and what were the results (program outputs). Present answers to the questions given in the lab description.
4. Conclusions – what you have learned?