C++ Standard Template Library (STL)

Task Summary

You will use containers, iterators, and algorithms of the C++ Standard Template Library (STL). The instructions below guide you through all steps, starting with the creation of a C++ project in Eclipse with the CDT plug-in.

Create C++ Project

Create a new C++ Project with *File – New – Other – C++ – C++ Project*. Click on *Next*, enter a project name (e.g. STLLab), and select *Executable – Hello World C++ Project* as *Project type* and *MinGW GCC* as *toolchain* (you may have to uncheck *Show project types and toolchains only…* to be able to do that). Click on *Finish*. If the project is not built automatically, build the project by first selecting the *STLLab* project in the *Project Explorer* and then selecting *Project – Build All*. The console should show:

…
Build complete for project STLLab
Time consumed: … ms.

Run the application by right-clicking on project *STLLab* and selecting *Run As – Local C/C++ Application*. The console should show: *Hello World!!!*

First Steps

Replace the content of the STLLab\src\STLLab.cpp file with the following:

```
#include <iostream>
#include <vector>  //vector class-template
using namespace std;
int main()
{
  vector<int> v;

  // add integers at the end of the vector
  v.push_back(2);
  v.push_back(3);
  v.push_back(4);
  v.push_back(7);
  v.push_back(5);

  // display info about v
  cout << "The size of v is: " << v.size() << "\nThe capacity of v is: " << v.capacity();
  vector<int>::const_iterator it;
  cout << "\nThe content of v is: ";
  for (it = v.begin(); it != v.end(); it++)
  {
        cout << *it << " ";
  }
  //============================================================
  return 0;
}
```

Save the file, then build and run the project as you did before. The console should show:

The size of v is: 5
The capacity of v is: 8
The content of v is: 2 3 4 7 5

Task A – Vector

Extend the code shown above by performing the following operations on the vector (note that more info regarding these operations can be found at http://www.cppreference.com). Add the extensions just before the `// display info about v` line. You may want to run the application each time after completing an operation to verify the content of the vector.

1) Insert 7 between the first and second element (use insert member function).
2) Replace the first element with 9 (use direct access with element index).
3) Copy the first two elements to the end of the vector (use push_back member function and direct access with element indexes).
4) Find the first element with value 3 (use find algorithm).
5) Erase the first element with value 3 (use erase member function).
6) Find the element with the smallest value and output the value to the console (use min_element algorithm).
7) Sort the vector (use sort algorithm).

8) Reverse the elements of the vector (use reverse algorithm).
9) Count the elements with value 9 and output the count to the console (use count algorithm).
10) Sum up the first four elements and output the result to the console (use accumulate algorithm and you may have to include <numeric>).

Task B – Set

Copy the whole code (from `vector<int> v;` to the line with all the `=====`) to after the line with all the `=====`. In the copied code, replace `vector<int> v;` with `set<int,less<int> > s;` and update the rest of the copied code so that it works with the set. Do not forget to include <set>. Push_back() does not exist for sets, therefore use insert(). Also note that capacity is not defined for sets and will therefore have to be removed from the code for displaying info. Some of the following operations are slightly different for sets compared to vectors:

1) Cannot insert at a specific location since this concept does not exist for sets. Therefore, just insert 7 (use insert member function). Note that 7 will not be added to the set because it already exists in the set.
2) Direct access with element indexes does not exist for sets. Therefore erase the first element (use erase member function), and then insert 9.
3) Cannot copy elements since each value exists only once in the set. Skip this operation.
4) Same as #4 in Task A.
5) Same as #5 in Task A.
6) Same as #6 in Task A.
7) Sorting does not make sense for a set since it is already sorted. Skip this operation.
8) Reverse also does not make sense. Skip this operation.
9) Same as #9 in Task A. Note that the maximum count is always 1.
10) Almost the same as #10 in Task A, however, the set contains only four elements at this point, so you may sum up the content of the whole set.

Task C – Multimap

Again copy the whole code (from `vector<int> v;` to the line with all the `=====`) to after the second line with all the `=====`. In the copied code, replace `vector<int> v;` with `mp_type mp;` and update the rest of the copied code so that it works with the multimap. Do not forget to include <map> and define mp_type by adding `typedef multimap<int,string,std::less<int> > mp_type;` before the main function. Push_back() does not exist for multimap, therefore use insert(). Note that you have to pass a mp_type::value_type(key, value) into the insert function (e.g., to add 2 to the multimap use mp_type::value_type(2,"two")). Also note that capacity again is not defined for multimap and will therefore have to be removed from the code for displaying info. Furthermore, the content of multimap is not accessed with *iterator but with iterator->first (for the key) and iterator->second (for the value). Some of the following operations are slightly different for multimap compared to vectors:

1) Cannot insert at a specific location since this concept does not exist for multimap. Therefore, just insert 7 (use insert member function). Note that 7 will be added to the multimap as duplicates are allowed.
2) Direct access with element indexes does not exist for multimap. Therefore erase the first element (use erase member function), and then insert 9.
3) Copy the first two elements again into the multimap (use an iterator, store the content of the first two elements in mp_type::value_type objects, and then use insert to add to the multimap).
4) Find all elements with value 3 (use equal_range member function).
5) Erase all elements with value 3 (use erase member function).
6) Same as #6 in Task A.
7) Sorting does not make sense for a multimap since it is already sorted. Skip this operation.
8) Reverse also does not make sense. Skip this operation.
9) Count the elements with value 9 and output the count to the console (use count member function).
10) Sum up the first four elements and output the result to the console (use iterator).

Submission of your solution

Archive your STLLab project with your solutions to Task A, Task B, and Task C by right-clicking on *STLLab* and selecting *Export – General – Archive File*. Make sure that all of *STLLab* is checked and *save in zip format* to *archive file* LabSTL<YourFirstName><YourLastName>.zip. Submit your zip file to Virtual Campus at https://maestro.uottawa.ca/index.asp?LANG=EN. The deadline for submission of your zip file to Virtual Campus is 13:00 on November 14, 2011.