SCIT-EIS-UOW
CSCI251/CSCI851 Advanced Programming
Spring 2021

Laboratory Exercise 12

Note that lab exercises marked with a * are effectively extension exercises.

1 Task One: Warm–up exercises

1. **Container, iterator, function**. Look at the source in `One.cpp` and in `Two.cpp`.
   (a) How do the two files differ.
   (b) Compile and run each of the programs.
   (c) What do the programs do?
   (d) Why do the results differ? Is this a good thing? Why or why not?
   (e) Would you expect this behaviour with most sorting algorithms? Why or why not?
2. **Container efficiency**. The program in `Three.cpp` times how long it takes to populate two vectors with some command line specified number of elements. One of the vectors is populated with an element being added one at a time, while in the other we use reserve to set space aside in advance.
   (a) Compile with C++11.
   (b) How does the time taken for each of these methods compare across a range of input sizes? In testing it makes sense to start with small values until you can gauge roughly how long some number would likely take.
   (c) What happens if you use 200000000 as the input?

2 Task Two: Containers

In this section we are going to explore some containers.

1. We firstly need a simple class. The file `Thing.cpp` contains a class `Thing` that you can use.
2. Set up containers of each of the following, each ready to accept `Thing` objects. Don't populate them yet, that's part of the next activity.
   (a) `vector`.
   (b) `deque`.
   (c) `list`.
3. Add timer code as in `Three.cpp` to test how long is takes to carry out each of the following activities, where appropriate, on each of the containers. Following `Three.cpp` you should accept the command line argument `number` to make it easier to see how the operations compare across appropriate ranges.

   (a) `push_back`.
   (b) `push_front`.

4. The operation `push_front` won't work for vector. You can insert elements at the start but you need to use insert and an iterator. So we might need to do something like the following, for vector of Things `vThing` and Thing instance `myThing`.

   ```
   vThing.insert(vThing.begin(),myThing);
   ```

   How efficient is this? Again you should start testing with small values for number.

5. For each of the containers define an iterator to step through as if to output each element. Make sure you understand how to output the elements, but carry out the timing with the output part commented out. Does the relationship between the timing seem very stable?

3 Task Three: Algorithms

1. Look at using some of the generic algorithms below. The algorithms library is at

   `http://en.cppreference.com/w/cpp/algorithm`

   You should look at at least read the following, noting the requirements for each, complexity for each, and giving an example of how they operate. Some of these could be used with the containers from the previous parts.

   (a) `sort`.
   (b) `merge`.
   (c) `partial_sum`.
   (d) `shuffle`.
   (e) `next_permutation`.

2. * One sorting algorithm you are unlikely to use seriously is Bogosort. There is an implementation in `Bogosort.cpp`. Look at the various operations included to make sure you understand what is going on. Run this algorithm a few times with sensible values of NUM to get some idea of the cost.