Assignment STL: Using the C++ Standard Template Library (STL)

COSC 2336: Data Structures and Algorithms

Fall 2021

Objectives

- Understand how templates are used in HLL like C++ to provide generic containers.
- Apply ideas of algorithmic complexity to understand tradeoffs of using different STL containers.
- Familiarize yourself with using basic containers, list, vector, set, maps, and abstractions such as queue, stack, priority queue.

Description

This assignment is open ended. I have not given specific tasks to do, but described tasks you can complete to get points for htis assignment.

Overview and Setup

For this assignment you will be given the following files that you will be using and adding code to for this assignment.

File Name	Description
`assg-tests.cpp`	Unit tests for the Set class you are to implement.

As usual, before starting on the assignment tasks proper, you should make sure you have completed the following setup steps.

1. Copy the assignment repository on GitHub using the provided assignment invitation link for 'Assignment Classes' for our current class semester and section.
2. Clone the repository using the SSH url to your local class DevBox development environment.
3. Checkout the 'origins/feedback' branch to your locak working DevBox repository.
4. Configure the project by running the `configure` script from a terminal.
5. Confirm that the project builds and runs, though no tests will be defined or run initially. If the project does not build on the first checkout, please inform the instructor.
6. You should create the issue for Task 1 and/or for all tasks for the assignment now before beginning the first task. On your GitHub account, go to issues, and create it/them from the issue templates for the assignment. Also you should close the initial Pull request that should be automatically opened for you, so that you can open your own when commiting your work.

There are actually no files beyond a stub test file with an example of a test of an STL list. You will have to add in your own tests and/or files for this assignment.

Assignment Tasks

For this assignment demonstrate using the C++ standard template library containers and algorithms for points. You need to come up with your own tasks, add the code and files to the project, and get them to build and run. Some suggested tasks.

1. Reimplement the queue assignment functions but use STL queue and stack objects instead of our hand built ones.
2. Demonstrate using maps and ordered maps. What is the backing store used for each? Show adding and iterating over the itmes in the maps.
3. Demonstrate using list and vector. What is the difference. Show using multiple member methods to add items to these containers and get information about the containers. Show how vector can be accessed using operator[].
4. Demonstrate using the set STL container.
5. Demonstrate using a priority queue, as we did for the queue assignment. Create a priority queue of items and show that items are removed by priority.
6. The STL library also contains algorithms in addition to data structures and containers. Demonstrate some of these, especially how to sort STL containers using STL sorting methods.

Program Style

At some point you will be required to follow class style and formatting guidelines. The VSCode environment has been set up to try and format your code for some of these guidelines automatically to conform to class style requirements. But not all style issues can be enforced by the IDE/Editor. The instructor may give you feedback in your pull comments and/or create issues for you for the assignment that you need to address and fix. You should address those if asked, and push a new commit that fixes the issue (or ask for clarification if you don't understand the request). In general the following style/formatting issues will be required for programs for this class:

1. All programs must be properly indented. All indentation must be consistent and lined up correctly. Class style requires 2 spaces with no embedded tabs for all code indentation levels. The editor style checker should properly indent your code when you save it, but if not you may need to check or correct this if code is misaligned or not properly indented.
2. Variable and function names must use `camelCaseNameingNotation`. All variable and function names must begin with a lowercase letter. Do not use underscores between words in the variable or function name. Often function names will be given to you, but you will need to create variables, and maybe some functions, that conform to the naming conventions.
   - Global constants should be used instead of magic numbers. Global constants are identified using `ALL_CAPS_UNDERLINE_NAMING`.
   - User defined types, such as classes, structures and enumerated types should use camel case notation, but should begin with an initial upper case letter, thus `MyUserDefinedClass`.
3. You are required to use meaningful variable and function names. Choosing good names for code items is an important skill. The code examples and starting code tries to give examples of good and meaningful names. In general, do not use abbreviations. Single variable names should be avoided, except maybe for generic loop index variables `i`, `j`, etc. Make your code readable, think of it as writing a document to communicate with other developers (and with your instructor who will be evaluating your code).
4. There are certain white space requirements. In general there should usually never be more than 1 blank line in a row in your code. Likewise there should usually not be more than 1 blank space on a line. There should be 1 blank space before and after all binary operators like `+`, `*`, `=`, `or`.
5. Function documentation is required for all regular functions and all class member functions. You need to follow the correctly formatted Doxygen function documentation format. We will use function documentation generation, and you should be sure your documentation can be built without emitting warnings or errors. Likewise all files should have a file header documentation at the top. You should edit the file header of files where you add in new code (not simply uncommenting existing code). Make sure the information has your correct name, dates, and other information.
6. Practice using proper Git commit messages. You should refer to issues and tasks correctly in commit messages.

Additional Information

The following are suggested online materials you may use to help you understand the tools and topics we have introduced in this assignment.

- Geeks for Geeks C++ Standard Template Library Tutorial
- C PlusPlus .com STL Reference