COMP 3512 Assignment #3: Implementing a heap and using a design pattern

Christopher Thompson
cthompson98@bcit.ca

BCIT CST — due on Friday 07 December 2018 before 23:59:59 PM

Introduction

Assignment 3 is ready! For this take-home coding project, you will implement a simple heap and then use it, with one of the design patterns we cover in class, to solve a small programming dilemma of your choice.

Setup

Please complete the following:

1. Create a new project in CLion. Let's call it Heap. Choose C++ Executable for the project type and ensure you select C++14 as the Language Standard.
2. Examine the project files that are created. Note there is a file called main.cpp that contains a main method stub. Remember that CLion uses cmake, which is a build tool similar to ant.
3. Open the CMakeLists.txt file and ensure you set the correct compiler flags: -Wall -Wextra -pedantic -std=c++14 and so on...
4. Add this project to version control. From the VCS menu in CLion select Import into Version Control | Create Git Repository... to add the project to a repo.
5. Add the project to GitHub by returning to the VCS menu and selecting Import into Version Control | Share Project on GitHub. Call it Heap, make sure it's private, and add a first commit comment. It's fine to add all the files for your first commit.
6. Visit GitHub and ensure your repository appears. Add me as a collaborator. In GitHub, I am known as chris-thompson. You'll recognize my avatar.
7. Ensure you commit and push your work frequently. You will not earn full marks if you don't.

Requirements

1. Implement a heap template class:
   (a) Recall that a heap is a way to organize the elements of a range that allows for fast retrieval of the element with the highest value at any moment (by popping the head), even repeatedly, while allowing for fast insertion of new elements (by pushing an element onto a heap).
   (b) Recall that a template's code must all be in the hpp file. Create a file called heap.hpp. Ensure your heap provides the following public interface:
      i. A constructor that accepts any STL sequence of elements and copies them to your heap.
      ii. A method called push which accepts an element and pushes it into the heap.
      iii. A method called pop which accepts no parameters and removes and returns the root (max) element from the heap.

iv. A private method called heapify which accepts no parameters and returns nothing, which ensures the elements in the heap are in heap form. This method should be called internally after any element is popped or pushed to your heap.
v. A method called size which returns the number of elements in the heap.
vi. A method called is_empty which returns true if the heap is empty, else false.
vii. A method called clear( ) which delete the elements in the heap, ensuring no memory leaks remain.
viii. An overloaded insertion operator so we can print the heap to standard output.

2. We will cover several design patterns during the final week of this course. Please select one design pattern and, using the heap, implement the design pattern in a small, simple program that demonstrates its correct use for a specific use case.

3. That's it. Yes, that's it!

Grading

This take-home assignment will be marked out of 10. For full marks this week, you must:

1. (2 points) Commit and push to GitHub after each non-trivial change to your code

2. (6 points) Successfully write a program that correctly and fully implements these requirements using an object oriented solution

3. (2 points) Write code that is commented and formatted correctly using good variable names, efficient design choices, atomic functions. Ensure your code has been tested thoroughly by including a full suite of unit tests.

Good luck, and have fun.