COMP 3512 Lab 8

Object Oriented Programming in C++

Nov 16 11:59PM

1 Instructions

This week we will be creating a surveillance program that will help us track the whereabouts of the people of Springfield. We will be using STL containers, iterators and algorithms. You will create a program that reads a text file containing people's names and the locations they have visited throughout the day. The program will then print out a sorted list of people in the city, which people have visited, or not visited specific locations throughout the day.

2 Set up your lab

Start by creating a new project:

1. Download the peoplePlaces.txt file from the Learning Hub. Ensure you select C++14 as the Language Standard.

2. Remember that CLion uses cmake, which is a build tool similar to ant. cmake uses a file called CMakeLists.txt. **Did you remember to set the correct compiler flags?**

3. Add this project to version control and GitHub. From the VCS menu in CLion select Import into Version Control — Create Git Repository... to add the project to a repo.

4. Add the project to GitHub by returning to the VCS menu and selecting Import into Version Control — Share Project on GitHub. Call it Lab8, make sure it's private, and add a first commit comment. It's fine to add all the files for your first commit.

5. Visit GitHub and ensure your repository appears. Add me as a collaborator. In GitHub, I am known as jeffbcit. You'll recognize my avatar.

3 Requirements

Remember to create a separate source (.cpp) and header (.hpp) file for each class, plus an additional source (.cpp) file that contains the main method. The text file will be in the following format:

```
Person1 Location1 Location2 Location3 Location4
Person2 Location2 Location3 Location4
Person3 Location1 Location3 Location4
Person4 Location1 Location2
Person5 Location1 Location2 Location4
```

1. Read the list of people and places from peoplePlaces.txt
2. Your program must output the following:
   (a) Print out a sorted list of all the people. People with names starting with A appear first, Z last.
   (b) List the people who visited both Krusty-Burger and Tavern
   (c) List the people who did NOT visit Krusty-Burger and Home
   (d) List the people who visited Krusty-Burger and School but did NOT visit Tavern and Home
   (e) List the people who visited all locations and remove them from your list
   (f) Print out a sorted list of all the people after removal
3. Implementation is generally open as long as you satisfy the above output requirements. However, we should use STL algorithms where possible.
   (a) I know we love vectors, but can you think of a way to use STL maps?
   (b) We need to sort people's names, STL maps provide automatic sorting don't they?
   (c) Look up STL algorithms find, remove, `for_each`, sort. These algorithms might help you instead of having to code the operations manually.
   (d) Print the results of your surveillance to the screen. The format of the output should be similar to the following:

```
Everybody in the city
//print out everybody

Visited both Krusty-Burger and Tavern
Person2
Person6
Person9
```

```
Did NOT visit Krusty-Burger and Home
Person7
Person13
Person20

Visited Krusty-Burger and School but did NOT visit Tavern and Home
Person9
Person10

Removing visited all locations
Person1
Person3

Everybody in the city after removing people who visited all locations
//everybody minus people who visted all locations
```

4. Add additional people and places into peoplePlaces.txt to test your code works
5. Ensure all member variables are private or protected. Do not use global or public members to store instance data.
6. Ensure you are not duplicating code.
7. After you submit your Lab to GitHub, send me a private message on Slack telling me you're finished, along with your student number, gitName, and collaboration url. ie: "Jeff I uploaded my work to gitHub. My student number is A00XXXXXX and my gitName is YYYYY, my git collaboration url is: ZZZZZZZZZZZZZZZ"
8. That's it. Invite me as a collaborator if you haven't yet, and let's mark it!

4 Grading

This lab 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. (3 points) Successfully implement the requirements exactly as described in this document
3. (3 points) Successfully test your code. Do you require unit tests?
4. (2 points) Write code that is consistently commented and formatted correctly using good variable names, efficient design choices, atomic functions, constants instead of magic numbers, etc.