COMP 3522 Lab 6

Object Oriented Programming in C++

Due Nov 1st 11:59PM

1 Instructions

This week you will create a Dictionary that reads a text file filled with words and definitions. The program will prompt the user to print the contents of the dictionary to the screen, or enter a new word and definition. Any non duplicate words and their definition will be added to the text file

2 Set up your lab

Start by creating a new project:

1. Create a new project in CLion. Let's call it Lab6. 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. 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 Lab6, 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.

1. Create a text file named dictionary.txt
   - (a) This dictionary text file will contain the words and their associated definitions
   - (b) You may store the words and definitions in this text file however you choose. Some examples may be ”word-definition” or word *newline* definition.
2. Declare and define a dictionary class
   - (a) Use a STL map to store your words and definitions
   - (b) Use an iterator to print the contents of the STL map
   - (c) Include any additional functions you need to achieve the requirements below
3. Main function
   - (a) Create an instance of the dictionary class and pass it the dictionary text file
   - (b) Extract the words and definitions from dictionary.txt and store them in a STL map within your dictionary class
   - (c) Create a ”menu” that provides the user with these options:
     - i. 1 - Print dictionary
     - ii. 2 - Find word definition
     - iii. 3 - Enter new word and definition
     - iv. 4 - Exit
   - (d) If the user inputs ’1’, all the words and definitions are printed out in the format ”word - definition for word”. The program then returns to the ”menu”.
   - (e) If the user inputs ’2’ they are prompted to enter a word to find in the dictionary. Wait for user input
   - (f) If the word exists, print out the definition and return to the menu
   - (g) If the word does not exist, print out ”the word doesn’t exist” and return to the menu
   - (h) If the user inputs ’3’, they are prompted to enter a word. Provide text telling the user to Enter a new word. Wait for user input.
   - (i) If the word already exists in the dictionary, tell the user the word exists and tell them to enter a new word. Wait for user input.

(j) If the word is new, tell them to enter a definition. Provide text telling the user to enter a definition. Wait for user input.

(k) After they have entered the definition, tell them their new word and definition has been added to the dictionary

(l) The program will now return to the "menu"

(m) If the user inputs '4' in the menu, the program will terminate

4. New words and definitions added by the user must be added to the STL map and saved in the dictionary.txt file
5. Include the dictionary.txt in your submission. Put it in the base directory of your project, not the debug build directory. Use "../dictionary.txt" to access your dictionary when it is in the base directory
6. Ensure all member variables are private or protected. Do not use global or public members to store instance data.
7. Ensure you are not duplicating code.
8. 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"
9. 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.