Lab 12: C++ Tricks

Nathan Jarus

July 12, 2017

Introduction

This lab will have you use some C++ features and things from the STL. Do not forget to compile with `-std=c++11` if you are using a C++ 11 feature!

Problem 1: A better phonebook

For this problem, you will create a C++ source file called `lookup.cpp`, as well as other `.cpp` and `.h` files as you desire.

Write a program that takes up to two command line arguments. The first argument is a filename for a telephone book (use `phonebook.txt` as an example). The second argument is a name to look up the number of; if no second argument is given, list the entire phonebook.

Your program should do a bit of error handling to ensure the user is using the program properly. If the user provides too many or two few command line arguments, your program should print out a helpful usage message. If the user provided an OK number of arguments, your program should check that the file opened properly, etc. Its good practice.

Use a `map<string, tuple<int, int, int> >` to store the contents of the phonebook as you read it from the file. You can use `make_tuple` to, well, make a tuple.

You may assume that the phonebook file will be formatted properly. Theres no need to check for a bunch of error conditions related to the structure of the phonebook file.

You may also assume that the names in the phonebook are all unique.

If the requested name is not present in the phonebook file, then act like we didnt search for a name at all. Just print out the contents of the phonebook.

Problem 2: Area Codes

For this problem, you will create a C++ source file called `area_codes.cpp`.

Write a program that takes one command line argument: the filename for a telephone book (see `phonebook.txt` for an example). The program will output a list of unique telephone area codes from the numbers in the file.

Again, your program should do a bit of error handling to ensure the user is using the program properly. If the user provides too many or two few command line arguments, your program should print out a helpful usage message. If the user provided an OK number of arguments, your program should check that the file opened properly, etc.

Use a `std::set` to store the area codes. (This is basically a map, but with just keys. Look it up on http://www.cplusplus.com/reference/set/set/!)

You can borrow the same phonebook-loading logic you wrote for Problem 1.

Epilogue

Please commit your code and push it to GitLab!