Lab 9: Sort using tally via STL map class

Monday, 9 April 2018

The map class of the standard template library can be used to sort a collection of entries. It is especially useful when there are many duplicates in the collection.

Visit your favorite website for documentation and research the functionality of the map class. Find out as much as you can about how it works behind the scenes, as well as the class functions available for you to use.

Lab Assignment:

1. You will generate all of the code for this assignment, so begin by creating a new project with your driver file named 'main.cpp.' Then add functions that allow you to read and write strings from/to a file.
2. Then, take the words that you read in from the example.txt that was provided in the repo and tally the strings in the file by inserting those strings into a map<string,int> object. Here, the string part of the map will be the individual words read in from the file, and the int portion of the map is the number of times the string part has been inserted in the map. (Hint: figure out what the statement ++tally[item]; does when tally is a map<string,int> and item is a string).

   Behind the scenes, the map container stores objects in a balanced binary search tree(research what it means for a tree to be balanced). The map iterator uses an inorder traversal to step through the tree objects, so the objects accessed by the iterator are encountered in sorted order. Think back to last lab and the three different traversal methods we discussed(pre/post/inorder).

3. Iterate through the tally map and write each string the correct number of times to the output file named output.txt. This means that the size of the output file will be the exact same as the input file but the words in output.txt will be sorted.

Turn In:

Turn in your main.cpp and output.txt files in the top level of your repo in your main branch. Please follow the file naming guidelines outlined in the instructions. This lab is due on the 15th of April 2018 at 11:59PM.

see next page for criteria

Criteria:

1. Read and write functions work as described. 5 points
2. Properly store words from input file into a map object. 10 points
3. Sorted words properly written to file using a map iterator. 5 points