**Objectives**

1. Illustrate use of the C++ STL map associative container.
2. Practice use of strings.
3. Introduce the concept of a filter.

**Background**

One of the major tools in the command line shells is the concept of a “filter”: a program that reads data from standard input (a.k.a., cin in C++), does something to that text, and then outputs the result to standard output (a.k.a., cout in C++). It’s a “filter” because it’s “filtering” the input.

For this lab, we will write a filter that reads text from standard input and then writes a frequency table to standard output. Nothing particularly earth-shattering about a frequency table: it’s just a table that shows the count of the number of words in a text.

This is a classic application of the dictionary data structure. Or as we know of it in C++: the map associative container.

**Instructions**

Create a new project in favorite IDE.

Let’s get things started by bringing in the required header files:

#include <string>

#include <map>

#include <iostream>

using namespace std;

So… what sort of map are we using. We want to keep track of the number of occurrences of each word in the file

We first need to introduce the concept of a type alias. It gets painful to have to enter map<string,int> when we need to declare a map. So… we have a way to do this using the using construct:

using FreqTable = map<string,int>;

Going forward, we use FreqTable as our data type.

We will declare prototypes for two functions that we will use to build the frequency table:

void buildFreqTable(FreqTable &);

void printTable(FreqTable &);

For the main function, just declare a variable for frequency table and rock and roll:

int main() {

FreqTable wordFreqTable;

buildFreqTable(wordFreqTable);

printTable(wordFreqTable);

return 0;

}

Let’s implement our helper functions.

We’ll build the frequency table by reading from cin until someone enters an “End-of-File” when they are done. We check to see if the word entered is or isn’t in the map. If so, then just increment the count and it not, then add it with count set to 1:

void buildFreqTable(FreqTable &aFreqTable) {

string aWord = "";

while (cin >> aWord) {

auto keyiter = aFreqTable.find(aWord);

if (keyiter != aFreqTable.end()) {

aFreqTable[aWord]++;

}

else {

aFreqTable[aWord] = 1;

}

}

}

Thing to remember is that a dictionary is a container like a list, stack, or queue and we can use iterators on dictionary just like we did with those containers. The map find() method will return an iterator that is positioned where it found an entry or the “end” of the list of keys. So, we use the find function, and if it’s not at the end, increment that entry otherwise add it to the list.

Now for printing the frequency table:

void printTable(FreqTable &aFreqTable) {

cout << "Word frequency table:" << endl;

for (const auto &pair : aFreqTable) {

cout << pair.first << " : " << pair.second << endl;

}

}

Remember that entries in a map are stored in “key-values” pairs, so the loop here gets a constant reference to each pair in the dictionary. We use the pair class first and second properties to get the key and value.

Build and run your program and then open your output file in a text editor.

Did you see the problem with this code? What about lowercase and uppercase? Or punctuation? You will have a homework problem based on this code where you will need to solve this problem.