

Adam Camerer
Mukund Telukunta
CS1585
07/23/2022

Project 2 Results

Project B: C++ Standard Library

Approach

Using `cstdlib`, I relied primarily on `map` for storage, `cctype` for filtering words, `fstream` for reading files, and `string` for storing the 'key' of the string, and `iostream` to output the results on the console.

For the project, I used a `map` to store occurrences of unique words, and then outputted them using an iterator. Since iterators are more generalizable, and I wanted to apply the knowledge I had learned, I used them to go through the `map` and output the results.

To not have to scroll as much, I outputted 3 key-value pairs on each line, separated by two tabs.

```
//instantiate iterator for map
template<typename K, typename V>
using MapIterator = typename std::map<K,V>::const_iterator;

//Print map of words (when finished)
template<typename K, typename V>
void PrintMap(const std::map<K,V>& m)
{
    int i = 0;
    for (MapIterator<K, V> iter = m.begin(); iter != m.end(); iter++)
    {
        std::cout << "Key: " << iter->first << " " << "Values: " << iter->second;
        if(i%3==2) std::cout<<std::endl;
        else std::cout<<"\t\t";
        i++;
    }
}
```

To process the words, I first created a `fileStream` object for the input file, and then iterated through each 'string' in the file (characters non separated by whitespace), which made it possible to select individual words

```

int main(void)
{
    //file
    static const char* fileName = "4507.txt.utf-8";

    // Will store the word and count.
    map<string, unsigned int> wordsCount;

    {
        // Begin reading from file:
        ifstream fileStream(fileName);

        // Check if file opened
        if (fileStream.is_open())
            while (fileStream.good())
            {
                // Store the next word in the file in a local variable.
                string word;
                fileStream >> word;
            }
    }
}

```

Once a 'word' was selected, I would begin processing it to remove 'basic' punctuation, i.e. not double hyphens, since that required a more specialized technique. I ignored web addresses (https), since they are usually unique (web addresses aren't even real words). It would be more difficult to add the : and / characters than ignore them altogether.

Using ispunct from ctype, I got a larger pool of characters to remove. Then, I whittled down the pool of characters, selecting all at the beginning or end of the word, or non hyphens between the beginning and end. Then, I removed the characters, and continued on.

After this, I then relooped over the loop again, converting all uppercase characters to lowercase. If I put the punctuation removal and lowercase in one loop, it would be more confusing.

Removing double hyphens was more complex. Since 'basic' punctuation was already removed, it made the job easier. To remove them, two words were created, so each would need to be added to the tally separately, and not as one big glob. To do this, I selected only words with double hyphens, and then took the substring before the double hyphen, added it to the map, and then only kept the part after the double hyphen.

```

74         if( !(word.find(webAddress) != std::string::npos) )
75         for(int i = 0; i<len; i++)
76         {
77             //delete punctuation (except in special cases)
78
79             if( ispunct(word[i]) && ( i==0 || i == len-1 || (word[i] != '-' ) ) )
80             {
81                 word.erase(i,1);
82                 len = word.size();
83                 i--;
84             }
85         }
86
87         //convert uppercase letters to lowercase
88         for(int i = 0; i<len; i++)
89         {
90             word[i] = tolower(word[i]);
91         }
92
93
94         //Deal with double hyphen
95         for(int i = 0; i<len; i++)
96         {
97             if( (word[i]=='-' && word[i+1]=='-') )
98             {
99
100                 string lower = word.substr(0,i);
101                 addToMap(wordsCount, lower);
102                 word.erase(0, i+2);
103                 len -= i+2;
104                 i=0;
105             }
106
107         }

```

After this, I added the filtered word to the map. Once all the words were filtered and mapped, I printed the map. If the file wouldn't open, an error message would be printed. After this, the program ended.

```
        //add (remaining) word to map
        addToMap(wordsCount, word);
    }
    else //File not openable
    {
        cerr << "Couldn't open the file." << endl;
        return EXIT_FAILURE;
    }

    // Print the words map.
    PrintMap(wordsCount);
}

return EXIT_SUCCESS;
}
```

Results

```

Key: volunteers Values: 6      Key: wage Values: 1      Key: wages Values: 2
Key: wait Values: 1           Key: waits Values: 1     Key: wake Values: 1
Key: waking Values: 1         Key: walk Values: 1      Key: walking Values: 1
Key: walks Values: 1          Key: walls Values: 1     Key: wander Values: 3
Key: want Values: 1           Key: wants Values: 1     Key: warranties Values: 3
Key: warranty Values: 2       Key: was Values: 6       Key: wash Values: 1
Key: washed Values: 1         Key: watch Values: 1     Key: watchfulness Values: 1
Key: wavering Values: 1       Key: waves Values: 1     Key: way Values: 6
Key: ways Values: 1           Key: we Values: 16       Key: weak Values: 5
Key: weaker Values: 3         Key: weakest Values: 1   Key: weakly Values: 1
Key: weakness Values: 12      Key: wealth Values: 2    Key: weapons Values: 1
Key: weave Values: 1          Key: web Values: 6       Key: weed-seeds Values: 1
Key: weeding Values: 1        Key: weeds Values: 1     Key: well Values: 4
Key: well-directed Values: 1  Key: well-poised Values: 1 Key: were Values: 1
Key: west Values: 1           Key: what Values: 12     Key: whatever Values: 3
Key: whatsoever Values: 3     Key: wheel Values: 1     Key: when Values: 13
Key: whenever Values: 1       Key: where Values: 6     Key: wherever Values: 1
Key: whether Values: 5        Key: which Values: 80    Key: whichever Values: 1
Key: while Values: 5          Key: whilst Values: 1    Key: whims Values: 1
Key: whine Values: 1          Key: who Values: 47      Key: whole Values: 2
Key: wholesome Values: 1      Key: whom Values: 1      Key: whose Values: 5
Key: wide Values: 2           Key: wider Values: 3     Key: widest Values: 2
Key: wielder Values: 1        Key: wields Values: 1    Key: wild Values: 1
Key: will Values: 59          Key: will-o-the-wisps Values: 1 Key: willing Values: 2
Key: wills Values: 2          Key: winds Values: 1     Key: wisdom Values: 3
Key: wise Values: 3           Key: wisely Values: 2    Key: wish Values: 2
Key: wishes Values: 2         Key: with Values: 86     Key: within Values: 11
Key: without Values: 15       Key: woman Values: 1     Key: women Values: 2
Key: word Values: 2           Key: words Values: 2     Key: work Values: 47
Key: working Values: 1        Key: workpeople Values: 1 Key: works Values: 32
Key: workshop Values: 2       Key: world Values: 17    Key: worldly Values: 3
Key: worlds Values: 1         Key: worldwide Values: 1 Key: worries Values: 1
Key: worry Values: 1          Key: would Values: 10    Key: woven Values: 1
Key: wretchedly Values: 1     Key: wretchedness Values: 7 Key: wrinkles Values: 2
Key: writing Values: 3         Key: written Values: 3   Key: wrong Values: 4
Key: wrought Values: 1        Key: wwwgutenbergorg Values: 3 Key: ye Values: 2
Key: yea Values: 1           Key: years Values: 3     Key: yes Values: 1
Key: yet Values: 7            Key: york Values: 1      Key: you Values: 103
Key: your Values: 46          Key: yourself Values: 2  Key: youth Values: 3
Key: youthful Values: 1       Key: youthfulness Values: 1 Key: zenith Values: 1
Key: the Values: 1
Total unique words: 2284
make clean
make[1]: Entering directory '/home/user/DataStructures/Project2/2022-SS-303-Project2-ajc3xc'
make[1]: Leaving directory '/home/user/DataStructures/Project2/2022-SS-303-Project2-ajc3xc'
user@hostname:~/DataStructures/Project2/2022-SS-303-Project2-ajc3xc$

```

Compiler (Makefile)

```
.PHONY: clean
```

```
clean:
```

```
    -@rm -f wordMap
```

```
    -@rm -f *.o
```

```
    -@rm -f *.aux
```

```
    -@rm -f *.log
```

```
    -@rm -f *.dvi
```

```
program:
```

```
    g++ wordMap.cpp -o wordMap
```

```
    ./wordMap
```

```
    make clean
```

```
document:
```

```
    pdflatex document.tex
```