Benjamin Horn

C++

Lab 05 Intensive String Work

M07306790

//Benjamin Horn

// Laboratory #5: Strings and Arrays

//

// We will be exploring string arrays in this problem. See main()

// for examples of what we are going to do.

//

// You only need to complete the functions listed at the end of this file

// and add a few output statements to main()

#include <iostream>

#include <iomanip>

#include <string>

#include <fstream>

#include <cstdlib>

using namespace std;

// Prototypes

// The ReadStringsFile is complete

bool ReadStringsFile(string fileName, string ArrayOfStrings[],int &number\_of\_lines);

void PrintStringArray(int number\_of\_lines, string ArrayOfStrings[]);

int CountWordsInArray(int number\_of\_lines, string ArrayOfStrings[] );

int CountWordsInString(string InputString );

int CountCharactersInString(string InputString);

int CountCharactersInArray(int number\_of\_lines, string ArrayOfStrings[]);

int CountUpperCaseCharactersInArray(int number\_of\_lines, string ArrayOfStrings[]);

int CountLowerCaseCharactersInArray(int number\_of\_lines, string ArrayOfStrings[]);

int CountPunctuationCharactersInArray(int number\_of\_lines, string ArrayOfStrings[]);

string LongestLineOfTextInArray(int number\_of\_lines, string ArrayOfStrings[]);

string ShortestLineOfTextInArray(int number\_of\_lines, string ArrayOfStrings[]);

// main() is almost complete please add proper cout statements to print longest string,

// smallest string, etc..

// See Sample Output

// - You may need to modify the filename assignment below to point to your test file

int main()

{

int number\_of\_lines = 0;

int word\_count;

int character\_count;

int uppercase\_letter\_count;

int lowercase\_letter\_count;

int punctuation\_count;

string filename="testfile.txt"; // Modify this to point to your test file

string ArrayOfStrings[500]; // This means we can NOT have files with more than 500 lines in them.

string longest\_string;

string shortest\_string;

// Could use getline and cout to prompt and retrieve the filename from the command window

// instead of hard coding it above.

// Could replace the following with a loop to retry a different filename on a failure.

if((ReadStringsFile(filename, ArrayOfStrings, number\_of\_lines)) != true)

{

cout << "Can not Read File: " << filename << endl;

exit(-1);

}

cout<<"The File "<< filename<<" was evaluated. It has "<<number\_of\_lines<< " lines"<<endl;

character\_count = CountCharactersInArray(number\_of\_lines, ArrayOfStrings);

word\_count = CountWordsInArray(number\_of\_lines, ArrayOfStrings);

longest\_string = LongestLineOfTextInArray(number\_of\_lines, ArrayOfStrings);

shortest\_string = ShortestLineOfTextInArray(number\_of\_lines, ArrayOfStrings);

uppercase\_letter\_count = CountUpperCaseCharactersInArray(number\_of\_lines, ArrayOfStrings);

lowercase\_letter\_count = CountLowerCaseCharactersInArray(number\_of\_lines, ArrayOfStrings);

punctuation\_count = CountPunctuationCharactersInArray(number\_of\_lines, ArrayOfStrings);

cout<<" "<<endl<<" " <<endl;

cout<<"character\_count: "<<character\_count<<endl;

cout<<" "<<endl<<" " <<endl;

cout<<"word\_count: "<<word\_count<<endl;

cout<<" "<<endl<<" " <<endl;

cout<<"longest\_string: "<<longest\_string<<endl;

cout<<" "<<endl<<" " <<endl;

cout<<"shortest\_string: "<<shortest\_string<<endl;

cout<<" "<<endl<<" " <<endl;

cout<<"uppercase\_letter\_count: "<<uppercase\_letter\_count<<endl;

cout<<" "<<endl<<" " <<endl;

cout<<"lowercase\_letter\_count: "<<lowercase\_letter\_count<<endl;

cout<<" "<<endl<<" " <<endl;

cout<<"punctuation\_count: "<<punctuation\_count<<endl;

// Please call the proper function to output the array

cout<<" "<<endl<<" " <<endl;

PrintStringArray( number\_of\_lines, ArrayOfStrings);

return 0;

}

// The "incomplete" functions below are sometimes called "stub" functions. They are used

// as place holders while testing other (possibly) more important parts of the code.

void PrintStringArray(int number\_of\_lines, string ArrayOfStrings[])/////////DONE

{

cout<<" "<<endl<<" " <<endl;

for ( int i = 0; i < number\_of\_lines; i++)

cout<< ArrayOfStrings[i]<<endl;

cout<<" "<<endl<<" " <<endl;

}

// This function Counts all characters in Array.

// Please loop through array and use the function

// CountCharactersInString to help calculate the number of characters.

int CountCharactersInArray(int number\_of\_lines, string ArrayOfStrings[])//////////done

{

int numberOfChars = 0;

for ( int i = 0; i < number\_of\_lines; i++)

{

int amount= ArrayOfStrings[i].length();

numberOfChars+= amount;

}

return numberOfChars;

}

// This function counts "words" in the array.

// Please loop through array and use the function

// CountWordsInString to finish this function

int CountWordsInArray(int number\_of\_lines, string ArrayOfStrings[] )/////////////done

{

int word = 0;

for ( int i = 0; i < number\_of\_lines; i++)

{

word += CountWordsInString( ArrayOfStrings[i] );

}

return word;

}

// This function takes a string and counts the "words" in that string

//

// This function my require some research but a crude approxitamation

// of counting words can be attempted using isspace and the c\_str()

// function.

//

// Example Input String:

// String1 String2 String3

// would result in a word count of three (3)

//

// I do not want you to spend a lot of time on this.

int CountWordsInString(string InputString )///////////////done

{

int word = 0;

for ( int i = 0; i < InputString.length()-1; i++)

{

if ((isspace(InputString[i]) || (( ispunct(InputString[i])) && (isspace(InputString[i+1])))))

word ++;

}

return word;

}

// This function returns the count of the characters in a string

// This is a one line function..

// Hint: You will want to look at the length() function

int CountCharactersInString(string InputString)

{

int word = 0;

for ( int i = 0; i < InputString.length()-1; i++)

{

if ((isspace(InputString[i]) || (( ispunct(InputString[i])) && (isspace(InputString[i+1])))))

word ++;

}

return word;

}

// This function counts the number of upper case letters in the Array/File

// Look at the isupper() function to make this work

int CountUpperCaseCharactersInArray(int number\_of\_lines, string ArrayOfStrings[]) ///////////done

{

int count=0;

for ( int i= 0; i < number\_of\_lines; i++)

{

for (int j = 0; j < ArrayOfStrings[i].length(); j++)

{

if (isupper(ArrayOfStrings[i][j]))

count++;

}

}

return count;

}

// This function counts the number of lower case letters in the Array/File

// Look at the islower() function to make this work

int CountLowerCaseCharactersInArray(int number\_of\_lines, string ArrayOfStrings[]) //////////done

{

int count=0;

for ( int i= 0; i < number\_of\_lines; i++)

{

for (int j = 0; j < ArrayOfStrings[i].length(); j++)

{

if (islower(ArrayOfStrings[i][j]))

count++;

}

}

return count;

}

// This function counts the number of punctuation characters in the Array/File

// Look at the ispunct() function to make this work

int CountPunctuationCharactersInArray(int number\_of\_lines, string ArrayOfStrings[])////////DONE

{

int count=0;

for ( int i= 0; i < number\_of\_lines; i++)

{

for (int j = 0; j < ArrayOfStrings[i].length(); j++)

{

if (ispunct(ArrayOfStrings[i][j]))

count++;

}

}

return count;

}

// This function finds the Longest line of text in the Array

// You have to use the length() function on strings

string LongestLineOfTextInArray(int number\_of\_lines, string ArrayOfStrings[])////////////// done

{

int newest = 0;

int location;

for ( int i = 0; i < number\_of\_lines; i++)

{

if (ArrayOfStrings[i].length() > newest)

{

newest = ArrayOfStrings[i].length();

location = i;

}

}

return ArrayOfStrings[location];

}

// This function finds the Shortest line of text in the Array

// This function needs to be completed.

string ShortestLineOfTextInArray(int number\_of\_lines, string ArrayOfStrings[])///////done

{

int newest = 1000000000;

int location;

for ( int i = 0; i < number\_of\_lines; i++)

{

if (ArrayOfStrings[i].length() < newest)

{

newest = ArrayOfStrings[i].length();

location = i;

}

}

return ArrayOfStrings[location];

}

///////////done

// This function is complete and it uses material that we did not cover

// so please do not try to modify it.

bool ReadStringsFile(string fileName, string ArrayOfStrings[],int &number\_of\_lines)

{

string input\_line;

ifstream infile(fileName.c\_str()); // This might be useful for you.

if(!infile)

{

return false;

}

number\_of\_lines = 0;

while(getline(infile, input\_line))

{

ArrayOfStrings[number\_of\_lines] = input\_line;

number\_of\_lines = number\_of\_lines + 1;

}

return true;

}

The File testfile.txt was evaluated. It has 21 lines

character\_count: 2723

word\_count: 435

longest\_string: Ornare suspendisse. Est nostra Nisl augue dictumst interdum bi

bendum sociosqu libero facilisis Pulvinar tellus sem inceptos tristique nec vita

e ipsum consectetuer fames justo neque gravida parturient mi. Est. Ultrices nasc

etur torquent sociis. Facilisis nullam per. Pretium quisque Netus. Mauris turpis

pulvinar ligula quis dignissim congue elit.

shortest\_string: brakes are defective.

uppercase\_letter\_count: 58

lowercase\_letter\_count: 2210

punctuation\_count: 63

This file is randomly generated from a combination of the UNIX fortune command

and one of the Lorem Ipsum random filler text generators. Most if the text

is completely made up.

You will not be elected to public office this year.

The wiki is unfortunately living a life a bit in the shadows but it contains lot

s of good stuff

The fashion wears out more apparel than the man.

Unfortunately the partitioner in the install is a bit "grumpy"

It is, as far as I know safe, but does not always let you do what you want

computers are really hard to understand - one user has no problems and another a

ll the problems in the world with the same hardware....

You never hesitate to tackle the most difficult problems.

Your depth of comprehension may tend to make you lax in worldly ways.

You will stop at nothing to reach your objective, but only because your

brakes are defective.

Neque. Vitae. Ut fames aenean. Mi id habitant. Rhoncus sem eu eleifend et adipis

cing eleifend, mus tellus hac integer conubia.

Proin ridiculus, tincidunt sociosqu aptent faucibus sociis molestie purus leo pe

r sociis condimentum, ipsum hac nisl praesent. Dictum nullam metus mus blandit d

uis adipiscing elit habitant, est placerat dignissim, quam primis curabitur sagi

ttis quam quisque cursus odio aenean sem vel viverra.

At hendrerit ultrices condimentum pharetra velit urna feugiat augue leo tempor S

enectus pretium quis lectus. Libero quisque venenatis inceptos libero iaculis du

i ullamcorper risus penatibus scelerisque, lectus et. Viverra interdum arcu.

Potenti. Fermentum hendrerit imperdiet id ornare convallis morbi lacinia lacinia

blandit in. Per, pretium laoreet cum blandit. Senectus phasellus senectus. Ince

ptos molestie lobortis nostra viverra rutrum proin dis.

Aenean at sodales metus pellentesque turpis ad libero leo dignissim nunc. Per cu

rsus non vel mattis auctor sem gravida rhoncus in ornare elementum viverra conva

llis interdum pede sit dignissim sollicitudin praesent ridiculus.

Tortor dignissim, imperdiet gravida magnis cubilia cubilia torquent phasellus po

suere. Amet hymenaeos. Eleifend suscipit viverra eu adipiscing molestie porttito

r consectetuer platea metus. Amet id euismod. Curabitur lectus cubilia elementum

rhoncus nec etiam vehicula class conubia.

Ornare suspendisse. Est nostra Nisl augue dictumst interdum bibendum sociosqu li

bero facilisis Pulvinar tellus sem inceptos tristique nec vitae ipsum consectetu

er fames justo neque gravida parturient mi. Est. Ultrices nascetur torquent soci

is. Facilisis nullam per. Pretium quisque Netus. Mauris turpis pulvinar ligula q

uis dignissim congue elit.

Mus vel nunc pellentesque purus. Neque feugiat suscipit, ipsum a nisi. Mi. Matti

s. Torquent commodo. Pede. Taciti sed parturient Ultrices, dui.