

1 Introduction

Each exercise should be its own separate project.

Remember to follow the coding and documentation standards for the class listed on the MyClasses pages.

When you are ready to submit your work create a folder called Homework05 in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire Homework05 folder up into a single zip file and submit it.

2 Exercises

1. This exercise you will be simulating die rolls and storing the entire simulation into a single one-dimensional array. You will then be using that array to do some calculations in functions you will write. The main you must use is below. The program will ask the user for the number of rolls in the simulation, and then the number of sides to the dice. It will then create the array, populate it with the entire simulation, find the number of each roll in the simulation, the longest run length of each roll, and the number of straights.

```
int main() {
    srand(time(0));

    int size=getPositiveInteger("Input the number of rolls on the simulation: ");
    int diesides=getPositiveInteger("Input the number of sides to the die: ");

    int A[size];
    PopulateArray(A, size, diesides);

    for (int i = 1; i <= diesides; i++)
        cout << "Count of " << i << ": " << countValues(A, size, i) << endl;

    for (int i = 1; i <= diesides; i++)
        cout << "Max run of " << i << ": " << MaxRunOf(A, size, i) << endl;

    cout << "Count of Straights: " << countStraights(A, size, diesides) << endl;

    return 0;
}
```

A run of the program with 1000000 rolls is below.

```
Input the number of rolls on the simulation: 1000000
Input the number of sides to the die: 6
Count of 1: 167195
Count of 2: 166904
Count of 3: 167158
Count of 4: 166483
Count of 5: 166292
Count of 6: 165968
Max run of 1: 8
```

```
Max run of 2: 7
Max run of 3: 7
Max run of 4: 7
Max run of 5: 11
Max run of 6: 8
Count of Straights: 18
```

The functions do the following:

- (a) `dieRoll` takes in the number of sides to the die and returns a pseudo-random number between 1 and the number of sides.
- (b) `PopulateArray` takes in the array, its size, and the number of sides to the die and populates the array with pseudo-random numbers between 1 and the number of sides.
- (c) `countValues` takes in the array, its size, and the face value to the die and returns the number of occurrences of that face value in the simulation.
- (d) `MaxRunOf` takes in the array, its size, and the face value to the die and returns the maximum run of that roll in the simulation.
- (e) `countStraights` takes in the array, its size, and number of sides to the die and returns the number of straights in the simulation. A straight is a run of consecutive rolls from 1 to the number of sides of the die. So if the die has 6 sides a run will be 1, 2, 3, 4, 5, 6 in a row in the simulation.
- (f) `getPositiveInteger` takes in a string that will be a prompt for the user and return the value the user typed in if the value was positive and if the user types in 0 or a negative number the function will tell them that it was an invalid value and ask for input again until the user types in a positive integer.

Note that the `countValues`, `MaxRunOf`, and `countStraights` are to also make sure the array cannot be altered.

2. This exercise is to take numeric input from a file, load it into an array and then do some calculations on that data. The `cabdata.txt` file contains over half a million decimal numbers each on their own line. These numbers are real data, they are the distances (in miles) for every cab fair of the famous Yellow Cab Company of New York city for the month of June 2022. You will write a program that will bring in this data and store it in an array, then use the array to gather some statistics on the data. You are going to write this in general so that you could load in any file with this type of data. Hence you need to make this work with different file sizes. The easiest way to do this is to go through the file once to see how many numbers are in the file, then create an array of that size, reset the file and then read in the data to the array. The statistics you will print out to the console screen are the average trip length, the smallest trip distance, the longest trip distance, the number of trips over 10 miles, and the number of trips under a quarter mile.
3. Write a program that will take two filenames from the user. The first filename is for a text source file, i.e. a file with a bunch of words in it. The second filename is for a file

that will contain the results of the character analysis. Have the program read the source text file and determine the relative frequencies of all the alphabetic characters, in a case-insensitive way. The relative frequencies are of course the number of occurrences of the character in the text divided by the number of alphabetic characters in the text. Make sure you do not count any spaces or punctuation characters. Have the program print out to the console a chart of characters and relative frequencies. And have the program save a copy of these results to the file the user selected. For the charts, use a tab character between the character and its relative frequency. This will make it easier to copy and paste the results to a spreadsheet which you will be doing later in the lab. For example, if you run the program on the single paragraph test file I included you will get the following results.

```
Input the input filename: TestDoc001.txt
Input the output filename: TestResults.txt
```

```
A 0.0674419
B 0.0162791
C 0.0418605
D 0.0302326
E 0.137209
F 0.0116279
G 0.0186047
H 0.055814
I 0.0534884
J 0
K 0.00465116
L 0.0302326
M 0.0232558
N 0.072093
O 0.0767442
P 0.027907
Q 0
R 0.0744186
S 0.072093
T 0.104651
U 0.0302326
V 0.00465116
W 0.0139535
X 0
Y 0.0325581
Z 0
```

And the `TestResults.txt` file would have the following contents.

```
A 0.0674419
B 0.0162791
C 0.0418605
D 0.0302326
E 0.137209
F 0.0116279
G 0.0186047
H 0.055814
I 0.0534884
J 0
K 0.00465116
L 0.0302326
M 0.0232558
```

```
N 0.072093
O 0.0767442
P 0.027907
Q 0
R 0.0744186
S 0.072093
T 0.104651
U 0.0302326
V 0.00465116
W 0.0139535
X 0
Y 0.0325581
Z 0
```

Now it is time to analyze the real files. In the zip file for this homework there are five files, HGGT1.txt, HGGT2.txt, Holmes.txt, Shakespeare.txt, and TestDoc001.txt. The first two are the *The Hitchhiker's Guide to the Galaxy* trilogy by Douglas Adams. Holmes.txt contains the complete *Sherlock Holmes* novels and short stories by Arthur Conan Doyle. Shakespeare.txt contains the complete works of William Shakespeare. Finally the TestDoc001.txt file is just a short test file.

Run your program on the *The Hitchhiker's Guide to the Galaxy*, *Sherlock Holmes*, and the works of Shakespeare. Copy and paste the results into a spreadsheet. You probably already have either Microsoft Excel or LibreOffice Calc on your computer. If not, LibreOffice is a suite of office programs containing, a word processor, spreadsheet, presentation package and more, and it is all free. You can also use several online spreadsheets as well.

Once all three have been run and loaded into the spreadsheet, have the first column be the characters then the next three columns be the relative frequencies of the characters for the three separate works. Now make a bar chart of the three sets of data with the characters along the x -axis.

- (a) What similarities and differences do you see between the three authors and time periods?
- (b) Considering that the majority of the works of Shakespeare are plays and the other works are novels, what does the character analysis say about character usage between the two different types of literature?
- (c) Shakespeare did most of his work in the latter part of the 1500's into the early part of the 1600's, Doyle wrote at the end of the 1800's into the early 1900's, and Adams work was in the 1970's. What does your graph imply about the character usage in the English language over this 400 year period of time?

Once you have the document with chart image and answers to the above questions completed. Export it to a PDF and upload it with the program.

4. This is an extension of the previous exercise where we will now look at word frequencies and not just character frequencies. The top 25 most used words in the English language in order and all lowercase are listed below.

the of to and a in that is i it for as with you on was be he this not have are at if but
Revise your program to find the occurrences of each of these words in an input text
file. Be careful to remove the punctuation around the words. For example, one line of
the Sherlock Holmes work is

“I should like to meet him,”

If you break this up over white-space you will have

“I

should

like

to

meet

him,”

where you really want

I

should

like

to

meet

him

Also, you will want to use an equality to check for the word occurrences since you do
not want to count a “the” if the word is “there”. If you run the program on my sample
document you should get the following.

```
Input the input filename: TestDoc001.txt
Input the output filename: test
```

```
THE 0.0963855
OF 0.0120482
TO 0.0361446
AND 0.0120482
A 0.0120482
IN 0.0361446
THAT 0.0240964
IS 0.0240964
I 0
IT 0
FOR 0
AS 0
WITH 0.0120482
YOU 0
ON 0
WAS 0
BE 0
HE 0
THIS 0
NOT 0.0120482
```

```
HAVE 0
ARE 0.060241
AT 0
IF 0
BUT 0.0240964
```

As before, run the program on all three works, create a bar chart of the relative frequencies of each of the 25 words and answer the following questions.

- (a) What similarities and differences do you see between the three authors and time periods?
- (b) Considering that the majority of the works of Shakespeare are plays and the other works are novels, what does the analysis say about word usage between the two different types of literature?
- (c) Shakespeare did most of his work in the latter part of the 1500's into the early part of the 1600's, Doyle wrote at the end of the 1800's into the early 1900's, and Adams work was in the 1970's. What does your graph imply about the word usage in the English language over this 400 year period of time?