**CMPE 50 – Spring 2021**

**Lab #4 – File I/O and Formatting.**

*For each of the exercises, you need to write an algorithm and pseudo-code, in a comment section of the source code (.cpp). The algorithm and pseudo-code can cover only the key portion of the solution. There is no need to write algorithms for trivial code. Some examples of algorithms are given at the end of this assignment.*

**Exercise 1 (Arrays and File I/O):**

Write a program that takes its input from a file of characters. This file has a single character on each line. This program is supposed to read each character from the file, storing each char read into a char array. Once the whole file is read, output the array to a new file (can be named as LabOutput4-1.txt) modifying all the characters to upper case and stripping all whitespaces and punctuations including dots, Exclamation marks, Question marks and newlines. Use file error checking while performing file operations. **Sample data files are available from Canvas->Files->Lab 4 sample files** (LabInput4-1.txt).

**Exercise 2 (Arrays and File I/O)**

Write a program that reads in a list of integers into an array with base type int. Provide the facility to either read this array from the keyboard or from a file, at the user’s option. If the user chooses file input, the program should request a file name. You may assume that there are fewer than 50 entries in the array. Your program determines how many entries there are. The output is to be a 2-column list. The first column is a list of the distinct array elements; the second column is the count of the number of occurrences of each element. You need to output the result in a file and on the screen.

The list should be *sorted* on entries in the first column, largest to smallest. Please use the input file LabInput4-2.txt or allow the user to input from the keyboard.

Example:

For example, for the input

-3 4 1 1 3 4

the output should be

N Count

4 2

3 1

1 2

-3 1

Process:

This assignment consists of multiple components.

If you are unsure how to start, you may simplify and do a basic implementation first, and then add the enhancements (of course feel free to do everything at once!):

* Input from the keyboard vs. reading from a file
  + You can start with keyboard input
  + When reading from file, hardcode the file name
  + Ask the user for input (file name)
  + Have the user select which option in the beginning
* Output the two arrays in two columns
  + Output unsorted
  + Sort first, then output (remember, you may use a simple algorithm as selection sort to sort your array)
  + Add formatting (i.e. make columns look nice)
* Method of implementation:
* First read and print all the elements line by line.
* Extend the program to count occurrences (You can use an extra array to store the unique digits inside a list. Then compare the individual element from the unique list with the original list to find out the occurrences.) and print using formatting commands

ifstream ifs;

If you want to find the file using relative path, you need to know where you are running your code. In Eclipse, it is where your project is. Then the input file relative path is starting from the project directory. It’s easier to copy the LabInput4-2.txt file over to the project directory. If so,

ifs.open(“LabInput4-2.txt”);

Hint, to find out where your code is running at, use

ofstream ofs(“TestOut.txt”);

A file names TestOut.txt will be created at where your code is running.

**Exercise 3:**

Write a program that merges the numbers in two files and writes all the numbers into a third file. Your program takes input from two different files and writes its output to a third file. Each input file contains a list of numbers of type int in sorted order from the smallest to the largest. After the program is run, the output file will contain all the numbers in the two input files in one longer list in sorted order from largest to smallest in the format given below. Your program should define a function that is called with the two input-file streams and the output-file stream as three arguments. Please use the following two input files given in the Assignment Lab 4: LabInput4-3.txt and LabInput4-4.txt.

## Algorithm/Pseudo-code for Exercise 1

1. ifstream ifs;
2. char iarray[1000];
3. open the input file and check for error, i.e.

ifs.open(“LabInput4-1.txt”);

if (ifs.fail())

{

// file open failed. Probably the file does not exist, or

// the file is not in the correct location.

// To find out where the correction location is, you can

// create an output file and open it and check where the

// output file is. That is the place where input file should be.

Print some message to indicate the error.

exit(1); // include <cstdlib>

}

1. int count = 0;

while (ifs >> iarray[count++])

{

// empty body

// This code fragment reads the integers in the file ifs to the integer

// array iarray. Count is the number of integers read.

}

1. Convert each character in iarray[] to uppercase using toupper() and exclude ‘.’ (period), ? (question mark) and ‘!’ (exclamation) and output the array to a new file. (Hint: use ofstream to output the array, which is very similar to ifstream)

## Algorithm/Pseudo-code for Exercise 2

1. You need three integer arrays to solve this exercise: inputArray, uniqueArray, countArray. An integer variable “size” is used to keep track of

how many elements are in inputArray.

countArray is parallel array that has index moves along with uniqueArray

inputArray:

-3 4 1 1 4 3

uniqueArray:

-3 1 3 4

countArray:

1 2 1 2

1. Read the integers into inputArray. The number of elements is saved in variable “size”.
2. Sort the input array in ascending order.
3. Initialize the elements in countArray to be zero.
4. for (int i = 0; i < size; i++) // iterate through inputArray

{

for each element in inputArray, check if it is there in uniqueArray

already, if so, increment the countArray by 1

if not, copy the element to uniqueArray and set countArray to be 1

for(int j = 0; j < count; j++) {

…

}

}

1. output in the desired format

## Algorithm/Pseudo-code for Exercise 3

1. Open the two input files, check for file open errors
2. Declare an integer array that is large enough to hold the numbers from both files.
3. Read the first file into the integer array.
4. Read the second file into the integer array. Now the integer array contains numbers from both files.
5. Sort the integer array (using the selection sort from the textbook.)

ifstream ifs1(“LabInput4-3.txt”);

ifstream ifs2(“LabInput4-4.txt”);

// check fail()

int size = 0;

int inputArray[100];

while(ifs1 >> inputArray[size++])

{

}

// size--; ??

while(ifs2 >> inputArray[size++])

{

}