CS-212: LAB 2 – Arrays and File Handling

Learning Objective:

- Creating variables and passing them into functions (pass by value and by pointer)
- Using input and output data files
- Performing error checking on data files
- Passing arrays and FILE pointers into functions
- Assign data to an array
- Perform calculations on an array
- Printing data from an array to an output file
- Capture your output in a text file

This lab has three parts:

- Part-A: Setting up your header file
- Part-B: Converting header functions to use a file pointer
- Part-C: Writing all other functions for this lab
- There is an extra credit worth up to 10%

Lab 2 Setup:

1. Open a new terminal window to SSH into your account on the [bingweb.binghamton.com] server. When logged in, make sure you are in your home directory by typing the following command if you are not already there. The "~" is just a shortcut symbol that means home directory:
   cd ~

2. Change your current directory to the "CS212" directory by typing:
   cd CS212

3. Confirm that you are in the "CS212" directory by typing:
   pwd

4. Create a directory for Lab2.
   mkdir Lab2

5. Move into the new directory
   cd Lab2

6. Create a new c file for you to write your lab.
   > <Last Name><FirstInitial>_Lab2.c

   > <Last Name><FirstInitial>_Lab2_Functions.c

   > <Last Name><FirstInitial>_Lab2.h

   Example: I would create FoosJ_Lab2.c, FoosJ_lab2_Functions.c and FoosJ_Lab2.h

   Note: <Last Name><First Initial>_Lab2.c will only have your main function.

INTRODUCTION:

Your lab will assign integers to two arrays and then perform a number of tasks on those arrays. The numbers for the first array will be read in from an input file and the second array will filled with random numbers. All your output will be printed to an output file. No sample output is provided in these directions. It is up to you to create a professional looking results and desk check (check by hand) all your calculations.

PART A: Set up your header file

In your header file, insert the safeguards that prevent data from being initialized twice

Also add comments to show where you will insert your include statements, macros and constants as needed, and prototypes.

Example:

```
#ifndef HEADER_H
#define HEADER_H

// Includes

// Macros

// Constants

// Prototypes

#endif
```

NOTE: Do not use HEADER_H, the symbol should represent your file.
My header file would use FOOSJ_LAB2_H

Notice I used all uppercase letters.

PART B: Create functions to print your header

Both your code files should include your header file.

Copy the functions (from any of the "*Files and Arrays*" sample programs) into your functions c file.

There should be three functions:

- void PrintHeader(FILE * pFout)
- void PrintDivider(FILE * pFout, char symbol, int numberOf)
- void CenterString(FILE * pFout, char string[], int lengthToCenterAcross)

Notice that the first parameter is a FILE pointer. This will allow us to print our header to either an output file or the screen (by passing in stdout, this is defined in the stdio.h headerfile).

Note: Don't forget to add prototypes and macros to your header file

HINT: You must open your output file before call the PrintHeader function

PART C: Adding functions to your lab

The top of your <Last Name><FirstInitial>_Lab2.c file will have your IPO and the include for your header file. .

The top of your <Last Name><FirstInitial>_Lab2_Functions.c will NOT have an IPO but will need an include for your header file.

NOTE: It is your responsibility to determine what parameters you need and if they should be pass by value or pass by pointer.

NOTE: Function need to be written in order and properly numbered in your functions file (Except the main that is not in the functions folder). Functions from part A should go before the part B functions.

FUNCTION 1: the main function:

- In your main function, create two arrays of size 50. A constant or macro should be used for the 50. That constant or macro should be created in your header file.
- Declare your file pointers
- The rest of your main function should be mostly function calls. Some code is allowed to error check your data file.
- Function calls don't need to be documented; all other code does need to be documented

FUNCTION 2 and 3: OpenFile and CloseFile

- These functions can open/close both files at one time or you can open/close one file at a time

FUNCTION 4: ReadDataFromFile

- This function will read integers into your first array.
- The array may not be filled, so keep a counter

FUNCTION 5: FillArrayWithRandomNumbers

- This function should fill the array with random number that are between 1 and 500
- Since the array will be filled, there is no need for a counter

See Appendix A for code to create random numbers.

FUNCTION 6: PrintArray

- This function will print the array to your output file
- The array must be printed in rows of 10
- There needs to be a header for each column
- Every row needs to be labeled EX: ROW 1:
- Function must use formatters to space the data.

Note: This function will be called for both arrays

Sample Output Table:

```
                    ARRAY 2: Data from rand() function

Column:     1      2      3      4      5      6      7      8      9     10
--------+------+------+------+------+------+------+------+------+------+------+
Row 1:  |  403 |  217 |  269 |  495 |   97 |  325 |  198 |  111 |  130 |  428 |
Row 2:  |  251 |  212 |  436 |  293 |  454 |  205 |  120 |  296 |   74 |  339 |
Row 3:  |  151 |  369 |  177 |  165 |   89 |  297 |  386 |  210 |  426 |  228 |
Row 4:  |  487 |  328 |  296 |  107 |  322 |  244 |  283 |  371 |  206 |  412 |
Row 5:  |  150 |  456 |  475 |   86 |  101 |  428 |  142 |   72 |   75 |   67 |
```

Your table doesn't have to look as fancy, but all the labels and proper spacing must be there.

FUNCTION 7: CountEvenOddNumbers

- This function will count the number of even and odd numbers in the array

Note: This function will be called for both arrays

Hint: You have two outputs, so you will not use the function return

Hint: Use the modulus operator or use bit manipulators to check if a number is odd or even

FUNCTION 8: TotalArray

- This function will total all the numbers in the array
- The result should be returned via the return variable

Note: This function will be called for both arrays

FUNCTION 9: CalculateAverage

- This function will calculate the average number in each array
- Don't forget to protect against divide by 0 (think if-statement)
- If you do have divide by 0, return 0
- The result will be returned via a return variable

Note: This function will be called for both arrays

Hint: Use the result from the TotalArray function

FUNCTION: 10: CountNumberAboveAverage

- This function will count how many elements of the array are above the average
- Cast each array element to a double before making the comparison
- The result will be returned via a return variable

Note: This function will be called for both arrays

FUNCTION 11: PrintResults

- This function will print all the results to your output file
- Print the total, average, count above results, and even/odd counters
- Value with decimals should be shown to 3 places.

Your main function:

This function will call the functions in the following order:

- OpenFile
- PrintHeader
- ReadDataFromfile
- FillArrayWithRandomNumbers
- PrintArray for array 1 and array 2
- CountEvenOddNumbers for array 1 and array 2
- TotalArray for array 1 and array 2
- CalculateAverage for array 1 and array 2
- CountNumberAboveAverage for array 1 and array 2
- PrintResults for array 1 results and array 2 results
- CloseFile

NOTE: If you wish, you may call the functions for array 1 and then call the functions for array 2 or you can follow the order above. The order is up to you.

DEBUGGING TIPS:

Do Part A and then run the program to make sure it works.

For part B:

Write function 2 and 3. Then use printf to print out a few elements of your first array to make sure the array works. Then do the same with function 4

Write functions 4 through 9 one at a time. Then pass the results into PrintResults and verify that they work and that they are correct.

Function 5 will probably be the hardest.

Function 6-9 should only take 15-20 minutes to write. If it they take more than an hour, you may want to ask for help.

If you want to run or debug your program in Microsoft Visual Studio Community 2022, you will want to include the following code in your header files BEFORE your include statements:

```
#define _CRT_SECURE_NO_WARNINGS
```

Remove all debug code before submitting your lab

EXTRA CREDIT 10%

If your lab does not run with proper output, no extra credit may be earned.

Declare a third array in your main for 100 possible integers.

Write a function that will assign element 0 to 1500 (this must be a macro/constant declared in your header file) and use the following algorithm to assign all other elements (starting with element 1):

```
Array3[0] = 1500;
While previous element does not equal 1
   If the previous element is even:
      Current element is equal to previous element / 2
   Else
      Current element is equal to 3 * previous element + 1
```

This array may not fill, so you will need to use a counter to keep track of how many elements are in the array

Then call all the other functions for your third array.

You will receive minimal help from the CAs and your professor for the extra credit.

RUNNING YOUR PROGRAM

Your program must run in gcc on the unix operating system with 0 errors and 0 warnings.

To run your program:

```
gcc *.c -Wall
```

This will create an output file named a.out

Valgrind must run with 0 errors and 0 warnings.

To run Valgrind:

```
valgrind --leak-check=full ./a.out
```

FILES TO SUBMIT TO BRIGHTSPACE

<Last Name><FirstInitial>_Lab2.c
<Last Name><FirstInitial>_Lab2_Functions.c
<Last Name><FirstInitial>_Lab2.h
Lab2_Output.txt

APPENDIX A:

Code to generate a random number:

```c
// Include for the srand and rand function
#include <stdlib.h>

// Include for the time function
#include <time.h>

int main (void)
{

   int randomNumber;
   time_t t;

   // Intializes random number generator to the clock
   srand((unsigned int) time(&t));

   // Create a random number between 1 and 20
   randomNumber = rand() % 20 + 1;
```

Note: Don't forget to include the necessary header files.

Note: Your header file includes, and prototype should be in your header file.

ADDITIONAL PARTIAL CREDIT GRADING RUBRIC:

NOTE: The full rubric is on Brightspace under Misc

Infraction	Points Lost
Not checking for divide by 0	-10 points
Leaving debug code in the lab	-5 points
Checking for equal in the above function	-5 points
Not using formatters in the print table function	-5 points
Not controlling the decimal places to 3 places	-5 points
Extra Credit: Not using a while loop	-3 points

Note to Grader: Any partial credit outside of the rubric must be approved by the teacher.