Test 01

CSci 515 Spring 2015

<2015-02-24 Tue>

Dates:

Due: In Lab, Wednesday March 4, by 4:05 pm (lab end time)

Description

You have 2 hours (our regularly scheduled lab time) to complete the following tasks. Create a single file, named test01.cpp, in a visual studio project called Test01. Set up the Visual Studio projects using the normal settings for our class and labs, as we have practiced for the past 7 weeks.

Perform the following tasks:

1. In the main function of your program, write a statement that prompts the user for a seed value. Seed the C standard library random number generator with this seed value using the `srand` standard library function. Write an index controlled loop that uses an index to execute 10 times. Inside of your loop, generate a random number between 1 and 4, and display the random number to standard output. If the number is a 1, display a special message "I generated a 1!". The output from this task and loop in your main function should look like the first part of the example output given below.

2. After the previous task in your main function, write code to perform the following task. I have given you a file named "test-01-data.txt". This file has a single integer value on each line. Open this file, and inside of a sentinel controlled loop, read in the values from the file and display the values to standard output. Your sentinel controlled loops should simply be a while loop that executes until the read from the file fails. The output from reading the give file will be exactly as given in the example output shown below.

3. Write a function named `calculateHypotenuse`. The function will take two floating point values as parameters, named `a` and `b`. The function should calculate the hypotenuse of a right triangle using the `a` and `b` parameters, according to this formula:

$$c = \sqrt{a^2 + b^2}$$

e.g. the hypotenuse is the square root of adding together the square of a and b. This value should be calculated and returned (as a floating point result) as a result of calling your function. In your main function, show an example of calling your function with values of 3.0 and 4.0 for the `a` and `b` parameters.

4. Create an array of integers of size 10 named values. Initialize all of values to 0 in a loop in your main function. Create a function named `initNegative` that takes an array of integers and a size as its two input parameters. This will be a void function (it will return no result). This functions should initialize each value to be equal to the negative of the index, e.g. the value at index 0 will be 0, the value at index 1 will be -1, the value at index 2 will be -2, etc. Show an example of calling your function in your `main()` function. In your main function, write a loop that displays the contents of the array after calling your `initNegative` function. An example of the output from this task is shown below as the last part of the example output.

Your program output should look something close to the following when I run your program:

```
----- Task 1 -----
Enter a seed value: 42
0: 3
1: 1
I generated a 1!
2: 2
3: 2
4: 1
I generated a 1!
5: 3
6: 2
7: 1
I generated a 1!
```

```
8: 4
9: 4

----- Task 2 -----
Read value from file: 3
Read value from file: 1
Read value from file: 42
Read value from file: 9
Read value from file: 11
Read value from file: 12
Read value from file: 7

----- Task 3 -----
Hypotenuse of triangle with sides 3 and 4: 5

----- Task 4 -----
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
```

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions have function headers preceding them that document the purpose of the functions, and the input parameters and return value of the function.

Lab Submission

An eCollege dropbox has been created for this lab. You should upload your version of the lab by the end of lab time to the eCollege dropbox named `Lab 06 Processing Arrays`. Work submitted by the end of lab will be considered, but after the lab ends you may no longer submit work, so make

sure you submit your best effort by the lab end time in order to receive credit.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.

2. 40+ pts. Your program must have the required named function, that accepts the required input parameters and return the required values (if any).

3. 20+ pts. Your `displayArrayValues` function must correctly format the displayed output on standard output. Your program should work if the begin and end range are equal, and should show now output when begin is greater than the end specified.

4. 20+ pts. You must use I/O formatting to correctly display the output index ranges of the arrays as shown.

5. 20+ pts. Your main function must prompt the user as specified, and display the output formatted correctly as shown.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the lab to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions. Your function headers must be formatted exactly as shown in the style guidelines for the class.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits (e.g. no calls to system()).

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening `{` and closing `}` for control statement blocks on their own line, indented correctly for the level of the control statement block.

9. All control statement blocks (if, for, while, etc.) must have `{ }` enclosing them, even when they are not strictly necessary (when there is only 1 statement in the block).

10. You should attempt to use meaningful variable and function names in your program, for program clarity. Of course, when required, you must name functions, parameters and variables as specified in the assignments. Variable and function names must conform to correct `camelCaseNameingConvention` .

Failure to conform to any of these formatting and programming practice guidelines for this lab will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.