X414.20 Fundamentals of Software Development

Coding Lab 4 – File Input, Modularity

Please complete these lab steps only while in the Zoom meeting.

It is easiest to print these instructions so that you can refer to them during the lab. Your screen will be busy enough with both Zoom window and your Amazon Workspace window.

Task 1 – Change the Blood Pressure program to one that reads from a file

Last week, you wrote a program that read in a name and age and gave a blood pressure advisement. This week you will change that program to that it reads the data from a data file (text file) and then prints the results to an output file (another text file).

The original blood pressure program prompted the user for a name and an age, and then it displayed a short advisory report. In this new non-interactive version, it will not prompt the user for anything, instead obtaining its data from an input file. And it will also not display anything on the screen, instead writing the report to an output file.

The source code should be stored under: **c:\class\lab4task1.c**. The external filename of the input file will be **c:\class\bpdata.txt**. You will need to manually create this file on your AWS disk by creating it in the Dev C++ editor and saving it to disk under that name. The format for the data should be: name of the subject on the first line and age on the second line. There are no blank lines above the name and the name starts in column 1. Assume the name might have a space in it and that it can be up to 15 characters in length. An example would be:

```
Maria Elena
45
```

The output file will have an external name of **c:\class\lab4task1out.txt**. Using the sample data above, the output file would be formatted like this:

```
Patient first name: Maria Elena
Patient age:        45
BP warning level:   142.5
```

The Patient first name line has no blank lines above it.

Some things to think about:

1. For every external file, there is an internal file name you have to declare. Come up with a simple identifier to use for each of these.
2. Remember that a statement is needed to link the internal and external files, and there's also a code to indicate if you are reading from a file or writing to a file.
3. What type of input statement do you need to read a string (from a file) that might have a space?
4. What do you have to do toward the end of the program to ensure that the end of the file is not truncated?

Keep working on this until you think it is correct. Then **Ask for Help** so that I can check both your pseudocode design and your C program (for each student). **Do not proceed until I have checked and approved your program.**

Task 2 – Change the report format of the Blood Pressure program

In this task, you will change the format of the report that is written to the file to a columnar one. Using the sample data we used above, the new output file should look as follows:

```
Patient Name            Age     Bp Warning Level

Maria Elena              45           142.5
```

The name will start at the very left margin, just under the P in the header. The age will be right-justified where its one's digit is under the "e" in "Age". Note that the age can be anywhere from 1 to 3 digits. The blood pressure should be displayed with the decimal point right under the "g" in "Warning". The blood pressure will be displayed rounded to 1 decimal digit and could read below, equal to or above 100.0.

The output file will have an external name of **c:\class\lab4task2out.txt**. The source code file should be: **c:\class\lab4task2.c**

Some things to think about:

1. This is a columnar report, so think about what I said in the lecture about your formats. What should they NOT be.
2. Instead of trial and error, think about how you can use your Dev C++ editor to determine column positions and column widths.

Keep working on this until you think it is correct. Then **Ask for Help** so that I can check both your pseudocode design and your C program (for each student). **Do not proceed until I have checked and approved your program.**

Task 3 – Change the Blood Pressure program to use void functions

In this task, you will alter the program you did in Lab 3 Task 4 (last week) so that it uses top-down development and void functions.

With this exercise, you will not alter the user interface of the interactive program you wrote last week. Instead, you will "start over," approaching the program top-down. I want you to divide the program into a main function and 3 void functions. The first void function should deal with the header. The second void function should deal with the user input. And the third void function deals with the calculation and the display of results.

The source code should be stored in **c:\class\lab4task3.c**.

1. I'd like the functions to be defined below the main(), which is how most programmers do it. What do we need to do special since we are defining the void functions below main()?

Ideally, you would do this by creating a hierarchy chart and three N-S charts, but we'll let you practice that at home by yourself.

Keep working on this until you think it is correct. Then **Ask for Help** so that I can check both your pseudocode design and your C program (for each student). **Do not proceed until I have checked and approved your program.**

Task 4 – Mini Lecture then Change Blood Pressure program also use a function with an argument

I will give a mini-lecture discussing the topic of functions that return values. You will then use this to alter your blood pressure program to use a function you create called bpwarning().

Instead of having a statement such as:

```
bloodpressure = BASELINE + (age/2.0);
```

I'd instead like you to have a statement that calls the function bpwarning(). It should have a single argument, which is the age. The statement would look like this (substitute whatever identifier you happened to use in your program):

```
bloodpressure = bpwarning(age);
```

Your function definition should use whatever local variables and constants it needs. While it's possible to do everything in a single return() statement, I'd instead like you to use local variables. Remember, the variables that are declared locally do not at all need to be declared in the main(), only inside of bpwarning(). And the parameter used in the definitions (the one that "accepts" the age) can be any identifier you want; it doesn't have to match what's in the call (except the data types must match).

The source code should be stored in: **c:\class\lab4task4.c**.

Some things to think about:

1. Remember what data type to use for the parameter in the definition. It should match the data type of the argument.
2. I'd like you to do a prototype for this function, just like the others. In the prototype, you only need the data type in the parameter area.

Keep working on this until you think it is correct. Then **Ask for Help** so that I can check both your pseudocode design and your C program (for each student).

Task 5 – Submit your files in Canvas (breakout session)

1. Close Dev C++.
2. Re-open your Google Chrome. Be sure to choose Chrome from inside your AWS, not the one on your physical computer.
3. You should be in Canvas. If not, browse there and login.
4. Go to **Modules**. Then **Module 4**. If it doesn't display automatically, click the wedge next to it to open things up. Click on **Coding Lab – M4** – F**ile Input and Modularity**.

5. Click the big **Submit Assignment** button.
6. Under **File Upload**, click **Choose File**.
7. In the File name box, type C: and press ENTER. Then double-click the class folder to open it.
8. For each of the C files you generated tonight, highlight each of them (one at a time), click it and then click **Open** (or you could simply double-click the file).
9. The file should appear next to the **Choose File** button.
10. Do this for each additional C file from tonight plus your single output file. You will need to choose **+Add Another File** for each of these. **There should be a total of 6 files**: 2 output files, 4 source code files. No need to submit the data file.
11. You may add a comment to the **Comments…** area if you wish.
12. Then click **Submit Assignment**. The button will change to **Submitting** and then you will be returned to the Coding Lab screen. Assignment should be marked as Submitted! In the upper right corner.
13. Later, to check your score, click on **Grades** in the second-level menu on the left.
14. If I haven't graded your program yet, you will see an icon. Other, you will see a score. You can click on the name of the assignment to see my comments and to issue further comments or respond to mine.
15. Coding Labs will typically receive a score of 5 if you completed the lab and your submission was substantially correct. You may receive a score of 4 if there are some problems, and 3 if it is incomplete. You will receive a 0 if you did not participate in the lab. Ignore any "Total" scores that Canvas may show. They are meaningless.

Congratulations on completing your third Coding Lab!!! Please exit the breakout session to rejoin the entire class for final announcements.