X414.20 Fundamentals of Software Development

Coding Lab 3 – Input, Output & Expressions

<u>Please complete these lab steps only while in the Zoom meeting.</u>

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 0 – Learn Casual Pseudocode

For now, we will use pseudocode to design our programs. In later modules we will use more complex tools. Pseudocode is like an English version of code—something that describes what the code will do. There is a formal pseudocode that many Computer Science students learn, but we'll make it simple here and do what I call "casual pseudocode".

As you do the programs in this lab, I want you to do them in casual pseudocode first before writing in C code. For each action, we will use one of 4 pseudocode statements to describe what's going on:

- **Display** (a variable or some text)
- **Input from User** (into a variable)
- **Calculate** (a variable using a value or expression)
- **Print to a file** (a variable or some text)

So as you write your pseudocode version of the programs in this lab, use a combination of these actions. There are a couple of other actions that will need to happen (such as opening or closing a file), but we'll deal with that in the actual C coding.

Don't worry about precise syntax for these pseudocode commands. For instance, for Display you might say any of the following:

- ***`Display "City of Residence: " and then the city`***
- ***`Display the string "City of Residence" and then a colon and a space and then the actual city`***
- ***`Display "City of Residence: ", city`*** *(assuming city is a variable)*

Task 1 – Pseudocode for Blood Pressure Program (Breakout Group)

In casual pseudocode, write a program that determines a warning guideline for systolic blood pressure (the upper number) based on the age of the user. Assume the age is a whole number with no decimals. The formula for the maximum is 120 + half the age.

A sample dialog for the program would be:

```
Blood Pressure Program

This is just a guideline program. Please consult your doctor
for a more accurate analysis.

Please enter your age: 45

Thank you. You indicated that you are 45 years of age.
Seek medical attention if your systolic blood pressure is 142.5 mm Hg or higher.
```

By the way, all these lines line up on the left margin with no indentation.

Using casual pseudocode, write the program that will produce the output above. For ease, you may use your Dev C++ editor for this, but name the file **lab3task1design.txt**. Here are some steps to help you:

1. Determine what variables you will need. Typically, one for each thing that is input from the user, and one for each value that you calculate and eventually display.
2. Step through the sample dialog and use one of the 4 pseudocode actions for each line. Not all actions will immediately result in a line of output, for instance when calculating.
3. If you see a blank line in the sample dialog, feel free to say ***`Display blank line`*** as a pseudocode statement.
4. In the case of the program prompting the user for a value, that ends up being two pseudocode statements: one for ***`Display`*** and one for ***`Input from user`***.

OK, go ahead and write the pseudocode in your own editor. You may discuss this with your coding partner. Once you think you have everything correct, review it carefully, then signal me using **Ask for Help**. **Do not proceed until I have approved your pseudocode.**

Task 2 – Translate the Blood Pressure Pseudocode into C code (Breakout Group)

Now you will write your very first C program! You will use your approved pseudocode as a guide. Here are some things to think about as you are writing:

1. Remember what you need at the top of the program (in virtually every program) to bring in the header file that contains some of your basic input and output functions.
2. Earlier you thought about what variables you may have needed for this program. For each of those, what would be a good identifier? And think about the appropriate data type. Is it a whole number? Or is it a number that could possibly have a decimal?
3. The blood pressure number should be displayed to 1 decimal digit.
4. Remember how each program is structured and how there is a main function.
5. Think about each pseudocode statement and what it translates to in the C language.
6. Check your punctuation and spelling carefully as you write your C statements.
7. Remember the final statement in your main function. It wasn't in your pseudocode but you'll always need it as the last statement in every program you write. Do you recall what it is?
8. Depending on how you declared your variables, if you run the sample data, you may end up with a whole number for the blood pressure value. Use **Ask for Help** this if it happens.

Save your file as **lab3task2.c** in your class folder. Work with your coding partner but do your own typing in your workspace so that you get experience typing and debugging. Keep compiling and executing and fixing your errors (both of you in your own workspace) until both of your programs appear to work correctly. Try different input values to test your program. Once you are done, **Ask for Help** so that I can check your program. **Do not proceed until I have checked and approved your programs.**

Task 3 – Include a first name in the dialog (Breakout Group)

You are going to rework your pseudocode AND your program to include a first name. Assume the name might have a space in it, and it is a maximum length of 12 characters. The new sample dialog looks like this (all these lines line up on the left margin with no indentation):

```
Blood Pressure Program

This is just a guideline program. Please consult your doctor
for a more accurate analysis.

Please enter your first name: Maria Elena
Please enter your age: 45

Thank you, Maria Elena. You indicated that you are 45 years of age.
Seek medical attention if your systolic blood pressure is 142.5 mm Hg or higher.
```

First, go back to your pseudocode. Edit it so that it now includes the input of the name, and the display of the name as part of the output. Be sure to not write over the previously saved design file. Save this file as **lab3task3design.txt**

Then go ahead and modify your code to reflect the new specifications. It is best to save your old code again immediately under the new name (which will be **lab3task3.c**) so that you don't accidentally overwrite the old file on disk, and then start modifying the code in the new file.

Some things to think about:

1. What new variable are you introducing? What is its data type? And what about the length? Don't forget the rule we discussed in the lecture about string length.
2. There are two different ways to input a string. Because this name may have a space in it, we must use a particular one when writing this in C. Which one?
3. Think carefully how to arrange your `printf()` statement (that displays the name and the age) now that two variables will be included in it.

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 programs.**

Task 4 – Add Some Style! (Breakout Group)

You are now going to modify your C code so that it abides by the style guidelines I've mentioned in the module. Specifically:

- Your program should use a defined constant for the value 120. It should be called BASELINE. You may use either the **const** method or the **#define** method of setting up a defined constant. Then substitute the defined constant down in the code for the actual 120.
- Your program should use comments. Comment each variable declaration and each section of code. Id' say there are three major sections of code.
- Make sure code look nice with proper indentation from the curly braces and a judicious use of blank lines in between comments and groups of code.

Go ahead and modify your code to reflect these style guidelines. Again, it is best to save your old code again immediately under the new name (which will be **lab3task4.c**) so that you don't accidentally overwrite the old file on disk, and then start modifying that.

Keep working on this until done. Then **Ask for Help** so that I can check both your pseudocode design and your C program. **Do not proceed until I have checked and approved your programs.**

Task 5 – Fahrenheit to Celsius Conversion (Breakout Group)

You are going to write a program that asks the user for a number of degrees in Fahrenheit and then converts this to Celsius and displays it.

What is the formula for this? Well, I could tell you, but I want you to do the work. Use your search engine to seek out the formula for converting Fahrenheit to Celsius. **Do not Google actual C code that performs this. You need to learn how to translate from a formula to actual code yourself.**

Your sample dialog for this would be:

```
Temperature Conversion App

Please enter a temperature in degrees Fahrenheit: 98.6
98.6 degrees F is equivalent to 37.0 degrees Celsius.
```

Or another example:

```
Temperature Conversion App

Please enter a temperature in degrees Fahrenheit: 99
99.0 degrees F is equivalent to 37.2 degrees Celsius.
```

Write both the pseudocode design (first) and then the C program to accomplish this. The pseudocode design filename should be **lab3task5design.txt** and the C program should be **lab3task5.c**

Considerations for this program:

1. Use meaningful identifiers for this program. Not F and C (or f and c).
2. Comment your declarations and your code.

3. The formula for degree conversion involves two constants. While you could do the program with the actual constants, to get you in the habit of using defined constants I want one of the constants to be called FREEZEPOINT and the other to be called CONVFACTOR. You should be able to determine which is which.
4. Both temperatures should be displayed with 1 decimal digit. Simply use your format specifier in the output statement to do this.

You do not need to wait for my approval after doing the design to proceed to the C coding. However, do make sure that you do the pseudocode before processing to the C code, or you won't receive all the points for this lab.

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. **<u>Do not proceed until I have checked and approved your design and your programs.</u>**

Task 6 – Add File Output to the Previous Program

You are going to adapt the previous program to include file output. The sample dialog will change to:

```
Temperature Conversion App

Please enter a temperature in degrees Fahrenheit: 99
99.0 degrees F is equivalent to 37.2 degrees Celsius.

This information has been recorded in a log file called tempchange.txt
```

The above goes to the screen. And the following will be written to an output file on disk (no blank line at beginning of the file and no indentation within the file):

```
Group 1
F  99.00
C  37.22
```

The formatting of lines 2 and 3 are a code indicating Fahrenheit (F) or Celsius (C) and then a space. Followed by the temperature with three digits plus decimal point and 2 decimal digits. (Think carefully what the format has to be to accomplish this!) If the input temperature had been 102 degrees, the file would look like this:

```
Group 1
F 102.00
C  38.89
```

I want you to alter the pseudocode to do these extra actions in both the interactive and file output portions of the code. You will get to use `Print to a file` as one of your new actions several times.

Remember that your program is what is writing the output to the output file; you do not manually type it and save it.

Your pseudocode will need a couple of extra actions: `Open the file` and `Close the File`. Think about where these need to go.

Your pseudocode should be saved to **lab3task6design.txt** and the C program should be **lab3task6.c** Remember, your output file should also be written to your c:\class folder and should be named **tempchange.txt**.

Task 7 – 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 3**. If it doesn't display automatically, click the wedge next to it to open things up. Click on **Coding Lab – M3 – Input, Output and Expressions**.
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**: 5 .c source code files and 1 output 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 second Coding Lab!!! Please exit the breakout session to rejoin the entire class for final announcements.