CS 135 -- Lab 2

Math1 Project

The purpose of this project is to demonstrate integer and floating-point data types (§ 2.6 & 2.9), some of the functions in `cmath` (§ 3.9), and the stream manipulators in `iomanip` (§ 3.7).

1) Create a new folder: `Desktop > code_2020-08-31`

2) Let's look at the code, and discuss what it will do: ***Math1.gif***

3) Create a new folder named ***Math1*** within `code_2020-08-31`.
Start the EditPlus editor: `File > Open`
Browse to: `Desktop > code_2020-08-31 > Math1`
Create a new source code file named "Math1.cpp".

4) Type in the code exactly as shown in the ***Math1.gif*** image,
   - Put in correct file header AND function header.
   - Make sure you add correct "Modification History" details.
   - Include all "Programmed by:" info shown in *pgm_template.cpp*.
   - Compile it, and run it.

5) If it fails, what do you need to do get it to compile?

6) What is happending with the output values of *y* in the first example?
Explain with comments in your code.

7) Experiment with "uncommenting" the other "cout" output statements.
What happens? Explain with comments in your code.

Space Project

The purpose of this project is to demonstrate the proper way to do units conversions using "named constants" (§ 2.16), and format a table with the stream manipulators in `iomanip` (§ 3.7).

1) Create a new folder named ***Space*** within `code_2020-08-31`.
Start the EditPlus editor: `File > Open`
Browse to: `Desktop > code_2020-08-31 > Space`
Create a new source code file named "Space.cpp".

2) Use the code from program 2-16 (p. 56) from the textbook as an example of how to get started.
   a) Add steps to also convert the data to "miles" and "US tons".
      - Do "research" to find the correct conversion factors.
      - Use **"named conversion constants"**; that is one of the points of this project
      - Show the results in a table layout as shown here.

```
Body       Mass (kg)   Mass (US tons)  Distance(m)   Distance (mi)
-------------------------------------------------------------------
Sun       1.989e+030    2.1925e+027   1.49598e+011   9.29558e+007
```

   b) Use data in the ***solar_system_data*** table to update Sun values, and add lines for Earth & Moon. Add lines for your favorite planets from the ***solar_system_data*** table (5 points each).

   c) Finishing
      - Put in correct file header AND function header.
      - Make sure you add correct "Modification History" details.
      - Include all "Programmed by:" info shown in *pgm_template.cpp*.
      - Compile it, and run it.

3) Save all your code in `code_2020-08-31` Folder as a zip file, and save it to your flash drive.

Strings1 Project

The purpose of this project is to demonstrate the usage of `cin` versus `getline()` for input (§ 3.8).

1) Use the same folder we created last time: `Desktop > code_2020-08-31`

2) Let's look at the code, and discuss what it will do: ***Strings1.gif***

3) Create a new folder named ***Strings1*** within `code_2020-08-31`.
Start the EditPlus editor: `File > Open`
Browse to: `Desktop > code_2020-08-31 > Strings1`
Create a new source code file named "Strings1.cpp".

4) Type in the code exactly as shown in the ***Strings1.gif*** image,
• Put in correct file header AND function header.
• Make sure you add correct "Modification History" details.
• Include all "Programmed by:" info shown in *pgm_template.cpp*.
• Compile it, and run it.

5) If it fails, use the debugger.

6) Experiment with "uncommenting" the "cin" input statements, and "commenting out" the "getline" input statements.
What is the difference between using "cin" and "getline?
Answer this in the NOTES of the function comment header.

Chars1 Project

The purpose of this project is to demonstrate how `char` and `string` data types work (§ 2.7 & 2.8).
You will be patching together the code from 4 programs from the textbook to function as one program.

1) Create a new folder named ***Chars1*** within `code_2020-08-31`.
Start the EditPlus editor: `File > Open`
Browse to: `Desktop > code_2020-08-31 > Chars1`
Create a new source code file named "Chars1.cpp".

2) Write a program that is a combination of programs 2-12, 2-13, 2-14, and 2-15 (p. 49-54).
Put comments in your code indicating where each section came from.

   a) At the end of the code, add a few more lines to print out the numerical values of the first four characters of "`movieTitle`", as demonstrated by your instructor in class.
   Do not "hard-wire" the values. Use your string variable with subscripts.

   • Put in correct file header AND function header.
   • Make sure you add correct "Modification History" details.
   • Include all "Programmed by:" info shown in *pgm_template.cpp*.
   • Compile it, and run it.

3) Save all your code in `code_2020-08-31` Folder as a zip file.

Submission Instructions

Make sure the code has the appropriate file comment headers at the beginning, and function headers before each function. (samples on Canvas).
Compile and test it in the MinGW environment we have on campus. That is how it will be graded.

The zip file should contain the `code_2020-08-31` folder containing the four project folders:

- ***Math1***, ***Space***, ***Strings1***, ***Chars1***.

Each folder will contain the source code and executables for each program.

Submit the zip file to the Drop box on Canvas.