Lab 4

Introduction to C
CSCI 112, Spring 2015

Objectives

- Familiarize yourself with C math functions
- Practice program design and organization
- Practice using `enum` and `typedef`
- Practice using a `switch` construct
- More practice with loops, ifs, and a user input menu

Overview

You will design and write a program that asks the user to select a trigonometric function from a menu and you will then calculate and display the value of the selected function for every 15 degrees, starting from 0 degrees up to 90 degrees.

See the `Example Execution` section below for a detailed example.

Requirements

1. You should attempt to organize your code in a clean and logical manner. Do not place all of your code in your `main` function, but rather divide your program goal into subtasks and create a function for each of these subtasks.

2. You will display your user menu as part of a user-control loop.

   ```
   Please choose an option:  (0) Sine (1) Cosine (2) Tangent (3) QUIT
   Enter your choice >
   ```

   Each time you display the menu you will read in the user's response, process their request, display the results, and repeat the menu again. Your loop will terminate only once the user has selected the last option (`QUIT`). You may assume the user will only enter an integer. Do not change the order or numbering of this menu.

3. If a user selects an invalid option to the menu (e.g., 4), you should display an error and repeat the menu again. For example:

   ```
   4 is an invalid option.  Please try again.
   ```

   Observe that the error message should repeat back the option the user selected.

4. You are to use an `enum` named `MENU` as part of your command-line interface options. Specifically, you should create an `enum` for the four choices (`Sine, Cosine, Tangent, QUIT`). You should wrap this `enum` with a type definition, named `menu_t` (menu type).

5. Every time the user enters a value, you will use a `switch` statement to compare against each case of your `enum`. The numbers 0, 1, 2, 3 should not appear in your user menu control as **you must use your user-defined** `enum`. You will also refer to your `enum` value `QUIT` in your user-controlled menu loop, rather than 3.

> PROTIP: Because all `enum`s are stored internally as integers, you can use the `"%d"` formatting code to read in the user input in your `scan` statement, but you will store the value the user entered in a variable of type `menu_t`.

6. You are required to use two constants in your program:

   - `PI` is 3.14159
   - `LOOP_LIMIT` is 90

   These constants must be defined near the top of your program. Refer to these constants throughout your program and not the values directly. You may lose points if your retype these values throughout your program.

> PROTIP: Because you are using a constant to define the `LOOP_LIMIT`, we should be able to easily change your program to print out up to 360 degrees by only changing one value at the top of your program.

7. Format your results showing four places after the decimal point, as shown in the `Example Execution`, Achieve the indentation using a TAB in your `printf` function.

> PROTIP: The control character for TAB is `"\t"`.

8. To display the trigonometric results, you will use a `for` loop based on degrees, starting from zero and counting up to `LOOP_LIMIT` by step of 15 degrees.

9. The trigonometric functions you will use takes in a value in radians. Therefore you must convert each degree value to radians before passing to the trigonometric functions. The relationship between degrees and radians is: $radian = PI \cdot degree \,/\, 180$. If you fail to convert degrees to radians, you will produce incorrect values.

> PROTIP: To use the `sin`, `cos`, and `tan` functions, you will need to import the `math.h` library.

10. `tan(90.00)` is an infinite number. You must explicitly look for and handle the case of `tan(90.00)` and you must to display `UNDEFINED` instead. If you do not, you will display an incorrect value and may lose points.

Example Execution

```
$ ./program4

Please choose an option: (0) Sine (1) Cosine (2) Tangent (3) QUIT
Enter your choice > 4
4 is an invalid option.  Please try again.

Please choose an option: (0) Sine (1) Cosine (2) Tangent (3) QUIT
Enter your choice > 0
        sin(0) = 0.0000
        sin(15) = 0.2588
        sin(30) = 0.5000
        sin(45) = 0.7071
        sin(60) = 0.8660
        sin(75) = 0.9659
        sin(90) = 1.0000

Please choose an option: (0) Sine (1) Cosine (2) Tangent (3) QUIT
Enter your choice > 1
        cos(0) = 1.0000
        cos(15) = 0.9659
        cos(30) = 0.8660
        cos(45) = 0.7071
        cos(60) = 0.5000
        cos(75) = 0.2588
        cos(90) = 0.0000

Please choose an option: (0) Sine (1) Cosine (2) Tangent (3) QUIT
Enter your choice > -1
-1 is an invalid option.  Please try again.

Please choose an option: (0) Sine (1) Cosine (2) Tangent (3) QUIT
Enter your choice > 2
        tan(0) = 0.0000
        tan(15) = 0.2679
        tan(30) = 0.5773
        tan(45) = 1.0000
        tan(60) = 1.7320
        tan(75) = 3.7320
        tan(90) is UNDEFINED

Please choose an option: (0) Sine (1) Cosine (2) Tangent (3) QUIT
Enter your choice > 3
You chose QUIT. Thank you, come again!
```

Compile

Compile your program using this `gcc` command.

```
$ gcc lab4.c -o program4 -lm
```

The additional `"-lm"` flag is required because we are using the math library.

Test

Test your program! There are a number of choices you may make during your design and implementation. Make sure your test each one of your trigonometric functions, your `QUIT` condition, and any input integers that are out of bounds of the menu options.

PROTIP: Make sure you output what is expected, nothing more and nothing less. Do not change the order or the number associated with each of the menu options, otherwise the grading script will choose incorrectly.

Try to recreate the output of the `Example Execution` by making the same menu choices given in the example. Does your output look the same?

Resources

- You should attend the Friday Lab session to seek assistance from the TAs and CAs.
- For general questions, check the D2L `Lab 4 Help Forum` to see if another student has already asked your question. Otherwise, post your question on the D2L forum. This forum is intended for general questions that will benefit all students. Do not paste your code nor give away any specific answers to the lab on this forum.
- For specific questions, attend the weekly lab session or attend the TA's office hours.
- You are encouraged to use resources or tutorials on the internet to learn unix or C. Check the class resource list on D2L for some links to useful resources.

Submission

- Assigned: February 24th
- Lab Day: February 27th
- Due Date: March 6th, 8:00am

1. Make sure you included ample and informative comments – it is 20% of your grade!
2. Rename your C file to `last_first_lab4.c` and substitute your last and first name.

   Protip: If you fail to follow the above file naming conventions, your program may not be graded automatically and you will lose points.

3. Submit your `.c` file to the D2L dropbox Lab4.

   Protip: Submit only your C file. Do not submit your object file or your executable program. Do not archive (e.g., `zip`) your file.

Each student will complete and submit this assignment individually. Submit your work before the deadline as late work will receive a 50% penalty. Labs submitted more than 24 hours late will not be accepted. The late deadline is Saturday, March 7th, 8:00am.