Chapter 5 Lab

**User Defined Functions**

**Lab Objectives**

* Be able to write user-defined functions
* Be able to write function prototypes
* Be able to call functions
* Be able to create a separate library files for function reusability and document user-defined functions correctly

**Introduction**

Functions are commonly used to break a problem down into small manageable pieces. A large task can be broken down into smaller tasks (functions) that contain the details of how to complete that small task. The larger problem is then solved by implementing the smaller tasks (calling the functions) in the correct order.

This will also allow for efficiencies since the method can be called many times without rewriting the code each time.

**Task #1 void Methods**

1. Copy the file geometry.cpp. This program will compile, but when you run it , it doesn’t appear to do anything except wait. That is because it is waiting for user input, but the user doesn’t have the menu to choose from yet. We will need to create this.
2. Above the main, write the prototype for a function called PrintMenu that has no parameter list and does not return a value.
3. Below the main, write the function definition for PrintMenu. It should simply print out instructions for the user with a menu option for the user to choose from. The menu should appear to the user as:

This is a geometry calculator.

Choose what you would like to calculate.

1. Find the area of a circle
2. Find the area of a rectangle
3. Find the area of a triangle
4. Find the circumference of a circle
5. Find the perimeter of a rectangle
6. Find the perimeter of a triangle

Enter the number of your choice:

1. Add a line in the main method that calls the PrintMenu function as indicated by the comments.
2. Compile, debug and run. You should be able to choose any option, but you will always get 0 for the answer. We will fix this in the next task.

**Task #2 Value-Returning Functions**

1. Write a function prototype and a function definition for **CircleArea** that takes the radius of the circle and returns the area using the formula *A = πr2*.
2. Write a function prototype and a function definition for **RectangleArea** that takes in the length and width of the rectangle and returns the area using the formula *A = lw*.
3. Write a function prototype and a function definition for **TriangleArea** that takes in the base and height of the triangle and returns the area using the formula *A = ½ bh*.
4. Write a function prototype and a function definition for **CircleCircumference** that takes in the radius of the circle and returns the circumference using the formula *C = 2πr*.
5. Write a function prototype and a function definition for **RectanglePerimeter** that takes the length and the width of the rectangle and returns the perimeter of the rectangle using the formula *P = 2l + 2w.*
6. Write a function prototype and a function definition for T**rianglePerimeter** that takes the lengths of the three sides of the triangle and returns the perimeter of the triangle which is calculated by adding up the three sides.

**Task #3 Calling Functions**

1. Add lines in the main which will call the functions. The comments indicate where to place the function calls.
2. **Below, write some sample data and hand calculated results for you to test all 6 functions.**
3. Compile, debug, and run. Test out the program using your sample data.

**Task #4 Reusable and Well-documented Functions**

1. Create a header file called formulas.
2. Remove #pragma once as this is non-standard and will not work in every environment. Add the header file guards

#ifndef FORMULAS\_H

#define FORMULAS\_H

1. Cut the function prototypes for the six functions that calculate values from the geometry file and paste after the header file guards.
2. End the file with #endif
3. In the geometry file, add a preprocessor directive (after the other #include lines)’

#include “formulas.h”

1. Add comments for each of the function prototypes you just moved. They should include:
   * 1. A one-line summary of what the function does.
     2. A description of what the function requires to operate and what the result of that operation is.
     3. A listing of each parameter and what that parameter represents.
     4. If a value is returned, what that value represents
2. Create a new file called formulas.cpp.
3. Cut the function definitions that match the prototypes you just moved and paste them into the new file.
4. Copy the comments from each function prototype to the corresponding function definition.
5. Compile, debug, and run. It should operate as previously.