Lab 1: An Introduction to C Programming

CSE/IT 113

"Oh dear! She's stuck in an infinite loop, and he's an idiot! Well, that's love for you"

Professor Farnsworth, *Futurama*

"What we've got here is failure to communicate"

the Captain, *Cool Hand Luke*

---

1 Introduction

To this point, you have dealt with small C programs that printed their results to the screen. This action of printing to the screen is called *output*. There are other methods of output, such as putting data into files, writing data to the World Wide Web, sending data to a printer, and giving data to another application. Notice that all of these examples involve information or data leaving a program and going elsewhere ... *output*.

Lab 1 will deal with basic *input*. Conceptually, input is the same as output but in reverse: instead of information leaving your program, information is coming into your program from elsewhere. The information could be coming in from the keyboard, a file, the World Wide Web, or another program. Lab 1 will only address input from the keyboard.

In addition to input and output, Lab 1 will introduce the basic anatomy of a C program, as well as the very basic data types, variables, and functions.

2 Overview

In order to receive full credit for this lab, you must **write a simple geometry calculator program**. The program will read in a variety of inputs and perform some simple geometric calculations.

2.1 Sample output

The output of your program will look as follows, where the red text indicates input from the user and is not output from your program. The blue text are variables printed by `printf()` using tokens in the format string. The black text is the string literal part of format string in `printf()`

Important: A majority of the output must be calculated by your program. That is the point of this program. Do *not* print the exact output as given here as an example of the program in operation. Your program should perform the geometry operations based on the numerical values read in from the keyboard.

```
Welcome to Wile E. Coyote's Geometry Calculator!

Enter the height of a rectangle as a whole (integer) number:  2
Enter the width of a rectangle as a whole (integer) number:  3

Rectangle Calculations:
The area of a rectangle with height 2 and width 3 is 6
The perimeter of a rectangle with height 2 and width 3 is 10
The length of the diagonal of a rectangle with height 2 and width 3 is 3.605551

Enter the radius of a circle as a floating point number:  2.0

Circle Calculations:
The area of a circle with radius 2.0 is 12.566371
The circumference of circle with radius 2.0 is 12.566371

Enter the height of a right triangle as a floating point number :  1.0
Enter the base of a right triangle as a floating point number :  1.0
```

```
Triangle Calculations:

The area of a right triangle with height 1.0 and base 1.0 is 0.500000

The perimeter of a triangle with height 1.0 and base 1.0 is 3.414214


Enter the number of sides of a regular polygon as an integer:  8

Enter the length of the side of a regular polygon as a floating point number:
5.0


Regular Polygons:

The exterior angle of a regular polygon with 8 sides is 45.000000 degrees.

The interior angles of a regular polygon with 8 sides sums to 1080.000000 degrees.

Each interior angle of a regular polygon with 8 sides is 135.000000 degrees.

The area of a regular polygon with 8 sides, each 5.0 long is 120.710678
```

2.2 Getting Started

To help get you started, let's work with some code that takes in user input!

```c
#include <stdio.h>

/* These are function prototypes. It is way to declare
   the function before they are used. The compiler requires that
   all functions be declared before they are used.

   You will find the definition of the functions below main()

*/

int add(int m, int n);
void print_answer(int m, char op, int n, int answer);

int main(void)
{
        int m;
        int n;
```

```c
        int result;

        /*Ask user for input*/
        printf("Please enter 2 integers: ");
        scanf("%d %d", &m, &n);

        /* this calls the add function */
        result = add(m, n);

        /* this calls the print_answer function */
        print_answer(m, '+', n, result);

        return 0;
}

/* function definitions begin after main in this style of coding */

/**
* Adds two integer variables
* @param a, the first addend
* @param b, the second addend
* @returns the sum a + b
*/
int add(int a, int b)
{
        return a + b;
}

/**
* Prints out the integer equation and answer
* @param a, the first integer input
* @param op, the operator
* @param b, the second integer input
* @param answer, the answer to the equation
* @returns nothing
* @remarks none
*/
void print_answer(int a, char op, int b, int answer)
{
        printf("The answer to %d %c %d is %d.\n", a, op, b, answer);
}
```

In order to read in numbers from the user, we use the function `scanf()`. It works similarly to `printf()`:

```
scanf("%d %d", &m, &n);
```

The first parameter to `scanf()` is a format string, similar to the one used in `printf()`. The format string in the above example means that we are scanning *standard input* (what you type on the terminal) for two integers (`%d %d`) and saving them into the variables `m` and `n` respectively (`&m, &n`). The ampersand (&) is important when using `scanf()`. If you forget it, you will get errors when you try to compile your program. The & operator says to use the *address* of the variable, not its contents. For the time being, just accept the fact we need an ampersand to make `scanf()` work.

For input of floating point numbers from the user, you will use the token %lf (again just like what you do for `printf()` to print `double` data types). For example, to enter two floating point numbers, your `scanf()` statement will look something like this:

```
double x;
double y;
printf("Enter a two floating point numbers: ");
scanf("%lf %lf", &x, &y);
```

3 An approach to coding

While the final program asks for user input, a beginner's mistake is to code the input part of the program first. **This cannot be stressed enough – this is a terrible way to start coding the program**.

Beginner's want to code the input section of the program and then write the functions and test them. Experienced programmers know this is a waste of time and that input is the last thing you code. The first thing you code is the functions and you test them with hard-coded values. Only after all functions are tested and working properly is the input part of the program written.

For example, the beginner wastes all his/her time on input:

```c
#include <stdio.h>

int main(void)
{
        int height = 0;
        int width = 0;

        /* beginner mistake -- starts with input before he/she
         * test any functions */

        printf("Enter the height of a rectangle: ");
        scanf("%d", &height);

        printf("Enter the width of a rectangle: ");
        scanf("%d", &width);


        return 0;
}
```

Rather the correct way to code is to not worry about input at all. Write and test all functions that are required of you using hard coded values:

```c
#include <stdio.h>

/* function prototypes */
int area_rectangle(int h, int w);

int main(void)
{
        /* hard coded values for variables. These are
         * test cases with known output */

        int height = 1;
        int width = 1;

        /* test the function with known values
         * compare the output of the function with what
         * you as a programmer expect it to be
         */

        /* note the printf is split into three lines
         * for display purposes only */
```

```
        printf("The area of rectangle with height %d ", height);
        printf("and width %d ", width);
        printf("is %d\n", area_rectangle(width, height));
        return 0;
}

/* function definitions */

/* write comments as you write the function, you will
 * not want to comeback and write them */

/**
 * Calculates the area of a rectangle
 * @param h the height of the rectangle
 * @param w the width of the rectangle
 * @return the area of the rectangle
 */
int area_rectangle(int h, int w)
{
        return w * h;
}
```

It is only after all functions are tested with hard coded values that the input (`print()/scanf()` pairs) is added to the program. Start with one function and make sure it works correctly before moving on to the next function.

4 Requirements

For this lab, you will write a geometry calculator program with these "features":

1. In addition to main() you will write 11 separate functions:

   area_rectangle()

   perimeter_rectangle()

   diagonal_rectangle()

   area_circle()

   circumference()

   area_triangle()

   hypotenuse()

   perimeter_triangle()

exterior_angle()

interior_angle()

area_regular_polygon()

2. Use function prototyping (see comments in the examples)
3. Your code must have *comments*! See above and `doxygen.c` on Canvas for commenting style.
4. The source code file needs to follow the format of `doxgen.c`. You need a begininng header with your information, etc.
5. a script of your running code. Name the script file `lab1.script`. Lab 0 gives directions on how to create a script.
6. a README (an example README can be found on Canvas). Name the README file, just `README`.

5 Programming Notes

File Naming Conventions

Name your source code `lab1.c`, your README file `README`, and your script file `lab1.script`.

Rectangle

In a real calculator you would use a floating point type for the height and width of the rectangle. As we want you to practice with both input of integral and floating point types, the rectangle's height and width are of type `int`.

The function `diagonal_rectangle()` returns a `double`. You will have to include the standard library `math.h` to use the `sqrt()` function to calculate the diagonal of a rectangle. To compile add `-lm` to the end of the compile command. For example:
`gcc -Wall lab1.c -o lab1 -lm`

Circle

The radius is of type `double`.

Approximate π with the value of 3.141593 in your calculations. In your area calculation use the function `pow()` from `math.h` to calculate the radius squared.

Triangle

Both the height and base of the triangle are of type `double`

Variables cannot be named the same, even if they are different types. Height is used in both the rectangle and triangle calculations. You may want to use a prefix so you can distinguish between the height of the rectangle and the height of the triangle. For example:

```
int rect_height = 1;
int rect_width = 1;
double tri_height = 1.0;
```

You will have to calculate the length of the hypotenuse, using the Pythagorean theorem, before you can calculate the perimeter of the triangle. In the body of the `perimeter_triangle()` function, call `hypotenuse` to calculate the length of the hypotenuse.

Regular Polygons

The number of sides of a regular polygon is of type `int`, while the length of a side is of type `double`.

As a reminder, regular polygons have n sides all of which are of equal length. Additionally, all angles of a regular polygon are equal. This leads to some nice properties.

To determine the exterior angle of a regular polygon the formula is $\frac{360}{n}$, where n is the number of sides. The angle is in degrees.

The sum of the interior angles of a regular polygon is $sum = 180(n-2)$, where n is the number of sides. The angle is in degrees.

From the sum of the interior angles, it is easy to calculate what each interior angle is: $interior\_angle = \frac{180(n-2)}{n}$. The angle is in degrees. It is not necessary that you create a separate function for this, but feel free to if it is easier for you.

Finally, to calculate the area of a regular polygon, use the formula:

$area = \frac{s^2 \cdot n}{4 \cdot tan(\frac{\pi}{n})}$

where s is the length of a side, n, is the number of sides. The prototype for the function `tan()` is found in `math.h`. Use the same value for π you used for the circle calculations.

6 Submission guidelines

Create a tarball containing your source code, script, and your README file. To do this run the command in the directory containing your source code, script and README:

```
tar czvf cse113_firstname_lastname_lab1.tar.gz *.c *.script README
```

To check the tarball has the desired files, run the command:

```
tar tf cse113_firstname_lastname_lab1.tar.gz
```

You will see a list of the files in the tarball.

Upload the tarball to Canvas before the due date.