CpSc 1111 Lab 6
Conditional Statements, Loops, the Math Library, and Random Numbers

Overview

For this lab, you will use:

- one or more of the conditional statements explained below
- `scanf()` or `fscanf()` to read in integer values entered by the user (from standard input)
- the `sqrt()` function from the math library
- a random number generator
- a loop

Background Information

All programming languages provide constructs to execute a section of code conditionally. In this week's lab, you will implement decision making with the use of one or more of the following conditional statements. You will also learn how to use the random number generator `rand()` function.

`if` Statements

The simplest conditional statement is the `if` statement. It is used when we want the computer to maybe execute some code based on the truth value of some condition. If the condition is true, the code in the body of the `if` statement will execute; if it is false, the body of the `if` statement will be skipped and execution will continue with the code immediately after the `if` statement.

Structure:

```
if (condition)
{
   one-or-more statements;
}
```

Example of an `if` statement:

```
if (n < 2)
{
   printf("Hello\n");
}
```

The above code prints a message only if the value of `n` is less than two. Otherwise, it does nothing.

What does the following code print if the value of `n` is 1?

```
if (n < 2)
   printf("Hello ");
   printf("Tigers!!\n");
```

What does it print if the value of `n` is 2?

`if-else` Statements

The previous `if` statement included code that executes when a condition is true. If we want certain code to execute when a condition is true and other code that executes when a condition is false, we use `if-else` statements. Exactly one of the two possible branches will be taken with an `if-else` statement.

Structure:

```
if (condition)
{
   one-or-more statements;
}
else
{
   one-or-more statements;
}
```

Example:

```
if ((age >= 13) && (age <= 19)) {
   printf("You are a teenager.\n");
}
else {
   printf("You are not a teenager.\n");
}
```

`if-else-if` Statements

When there are more than two options, instead of nesting `if-else` statements, C provides us with the `if-else-if` construct.

Structure:

```
if (condition)
{
   one-or-more statements;
}
else if (condition)
{
   one-or-more statements;
}
else if (condition)
{
   one-or-more statements;
}
else
{
   one-or-more statements;
}
```

Note: The last `else` is optional and works well whenever there is a fall through case when all other conditions are false. If there is no fall through option needed, then it may make sense to not have that last `else` at all.

Example:

```
if ( (day > 2) && (day <= 6) ) {
   printf("weekday\n");
}
else if (day == 7) {
   printf("Saturday\n");
}
else {
   printf("Sunday\n");
}
```

Dangling `else` Problem

The compiler associates an `else`- part with the closest `if`. The following code illustrates this point. **What does it print?** (Remember that the compiler ignores formatting.)

```
int n = 5;
printf("hello\n");
if (n < 4)
   if (n > 0)
      printf("good\n");
else
   printf("bye\n");
```

Random Number Generators

`rand()` and `srand()`

The function `rand()` will generate a (pseudo) random number from 0 to RAND_MAX, which is a constant defined in `<stdlib.h>` with a value of 32,767.

You use the `rand()` function to return a value in a range by using modulo of the returned value by the range span + initial value of the range:

`rand() % range_span + initial value` where `range_span = (high - low + 1)`

For example, if you wanted 100 numbers from 0 to 99:	`(rand() % 100)`
If you wanted 100 numbers from 1 to 100:	`(rand() % 100) + 1`
1985 – 2016 (inclusive):	`(rand() % 32) + 1985`
10 – 30 (inclusive):	`(rand() % 21) + 10`

But, using `rand()` as explained above will produce the same set of "random" values every time you run it because it automatically seeds the random number generator with 1. You can seed the random number generator with the `srand()` function using something other than 1. The most common way to seed the random number generator is to use the time function, which is in the `<time.h>` library file. This will seed the random number generator with the current time, which will be different every time you run the program, thus producing different sets of "random" values every time you run the program.

The way to do this is by including the following line of code in your program, somewhere above the `rand()` function call:

```
srand( (int)time(0) );
```

Typically, you would only seed the random number generator once per every set of calls to `rand()`. Don't forget to #include both `<stdlib.h>` and `<time.h>` at the top of your file.

The `math.h` Library (same as in the Lab 4 write-up)

Another library provided by C is the `math.h` library. This library provides math functions that you can use, such as `pow()`, `sqrt()`, `abs()`, `cos()`, `sin()`, `tan()`, `log()`, among many others. Pages 485-492 of the "Programming in C" book list many of the functions in this library. You can also learn about the math library functions by typing `man` at the Unix command prompt, a space, the function name, and then pressing Enter. For example, to learn about sqrt you would enter: `man sqrt`

To get some experience using the math library, for this lab assignment, you will be using the `sqrt()` function.

One last thing for you to know - you will need to compile your program with an option (`l` for library, `m` for math), for example:

```
gcc -Wall main.c -lm
```

Lab Assignment

Reminder About Style, Formatting, and Commenting Requirements

- The top of your file should have a header comment, which should contain:
  - Your name
  - Date
  - Lab section
  - Lab number
  - Brief description about what the program does
  - Any other helpful information that you think would be good to have.
- Variables should be declared at the top of the main function, and should have meaningful names.
- One exception to the rule of using meaningful names for variables is to use `i` or `j` or `n` or some other single letter as a loop index variable name; this is a common, acceptable practice.
- Always indent your code in a readable way. Some formatting examples may be found here: https://people.cs.clemson.edu/~chochri/Assignments/Formatting_Examples.pdf

Write a program called `main.c` which will prompt the user to enter a three digit number on the keyboard. Your program will find the square root of the number using the `sqrt()` function from the math library (make it also an integer value). You can print that value to the screen so that you check to make sure it compiles and works; then remove or comment out that print statement and continue to the next step. That square root value will be used as the radius of a circle.

Your program will use a random number generator (RNG) to check if a randomly generated point falls within the circle with that radius and center at (0,0). Both values of the point (x,y) will be randomly generated. Don't forget to seed the random number generator first. Both times that `rand()` is called, the general formula will be:

```
(rand() % (total number of values within the range)) + (minimum value of the range)
```

The minimum and maximum values for x and y should be entered in by the user. The total number of values for x (or y) will be an expression based on the minimum and maximum values for x (or y). Put this much into a loop that goes 10 times (except for seeding the RNG – this only needs to be done once before the loop) and show the (x,y) values each time. Compile and run, and verify by inspection that both the x and y values of each of the 10 points fall within the correct range. Run it several times and you will see a different set of random (x,y) values each time it is run.

Once you feel confident that your random (x,y) values are within the correct range, ***comment out the line of code where it is seeded***, and then recompile and run again. The output on the following page shows what you should get when entering 100 as your 3-digit input number, x with a range of [-4, 12], and y with a range of [-2, 8] with an un-seeded RNG.

```
Enter a 3 digit number: 100
Enter a minimum x value: -4
Enter a maximum x value: 12
Enter a minimum y value: -2
Enter a maximum y value: 8
(  6,  8)
(  5,  0)
( 11,  2)
( -1,  4)
(  7, -1)
(  1,  5)
( -1,  1)
(  8,  2)
(  7,  8)
(  2, -2)
```

If your output matches what is shown above, you can continue with the rest of the program. You'll need the formula for determining if a point falls within a circle – you can Google this if you need to. Using an if-else statement inside that loop that goes 10 times, print to the user whether that point lies inside, on the circumference of, or outside the circle. Your finished output should look like the following:

```
Enter a 3 digit number: 100
Enter a minimum x value: -4
Enter a maximum x value: 12
Enter a minimum y value: -2
Enter a maximum y value: 8
(  6,  8) is on the circumference of the circle with radius 10
(  5,  0) is inside the circle with radius 10
( 11,  2) is outside the circle with radius 10
( -1,  4) is inside the circle with radius 10
(  7, -1) is inside the circle with radius 10
(  1,  5) is inside the circle with radius 10
( -1,  1) is inside the circle with radius 10
(  8,  2) is inside the circle with radius 10
(  7,  8) is outside the circle with radius 10
(  2, -2) is inside the circle with radius 10
```

Turn In Work

1. Before turning in your assignment, make sure you have followed all of the instructions stated in this assignment and any additional instructions given by your lab instructor(s). Always test, test, and retest that your program compiles and runs successfully on our Unix machines before submitting it.
2. Show your TA that you completed the assignment. Then submit your `main.c` program using the handin page: http://handin.cs.clemson.edu. ***Don't forget to always check on the handin page that your submission worked. You can go to your bucket to see what is there.***

Grading Rubric

If your program does not compile on our Unix machines or your assignment was not submitted on time, then you will receive a grade of zero for this assignment. Otherwise, points for this lab assignment will be earned based on the following criteria:

Functionality	60
Style, Formatting, & Commenting	10
Use of conditional statement	10
Use of loop	10
Use of sqrt() from math.h	5
Use of rand() from stdlib.h	5
Penalty for not following this lab's instructions (incorrect I/O, etc.)	-5 if warnings when compile, and other point deductions decided by grader