

## Step 4: Loops and Repetition

---

This assignment is done in class on (monday)02-03-14 AND (friday)01-31-14.

This assignment is due for monday class 02-03-14 AND for the friday class 01-31-14.

### Getting Started

---

By now you should be all logged on. You should still have a cse251 directory in your home directory from last week. Please change into that directory.

You should be able to do a pwd and see that you are in the cse251 directory at this time.

Remember, start gedit with this command line:

```
gedit &
```

Create a new program called looper.c with the following code to start:

```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

/*
 * Simple program to experiment with looping
 */

int main()
{
    printf("My looper program!\n");
}
```

Be sure it compiles and runs. The commands to compile and run are, of course:

```
gcc -o looper -lm looper.c
./looper
```

### 1. The while statement

---

Add this variable to your looper program before the printf statement:

```
int i = 1;
```

Now, after the printf statement, add this code:

```
while(i <= 10)
{
    printf("i=%d\n", i);
    i = i + 1;
}
```

Compile and run this. The output should look like this:

```
My looper program!
i=1
i=2
i=3
i=4
```

```
i=5
i=6
i=7
i=8
i=9
i=10
```

What this does: When the while statement is executed,  $i$  is less than or equal to 10, so the statement following the while is executed. In this case, there is a block following the while, so the block gets executed. The block prints the value of  $i$ , then increments  $i$  by 1. Then we return to the while and test again. Again,  $i$  (2, now) is less than or equal to 10, so the block gets executed again. This is a while loop.

Eventually,  $i$  will be incremented to a value of 11 and the test will fail. Then the program proceeds to the next statement.

Notice that I used an integer in this program. A common activity in C programs is counting. Counting is always safer with integers than floating point values.

Statements like  $i = i + 1$  are very common in C programs (and C++, C#, and Java, in fact). So, C has a shortcut for this. To add 1 to an integer, you can do:

```
i++; /* Equivalent to i = i + 1; */
```

There is a version that subtracts as well:

```
i--; /* Equivalent to i = i - 1; */
```

Change the  $i = i + 1$ ; to  $i++$ ; in your program. It should work the same way.

Now copy this program to a new program named factorial.c. In case you don't remember, the Unix command will be:

```
cp loop.c factorial.c
```

We're going to write a program to compute factorials. You can delete the printf statement in your program after you copy it.

Let's compute a factorial. That's a good looping activity. The factorial of a number is the product of all integers up to that number. So,  $3!$  (! means factorial) is  $3 * 2 * 1$  and 5 factorial is  $5 * 4 * 3 * 2 * 1$ .

Put this code in your program now, replacing the existing counting loop:

```
int f = 3;      /* Number we compute the factorial of */
int fac = 1;    /* Initial value of factorial */
while(f > 0)
{
    fac = fac * f;
    f--;
}

printf("Factorial = %d\n", fac);
```

When you compile and run this you should get an output of 6, which is  $3 * 2 * 1$ . Read over this code carefully and be sure you understand what it is doing.

Modify this code so it inputs the value to compute the factorial of and outputs the number entered and the factorial in this form:

```
Number to compute the factorial of: 4
4! = 24
```

Remember, to input an integer use `scanf("%d", &f);` Don't forget the `&`. Also, be sure your program outputs what you are computing the factorial of in some way.

Let's make this a little more complete. An odd fact is that 0! is explicitly defined to be 1. Add code to handle this *special case*.

Finally, add code to keep inputting and computing factorials until the user enters a negative number. When that happens, exit the program. Write this without using the `exit()` function. Consider using a Boolean flag to control when we are done. When you execute your program, it should work like this:

```
Number to compute the factorial of: 5
5! = 120
Number to compute the factorial of: 0
0! = 1
Number to compute the factorial of: 6
6! = 720
Number to compute the factorial of: -1
cbowen@ubuntu:~/cse251$
```

When working on your program, you may managed to get it stuck in a continuous loop, where the program does not stop executing. In that case, hit Ctrl-C, where you hold down the Ctrl key and hit a C. This is a break and should force your program to exit. If that does not work, closing the terminal and opening another should work.

Go to <http://www.cse.msu.edu/handin>. Select CSE 251 and Step 4 factorial.c and hand in the file factorial.c.

## 2. The for Statement

Copy the program `looper.c` to `sine.c`. We're going to create a program that works with sine waves.

Replace the body of the program with this code:

```
int main()
{
    double angle;
    int numSteps = 20;
    double maxAngle = M_PI * 2;
    int i;

    for(i = 0; i < numSteps; i++)
    {
        printf("%3d: \n", i);
    }
}
```

Compile and run this.

Modify this program so it outputs 0 to 20 instead of 0 to 19, but without changing the value of `numSteps`. You should be able to do this with one character.

I want to compute the value of sine for values from 0 to `maxAngle` ( $2 * \pi$ ). So, let's compute the angle first. Try this code inside the loop:

```
angle = i / numSteps * maxAngle;
printf("%3d: %5.2f\n", i, angle);
```

If you run this you'll get values of 0, then suddenly 6.28. Do you know why? The problem is that `i` and `numSteps` are both integers. So,  $3 / 20$  is 0 in integer math. We need to do our math as floating point values instead. Change the angle computation to this to force the values of `i` and `numSteps` to be treated as doubles:

```
angle = (double)i / (double)numSteps * maxAngle;
```

When you run this you should see nice steps from 0 to  $2\pi$ . We call this "casting" the value to a double. Note how I used fixed width output formats to make the output line up.

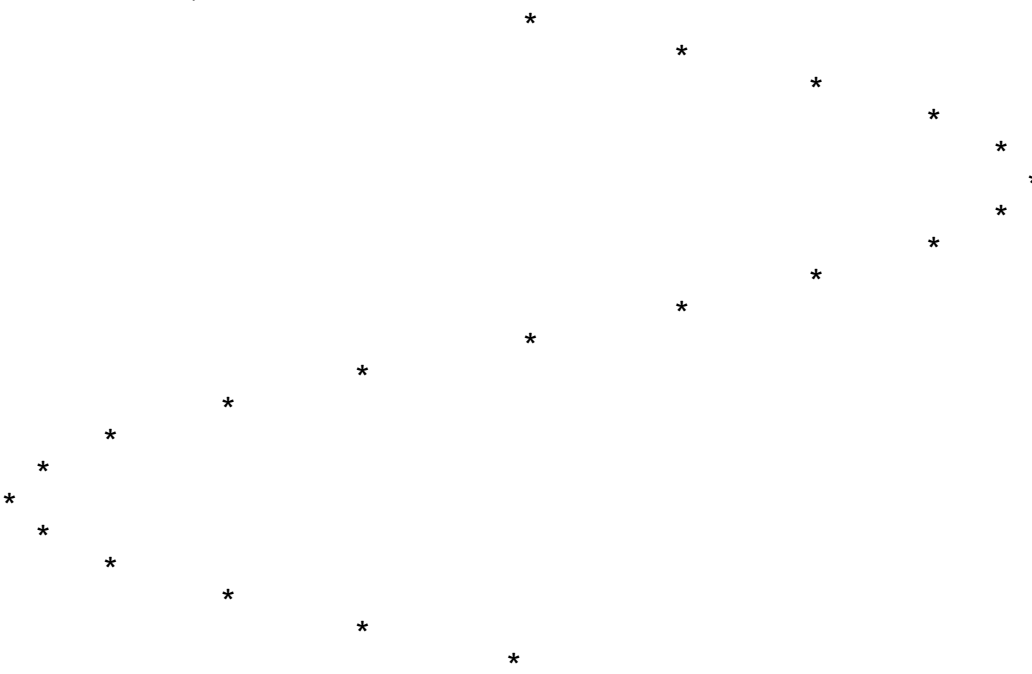
Now we can compute the value of sine for each angle:

```
angle = (double)i / (double)numSteps * maxAngle;
sinVal = sin(angle);
printf("%3d: %5.2f %6.3f\n", i, angle, sinVal);
```

Be sure you add a declaration for the new variable sinVal and what type should it be?

I want to plot the sine wave. I'll use the \* character to plot. Modify your program to generate the following output:

```
cbowen@ubuntu:~/cse251$ ./sine
0: 0.00
1: 0.31
2: 0.63
3: 0.94
4: 1.26
5: 1.57
6: 1.88
7: 2.20
8: 2.51
9: 2.83
10: 3.14
11: 3.46
12: 3.77
13: 4.08
14: 4.40
15: 4.71
16: 5.03
17: 5.34
18: 5.65
19: 5.97
20: 6.28
```




My plot starts right after the angle and is 60 characters wide. A few hints to help with this:

- Remember, if you leave off the `\n` on a `printf` statement, it does not return.
- Try first just creating a loop that outputs 20 spaces, then an asterisk.
- The first star in the above graph is at 0. The left-hand side of the graph is at -1 and the right hand side of the graph is at plus 1. In order to place the stars properly, you will have to scale the range of sinVal (-1 to 1) to the range 0 to 60. (e.g, `numSpaces = 30 + sinVal * 30`).
- Make sure to use floating point arithmetic. Remember that integer arithmetic truncates any remainder. If specifying numbers, make sure to use a decimal place (2.0, rather than 2, etc).

There are several ways to do this. All will require a for loop inside of the for loop that loops over the angles.

Once you have this working, change it to do 40 steps instead of 20. You may have to increase the size of your terminal window. This output should look something like this, except I have turned off the angles, but you should include them in yours:

```
cbowen@ubuntu:~/cse251$ ./sine
0: -----
1: -----
2: -----
3: -----
4: -----
5: -----
6: -----
7: -----
8: -----
9: -----
10: -----
11: -----
12: -----
13: -----
14: -----
```



```

15: -----
16: -----
17: -----
18: -----
19: -----
20: -----
21: -----
22: -----
23: -----
24: -----
25: -----
26: -----
27: -----
28: -----
29: -----
30: -----
31: -----
32: -----
33: -----
34: -----
35: -----
36: -----
37: -----
38: -----
39: -----
40: -----

```

### 3. do/while and final work for today

Modify your program sine.c to have the following features:

- Ask the user to input a number of steps. If a value less than 2 is entered, ask the user again. Implement this with a do/while loop.
- Plot the sine wave using the value the user provided.
- Instead of using \*, use / if the waveform is falling, \ if it is rising, or \* if the absolute value of the derivative is less than 0.1. The output should look something like this:

```

cbowen@ubuntu:~/cse251$ ./sine
Input the number of steps: -3
Input the number of steps: 20
0: 0.00
1: 0.31
2: 0.63
3: 0.94
4: 1.26
5: 1.57
6: 1.88
7: 2.20
8: 2.51
9: 2.83
10: 3.14
11: 3.46
12: 3.77
13: 4.08
14: 4.40
15: 4.71
16: 5.03
17: 5.34
18: 5.65
19: 5.97
20: 6.28

```

Some hints:

- You can determine the absolute value using the `fabs()` function. For example: `fabs(x)` will give me the absolute value of `x`.
- To include `\` in a character string, instead using two: `\\` as in **`printf("This is what a \\ will look like: \\n");`**

Go to <http://www.cse.msu.edu/handin>. Select CSE 251 and Step 4 sine.c and hand in the file sine.c.

## And We Are Done!

---

You should have turned in two short programs (factorial and sine wave). Then we are done for today.

[CSE 251](#)