Patterns

By:
Bhavik B.
Student of CS50

CS50: Introduction to Computer Science

An introduction to the intellectual enterprises of computer science and the art of programming.

This is CS50, Harvard University's introduction to the intellectual enterprises of computer science and the art of programming for majors and non-majors alike, with or without prior programming experience. An entry-level course taught by David J. Malan, CS50 teaches students how to think algorithmically and solve problems efficiently.

Topics include abstraction, algorithms, data structures, encapsulation, resource management, security, software engineering, and web development. Languages include C, PHP, and JavaScript plus SQL, CSS, and HTML. Problem sets inspired by real-world domains of biology, cryptography, finance, forensics, and gaming. As of Fall 2014, the on-campus version of CS50 was Harvard's largest course.

For more info about course visit following link:
https://www.edx.org/course/introduction-computer-science-harvardx-cs50x

[P.T.O]

hello, world!

Today I'm going to show you key concepts to solve #mario of CS50-2015 pset1 for absolute beginner. What is to be done is making a pattern here.

Step - 1
Let’s say we have to make a 5x3 rectangle.

```
#####
#####
#####
```

There are 3 rows & 5 columns in it. To understand how a loop works we have to make a simple program to print that rectangle.

Let’s do it. Let's say there are r=3 rows & c=5 columns to do that.
for (int r = 1; r <= 3; r++) that is row loop.
for (int c =1; c <= 5; c++) that is column loop.

```
/**
  * You can initialize as r = 0 & c = 0 & put condition as r < 3 & C < 5
  * Basic idea is you have to run row loop 3 times or say maximum row
  * Same you have to run column loop 5 times or say maximum column
  */
```

Now let's try to nest it. Loop within a loop. Run following program inside standard structure of C. See the output.

```
for (int c = 1; c <= 5; c++)
{
        for (int r = 1; r <= 3; r++)
        {
        printf("#");
        }
            printf("\n");
}
```

What happened here? It printed 5x3 rectangle instead of 3x5.

What do you understand by that?
Outer loop is for printing ROW & inner loop is for printing COLUMN.
So to debug program you need to switch the places of loops & it will work. That's the basic.

[P.T.O]

Step – 2
You should be able to make any rectangular pattern now.
Make a 5x5 square for further. In 5x5 square. Loops look like,

```
for (int r = 1; r <= 5; r++)
{
        for (int c = 1; c <= 5; c++)
        {
            printf("#");
        }
            printf("\n");
}
```

Now, let's say we have to make a triangular pattern. 5x5

```
#
##
###
####
#####
```

What changes to be done here? You can try by yourself 1st.
ANS: Column loop ~ Inner loop. for (int c = 1; c <= r; c++) that's it.

What do you learn from it?
If you've to make a triangular pattern, you should only change the condition of column loop from "c <= 5" to "c <= r". In short c <= 5 prints a rectangular pattern, whereas c <= r prints a triangular pattern.

Step – 3
Now let's try a mirror image of that pattern.

```
    #
   ##
  ###
 ####
#####
```

To make it more helpful, I've put 0s instead of spaces to learn precisely.

```
0000#
000##
00###
0####
#####
```

[P.T.O]

What changes have to be made?
Try by yourself.
ANS: A variable for a row & two different variables for column.

Why two different variables for column? Because we have to print 0 & # both. In previous pattern, after condition of column loop terminates it directly go to next statement which is printf("\n"); indicates to go to a new line. So there won't any extra variable required to print the space. But here is the different case.

For 0 let's take c (refers to Column) & for # let's take s (refers to Sharp)

```
for (int r = 1; r <= 5; r++)
{
        for (int c = ; c ; c )
        {
            printf("0");
        }

        for (int s = ; s ; s )
        {
            printf("#");
        }
            printf("\n");
}
```

Step – 4
How to fill in the blanks??

See the pattern again. The 1st ROW means r = 1 & loop is executing outer for loop 1st time. There are 4 0s & 1 # in 1st row. So, whichever loop carries 0 to print must run 4 times & loop carries # to print must run one time, before they terminate & move to r = 2. (2nd execution of outer loop)

It is clear that loop with variable C must run 4 times before it terminates & go to next loop with variable s.

You can run different trials & errors to make that work. But there comes the use of 0s instead of a blank space.

[P.T.O]

Step – 4A
There are 0s 1st row, so you can start with c = 4.
Once you start with c = 4, you must decrease it because 0s are decreased by one in the next row. Hence c--
Tricky part is the condition. You know that you've to run it 4 times, hence 4, 3, 2, 1. Let's assume that c >= 1 for the condition.
Why that loop must run 4 times? Because there are maximum 4 0s to print which are in 1st row. So we have started with c = 4, & decreased it by c--

Step – 4B
There is 1 #, so you can start with s = 1. Once you start with s=1, you must increase it because # are increased by one in next row. Hence s++
Tricky part is condition. How many times you should run the loop before it terminates?

See the last ROW. There are 5 # means loop "must be executed 5 times", hence 1, 2, 3, 4, 5. Let's assume that s <= 5 for the condition. Why that loop must run 5 times? Because there are maximum 5 # to print which are in last row. So we have started with s = 1, & increased it by s++

Now fill in the blanks in that loop & execute the code.
There is a bug right. It didn't print what we required. Why?

Step – 4C
Now go to step-2 & see what you already have learnt from it.
If you've to make a triangular pattern, you should change the condition from "c <= 5" to "c <=r ". If you use c>=1 & r<=5 as conditions, you'll end up with a RECTANGULAR pattern something like this, that is what exactly happened.
```
0000#####
0000#####
0000#####
0000#####
0000#####
```

It is clear from the desired output that we have triangular patterns of 0s & #. So we MUST use something like "c >= r" & "s <= r" in the conditions. ***But let me tell you that according to specific requirements of output, you have to change conditions like c >= r – 1 or s <= r + 2 & so on.***

[P.T.O]

Use following loops in program:

```
for (int r = 1; r <= 5; r++)
{
       for (c = 4; c >= r; c--)
       {
          printf("0");
       }

       for (s = 1; s <= r; s++)
       {
          printf("#");
       }
          printf("\n");
}
```

Output:

```
0000#
000##
00###
0####
#####
```

Step -5

Now talking about Mario. Count ROW & COLUMN of the desired output.

Height: 8

```
       ##
      ###
     ####
    #####
   ######
  #######
 ########
#########
```

Let’s make a change for simplicity. Instead of Spaces I can use 0s, to make it more understandable. But let me go a step more into it. I am using numeric instead of 0s, so that we don't need to count it.

[P.T.O]

```
Height: 8
1234567##
123456###
12345####
1234#####
123######
12#######
1########
#########
```

So, when height is 8 output looks similar.

Height, literally means numbers of ROWs here & Width means number of Columns. In output, 8 Rows & 9 Columns (7 + 2#)
Now, you should be able to do it on your own.

Try Hacker Edition of Mario if you like to push yourself more.

```
Height: 4
   #  #
  ##  ##
 ###  ###
####  ####
```

This diagram should help. Space is replaced by x & # is replaced by for loop variable that is printing # i.e. printf("%d", variable_name);

```
Height: 4
xxx4xx7
xx34xx78
x234xx789
1234xx78910
```

Thanks for reading.
Good Luck!

My name is Bhavik & this is CS50 :)

☺ ☺ ☺