

## LESSON 04 – NESTED LOOPS



In this lesson we will examine nested loops (a loop within a loop). Just like an if-statement, we can nest our loops to solve certain problems.

**Sections:**

- I. [BASIC STRUCTURE OF NESTED LOOPS ..... PAGE 1](#)
- II. [NESTED FOR-LOOP PATTERNS ..... PAGE 3](#)
- III. [MIXING IT UP ..... PAGE 5](#)

**I. BASIC STRUCTURE OF NESTED LOOPS:**

The following shows an example of a basic **nested while-loop**:

```
1  int x =0, y=0;
2  while(x < 3)
3  {
4      y = 0;
5      while(y < 3)
6      {
7          Console.WriteLine("x: " + x + ", y: " + y);
8          y++;
9      }
10     x++;
11 }
```

Sample Output:

```
x: 0, y: 0
x: 0, y: 1
x: 0, y: 2
x: 1, y: 0
x: 1, y: 1
x: 1, y: 2
x: 2, y: 0
x: 2, y: 1
x: 2, y: 2
```

Outer while-loop

Inner while-loop

On line 1 we declare two variables 'x' and 'y', one for our **outer while-loop** and one for our **inner while-loop** respectively. Our outer while-loop will cycle until 'x' equals '3', and you can see that on line 10 we are incrementing 'x' by '1' before the next iteration of this loop. Our **inner while-loop** will cycle until 'y' equals '3', and you can see that on line 8 we are incrementing 'y' by '1' before the next iteration of this loop.

The result of this program is that our outer while-loop cycles exactly 3 times. Our inner while-loop cycles 3 times on each iteration of the outer while-loop, resulting in a total of 9 cycles. Notice on line 4 we set the value of 'y' to '0' right before our inner while-loop executes. This is because after the inner-while loop finishes an execution (i.e., cycles 3 times), 'y' will equal '4'. Then when it is time to execute again (on the next iteration of the outer while-loop), we need to reset the value of 'y' back to '0' or this inner-loop will not execute since the condition will be false.

Because the above nested loops are definite (i.e., we know how many times we wish to execute our loops) we could use a **nested for-loop** instead:

```
for(int x = 0; x < 3; x++)
{
    for(int y = 0; y < 3; y++)
    {
        Console.WriteLine("x: " + x + ", y: " + y);
    }
}
```

Sample Output:

```
x: 0, y: 0
x: 0, y: 1
x: 0, y: 2
x: 1, y: 0
x: 1, y: 1
x: 1, y: 2
x: 2, y: 0
x: 2, y: 1
x: 2, y: 2
```

Outer for-loop

Inner for-loop

The above program works the exact same way. The benefit of using a **for-loop** instead of a while-loop in this case is that we do not need to increment or reset our variables during each iteration because the for-loop already does this for us.

## II. NESTED FOR-LOOP PATTERNS:

A common practice with nested for-loops is to make character patterns. Let's jump right in:

```
for (int x = 0; x < 5; x++)
{
    for (int y = 0; y < 5; y++)
    {
        Console.Write('*');
    }
    Console.WriteLine();
}
```

Output

```
*****
*****
*****
*****
*****
```

```
for (int x = 0; x < 5; x++)
{
    for (int y = 0; y <= x; y++)
    {
        Console.Write('*');
    }
    Console.WriteLine();
}
```

Output

```
*
**
***
****
*****
```

```
int z = 0;
for (int x = 0; x < 5; x++)
{
    for (int y = 0; y <= x; y++)
    {
        Console.Write(z);
        Console.Write('\t');
        z++;
    }
    Console.WriteLine();
}
```

Output

```
0
1    2
3    4    5
6    7    8    9
10   11   12   13   14
```

```
for (int x = 0; x < 5; x++)
{
    for (int y = 4; y >= 0; y--)
    {
        if (y > x)
            Console.Write(' ');
        else
            Console.Write('*');
    }
    Console.WriteLine();
}
```

Output

```
      *
     **
    ***
   ****
  *****
```

The best way to understand how these patterns are constructed is to analyze the code and try them out yourself. You can see that the outer for-loop will always cycle through the number of rows in the pattern, whereas the inner for-loop will always cycle through the character output for a single row.

Here are some other notes to be aware of:

- Notice how the inner for-loops of the last two patterns utilize the outer for-loop's 'x' variable.
- Notice how the second last pattern is making use of 3 variables: x, y & z
- Notice that the last pattern makes use of an if-statement to produce its pattern.

**Note:** The patterns above could have been created using while-loops, but since we are dealing with definite patterns, using for-loops makes more sense.

### III. MIXING IT UP:

We can nest as many loops and if-statements as we wish. Obviously, we like to keep our code simple, but there are times when nesting becomes very useful. Here is an example of where we read a file using a while-loop, then we use a for-loop to output a character a certain number of times based on the file input. Here is the input file:

#### Input File (input.txt):

```
3
i
2
c
6
s
0
```

Each pair of lines has an integer and a character. The integer indicates how many times the character should be outputted on a line. If a '0' is encounter, then we stop:

#### Sample Output:

```
iii
cc
ssssss
```

Here is the code to make this work:

```
1 // open input file
2 StreamReader sr = new StreamReader("input.txt");
3
4 // initialize variables
5 int num = 0;
6 char c = '\0';
7
8 // loop through file
9 while(true)
10 {
11     // get number of characters to output
12     num = Convert.ToInt32(sr.ReadLine());
13
14     // break out of while-loop if 'num' equals '0'
15     if(num == 0)
16         break;
17 }
```

```
18 // get character to output
19 c = Convert.ToChar(sr.ReadLine());
20
21 // output 'c' for 'num' times
22 for(int x = 0; x < num; x++)
23 {
24     Console.Write(c);
25 }
26 Console.WriteLine();
27 }
28
29 // close input file
30 sr.Close();
```

Notice on **line 9** we set our **while-loop** to loop forever:

```
while(true)
```

You may be thinking that this loop is going to go forever and crash your program! However, notice **lines 15 & 16**:

```
if(num == 0)
    break;
```

The **'break'** statement stops (breaks out) of the loop surrounding it. In our case, the **while-loop** is the surrounding loop so when we read a '0' in from the file our code will break out of the while-loop and the program will continue.

Finally, the inner **for-loop** is used to output the current row of characters (c) based on the current integer (num) which are both read in from the input file.