Mr. Bellavia

LESSON 02 - FOR-LOOPS

Sections:

I. FOR-LOOPS ..... PAGE 1
II. FOR-LOOP EXAMPLES ..... PAGE 4

In this lesson we will examine the use of for-loops and look at multiple examples demonstrating how their use and construction.

I. FOR-LOOPS:

Another way to loop your code is to use a **for-loop.** The benefits of using a for-loop instead of a while-loop is easily shown with an example:

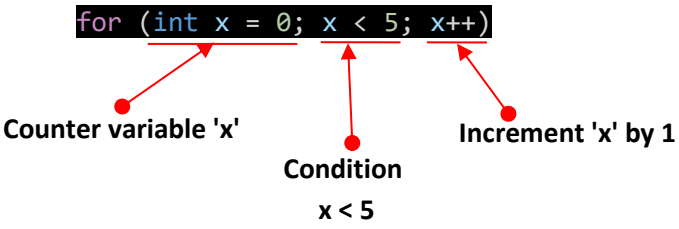
```
for (int x = 0; x < 5; x++)
{
    Console.WriteLine("I am a for loop!");
}
```

Sample **Output**:

```
I am a for loop!
I am a for loop!
I am a for loop!
I am a for loop!
I am a for loop!
```

For-Loops are **definite loops** (i.e., we can pre-determine how many times the loop should iterate/cycle). We can see that the above loop has 5 iterations (i.e., cycled 5 times). For-loops utilize a **counter variable**. In the above loop, we initiated an integer variable named **'x'** to a value of **'0'**. For-loops also contain a **check condition** (in this case: **x < 5**), and a way to **increment the counter variable** (in this case: **x++**). Remember that using **'x++'** is the same as using **'x = x + 1'**. The above for-loop cycles 5 times because the counter variable **'x'** starts at **0**, increments by **1** on every iteration **(x++),** and stops when it no longer is less than **5** (i.e. **x < 5**). To summarize:

Mr. Bellavia



Note: Each part of the for-loop statement is separated by a semi colon **';',** and the entire for-loop statement is inside brackets **( ).**

Like while-loops, we can print out the value of our counter (in this case 'x') on every iteration:

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

Note: Since we only have one line of code in our for-loop, we could have omitted the braces **{ }.** However, it may be good practice to keep the braces in case our code changes in the future and we add more than one line of code to this for-loop.

Sample **Output**:

```
0
1
2
3
4
```

Notice that **'x'** starts at **0** and goes up by **1** on every iteration. Once **'x'** equals **5** the condition (**x < 5**) is no longer true, and the loop stops (i.e., it stops on the beginning of the 6th iteration).

We do not have to increase by **'1'** on every iteration. Consider the following:

```
for (int x = 0; x < 5; x = x + 2)
{
    Console.WriteLine(x);
}
```

Sample **Output**:

```
0
2
4
```

In the above for-loop, the counter variable **'x'** increases by **'2'** on every iteration (because we have **'x = x + 2'** as our increment, not **'x++'**) and hence will only loop 3 times.

Now Consider the following:

```
for (int x = 1; x <= 5; x = x + 2)
{
    Console.WriteLine(x);
}
```

Sample **Output**:

```
1
3
5
```

The above for-loop initiates the counter variable **'x'** equal to **1 (x = 1)** and has its condition set to be less than or equal to 5 **(x <= 5).**

The possibilities for constructing a for-loop are endless! The following section will dive into some more for-loop examples.

II. FOR-LOOP EXAMPLES:

a) Finding all numbers divisible by 3:

The following for-loop will print out all numbers that are **divisible** by **3** from **0** up to but not including **20**:

```
for (int x = 0; x < 20; x++)
{
    if(x % 3 == 0)
        Console.WriteLine(x);
}
```

Sample **Output**:

```
1
3
5
```

Inside this for-loop we have an **if-statement** that checks to see if our counter variable is divisible by **3** (remember the modulus (%) operator: x % 3 must equal 0 to be divisible by 3). However, what we have really done is outputted every 3rd number, so we could have done this instead:

```
for (int x = 0; x < 20; x = x + 3)
{
    Console.WriteLine(x);
}
```

The above for-loop would have given the same result. So why use x % 3? Well consider the following for-loop:

```
for (int x = 1; x < 20; x = x + 3)
{
    Console.WriteLine(x);
}
```

This loop starts at '1' so printing every 3rd number would give you:

Sample **Output**:

```
1
4
7
10
13
16
19
```

Obviously, these numbers are not divisible by 3.

b) Print every second number backwards from 20 to 0:

```
for (int x = 20; x >= 0; x = x - 2)
{
    Console.WriteLine(x);
}
```

Sample **Output**:

```
20
18
16
14
12
10
8
6
4
2
0
```

In this example, we start our counter variable **'x'** at **20.** Our condition this time is checking to make sure that **'x'** is greater or equal to **0** on every iteration **(x >=0).** Finally, we **decrement** (count backwards) by a value of **'2' (x = x – 2)**.

c) Ask the user for two integers and find the average of all numbers in between:

```
int total = 0, count = 0, x = 0, y = 0;
double avg = 0;

Console.Write("Enter a value for x: ");
x = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter a value for y: ");
y = Convert.ToInt32(Console.ReadLine());

for (int z = x; z <= y; z++)
{
    Console.WriteLine(z);
    total = total + z;
    count++;
}
avg = (double)total / count;
Console.WriteLine("Average: " + avg);
```

Sample **Input** & **Output**:

```
Enter a value for x: 3
Enter a value for y: 10
3
4
5
6
7
8
9
10
Average: 6.5
```

Many things to notice here:

- Our counter variable **'z'** starts at **'x'** which the user inputted as **'3'**.
- The condition goes up to and including **'y'** which the user has inputted as **'10'**.
- We have an accumulator named **'total'** which keeps adding **'z'** to itself on every for-loop iteration.
- We have a counter variable named **'count'** which counts how many times the loop has iterated.
- We find the average after the for-loop has finished by dividing **'total'** by **'count'**.