## **Unary Operators**

#### WE'LL COVER THE FOLLOWING ^

- Unary Operators Example
  - Code Explanation

In the previous lesson we discussed the basic operators used in C#. One of the types listed was the **unary** operators. Let's learn about them now.

# **Unary Operators Example** #

Take a look at the example below which uses the **unary** operators.

```
using System;
class UnaryOps
    public static void Main()
        int unOps = 5;
        int preIncrement;
        int preDecrement;
        int postIncrement;
        int postDecrement;
        int positive;
        int negative;
        bool temp;
        preIncrement = ++unOps;
        Console.WriteLine("pre-Increment: {0}", preIncrement);
        preDecrement = --unOps;
        Console.WriteLine("pre-Decrement: {0}", preDecrement);
        postDecrement = unOps--;
        Console.WriteLine("Post-Decrement: {0}", postDecrement);
        postIncrement = unOps++;
        Console.WriteLine("Post-Increment: {0}", postIncrement);
        Console.WriteLine("Final Value of unOps is: {0}", unOps);
```

```
positive = +postIncrement;
Console.WriteLine("Positive: {0}", positive);

negative = -postIncrement;
Console.WriteLine("Negative: {0}", negative);

temp = false;
temp = !temp;
Console.WriteLine("Logical Not: {0}", temp);
}
```







[]

### Code Explanation #

When evaluating expressions:

- postIncrement (x++) and postDecrement (x--) operators return their current value and then apply the *operators*.
- preIncrement (++x) and preDecrement (--x) operators apply the *operators* to the variable **prior** to returning the *final* value.

### In the code above:

- In line 7 the unOps variable is *initialized* to **5**.
- When the preIncrement (++x) operator is used in line 16, unOps is incremented to 6 and the value 6 is assigned to the preIncrement variable.
- The preDecrement (--x) operator in line 19 turns unOps back to 5 and then assigns the value to the preDecrement variable.
- When the postDecrement (x--) operator in line 22 is used, the value of unOps, 5, is placed into the postDecrement variable and then unOps is decremented to 4.
- Next, in line 25 the postIncrement (x++) operator moves the current value of unOps, 4, to the postIncrement variable and then increments unOps to 5.
- In line **30** the variable positive is set equal to postIncrement variable which is equal to **4**.
  - The **plus** operator (+) in front doesn't affect the value of a number,

assigning positive with the same value as postIncrement, 4.

- In line **33** the variable negative is also set equal to postIncrement variable which is equal to **4**.
- Applying the **minus** (-) operator to a *positive* number results in a *negative* number.
  - Hence, in line **33** negative will equal **-4**, instead of **4**.

**Note:** The minus operator ( - ), which is not the same as the preDecrement operator ( -- ), doesn't change the value of postIncrement — it just applies a sign **negation**.

The *logical* **negation** operator (!) is a logical operator that works on **bool** values, changing true to false or false to true.

In the case of the temp variable above

- In line **36** the value is *initialized* to false
- The next line **37** applies the *logical* **negation** operator, (!), which **returns true** and reassigns the new value, **true**, to **temp**. Essentially, it is toggling the value of the bool variable, **temp**.

Now that you've understood **unary** *operators* lets take a look at the example for **binary** *operators* in the next lesson.