# Logical Operators

**The goal for this exercise** is to understand how to use logical operators.

Just as we’ve already seen with arithmetic/mathematical, and then relational/comparison operators, we need to know how to evaluate expressions that contain logical operators.

The following table contains a summary of the three logical operators.

|  |  |  |
| --- | --- | --- |
| **C# Syntax** | **Name** | **Meaning** |
| **!**A | Logical NOT | Reverses the logical true/false value:  If A is true, then !A is false.  If A is false, then !A is true.  Note that this is a *unary* operator, meaning that it takes only 1 operand – the “A”. |
| A **&&** B | Logical AND | Logical AND evaluates to true ONLY IF BOTH A and B are true.  Otherwise it evaluates to false.  Both this and the ORs are binary operators, so they take two arguments – A, B |
| A **||** B | Logical OR | Logical OR evaluates to true if either A, or B, or both A and B are true.  Logical OR evaluates to false ONLY IF BOTH A and B are false. |
| A **^** B | Exclusive OR (XOR) | Exclusive OR evaluates to true if ONLY ONE of A or B are true. Exclusive OR evaluates to false if A and B are both false, or if both A and B are true. |

**Example:**

Assume that the following variables have been declared:

**int x = 5;**

**int y = 2;**

**int z = 10;**

We want to evaluate whether the following expression is true or false:

**x > 4 && z <= 10**

// > and <= have the same precedence, so

// the left-to-right associativity of the operators

// tells us that > goes first ,

// so substitute the value of x

**5 > 4 && z <= 10**

// 5 is greater than 4 – it's **true**

**true && z <= 10**

// <= goes next, so substitute the value of z

**true && 10 <= 10**

// 10 is clearly <= 10, therefore it's **true**

**true && true**

// && is the only one left

// true AND true is true, so we get:

**true**

Thus, we can see that x > 4 && z <= 10 is true, in this case.

**What you need to do for this exercise:** Do these other examples (keep the same variable values) in the same style/format as the above example. If you are required to submit this exercise to the instructor (as indicated on the course web page), you should do so by including this answer in a comment, in the provided starter project within the class named Logical\_Operators. Make sure to CLEARLY label this with the exercise number, so that your instructor can quickly and easily find it. Handing this exercise in, in a separate file, will result in a point penalty.

1. y < 1 || y > 1
   1. If **y** is an integer, can you find a better way to express the above?  
      When looking for a different/better way to express this, think when the expression will be true, and when it won't. Once you've got that figured out, think about how you might explain it to a friend – what's a quick, 1 sentence summary of when the expression evaluates to true? Lastly, think about how you would write that in C#.  
      **When handing your answer to this part, put in something that clearly states what your new, simplified expression (in C# is)**
2. y < 2 || x > 4
3. z > 8 || y < 3
4. !(z > 8)
5. !( z > 8 || y < 3)
6. !( z > 4 && y == 2) || (z ==10 && y <= 3)
7. y < z ^ z < x
8. z < x ^ y < z
9. 10 < 20 ^ 20 < 30
10. 20 == 10 ^ 20 < 10