Mr. Bellavia

LESSON 04 - IF-STATEMENTS WITH LOGICAL OPERATORS

In this lesson we will examine how to use logical ANDs '&&' and logical ORs '||' inside our if-statement conditions to simplify our code when dealing with more complex decision making.

Sections:

I. LOGICAL OPERATORS:

When using if-statements, we can combine our conditions into a single if-statement with the following logical operators:

Logical Operator	Description
&&	AND
||	OR

To get a grasp on how this works let's first consider an example without the use of logical operators. The following program asks the user for a number and determines if that number is between 0 and 100:

```
int x = 0;
Console.Write("Please enter an integer: ");
x = Convert.ToInt32(Console.ReadLine());

if(x >= 0)
{
    if(x <= 100)
        Console.WriteLine("Your number is between 0 and 100.");
    else
        Console.WriteLine("Your number is not between 0 and 100.");
}
else
    Console.WriteLine("Your number is not between 0 and 100.");
```

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

```
Please enter an integer: 5

Your number is between 0 and 100.
```

You can see from the above code this is a simple program that determines if the user entered a number between 5 and 10. However, we could simplify this code even further with the use of logical operators which we will do in the next sections.

II. LOGICAL AND '&&' OPERATOR:

Using the example from the previous section, we can combine the two conditions into one if-statement condition using the logical AND '&&' operator:

```
int x = 0;
Console.Write("Please enter an integer: ");
x = Convert.ToInt32(Console.ReadLine());

if(x >= 0 && x <= 100)
    Console.WriteLine("\nYour number is between 0 and 100.");
else
    Console.WriteLine("\nYour number is not between 0 and 100.");
```

Notice how we used the **double ampersand '&&'** to join two conditions together. Our if-statement in this case now reads: if x is greater than or qual to zero **AND** x is less than or equal to 100. The program runs the exact same, but the use of the double ampersand has simplified our code. Not only do we have a smaller number of lines of code, but we are also no longer repeating code (i.e., the code in the last section had 3 Console.WriteLine() statements whereas now we only have 2).

Mr. Bellavia

III. LOGICAL OR '||' OPERATOR:

Like the logical AND, we can use a logical OR '||'. Consider the same example as before except now with a logical OR:

```
int x = 0;
Console.Write("Please enter an integer: ");
x = Convert.ToInt32(Console.ReadLine());

if(x < 0 || x > 100)
    Console.WriteLine("\nYour number is not between 0 and 100.");
else
    Console.WriteLine("\nYour number is between 0 and 100.");
```

Notice how we used the **double bar '||'** to join two conditions together. Our if-statement in this case now reads: if x is less than zero **OR** x is greater than 100. The program runs the exact same, except we have flipped the logic where the condition now tests if 'x' is not in the range of 0 and 100 and displays the appropriate message.

IV. AN IN-DEPTH EXAMPLE:

Let's look at a more in-depth example where we can use logical operators:

According to the USGA (United States Golf Association) the following table shows the current yardage guidelines for par ratings for golf courses:

	Men	Women
Par 3	Up to 250 yards	Up to 210 yards
Par 4	251 to 470 yards	211 to 400 yards
Par 5	471 to 690 yards	401 to 575 yards
Par 6	691 yards+	576 yards+

Let's write a program that asks the user for their gender (male or female), then ask for the hole length (yards) and output back to the user whether the hole is a par 3, 4, 5 or 6:

```
Console.WriteLine("---PAR RATING---");

string? gender = "";
Console.Write("\nAre you male 'm' or female 'f: ");
gender = Console.ReadLine();
gender = gender.ToLower();

int yards = 0;
Console.Write("\nPlease enter the number (integer) of yards for the hole: ");
yards = Convert.ToInt32(Console.ReadLine());

int par = 0;
if((gender == "m" && yards > 0 && yards <= 250) || (gender == "f" && yards > 0 && yards <= 210))
    par = 3;
else if((gender == "m" && yards >= 251 && yards <= 470) || (gender == "f" && yards >= 211 && yards <= 400))
    par = 4;
else if((gender == "m" && yards >= 471 && yards <= 690) || (gender == "f" && yards >= 401 && yards <= 575))
    par = 5;
else if((gender == "m" && yards >= 691) || (gender == "f" && yards >= 576))
    par = 6;
else
    par = 0;

if(par == 0)
    Console.WriteLine("\nYou did not enter valid information!");
```

```
else
    Console.WriteLine("\nYour hole is a par " + par);
```

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

```
---PAR RATING---

Are you male 'm' or female 'f: m

Please enter the number (integer) of yards for the hole: 445

Your hole is a par 4
```

As you can see, we really take advantage of the logical operators here. Let's break down our first if-condition:

```
if((gender == "m" && yards > 0 && yards <= 250) || (gender == "f" && yards > 0 && yards <= 210))
    par = 3;
```

Notice that we can combine as many ANDs **'&&'** and ORs **'||'** as we wish. Also notice the use of brackets **'( )'**. Remember that due to BEDMAS everything inside the brackets are equated separately from everything else. Therefore, our first set of brackets we have:

```
(gender == "m" && yards > 0 && yards <= 250)
```

Here we are testing if the **'gender'** is male **AND** if the **'yards'** is between 1 **AND** 250. This entire statement is enclose in it's own set of brackets **'( )'.**

Next, we have a logical OR **'||'** separating our two sets of brackets (left set of brackets is testing male, right set of brackets is testing female).

Finally, we have our second set of brackets:

```
(gender == "f" && yards > 0 && yards <= 210)
```

Here we are testing if the **'gender'** is female **AND** if the **'yards'** is between 1 **AND** 210. This entire statement is enclose in it's own set of brackets **'( )'.**

Overall, what we see is that the variable **'par'** will be set to **'3'** if the 'gender' is **male** and the **'yards'** are between **1** and **250** **OR** if the 'gender' is **female** and the **'yards'** is between **1** and **210.**

All the remaining if-statement conditions work in a similar manner. Making use of the logical operators in such a way has allowed us to keep our code clean and our logic in one place. The point of this program is to determine the 'par' variable. If we did not use logical operators like above, then our code could have gotten very messy. For example, we may have had something like this:

```
if (gender == 'm')
{
   if(yards > 0)
   {
      if(yards <= 250)
      {
         par = 3;
      }
      else if(yards <= 470)
      {
         par = 4;
      }
      .
      .
      .
   }
}
else if (gender == 'f')
{
   if(yards > 0)
   {
      if(yards <= 210)
      {
         par = 3;
      }
      else if(yards <= 400)
      {
         par = 4;
      }
      .
      .
      .
   }
}
```

The above approach is **not ideal.** There are way more lines of code, and we are repeatedly setting the 'par' variable in different places (for example, 'par = 3' appears twice, once in the male if-block and once in the female if-block).

Keeping your code and logic simple and efficient takes practice. Try your best to be aware of repeating code/logic and make use of logical operators when you can.