Combining boolean expressions in C#

José Augusto Thomas | Unity Expert

Using bools as expressions

As we've seen in the lecture on Conditional Statements, they analyze nothing but an expression. An expression always returns a boolean value, that is, can be either **true** or **false**.

In our example, we looked at age being at least 21 in order to allow a person to buy alcoholic beverage.

```
if (age >= 21) {
        //body of our if statement
}
```

So, base on this example, *age >= 21* is an expression that will return either **true** or **false**.
Since age >= 21 returns a boolean value, it can be obviously assigned as the value of a bool variable. Therefore, the following code is valid:

```
bool anExpression = age >= 21;
```

This bool is going to hold either true or false. This bool value can now be used as the expression or the if statement. We can rewrite the code given by I in the following way without loosing any functionality:

```
bool anExpression = age >= 21;
if (anExpression) {
        //body of our if statement
}
```

Combining bools to form more sophisticated expressions

So now we're going to see how to combine several expression together by using logical elements. By combining expressions, I mean checking if a person is at least 21 years old, male and with a valid ID, for example. We can implement a structure just by using if statements one inside another and it would look like this (assume we have defined variables for gender and ID's validity):

```
if (age >= 21) {
        if (gender == "Male") {
              if (idSituation == "valid") {
                    // code to be executed in case these three
                    // conditions are valid
              }
        }
}
```

So we need to pass through these three expression in order to execute the body of the most inner statement, since the body of the other two are if statement themselves.

We can reduce this to just one if statement by making usage of logical operands. The main ones are as follows:

- && - logical AND (All combined expressions must be true in order to execute the body of the if statement).
- ll - logical OR (At least one of the combined expressions must be true in order to execute the body of the if statement).
- && and ll can be combined together to create unique combinations.

Now, let's say that we want to use logical operand && to combine those three expressions together. The code would look like this:

```
if (age >= 21 && gender == "Male" && idSituation == "valid")
{
        // code to be executed in case this chained expression
        // is true
}
```

Moving a little bit further. Let's say that people more than 60 years old don't have to have a valid ID checked, but people between 21 and 60 years old still do. This this we would have 2 ands sided with 1 or operation:

```
if ((age >= 21 && idSituation == "valid") || age > 60) {
        // code to be executed in case this chained expression
        // is true
}
```

Note that the first AND operation is enclosed in parenthesis, that means that condition will be verified as a whole before compared to the OR statement.
Remembering: AND - all conditions must be true; OR - at least one has to be true.
So this is the same as comparing **(true) ll false** or maybe (**true) ll true** whereas the first bool value came from the AND verification.