

Forum:[Hack Forums](#)

Thread: [Conditional Statements - Padrino's Tutorials: Part 3](#)

padrino - June 18, 2025, 5:15 am

Introduction

Ok, so we've learnt how to get our programs running and we now know how to use variables and data types correctly. Next up, we're going to be learning about conditional statements in C#. Essentially, these act as logical gateways that allow our code to take certain paths depending on some condition we set. We actually used one of them in our last tutorial, but here we will go into more details. This will be a relatively short tutorial, but will provide a good baseline for building on what we learnt last week. So, let's get into it:

If Statements

If statements are a pretty easy concept to grasp, but they are a super important foundation of C# and indeed most if not all programming languages. So what do they look like? Here's an example building on what we did last week:

```
Code    //boolean
        bool example = true;

        //if statement
        if(example){
            Console.WriteLine("looks like the boolean was true.");
        }else{
            Console.WriteLine("looks like the boolean was false");
```

As you can see we start the If statement with, you guessed it, the if keyword. We then follow this with a parenthesis and within those, we give the condition we want to evaluate. In the example above, we gave it a boolean variable we set at the start of the code: (example). We then follow that with a set of braces {}, and within those we put whatever code we want to run in the event that the condition evaluates to true. In this example, we just tell it to output a line to the console like we have done in previous tutorials: Console.WriteLine("looks like the boolean was true.");.

Now, we could leave it there, and if the condition evaluated to false, the program would just ignore the code within the if statement and carry on. And sometimes that's fine, but sometimes we want to have the code to something else specifically if the condition evaluates to false. That's where the else key word comes in. Essentially we're telling the code to evaluate the condition in the if statement, if it's true, do something, otherwise, do something else. In the example above, we just get it to output to the console again: Console.WriteLine("looks like the boolean was false");

Now, that's great, but what if there's more than 2 possible outcomes? That's where we can use the else if keywords. Here's a more complex example below:

```
Code    //generate a random number and assign it to an integer variable
        Random randNumber = new Random();
        int number = randNumber.Next(1, 15);

        //if statement to check the numbers value
        if(number >= 1 && number < 5){
```

```

    Console.WriteLine("The random number was less than 5");
}else if (number >= 5 && number <10){
    Console.WriteLine("The random number was less than 10");
}else{
    Console.WriteLine("The random number was greater than 10");
}

```

The code above may look intimidating but it's actually pretty simple if you break it down. First we're simply generating a random number between 1 and 15 and assigning it to the variable name number. We then use that number in our if statement and check its value. The first if checks if the number is greater than or equal to 1 and less than 5 if it is (i.e. it evaluates to true) then the code will output the message: The random number was less than 5. We then have another specific condition we want to check, so we use the else if keywords and check if the value is equal to or greater than 5 and less than 10. If it is, we output the message: The random number was less than 10. Finally, we want to put a line where it will output a message for all other eventualities. We know that the only outcome here is if the number is greater than 10 so we don't need to provide a specific condition so we just use the else keyword to output the message: The random number was greater than 10.

You can run this code as many times as you like and you will see that you get a different output depending on what number is generated due to the if statement we have implemented. You can add as many conditions as you like, there's really no limit on the number of else ifs you can use. However, it's not the most efficient as it will evaluate every condition before continuing on to the next block of code. Sometimes a switch statement is needed - and we'll go over those below.

Switch Statements

Switch statements are very much like if statements, they evaluate a condition and use it to choose which code block to execute, but in large applications with a lot of conditions, they're far more efficient, not to mention easier to read and follow. Below is an example of a switch statement in use:

```

Code    //generate a random number and assign it to an integer variable
        Random randNumber = new Random();
        int number = randNumber.Next(1, 10);

        //switch statement to check the numbers value
switch (number)
{
    case 1:
        Console.WriteLine("The random number is 1.");
        break;
    case 2:
        Console.WriteLine("The random number is 2.");
        break;
    case 3:
        Console.WriteLine("The random number is 3.");
        break;
    case 4:
        Console.WriteLine("The random number is 4.");
        break;
    case 5:
        Console.WriteLine("The random number is 5.");
        break;
    case 6:
        Console.WriteLine("The random number is 6.");
}

```

```

        break;
    case 7:
        Console.WriteLine("The random number is 7.");
        break;
    default:
        Console.WriteLine("The random number is greater than 7.");
        break;
}

```

So how does a switch statement work? Well, it's very similar to the if statements we've just been looking at. As you can see from the code above, we generate a random number between 1 and 10 and assign it to the number variable. We then declare the switch statement using the switch keyword and pass it a value to evaluate, in this case it's the number variable. We then provide it with a number of cases, each one outlining a specific condition relating to the value we have passed into it. In our example each case is used to check what the number is, and then output a string to the console to say what number was generated. For example, the specific case of our variable number being 6 the console will output The random number is 6. You will also notice that there's a default case right at the end - that will execute when no other condition is met - much like the else section in our if statements. The final thing to spot is the break lines - these tell the code to exit the switch statement and to not evaluate any other conditions.

You can also use these switch statements with other data types for example if you wanted to evaluate a string. For example:

```

Code    //assign name value
        string number = "padrino";

        //switch statement to check name
        switch (number)
        {
            case "bastille":
                Console.WriteLine("username is bastille.");
                break;
            case "omni":
                Console.WriteLine("username is omni.");
                break;
            case "padrino":
                Console.WriteLine("username is padrino.");
                break;
            default:
                Console.WriteLine("username not known");
                break;
        }

```

Conclusion

That's it! You've learnt how to use conditionals in your code, feel free to take the code examples I've given you and use them and edit them. Play around and get to grips with how they work. Next week we'll be going further into the fundamentals and using loops - exciting.

I hope you found this tutorial useful, and as always my DM's are always open for anyone that wants advice or help with their code.

Happy Coding!

Amazing high quality content right here. Very well explained with all major points narrowed down, Always high quality. Thank you for knowledge contribution.

padrino - June 23, 2025, 11:59 am

(Jun 23, 2025 02:04 AM)ro0ti Wrote: [\[CITE\]](#) Amazing high quality content right here. Very well explained with all major points narrowed down, Always high quality. Thank you for knowledge contribution.

My pleasure mate! Next one will be dropping fairly soon.

Lookdadon - June 24, 2025, 2:08 pm

just an small addon:

to avoid repeatedly if/else, operator can be used and we can use use implicit typing and its simplify the whole process and let the compiler defining its type and is strong as named variable.

Code //diff way to instanciate but same thing

//bool instanciated with no value is default false

//as same as int is default 0

```
bool isSkyRed;
```

```
var isSkyRed = false;
```

```
bool isSkyRed = false;
```

```
string myResponse1 = "You are right";
```

```
var myResponse2 = "Wrong answer";
```

```
//operator: if True myResponse1 else myResponse2
```

```
if(isSkyRed) ? myResponse1 : myResponse2;
```

```
so usefull
```

and nice tut on c#?

padrino - June 26, 2025, 5:31 am

(Jun 24, 2025 02:08 PM)Lookdadon Wrote: [\[CITE\]](#) just an small addon:

to avoid repeatedly if/else, operator can be used and we can use use implicit typing and its simplify the whole process and let the compiler defining its type and is strong as named variable.

Code //diff way to instanciate but same thing

//bool instanciated with no value is default false

//as same as int is default 0

```
bool isSkyRed;
```

```
var isSkyRed = false;
```

```
bool isSkyRed = false;
```

```
string myResponse1 = "You are right";
```

```
var myResponse2 = "Wrong answer";
```

```
//operator: if True myResponse1 else myResponse2
```

```
if(isSkyRed) ? myResponse1 : myResponse2;
```

```
so usefull
```

and nice tut on c#?

Thanks! And yeah so for anyone following this thread and wanting to know more what's being referenced here is something called a ternary operator. They're super handy for 1 line conditionals.

Thanks for the assist!

Dirty - June 28, 2025, 8:15 am

(Jun 26, 2025 05:31 AM)padrino Wrote: [\[CITE\]](#) Thanks! And yeah so for anyone following this thread and wanting to know more what's being referenced here is something called a ternary operator. They're super handy for 1 line conditionals.

Thanks for the assist!

Personally not a big fan of ternary operators, mostly from the readability standpoint I think they basically add close to nothing. A singular if-else statement isn't that unreadable and a nesting of ternary operators would make me block a PR for it because the readability plummets every nest you do.

padrino - June 28, 2025, 4:24 pm

(Jun 28, 2025 08:15 AM)SIM-3719807 Wrote: [\[CITE\]](#) Personally not a big fan of ternary operators, mostly from the readability standpoint I think they basically add close to nothing. A singular if-else statement isn't that unreadable and a nesting of ternary operators would make me block a PR for it because the readability plummets every nest you do.

Oh for sure I wouldn't nest a ternary. That would be a nightmare for readability. But I often use them for setting variables based on a conditional and simple things like that etc.

I think they absolutely have their uses

Lookdadon - June 30, 2025, 4:39 am

So much high quality lessons u are giving here. Anyone who wanna learn how to "code" have it served and ready?