Conditionals

+4

Published Mar 2, 2023 · **Updated Oct 19, 2024**

Contribute to Docs →

In C#, **conditionals** compare inputs and return a boolean value indicating whether it evaluates to `true` or `false`.

Conditional statements include the `if`, `else` and `else if` statements. A shorthand for the `if` / `else` statement is the conditional or ternary operator.

If Statement

An `if` statement evaluates a condition that, when true, will run a code block that follows.

In the example below, three variables are declared and assigned values. Then, an `if` statement checks for a condition; if it evaluates to true, then the variable of boolean type will change:

```
var input1 = 10;
var input2 = 10;
var output = false;

if (input1 == input2) {
  output = true; // Sets the output from false to true.
}
```

If the code above returned `true`, the code block below will print a statement to the console:

```
if (output == true) {
  Console.WriteLine("I returned true");
}
```

Note `==` means equal and `!=` means not equal.

Else Statements

An `else` statement is combined with the `if` statement. In the case that the condition following the `if` statement returns `false`, the code block following the `else` statement will run.

In the example below, three variables are assigned values:

```
var input1 = 10;
var input2 = 10;
var output = false;

// If the input variables are not equal, the output will be set to true.
if (input1 != input2) {
  output = true;
} else {
  output = false;
}

if (output == true) {
  // If the output is true, the following string will be printed.
  Console.WriteLine("I returned true");
} else {
  // Otherwise, the string within this else block will be printed.
  Console.WriteLine("I returned false");
}
```

Since the output is `false`, this will output:

```
I returned false
```

Else If Statements

An `else if` statement comes after an `if` statement and is used if an extra comparison is needed before an `else` statement.

```
// Four variables are declared here.
var input1 = 10;
var input2 = 10;
var input3 = 5;
var output = false;

// If input1 is equal to input3 then set the variable of output to true.
if (input1 == input3) {
  output = true;
// If input1 is equal to input2, then set the variable of output to true as well.
} else if (input1 == input2) {
  output = true;
// If the two conditions above are false, the else code block will run.
} else {
  output = false;
}

if (output == true) {
  Console.WriteLine("I returned true");
} else if (output == false) {
  Console.WriteLine("I returned false");
} else {
  Console.WriteLine("Error");
}
```

Above, the `else if` condition was true so the output was reassigned a value of `true`. This would run the code block in the first `if` block which will output:

```
I returned true
```

Conditional Operator

The conditional operator `?:` also known as the ternary operator, checks a boolean output and returns one of two results depending on whether the condition is true or false. The ternary operator can be read in pseudocode as follows:

```
Is this condition true ? Run this if yes : Run this if no;
```

In the example below, the condition that is checked is if `input1` is equal to 10. If that condition is true, it returns the first string. Otherwise, it returns the second string:

```
string getInput1(int input1) => input1 === 10 ? "I returned true" : "I returne

Console.WriteLine(getInput1(10)); // Output: "I returned true"
Console.WriteLine(getInput1(5)); // Output: "I returned false"
```

Codebyte Example

Run the following codebyte example to understand how conditionals work in C#:

Code

Output

```
using System;

class Program
{
    static void Main()
    {
        int number = 10;

        // Using if-else conditiona
        if (number > 0)
        {
            Console.WriteLine("The 
        }
        else if (number < 0)
```

Run

All contributors

- Anonymous contributor
- @CaupolicanDiaz
- @Christine_Yang
- Anonymous contributor
- Anonymous contributor

Contribute to Docs

- Learn more about how to get involved.
- Edit this page on GitHub to fix an error or make an improvement.
- Submit feedback to let us know how we can improve Docs.

Learn C# on Codecademy

Career path

Full-Stack Engineer

A full-stack engineer can get a project done from start to finish, back-end to front-end.

Includes **51 Courses**

With **Professional Certification**

Beginner Friendly **150** hours

Free course

Learn C#

Learn Microsoft's popular C# programming language, used to make websites, mobile apps, video games, VR, and more.

Beginner Friendly **23** hours

Back to top