If-Else Statement

This lesson discusses if-else statements in detail including nested-ifs and if-else if-else statements using examples

WE'LL COVER THE FOLLOWING

Programming in general often requires a *decision* or a *branch* within the code to account for how the code operates under different inputs or conditions.

Within the **C#** programming language, the simplest and sometimes the most useful way of creating a branch within your program is through an `If-Else` statement.

If-Else Block

As with most of C#, the if statement has the same syntax as in C, C++, and Java. Thus, it is written in the following form:

```
if (condition)
{
  // Do something
}
else
{
  // Do something else
}
```

if-else

> **Note:** The two execution sections are *mutually exclusive*. Meaning only **one** of the two sections can execute at a time based on the result of the boolean expression.

The `if` statement evaluates its **condition** expression to determine whether to execute the `if`-body. Optionally, an `else` clause can immediately follow the `if` body, providing code to execute when the condition is `false`.

If-else Example

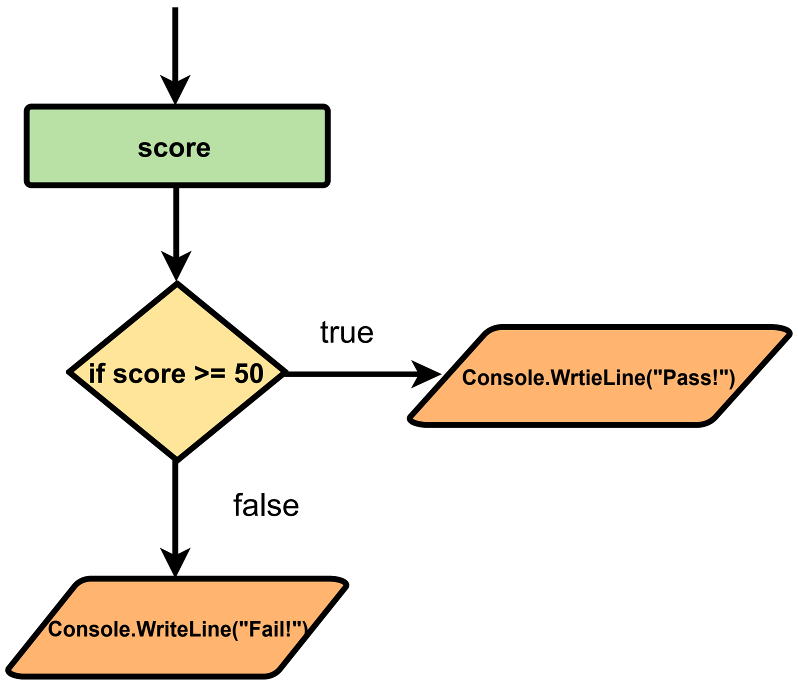
Let’s take a look at an example implementing the `if-else` statement.

In the example below, you’ll pass a value of **score** for a subject and the program will output whether you *passed* or *failed* the course depending on the score.

> **IMPORTANT NOTE:** In the example below before you click `RUN` to execute the code click the `>_STDIN` button first. An input bar will appear. Enter any value of score there and then click the `RUN` button. If you don’t enter the value first a run time **ERROR** will occur.

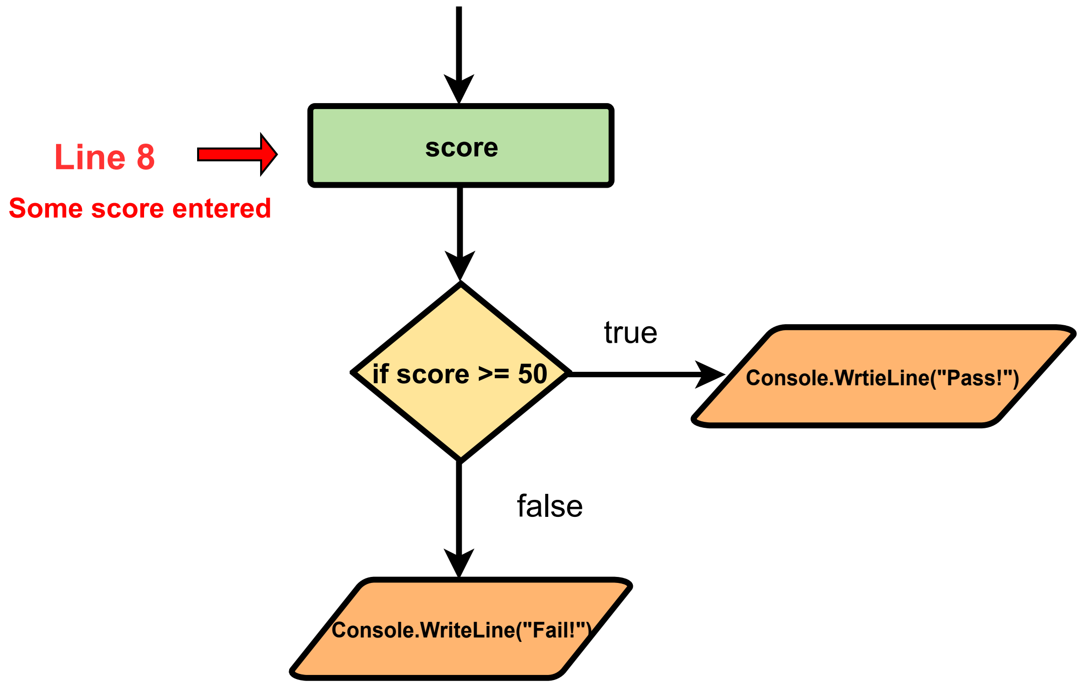
```
using System;

class IfElseExample
{
  static void Main(){
    double score;
    score = double.Parse(Console.ReadLine());
    Console.WriteLine("Your score: {0}",score);
    if (score >= 50) // If score is greater or equal to 50
    {
      Console.WriteLine("Pass!");
    }
    else // If score is not greater or equal to 50
    {
      Console.WriteLine("Fail!");
    }
  }
}
```



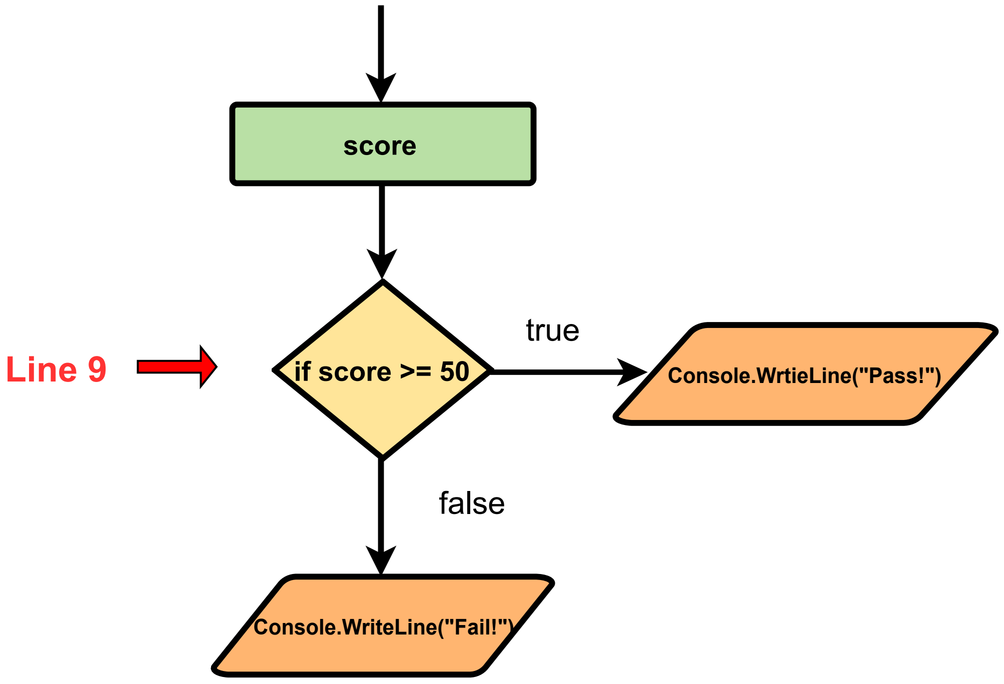
Flowchart of if-else variation

1 of 6



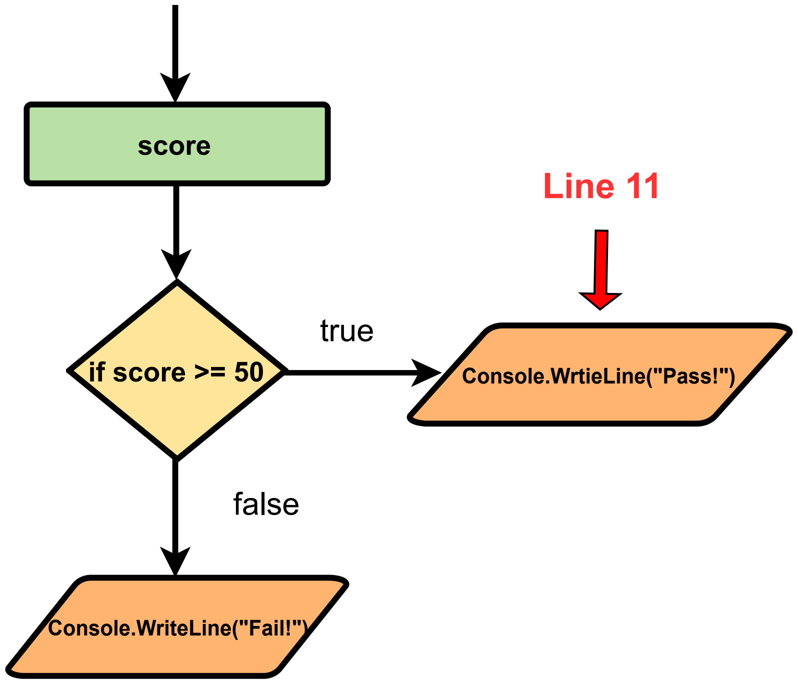
Flowchart of if-else variation

2 of 6



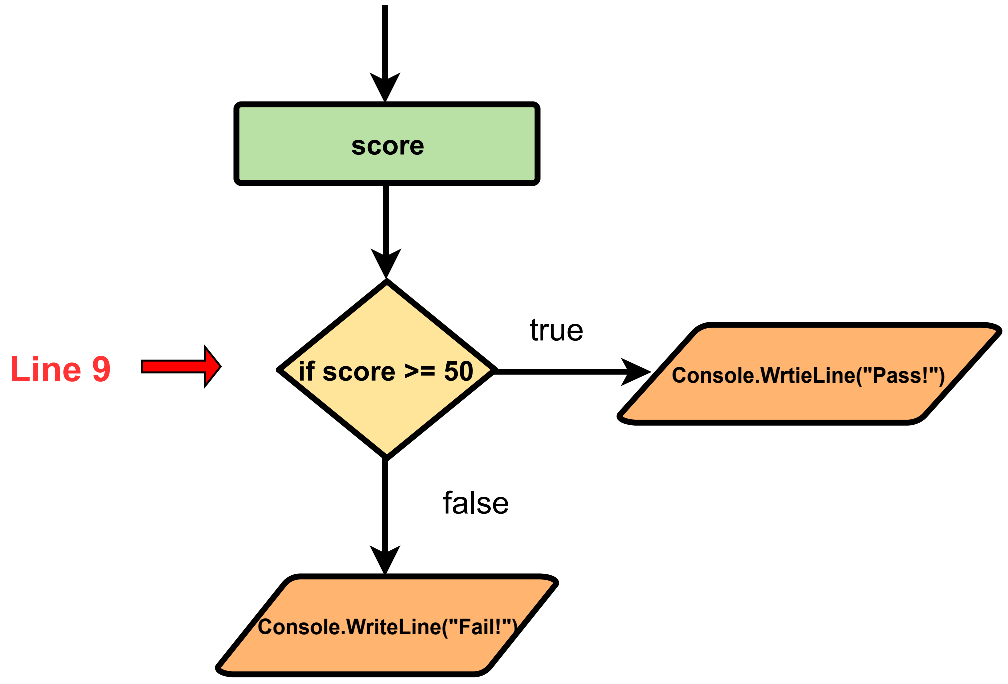
Flowchart of if-else variation

3 of 6



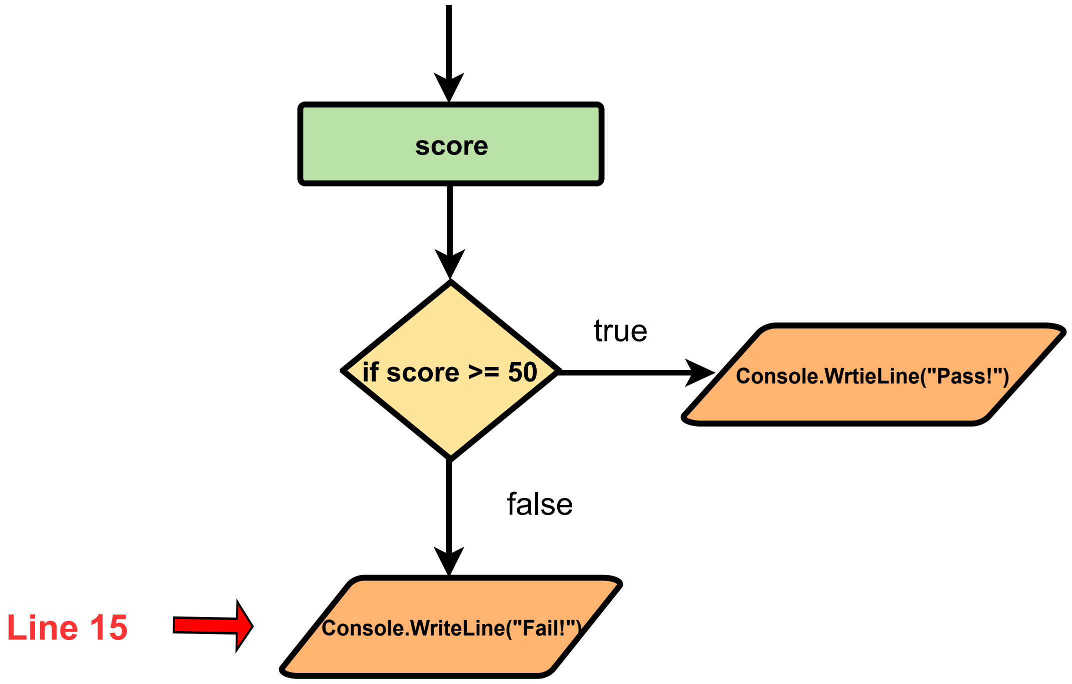
Flowchart of if-else variation

4 of 6



Flowchart of if-else variation

5 of 6



Flowchart of if-else variation

6 of 6

Code Explanation

In this example

- We first enter a **score** through the console.
- In line 7 we parse and convert the score to type `double`.
- In the line **9** we have the code `score >= 50` inside the `if` statement.
    - This can be seen as a **boolean** condition, where if the *condition* is evaluated to equal **true**, then the code that is in between the `if { }` runs.
- If the `score` entered was **60** then the *output* would display **Pass!** since the *parameter* value of **60** is **greater** or **equal** to **50**.
- However, if the `score` entered was **30** then the output would display **Fail!** since the value **30** is **not greater** or **equal** to **50**, thus the code in between the `else { }` runs instead of the `if` statement.

Nested-If

Here one If-else block is nested within another. One has to be careful in closing the inner if before closing the outer if.

```
if(condition1)
{
  //execution statement(s)
 if(condition2)
 {
  //execution statement(s)
 }
 else
 {
   //execution statement(s)
 }
}
else
{
  //execution statement(s)
}
```

Nested-if

In the code block above if

- `condition1` is **true** then
    - after executing the **execution statement(s)** code flows to the `condition2` check.
    - Based on the result either **execution statement(s)** of the nested `if` or `else` are executed.
- `condition1` is **false** then
    - code flows to `else` statement in **line 13** and the **execution statements** of this `else` get executed.

Else-If Statement

This is the complex **nested** form of `if-else` block. Here the `else` statement gets replaced by **one** or **many** `else-if` statements and there could be an optional ending `else` statement.

Let's take a look at an example below to better understand the concept of **nested ifs**.

In this example, we'll modify the *score testing* example from above to include more checks using **nested ifs**.

> **IMPORTANT NOTE:** In the example below before you click `RUN` to execute the code click the `>_STDIN` button first. An input bar will appear. Enter any value of score there and then click the `RUN` button.

```
using System;

class NestedifExample
{
    static void Main()
    {
        double score;
        score = double.Parse(Console.ReadLine()); //reading score value from command line and
        Console.WriteLine("Your score: {0}",score);
        if (score > 100) // If score is greater than 100
        {
            Console.WriteLine("Error: score is greater than 100!");
        }
        else if (score < 0) // Else If score is less than 0
        {
            Console.WriteLine("Error: score is less than 0!");
        }
        else if (score >= 50) // Else if score is greater or equal to 50
        {
            Console.WriteLine("Pass!");
        }
        else // If none above, then score must be between 0 and 49
        {
            Console.WriteLine("Fail!");
        }
    }
}
```

Code Explanation

- All the statements in the code above will run in order from the top all the way to the bottom until a *condition* has been met.
- In this new update of the code, we've added **two** new branches
  - In line **10** we check if `score` is greater than **100**
  - In line **14** we check if `score` is less than **0**
- Now if we enter a `score` of **110** the output displayed would be: **Error: score is greater than 100!**
- If we if we enter a `score` of **-20** the output displayed would be: **Error: score is less than 0!**

This marks the end of our discussion on `if-else` statements. Let's take a look at **switch** statements in the next lesson!