Conditional Statement

A statement that can be executed based on a condition is known as a "Conditional Statement". The statement is often a block of code. C# conditional statements allow you to branch your code depending on a certain value of an expression or condition. C# has two constructs for branching code, the ***if*** statement and the ***switch*** statement.

C# supports the usual logical conditions from mathematics:
Less than: a < b
Less than or equal to: a <= b
Greater than: a > b
Greater than or equal to: a >= b
Equal to a == b
Not Equal to: a!= b

You can use these conditions to perform different actions for different decisions.

C# has the following conditional statements:

1. *Use **if** to specify a block of code to be executed, if a specified condition is true*
2. *Use **if-else** to specify a block of code to be executed, if the same condition is false*
3. *Use **if and else if** to specify a new condition to test, if the first condition is false*
4. *Use **switch** to specify many alternative blocks of code to be executed*

1. The if Statement

Use the if statement to specify a block of C# code to be executed if a condition is True other block will not executed if the condition is false.

Syntax

```
if (condition)
{
  // block of code to be executed if the condition is True
}
```

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.

Example-- In the example below, we test two values to find out if 20 is greater than 18. If the condition is True, print some text:

```
if (20 > 18)
{
  Console.WriteLine("20 is greater than 18");
}
```

OUTPUT-
20 is greater than 18

We can also test variables:

```
int x = 20;
int y = 18;
if (x > y)
{
  Console.WriteLine("x is greater than y");
}
```

OUTPUT-
20 is greater than 18

Example explained
In the example above we use two variables, x and y, to test whether x is greater than y (using the > operator). As x is 20, and y is 18, and we know that 20 is greater than 18, we print to the screen that "x is greater than y".

2. The if-else Statement

Use the if-else statement to specify a block of code to be executed with if and else conditions. If block will be executed if the condition is True otherwise else block will be executed in case condition is False.

Syntax

```
if (condition)
```

```
{
  // block of code to be executed if the condition is True
}
else
{
  // block of code to be executed if the condition is False
}
```

Example-

```
int time = 20;
if (time < 18)
{
  Console.WriteLine("Good day.");
}
else
{
  Console.WriteLine("Good evening.");
}
```

Outputs

Good evening.

Example explained

In the example above, time (20) is greater than 18, so the condition is False. Because of this, we move on to the else condition and print to the screen "Good evening". If the time was less than 18, the program would print "Good day".

3. The if with else-if Statement

Use the if with else-if statement to specify multiple conditions and execute one specific block of code, depends on true condition. First compiler will check whether the if condition is true or false. So if, if condition is true then if block will execute otherwise next else if blocks will be tested and execute that block which will true. If no condition is true then else block will execute. Else block is optional.

Syntax

```
if (condition1)
{
  // block of code to be executed if condition1 is True
}
else if (condition2)
{
  // block of code to be executed if the condition1 is false and condition2 is True
}
else if (condition3)
{
  // block of code to be executed if the condition1 and condition2 is false and
condition3 is True
}
else
{
  // block of code to be executed if the condition1, condition2 and condition3 is False
}
```

Example-

```
int time = 22;
if (time < 12)
{
  Console.WriteLine("Good morning.");
}
else if (time < 16)
{
  Console.WriteLine("Good afternoon.");
}
else if (time < 20)
{
  Console.WriteLine("Good evening.");
}
else
{
  Console.WriteLine("Good night.");
}
```

Outputs
Good night.

Example explained
In the example above, time (22) is greater than 12, so the first condition is False. The next condition (time is greater than 16), in the else if statement, is also False, The next condition (time is greater than 20), in the else if statement, is also False, so we move on to the else condition since condition1, condition2 and condition3 are False - and print to the screen "Good night".

However, if the time was 14, our program would print "Good afternoon."

4. Switch construct

The switch statement allows you to handle program flow based on predefined sets of conditions. It takes a ***switch*** argument followed by a series of ***case*** clauses. The syntax of the switch construct is as in the following:

Syntax

```
switch(argument)
{
    case 1:
    // Do anything
    break;
    case 2:
    // Do anything
    break;
    default:
    // Do anything
    break;
}
```

When the expression in the switch argument is evaluated, the code immediately following the case clause executes and it marks the end of statements by the ***break*** clause. If the expression evaluates to none of the other clause then you can include a ***default*** clause.

Note: The order of case doesn't matter; you can put the default case first.

The following example performs some short math operations like addition, multiplication and so on. You can ask the user at run time in form of choices what operation he wants to do. Then we read two numeric values from the console and execute the operation as selected earlier.

```
Static void Main(string[] args)
    {
      int x,y;
      Console.WriteLine("Enter two Integers");
      x = int.Parse(Console.ReadLine());
      y = int.Parse(Console.ReadLine());
      Console.WriteLine("Operations\n----------------------------------\n");
      Console.WriteLine("1= Addition\n2= Substraction \n3= Multification");
      Console.Write("Enter the operation code::");
      int op = int.Parse(Console.ReadLine());

      switch (op)
      {
        case 1:
          Console.WriteLine("Add=" + (x + y));
          break;
        case 2:
          Console.WriteLine("Subs=" + (x - y));
          break;
        case 3:
          Console.WriteLine("Multiply=" + (x * y));
          break;
        default:
          Console.WriteLine("wrong choice");
          break;
      }
      Console.ReadKey();
    }
```

Output

Depends on the user inupt

For practice-

Example 1

```
int data=10;
if (data !=5)
 {
  Console.WriteLine("value is not equal");
 }
```

Example 2

For example in the following program we want to determine whether or not a string is longer than zero characters:

```
static void Main(string[] args)
{

  string str = "welcome home";

  if (str.Length > 0)
    Console.WriteLine("lenght is > 0");
  else
    Console.WriteLine("lenght is < 0");

  Console.ReadKey();
 }
```

Example 3

```
int x=10, y=20,z=22;
if (x >y)
  {
    Console.WriteLine("x is Highest");
```

```
  }
else if (x > z)
  {
    Console.WriteLine("x is Highest");
  }
else if (y > z)
  {
    Console.WriteLine("y is Highest");
  }
else
  {
    Console.WriteLine("z is Highest");
  }
```

Example 4

```
switch (country)
  {
     case "india":
     goto case "Canada";
     case "USA":
     break;
     case "Canada":
     break;
  }
```