**Conditionals (if statement)**

C# provides many decision making statements that help the flow of the C# program based on certain logical conditions. C# includes the following decision making statements.

1. if statement
2. if-else statement
3. switch statement
4. Ternary operator :?

**if statement**

Syntax:

if(boolean expression)

{

// execute this code block if expression evalutes to true

}

The 'if' statement contains boolean expression inside brackets followed by a single or multi line code block. At runtime, if a boolean expression is evaluates to true then the code block will be executed.

Consider the following example where the if condition contains true as an expression.

if(true)

{

Console.WriteLine("This will be displayed.");

}

if(false)

{

Console.WriteLine("This will not be displayed.");

}

The if statement can contain boolean expression. An expression which returns either true or false. Following example uses logical expression as a condition:

int i = 10, j = 20;

if (i > j)

{

Console.WriteLine("i is greater than j");

}

if (i < j)

{

Console.WriteLine("i is less than j");

}

if (i == j)

{

Console.WriteLine("i is equal to j");

}

**if-else statement**

C# also provides for a second part to the if statement, that is else. The else statement must follow if or else if statement. Also, else statement can appear only one time in an if-else statement chain.

Syntax:

if(boolean expression)

{

// execute this code block if expression evalutes to true

}

else

{

// always execute this code block when above if expression is false

}

As you can see in the above syntax, the else statement cannot contain any expression. The code block that follows else statement will always be executed, when the 'if' condition evaluates to be false.

int i = 10, j = 20;

if (i > j)

{

Console.WriteLine("i is greater than j");

}

else

{

Console.WriteLine("i is either equal to or less than j");

}

**else if statement:**

The 'if' statement can also follow an 'else' statement, if you want to check for another condition in the else part.

static void Main(string[] args)

{

int i = 10, j = 20;

if (i > j)

{

Console.WriteLine("i is greater than j");

}

else if (i < j)

{

Console.WriteLine("i is less than j");

}

else

{

Console.WriteLine("i is equal to j");

}

}

You can use multiple else-if statements in a single 'if' statment chain. Also, you can remove the curly brackets, when the 'if' block has only one line to execute:

int i = 10, j = 20;

if (i > j)

Console.WriteLine("i is greater than j");

else if (i < j)

Console.WriteLine("i is less than j");

else if (i == j)

Console.WriteLine("i is equal to j");

**Nested if statements:**

C# allows nested if else statements. The nested 'if' statement makes the code more readable.

int i = 10;

if (i > 0)

{

if (i <= 100)

{

Console.WriteLine("i is positive number less than 100");

}

else

{

Console.WriteLine("i is positive number greater than 100");

}

}

**Points to Remember :**

1. if-else statement controls the flow of program based on the evaluation of the boolean expression.
2. It should start from the if statement followed by else or else-if statements.
3. Only one else statement is allowed in the if-else chain.
4. Multiple else-if statements are allowed in a single if-else chain.
5. Nested if-else statement is allowed.