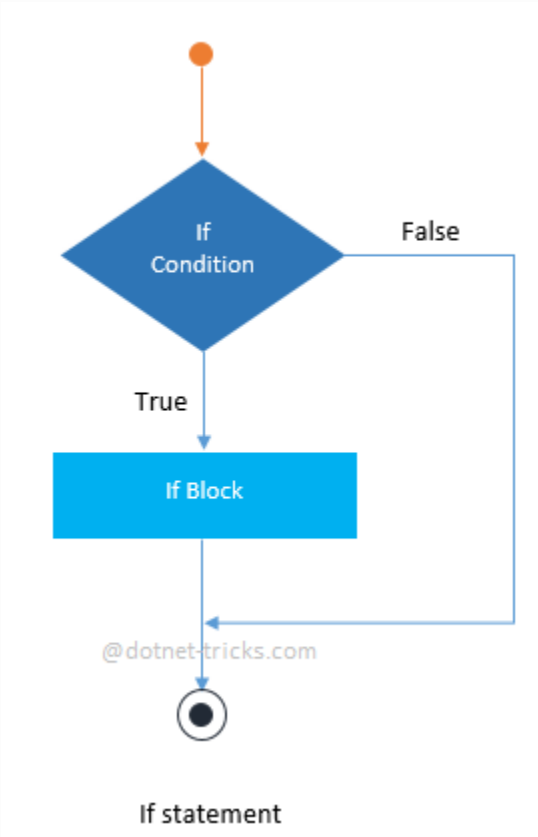
UNDERSTANDING DECISION MAKING STATEMENTS IN C#

Decision making statements help you to make decision based on certain conditions. These conditions are specified by a set of decision making statements having boolean expressions which are evaluated to a boolean value true or false. There are following types of decision making statements in C#.

1. If statement

An if statement consists of a boolean expression which is evaluated to a boolean value. If the value is true then if block is executed otherwise next statement(s) would be executed.



If statement

You can have multiple if statement as shown below-

```
1. public class Example
2. {
3.  static void Main()
4.  {
```

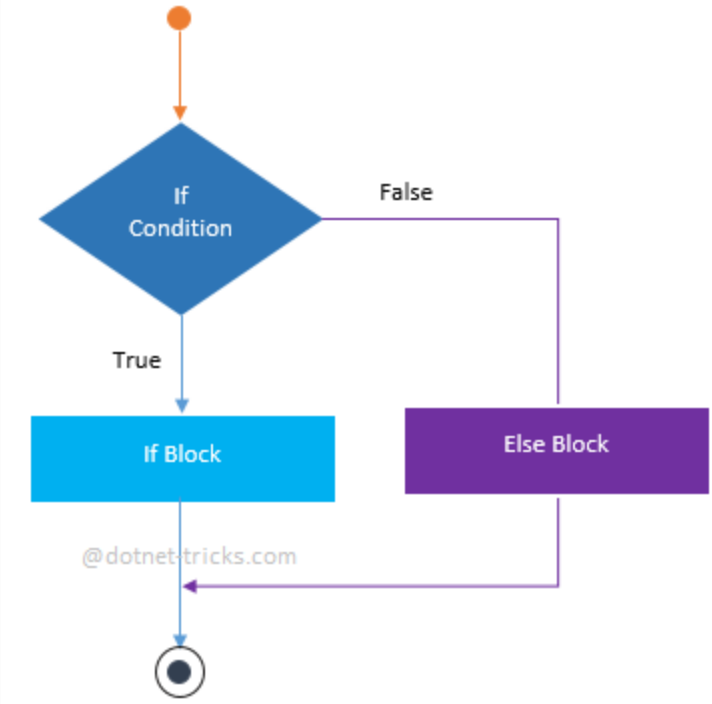
```
int a = 5, b = 2;
int result = a / b;

if (result == 2)
{
 Console.WriteLine("Result is 2");
 }
 if (result == 3)
 {
 Console.WriteLine("Result is 3");
 }
 }
}
/* Output
 Result is 2
 */
```

You can also do nesting of if statement means an if statement inside another if that is called nested if statement.

2. If-Else statement

An if-else statement consists of two statements – if statement and else statement. When the expression in an if-statement is evaluated to true then if block is executed otherwise the else block would be executed.



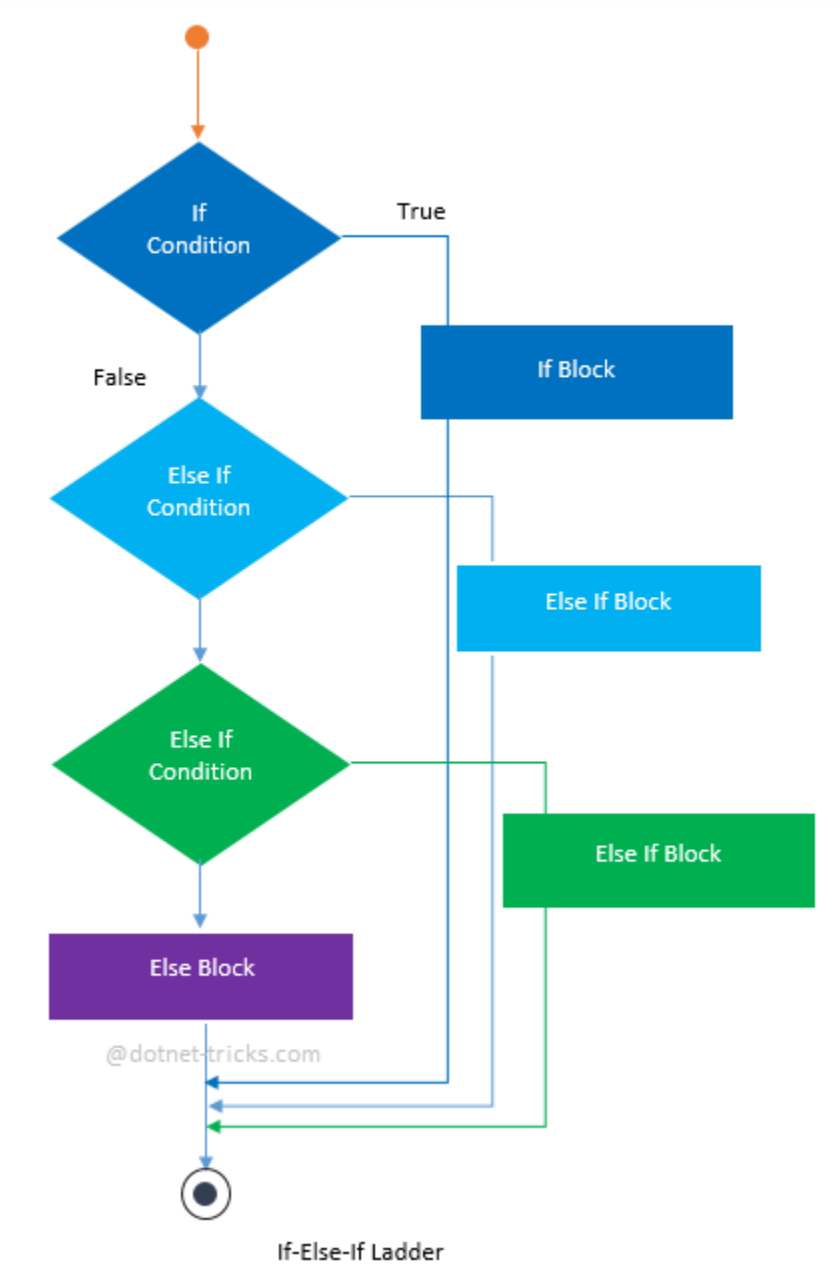
If-Else statement

```
1. public class Example
2. {
3.  static void Main()
4.  {
5.  int a = 5, b = 6;
6.  int result = a - b;
7.  if (result > 0)
8.  {
9.  Console.WriteLine("Result is greater than zero");
10.  }
11.  else
12.  {
13.  Console.WriteLine("Result is smaller than or equal to zero");
14.  }
15.  }
16. }
17. /* Output
18. Result is smaller than or equal to zero
```

```
19.   */
```

3. If-Else-If statement or ladder

The If-Else-If ladder is a set of statements that is used to test a series of conditions. If the first if statement meet the result then code within the if block executes. If not, control passes to the else statement, which contains a second "if" statement. If second one meet the result then code within the if block executes. This continues as a series of else if statements. A default else code block may execute when no condition has been evaluated to true.



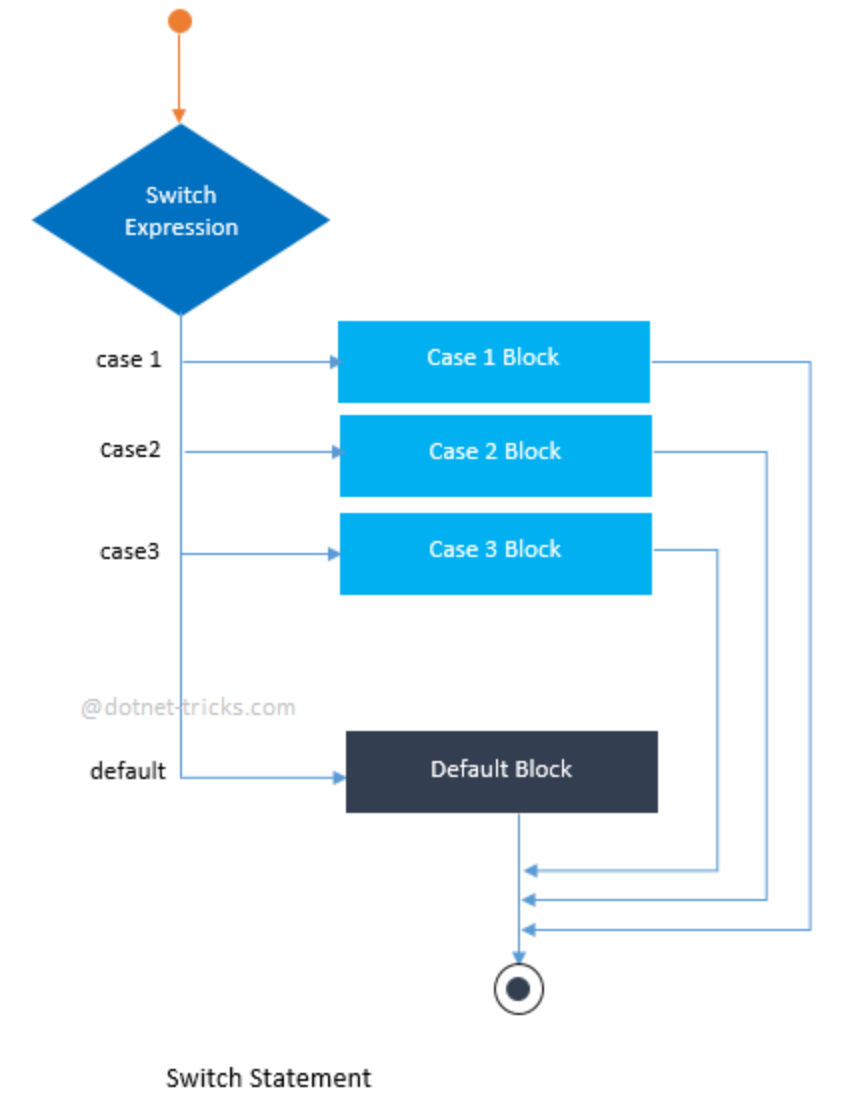
If-Else-If Ladder

If-Else-If ladder must contain more specific case at the top and generalize case at the bottom.

```
1. public class Example
2. {
3.  static void Main(string[] args)
4.  {
5.  char grade = 'B';
6.
7.  if (grade == 'A')
8.  {
9.  Console.WriteLine("Excellent!");
10.  }
11.  else if (grade == 'B')
12.  {
13.  Console.WriteLine("Well done");
14.  }
15.  else if (grade == 'D')
16.  {
17.  Console.WriteLine("You passed");
18.  }
19.  else if (grade == 'F')
20.  {
21.  Console.WriteLine("Better try again");
22.  }
23.  else
24.  {
25.  Console.WriteLine("You Failed!");
26.  }
27.  }
28. }
29. /* Output
30.  Well done
31. */
```

4. Switch statement

Switch statement acts as a substitute for long If-Else-If ladder that is used to test a series of conditions. A switch statement contains one or more case labels which are tested against the switch expression.



Switch Statement

When one case matches the value with the result of switch expression, the control continues executing the code from that label. When no case label contains a matching value, control is transferred to the default section, if it exists. If there is no default section, no action is taken and control is transferred outside the switch statement.

```
1. public class Example
2. {
3.  static void Main(string[] args)
4.  {
5.  char grade = 'B';
6.
7.  switch (grade)
```

```
{
case 'A':
 Console.WriteLine("Excellent!");
 break;
 case 'B':
 case 'C':
 Console.WriteLine("Well done");
 break;
 case 'D':
 Console.WriteLine("You passed");
 break;
 case 'F':
 Console.WriteLine("Better try again");
 break;
 default:
 Console.WriteLine("You Failed!");
 break;
 }
 }
}
/* Output
 Well done
*/
```

Key points about Switch statement

31. Each case label specifies a constant value.
32. A switch statement can have multiple switch sections, and each section can have one or more case labels.
33. Unlike C and C++, C# does not allow continuing execution from one switch section to the next. It means each switch section must be separated by a break or other jump statement such as goto, return and throw.
34. Unlike If-Else-If ladder, it is not mandatory to put more specific case at the top and generalize case at

Source : http://www.dotnet-tricks.com/Tutorial/csharp/PVO1280414-Understanding-decision-making-statements-in-C#.html