

## Program Control Structure

- is responsible for the flow of execution of Program.

### There are 3 types of Program Control Structure

#### 1. Sequence/ Sequential

- **Simplest** form of Program Control Structure.
- Example is Basic Input/Output.

#### 2. Conditional

- also known as **Selection Control Structure**.
- It allows user to select a certain course of action based on the condition being evaluated.
- Examples are **if, if-else, nested if and switch case statements**.

#### 3. Repetition

- also known as **Iteration**.
- It allows user to execute statements several times.

Examples are **for loop, do-while loop, while loop, for each loop, nested loop**.

## Conditional Constructs C#

- Conditional constructs are used to transfer execution control to the correct path based on comparison result.
- The conditional constructs determine runtime that which statements need to be executed. It uses comparison result for determining correct path of execution.
- **If else, switch case** are used for comparing value.



## 1.1 C# If Else Constructs

- The if... else construct is used for determining the flow of program based on returning expression value.
- It evaluates the comparison operator and based on value executes the statements.
- For example, if you want to execute a piece of code when the requirements meet then **if... else** construct determine which piece of code will be executed.
- **else** is default condition and executes when no if condition matches.

### 1.1.1 if — Conditional Statement

The main format (or syntax) of the conditional statement **if** is as follows:

```
if (Boolean expression)
{
    Body of the conditional statement;
}
```

The syntax includes: **if-clause**, **Boolean expression** and **body of the conditional statement**.

The **Boolean expression** can be a **Boolean variable** or **Boolean logical expression**.

The **body of the statement** is the part locked between the curly brackets { }. It may consist of one or more operations (statements). When there are several operations, we have a complex block operator, i.e. series of commands that follow one after the other, enclosed in curly brackets.

**Note:** Always put curly brackets { } for the body of “if” blocks even if they consist of only one statement!

## Sample Program

```
using System;
```

```
namespace Bigger2ndNumber
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```
            Console.WriteLine("Enter two numbers.");
```

```
            Console.Write("Enter first number: ");
```

```
            int firstNumber = Convert.ToInt32(Console.ReadLine());
```

```
            Console.Write("Enter second number: ");
```

```
            int secondNumber = Convert.ToInt32(Console.ReadLine());
```

```
            int biggerNumber = firstNumber;
```

```
            if (secondNumber > firstNumber)
```

```
            {
```

```
                Console.WriteLine("The bigger number is: {0}", secondNumber);
```

```
            }
```

```
            Console.ReadKey(true);
```

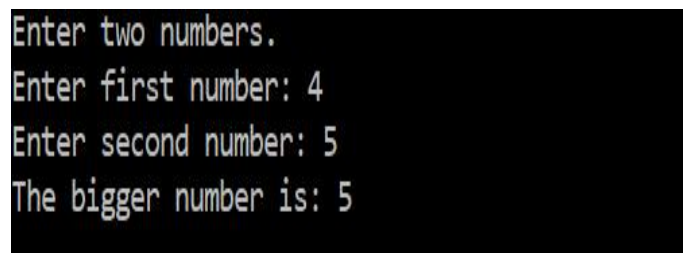
```
        }
```

```
    }
```

```
}
```

## Output

If we start the example and enter the numbers 4 and 5 we will get the following result:



```
Enter two numbers.  
Enter first number: 4  
Enter second number: 5  
The bigger number is: 5
```

And if we enter the numbers 5 and 4 we will get the following result:

```
Enter two numbers.  
Enter first number: 5  
Enter second number: 4
```

### 1.1.2 if-else — Conditional Statement

In C#, as in most of the programming languages there is a conditional statement with **else clause**: the if-else statement. Its format is the following:

```
if (Boolean expression)  
{  
    Body of the conditional statement;  
}  
else  
{  
    Body of the else statement;  
}
```

The format of the if-else structure consists of the reserved word **if**, **Boolean expression**, **body of a conditional statement**, reserved word **else** and **else-body statement**.

The **body of else-structure** may consist of one or more operators, enclosed in curly brackets, same as the body of a conditional statement.

This statement works as follows:

- the expression in the brackets (a Boolean expression) is calculated.
- The calculation result must be Boolean — true or false. Depending on the result there are two possible outcomes.
- If the Boolean expression is calculated to true, the body of the conditional statement is executed and the else-statement is omitted and its operators do not execute.
- Otherwise, if the Boolean expression is calculated to false, the else-body is executed, the main body of the conditional statement is omitted and the operators in it are not executed.

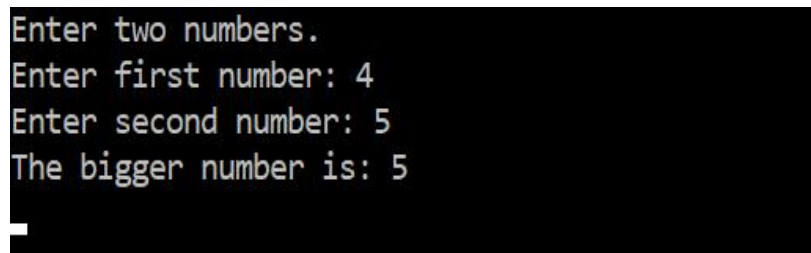
## Sample Program

```
using System;

namespace BiggerNumberIfElse
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Enter two numbers.");
            Console.Write("Enter first number: ");
            int firstNumber = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter second number: ");
            int secondNumber = Convert.ToInt32(Console.ReadLine());
            int biggerNumber = firstNumber;
            if (secondNumber > firstNumber)
            {
                Console.WriteLine("The bigger number is: {0}", secondNumber);
            }
            else
            {
                Console.WriteLine("The bigger number is: {0}", firstNumber);
            }
            Console.ReadKey(true);
        }
    }
}
```

## Output

If we start the example and enter the numbers 4 and 5 we will get the following result:



```
Enter two numbers.
Enter first number: 4
Enter second number: 5
The bigger number is: 5
```

And if we enter the number 5 and 4 we will get the following result:

```
Enter two numbers.  
Enter first number: 5  
Enter second number: 4  
The bigger number is: 5
```

### 1.1.3 Ladderized if (if...else if...else) – Conditional Statement

An **if** statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.

When using if, else if and else statements there are few points to keep in mind.

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of the remaining else if's or else's will be tested.

The syntax of an **if...else if...else** statement in C# is:

```
if(boolean_expression 1)
{
    /* Executes when the boolean expression 1 is true */
}
else if( boolean_expression 2)
{
    /* Executes when the boolean expression 2 is true */
}
else if( boolean_expression 3)
{
    /* Executes when the boolean expression 3 is true */
}
else
{
    /* executes when the none of the above condition is true */
}
```

## Sample Program

```
. using System;
. using System.Collections.Generic;
. using System.Linq;
. using System.Text;
.
. namespace if_else
. {
.     class Program
.     {
.         static void Main(string[] args)
.         {
.             int opt, num1, num2;
.             float result;
.
.             label:
.
.             Console.WriteLine("\n\tMenu");
.             Console.WriteLine("\nPress 1 for add");
.             Console.WriteLine("Press 2 for subtraction");
.             Console.WriteLine("Press 3 for multiplication");
.             Console.WriteLine("Press 4 for Division");
.
.             Console.Write("\n\nEnter first number:\t");
.             num1 = Convert.ToInt32(Console.ReadLine());
.
.             Console.Write("Enter second number:\t");
.             num2 = Convert.ToInt32(Console.ReadLine());
.
.             Console.Write("\nEnter your option:\t");
.             opt = Convert.ToInt32(Console.ReadLine());
.
.             if (opt == 1)
```

```

.      {
.          result = num1 + num2;
.          Console.WriteLine("\n{0} + {1} = {2}", num1, num2, result);
.      }
.      else if (opt == 2)
.      {
.          result = num1 - num2;
.          Console.WriteLine("\n{0} - {1} = {2}", num1, num2, result);
.      }
.      else if (opt == 3)
.      {
.          result = num1 * num2;
.          Console.WriteLine("\n{0} x {1} = {2}", num1, num2, result);
.      }
.      else if (opt == 4)
.      {
.          result = (float)(num1 / num2);
.          Console.WriteLine("\n{0} / {1} = {2}", num1, num2, result);
.      }
.      else
.      {
.          Console.WriteLine("Invalid option. Try again");
.          goto label;
.      }
.      Console.ReadLine();
.    }
.  }
. }

```



## Output

```
Menu

Press 1 for add
Press 2 for subtraction
Press 3 for multiplication
Press 4 for Division

Enter first number : 16
Enter second number : 5

Enter your option: 4
16 / 5 = 3.2__
```

If you have more than one if construct, then you can use else if construct for evaluating expression. The C# also supports nested if else construct.

If you want to evaluate certain condition based on previous if, then you can use **nested if-else constructs** in C# programming.

### 1.2 C# Switch Case Constructs

**switch case** is also another condition constructs in C# programming that evaluate the condition as if else but the only difference is that it makes the program simpler and easier.

It is used when there is multiple if condition in a program.

It also includes a default value in Default statements. If no any case matches, then **default** statements executes and run the code.

The structure **switch-case** chooses which part of the programming code to execute based on the calculated value of a certain expression (**most often of integer type**).

The format of the structure for choosing an option is as follows:

```
switch (integer_selector)  
{  
  case integer_value_1:  
    statements;  
    break;  
  case integer_value_2:  
    statements;  
    break;  
  // ...  
  default:  
    statements;  
    break;  
}
```

The **selector** is an expression returning a resulting value that can be compared, like a number or string.

The **switch operator** compares the result of the selector to every value listed in the **case labels** in the body of the switch structure.

If a match is found in a **case label**, the corresponding structure is executed (simple or complex).

If no match is found, the **default** statement is executed (when such exists).

The value of the **selector** must be calculated before comparing it to the values inside the switch structure.

**The labels should not have repeating values; they must be unique.**

As it can be seen from the definition above, every case ends with the operator **break**, which ends the body of the switch structure.

The C# compiler requires the word **break** at the end of each **case-section** containing code.

If no code is found after a case-statement, the break can be omitted and the execution passes to the next case-statement and continues until it finds a break operator.

After the **default** structure break is **obligatory**.

### Sample Program

```
. using System;
. using System.Collections.Generic;
. using System.Linq;
. using System.Text;
.
. namespace switch_case
. {
.     class Program
.     {
.         static void Main(string[] args)
.         {
.             int opt, num1, num2;
.             float result;
.
.             label:
.
.             Console.WriteLine("\n\tMenu");
.             Console.WriteLine("\nPress 1 for add");
.             Console.WriteLine("Press 2 for subtraction");
.             Console.WriteLine("Press 3 for multiplication");
.             Console.WriteLine("Press 4 for Division");
.
.             Console.Write("\n\nEnter first number:\t");
.             num1 = Convert.ToInt32(Console.ReadLine());
.
.             Console.Write("Enter second number:\t");
.             num2 = Convert.ToInt32(Console.ReadLine());
.
.             Console.Write("\n\nEnter your option:\t");
.             opt = Convert.ToInt32(Console.ReadLine());
.
.         }
.     }
. }
```

```

.         switch (opt)
.         {
.             case 1:
.                 result = num1 + num2;
.                 Console.WriteLine("\n{0} + {1} = {2}", num1, num2, result);
.                 break;
.
.             case 2:
.                 result = num1 - num2;
.                 Console.WriteLine("\n{0} - {1} = {2}", num1, num2, result);
.                 break;
.
.             case 3:
.                 result = num1 * num2;
.                 Console.WriteLine("\n{0} * {1} = {2}", num1, num2, result);
.                 break;
.
.             case 4:
.                 result = (float)num1 / num2;
.                 Console.WriteLine("\n{0} / {1} = {2}", num1, num2, result);
.                 break;
.
.             default:
.                 Console.WriteLine("\nInvalid option. Please try again.");
.                 goto label;
.         }
.         Console.ReadLine();
.     }
. }
.

```

## Output

### Menu

Press 1 for add

Press 2 for subtraction

**Press 3** for multiplication

**Press 4** for **Division**

**Enter** first number : 22

**Enter** second number : 8

**Enter** your option: 422 / 8 = 2.75\_\_

#### Guideline to use Switch case:

- If you are accepting char value then write it within single quotes as follow: case '1', case 'b', case 'c', case 'k' etc.
- If you are accepting string value then write it within double quotes as follow: case "add", case "sub", case "mul", case "div" etc.

#### Engaging Activity

**Q1:** Write a program in which accept a number from the user and identify whether the number is even or odd.

**Answer:**

```
. using System;
. using System.Collections.Generic;
. using System.Linq;
. using System.Text;
.
. namespace even_number
. {
.     class Program
.     {
.         static void Main(string[] args)
.         {
.             int num;
.             Console.Write("Enter your number:\t");
.             num = Convert.ToInt32(Console.ReadLine());
.
.             if (num % 2 == 0)
.             {
.                 Console.WriteLine("{0} is Even Number", num);
```

```

.     }
.     else
.     {
.         Console.WriteLine("{0} is Odd number", num);
.     }
.     Console.ReadLine();
. }
. }
. }

```

### Output

Enter your number : 5656 is Even Number\_\_

**Q2:** Write a program in which accept a number from the user between 1 to 7 and display corresponding days starting with Monday.

**Hint:** Use switch case

### Answer:

```

. using System;
. using System.Collections.Generic;
. using System.Linq;
. using System.Text;
.
. namespace find_day
. {
.     class Program
.     {
.         static void Main(string[] args)
.         {
.             int opt;
.             label:
.             Console.Write("\n\nEnter your option (1-7) for days. 1 for Monday:\t");
.             opt = Convert.ToInt32(Console.ReadLine());
.

```

```

.      switch (opt)
.      {
.          case 1:
.              Console.WriteLine("Monday");
.              break;
.          case 2:
.              Console.WriteLine("Tuesday");
.              break;
.          case 3:
.              Console.WriteLine("Wednesday");
.              break;
.          case 4:
.              Console.WriteLine("Thursday");
.              break;
.          case 5:
.              Console.WriteLine("Friday");
.              break;
.          case 6:
.              Console.WriteLine("Saturday");
.              break;
.          case 7:
.              Console.WriteLine("Sunday");
.              break;
.          default:
.              Console.WriteLine("Invalid option. Please try again\n");
.              goto label;
.      }
.      Console.ReadLine();
.  }
.  }
.  }

```

**Output**

**Enter** your option <1-7> for days. 1 for **Monday** : 9

**Invalid** option. **Please** try again

**Enter** your option <1-7> for days. 1 for **Monday** : 2

**Tuesday**\_\_