Module – 2

8.Operators and Expressions

C# provides a number of operators. Many of them are supported by the built-in types and allow you to perform basic operations with values of those types. Those operators include the following groups:

- **Arithmetic operators** that perform arithmetic operations with numeric operands
    - Arithmetic Operators - +, -, /, *, %,++(Incremental), --(Decremental) etc.

- Assignment Operators – They are used for assigning a value to a variable.
    - Assignment Operators - =, +=, -=, *=, /=, etc.

- **Boolean logical operators** that perform logical operations with bool operands.
    - Logical Operators - &&, ||, !

- **Bitwise and shift operators** that perform bitwise or shift operations with operands of the integral types.
    - Bitwise Operators - &, |, ^, >>, <<, etc.

- **Relational Operators** They are used to compare operands to obtain a result.
    - Relational Operators - ==, !=, <, >, >=, <=, etc.

- **Misc Operators** –They perform miscellaneous tasks like giving type of operand, size of operand etc.
    - Misc Operators – size of(), type of(), ?:(Ternary), etc.

9.Loop Iteration

The following statements repeatedly execute a statement or a block of statements:

- **for statement**: executes its body while a specified Boolean expression evaluates to true.
    - Syntax -

```
for (statement1; statement2; statement3;)
{
    //code to be executed
 }
```

- **The foreach statement**: enumerates the elements of a collection and executes its body for each element of the collection.
    - Syntax -

```
foreach (type variablename in arrayname)
 {
 //code to be executed
 }
```

- **The do statement**: conditionally executes its body one or more times.
    - Syntax -

```
do
 {
        //code to be executed
 }
while(condition)
```

- **The while statement**: conditionally executes its body zero or more times.
    - Syntax -

```
while(condition)
{
    // code to be executed
 }
```

- **break**: It is used to jump out of a loop when a particular condition is met.
- **Continue**: It is used to continue the loop but breaking that particular iteration when continue statement is executed

10.Understanding Arrays

- You can store multiple variables of the same type in an array data structure. You declare an array by specifying the type of its elements.
- An array can be single-dimensional, multidimensional or jagged.
- The number of dimensions and the length of each dimension are established when the array instance is created. These values can't be changed during the lifetime of the instance.
- The default values of numeric array elements are set to zero, and reference elements are set to null.
- A jagged array is an array of arrays, and therefore its elements are reference types and are initialized to null.
- Arrays are zero indexed: an array with n elements is indexed from 0 to n-1.
- Array types are reference types derived from the abstract base type Array. All arrays implement IList, and IEnumerable. You can use the foreach statement to iterate through an array. Single-dimensional arrays also implement IList<T> and IEnumerable<T>.

Syntax:

< Typeof array > [ ] < arrayName >;

- Here first we must specify the data type along with the brackets [].
- After that we must specify the Name of the Array.

11.Defining and Calling Methods

- A method is a code block that contains a series of statements.
- A program causes the statements to be executed by calling the method and specifying any required method arguments.
- In C#, every executed instruction is performed in the context of a method.
- The Main method is the entry point for every C# application and it's called by the common language runtime.

Syntax:

```
< Access Specifier > < Return Type >  < Method Type >  ( < Parameter List > )
        {
                // method body
        }
```

- Access Specifier – describes the access of the method to classes in the program ex- public, private, protected, Internal.
- Return Type – can be a type of value which a method returns or it also can be void if a method returns nothing.
- Method name – method receives a user defined name.
- Parameter list – it is the list of parameters passed on as argument when it is called.

- Different type of parameters in method

    - Value type – It is the normal C# value parameter which means value is directly passed.
    - Ref. Type – References of a variable is passed as an argument which are assigned first. Argument passed consists of 'ref' keyword first. Any changes made in this argument in method will reflect on the variable in the calling method.
    - Optional or type – These are the type of arguments which may be passed only if operations on them are required otherwise the function consists default values of these parameters. This parameter should only be passed after required parameters

12.Working with strings

- A string is an object of type String whose value is text.
- Internally, the text is stored as a sequential read-only collection of Char objects.
- There is no null-terminating character at the end of a C# string.

- Use of various string methods

Property	**Description**
Copy()	Creates a new instance of string with the same value as a specified string.
Compare()	Used to compare the two string objects
Contains(String)	Returns a value indicating whether a specified substring occurs within this string.
Equals()	Determines whether two string objects have the same value.
Trim()	Returns a new string in which all leading and trailing occurrences of a set of specified characters from the current String object are removed.
ToString()	Converts the value into string type.
ToUpper()	Converts the value into uppercase.
ToLower()	Converts the value into lowercase.
ToCharArray()	Copies the characters in this instance to a Unicode character array.
Substring()	Retrieves a substring from this instance.
Split()	Returns a string array that contains the substrings in this instance that are delimited by elements of a specified string or Unicode character array.

13.Working with Datetime

- Represents an instant in time, typically expressed as a date and time of day.

- Properties of DateTime –
    - It gives us date and time
    - It gives us year, day of week, day of year, time of day.
    - Now() property gives the current date and time.