Module 2 Theory

1) An **operator** is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C# has rich set of built-in operators and provides the following type of operators −

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

2) A **loop statement** allows us to execute a statement or a group of statements multiple times.

C# provides the following type of loops −

- while loop - It repeats a statement or a group of statements while a given condition is true. It tests the condition before executing the loop body.
- for loop - It executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
- do...while loop - It is similar to a while statement, except that it tests the condition at the end of the loop body
- nested loops - You can use one or more loop inside any another while, for or do..while loop.

Loop **control statements** change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

C# provides the following control statements --

- break statement - Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
- continue statement - Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

A loop becomes **infinite loop** if a condition never becomes false. The for loop is traditionally used for this purpose. Since none of the three expressions that form the for loop are required, you can make an endless loop by leaving the conditional expression empty.

When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but programmers more commonly use the for(;;) construct to signify an infinite loop.

3) An **array** stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type stored at contiguous memory locations. A specific element in an array is accessed by an index.

To declare an array in C#, you can use the following syntax −

<Data type>[] <array name>;

where,

- datatype is used to specify the type of elements in the array.
- [ ] specifies the rank of the array. The rank specifies the size of the array.
- arrayName specifies the name of the array.

Declaring an array does not initialize the array in the memory. When the array variable is initialized, you can assign values to the array.

Array is a reference type, so you need to use the new keyword to create an instance of the array.When you create an array, C# compiler implicitly initializes each array element to a default value depending on the array type.

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array.

- Multi-dimensional arrays - C# supports multidimensional arrays. The simplest form of the multidimensional array is the two-dimensional array.
- Jagged arrays - C# supports multidimensional arrays, which are arrays of arrays.
- Passing arrays to functions - You can pass to the function a pointer to an array by specifying the array's name without an index.
- Param arrays - This is used for passing unknown number of parameters to a function.
- The Array Class - Defined in System namespace, it is the base class to all arrays, and provides various properties and methods for working with arrays.

4) A **method** is a group of statements that together perform a task. Every C# program has at least one class with a method named Main.

To use a method, you need to −

- Define the method
- Call the method

When we define a method, we basically declare the elements of its structure. Following are the various elements of a method −

- Access Specifier − This determines the visibility of a variable or a method from another class.
- Return type − A method may return a value. The return type is the data type of the value the method returns. If the method is not returning any values, then the return type is void.
- Method name − Method name is a unique identifier and it is case sensitive. It cannot be same as any other identifier declared in the class.

- Parameter list − Enclosed between parentheses, the parameters are used to pass and receive data from a method. The parameter list refers to the type, order, and number of the parameters of a method. Parameters are optional; that is, a method may contain no parameters.
- Method body − This contains the set of instructions needed to complete the required activity.

We can call a method using the name of the method. We can also call public method from other classes by using the instance of the class. A method can call itself . This is known as recursion.

When method with parameters is called, we need to pass the parameters to the method. There are three ways that parameters can be passed to a method −

- Value parameters - This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.
- Reference parameters - This method copies the reference to the memory location of an argument into the formal parameter. This means that changes made to the parameter affect the argument.
- Output parameters - This method helps in returning more than one value.

5) In C#, we can use **strings** as array of characters, However, more common practice is to use the string keyword to declare a string variable. The string keyword is an alias for the System.String class.

We can create string object using one of the following methods −

- By assigning a string literal to a String variable
- By using a String class constructor
- By using the string concatenation operator (+)
- By retrieving a property or calling a method that returns a string
- By calling a formatting method to convert a value or an object to its string representation

Strings have various properties and methods that help in working with the string objects.

6) We use the **DateTime** when there is a need to work with the dates and times in C#.We can format the date and time in different formats by the properties and methods of the DateTime.

The value of the DateTime is between the 12:00:00 midnight, January 1 0001 and 11:59:59 PM, December 31, 9999 A.D.

We have different ways to create the DateTime object. A DateTime object has Time, Culture, Date, Localization, Milliseconds.

The DateTime has the Date and Time property. From DateTime, we can find the date and time. DateTime contains other properties as well, like Hour, Minute, Second, Millisecond, Year, Month, and Day.

The other properties of DateTime are:

- We can get the name of the day from the week with the help of the DayOfWeek property.
- To get the day of the year, we will use DayOfYear property.
- To get time in a DateTime, we use TimeOfDay property.
- Today property will return the object of the DateTime, which is having today's value. The value of the time is 12:00:00
- The Now property will return the DateTime object, which is having the current date and time.
- The Utc property of DateTime will return the Coordinated Universal Time (UTC).
- The one tick represents the One hundred nanoseconds in DateTime. Ticks property of the DateTime returns the number of ticks in a DateTime.
- The Kind property returns value where the representation of time is done by the instance, which is based on the local time, Coordinated Universal Time (UTC). It also shows the unspecified default value.