RKIT Module – 2

Name: Adhyaru Keyur D.

Operators and Expressions

Ternary Operator

- ✓ Decision making operator?**:** which is called the conditional Operator or ternary Operator
- ✓ **Syntax:** condition **?** Statement 1(true) : Statement 2(false)
- ✓ If condition become true, then statement 1 Execute else statement 2 execute.

Nested Ternary Operator

- ✓ In the Nested Ternary operator, we must pass second conditional expression.
- ✓ **Syntax:** condition **? condition ? Statement 1(true) : Statement 2(false)** : Statement 2(false)

Increment and Decrement Operators

- ✓ Increment operator is used to increment the value by 1.
- ✓ Decrement Operator is used to decrement the value by 1.

Loop Iteration

For Loop

- ✓ for loop is used to execute block of code multiple times.
- ✓ Now to use for loop we have one Keyword: **for**

Syntax:

```
For (initializer; condition; iterator)

{

        //code

}
```

Initializer:

- ✓ used to initialize a variable that will be local to a for loop.
- ✓ It can also be zero or more assignment statements

Condition:

- ✓ The condition is a Boolean expression that will return either true or false.
- ✓ If an expression evaluates to true, then it will execute the loop again; otherwise, the loop is exited.

Iterator:

- ✓ The iterator defines the incremental or decremental of the loop variable.

Foreach Loop

- ✓ The foreach loop is used to iterate over the elements of the collection.
- ✓ The collection may be an array or a list.
- ✓ It executes for each element present in the array.

Syntax:

```
foreach(data_type var_name in collection_variable)
{
    // code
}
```

while Loop

- ✓ execute the block of code until the specify condition not become false.

Syntax:

```
While(condition)
{
        //code
}
```

- ✓ While loop start with **while** keyword.
- ✓ For loop contains the initialization part but, in the while, loop we have to initialization before use.
- ✓ To break the loop, we can use the break keyword.

Do-while Loop

- ✓ Do while loop execute the code before the condition check.
- ✓ It means first code will be executing and at the end it will check the condition.
- ✓ It will execute the block until condition become false.

Syntax:

```
do {
//code
} While(condition);
```

- ✓ Do-While loop start with **do** keyword.
- ✓ **Note:** Above mention loops can be nested.

Break statement

- ✓ The break statement is used to terminate the loop or statement.
- ✓ After that, the control will pass to the statements that present after the break statement.
- ✓ If the break statement presents in the nested loop, then it terminates only those loops which contains break statement.

Syntax: break;

- ✓ Using **break** keyword, we can break the loop or statements.

Continue statement

- ✓ This statement is used to skip over the execution part of the loop on a certain condition.
- ✓ After that, it transfers the control to the beginning of the loop.
- ✓ Basically, it skips its following statements and continues with the next iteration of the loop.

Syntax: continue;

- ✓ Using **continue** keyword, we can continue the next iteration of loop.

goto statement

- ✓ This statement is used to transfer control to the labelled statement in the program.
- ✓ The label is the valid identifier and placed just before the statement from where the control is transferred.

Understanding Arrays

- ✓ An array is a group of like-typed variables that are referred to by a common name. And each data item is called an element of the array.
- ✓ The data types of the elements may be any valid data type like char, int, float, etc.
- ✓ the elements are stored in a sequence.
- ✓ Length of the array specifies the number of elements present in the array.
- ✓ In C# the allocation of memory for the arrays is done dynamically.
- ✓ An Arrays are kind of objects; therefore, it is easy to find their size using the predefined functions.
- ✓ The variables in the array are ordered and each has an index beginning from 0.
- ✓ A C# array variable can also be declared like other variables with [] after the data type.
- ✓ C# array is an object of base type System.Array.
- ✓ A jagged array elements are reference types and are initialized to null.
- ✓ Array elements can be of any type, including an array type.

Syntax :

< Data Type > [ ] < NameArray >

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

- ✓ In C# there are three Types of Array.

1. One dimensional Array
2. Multidimensional Arrays
3. Jagged Arrays

One Dimensional Array

- ✓ In this array contains only one row for storing the values.
- ✓ All values of this array are stored contiguously starting from 0 to the array size.
- ✓ **Example:** int[] arrayint = new int[2];

Multidimensional Arrays

- ✓ The multi-dimensional array contains more than one row to store the values.
- ✓ It is also known as a Rectangular Array in C# because it's each row length is same.
- ✓ It can be a 2D-array or 3D-array or more.
- ✓ To storing and accessing the values of the array, one required the nested loop.
- ✓ **Example:** int[,] intarray = new int[4, 2];

<u>**Jagged Arrays**</u>

- ✓ An array whose elements are arrays is known as Jagged arrays it means “array of arrays “.
- ✓ The jagged array elements may be of different dimensions and sizes.
- ✓ It’s possible to mix jagged and multidimensional arrays.
- ✓ **Example:** int[][] arr1 = { new int[] { 1, 3, 5, 7, 9 }, new int[] { 2, 4, 6, 8 } };

Defining and Calling Methods

- ✓ Methods are generally the block of codes or statements in a program that gives the user the ability to reuse the same code which ultimately saves the excessive use of memory, acts as a time save.
- ✓ more importantly, it provides a better readability of code.
- ✓ So basically, a method is a collection of statements that perform some specific task and return the result to the caller. A method can also perform some specific task without returning anything.

Method Declaration

- ✓ Method declaration means the way to construct method including its naming.

Syntax:

- <Access_Modifier> <return_type> <method_name>([<param_list>])

Method Calling

- ✓ Method Invocation or Method Calling is done when the user wants to execute the method.
- ✓ The method needs to be called for using its functionality.
    - A method returns to the code that invoked it when:
    - It completes all the statements in the method
    - It reaches a return statement
    - Throws an exception

Method Parameters

- ✓ There might be certain situations the user wants to execute a method but sometimes that method requires some value inputs in order to execute and complete its tasks.
- ✓ These input values are known as Parameters in a computer language terms.
- ✓ Now, these parameters can be either int, long or float or double or char. However, it depends upon the user requirements.
- ✓ The methods in C# can be classified into different categories based on return type as well as input parameters.

Advantages of using the Methods:

- ✓ There are many advantages of using methods. Some of them are listed below:
    - It makes the program well structured.
    - Methods enhance the readability of the code.
    - It provides an effective way for the user to reuse the existing code.
    - It optimizes the execution time and memory space.

Strings

A string is a series of characters that is used to represent text.

It can be a character, a word or a long passage surrounded with the double quotes ".

C# provides the String data type to store string literals.

There **two** ways to declare a string variable in C#.

Using **System.String** class and using **string** keyword.

Both are the same and make no difference.

Properties

Property	Description
Chars[Int32]	Gets the char object at a specified position.
Length	Gets the number of characters in the current string object.

Use of various string methods

Property	Description
Copy(String)	Creates a new instance of string with the same value as a specified string.
Compare()	Used to compare the two string objects
CompareTo()	
Contains(String)	Returns a value indicating whether a specified substring occurs within this string.
Equals()	Determines whether two string objects have the same value.
indexOf()	Return the position of specify char or string. Return -1 if char does not found.
Insert(int32, String)	Insert the new string at the given position.
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.
TrimEnd(Char[])	Removes all trailing occurrences of a set of characters specified in an array from the current String object.
TrimStart(Char[])	Removes all leading occurrences of a set of characters specified in an array from the current String object.
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.

Date Time Class

- ✓ C# DateTime is a structure of value Type like int, double etc.
- ✓ It is available in System namespace and present in mscorlib.dll assembly.
- ✓ DateTime helps developer to find out more information about Date and Time like Get month, day, year, week day.
- ✓ It also helps to find date difference, add number of days to a date, etc.
- ✓ It initializes a new instance of DateTime object.
- ✓ At the time of object creation we need to pass required parameters like year, month, day, etc

Example:

- DateTime date1 = new DateTime(2015, 12, 25);