Advanced Programming in C#

C# - Strings

- A **string** is an ordered sequence of characters, enclosed in double quotation marks.
- In C#, you 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.

Creating a String Object

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

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

Declaring strings in C#

```
string VariableName;
```

- where:

– **`string`** is a string type.

– **`VariableName`** specifies the name of the string variable.

Example:

```
    string first_name; //declaring a string
    variable named first_name
```

The following example demonstrates some of the above:

```
static void Main(string[] args) {
string fname, lname; //first method
fname = "Rowan";
lname = "Atkinson";

char []letters= { 'H', 'e', 'l', 'l','o' };
string [] sarray={ "Hello", "From", "Computer",
"Science" };

string fullname = fname + lname;  //Third method
Console.WriteLine("Full Name: "+ fullname);
string greetings = new string(letters); //second
method
Console.WriteLine("Greetings: "+ greetings);


  Console.WriteLine("Greetings:  "+ greetings);
  Console.Write("Message: ");
  for (int i = 0; i < sarray.Length; i++)
    {
      Console.Write(sarray[i] + " ");
    }
    Console.WriteLine();
    //fourth method
    string message = String.Join(" ", sarray);
    Console.WriteLine("Message2 " + message);
  }
```

- When the above code is compiled and executed, it produces the following result

```
Full Name: RowanAtkinson

Greetings: Hello

Message: Hello From Computer Science

Message2: Hello From Computer Science
```

Properties of the String Class

- The String class has the following two properties

No.	Property & Description
1.	**Chars[int]** Gets the Char object at a specified position in the current String object.
2.	**Length** Gets the number of characters in the current String object.

```
string s1 ="Department of Computer Science";
 char ch1 = s1[2];   //ch1 will be 'p'

 int size = s1.Length;
 Console.WriteLine("Size of the string = "
+size);   //will be 30
```

Methods of the String Class

- The String class has numerous methods that help you in working with the string objects.
- The following table provides some of the most commonly used methods:

No.	Methods & Description
1.	**public static int Compare(string strA, string strB)** Compares two specified string objects and returns an integer that indicates their relative position in the sort order. (0 if equal)
2.	**public static string Concat(string str0, string str1)** Concatenates two string objects.

3.	**public bool Contains(string value)** Returns a value indicating whether the specified String object occurs within this string.
4.	**public static string Copy(string str)** Creates a new String object with the same value as the specified string.
5.	**public bool EndsWith(string value)** Determines whether the end of the string object matches the specified string.
6.	**public int IndexOf(char value)** Returns the zero-based index of the first occurrence of the specified Unicode character in the current string.
7.	**public int LastIndexOf(char value)** Returns the zero-based index position of the last occurrence of the specified Unicode character within the current string object.
8.	**public string Remove(int startIndex, int count)** Removes the specified number of characters in the current string beginning at a specified position and returns the string.
9.	**public string Replace(char oldChar, char newChar)** Replaces all occurrences of a specified Unicode character in the current string object with the specified Unicode character and returns the new string.
10.	**public string ToLower()** Returns a copy of this string converted to lowercase.
11.	**public string ToUpper()** Returns a copy of this string converted to uppercase.
12.	**public string Substring(int start)**

	Returns a substring from the original string starting at start to the end.

Examples

- The following example demonstrates some of the methods mentioned above:

Comparing Strings

```
string str1 = "This is test";

string str2 = "This is text";


if (String.Compare(str1, str2) == 0)

{

Console.WriteLine(str1 + " and " + str2 +  " are
equal.");

}

else

{

Console.WriteLine(str1 + " and " + str2 + " are
not equal.");

}
```

- When the above code is compiled and executed, it produces the following result:

```
This is test and This is text are not equal.
```

String Contains String

```
string str = "This is test";

if (str.Contains("test"))
  {
 Console.WriteLine("The sequence 'test' was found.");
  }
```

- When the above code is compiled and executed, it produces the following result:

```
The sequence 'test' was found.
```

Getting a Substring

```
string str = "Last night I dreamt of San Pedro";
    Console.WriteLine(str);
    string substr = str.Substring(23);
    Console.WriteLine(substr);
```

- When the above code is compiled and executed, it produces the following result:

```
Last night I dreamt of San Pedro
San Pedro
```

String Concatenation

- The + operator can be used between strings to combine them. This is called concatenation:

```
string firstName = "John";
string lastName = "Steven";
string name = firstName + lastName;
Console.WriteLine(name);
```

- When the above code is compiled and executed, it produces the following result:

```
John Steven
```