String in C#

A string is a sequence of characters. We can use string keyword to create Strings in C#. The string keyword is alias name for System.String class in C#. It is recommended to use string as it works even without using System namespace.

For example,

```
string name = "John David";
```

Here, we have created a string named name and assigned the text "John David". We need to use double quotes to represent strings in C#.

A string variable in C# is not of primitive types like int, char, etc. Instead, it is an object of the String class.

Example Program: Create a string variable and display the same

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string name = "John David";
            Console.WriteLine("Hello " + name);
            Console.ReadKey();
        }
    }
}
```

Output: Hello John David

String Operations

C# string provides various methods to perform different operations on strings.

Get the Length of a string

To find the length of a string, we can use the Length property.

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string name = "Welcome";
            Console.WriteLine(name.Length);
            Console.ReadKey();
```

```
        }
    }
}
```

Output: 7

Extract a particular character from a string

In C#, the characters in a string are indexed starting from 0. Hence, the characters can be accessed or extracted via index. For example,

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string data = "Welcome";
            //to access W
            Console.WriteLine(data[0]);
            //to access c
            Console.WriteLine(data[3]);
            Console.ReadKey();
        }
    }
}
```

Output:
W
C

Extract all characters from a string

1. for loop with index

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string data = "Welcome";
            for(int i = 0; i < data.Length; i++)
            {
                Console.WriteLine(data[i]);
            }
            Console.ReadKey();
        }
```

```
    }
}
```

Output:

```
W
e
l
c
o
m
e
```

2. foreach loop

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string data = "Welcome";
            foreach (char ch in data)
            {
                Console.WriteLine(ch);
            }
            Console.ReadKey();
        }
    }
}
```

Output:

```
W
e
l
c
o
m
e
```

Exercise Question: Write a program to store the given text data and count the number vowels and display the vowels and count.

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
```

```
            Console.Write("Enter the Text: ");
            string data = Console.ReadLine();
            int count = 0;
            foreach (char ch in data)
            {
               if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'){
                    Console.Write(ch + ",");
                    count++;
                }
            }
            Console.WriteLine("\nTotal:" + count);
            Console.ReadKey();
        }
    }
}
```

Output:

```
Enter the Text: welcome to cse
e,o,e,o,e
Total: 5
```

Concatenate Two Strings

We can join two strings in C# using the Concat() method.

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string firstname = "John";
            string secondname = " David";
            string name = string.Concat(firstname, secondname);
            Console.WriteLine(name);
            Console.ReadKey();
        }
    }
}
```

Output:

```
John David
```

Compare Two Strings

We can make comparisons between two strings using the Equals() method. The Equals() method checks if two strings are equal or not.

The string Equals() method will perform case-sensitive string comparison.

For example,

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string a = "Hello";
            string b = "Hello";
            if (a.Equals(b))
                Console.WriteLine("Equal");
            else
                Console.WriteLine("Not Equal");
            Console.ReadKey();
        }
    }
}
```

Output:

```
Equal
```

The case-insensitive Comparison

If we want to perform case insensitive string comparison, we need to use the `StringComparison`.OrdinalIgnoreCase property as second argument inside the Equals method.

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            string a = "Hello";
            string b = "hello";
            if (a.Equals(b,StringComparison.OrdinalIgnoreCase))
                Console.WriteLine("Equal");
            else
                Console.WriteLine("Not Equal");
            Console.ReadKey();
```

```
        }
    }
}
```

Output:
Equal

Replace characters in a string

The **Replace()** method returns a new string by replacing each matching character/substring in the string with the new character/substring.

Replace(string oldValue, string newValue)

Example:

```
using System;
class Program
    {
        static void Main(string[] args) {
            string data = "Welcome";
            //replace 'e' with $
            data = data.Replace('e', '$');
            Console.WriteLine(data);
            Console.ReadKey();
        }
 }
```

Output: W$lcom$

Remove characters in a string

The String **Remove()** method removes a specified number of characters from the string.

Remove(int startIndex, int count)

```
using System;
class Program
    {
        static void Main(string[] args) {
            string data = "Sample Text";

            //remove Text
            Console.WriteLine(data.Remove(6));

            //remove Sample
            Console.WriteLine(data.Remove(0,7));

            Console.ReadKey();
        }
 }
```

```
Output:
Sample
Text
```

Strings are Immutable

This means, once we create a string, we cannot change that string.

For example, consider the following code.

```
string a = "Welcome ";
string b = "to cse";
a.Concat(b);
Console.WriteLine(a);
```

```
Output: Welcome
```

Here, we are using the Concat() method to add the string "Welcome " to the string “to cse”. But, the variable a still gives the old string “Welcome ”.

The reason behind this is, strings in C# are immutable objects. Hence, it creates a new object whenever the string are modified.

The new modified string object, "Welcome to cse", is released for garbage collection because no other variable holds a reference to it.

String interpolation

- In C#, we can use string interpolation to insert variables inside a string.
- For string interpolation, the string literal must begin with the $ character and the variables can be inserted via curly braces { }, for example,

```
string msg=$"The number is {num}";
```

Notice that,

- the string literal starts with $
- the name variable is placed inside the curly braces {}

Example Program

```
using System;
namespace DemoApplication
{
    class Program
    {
        static void Main(string[] args) {
            int num=100;
            string msg=$"The number is {num}";
```

```
                Console.WriteLine(msg);
                Console.ReadKey();
            }
        }
    }
```

Output: The number is 100

How to create an array of strings in C#?

An array of strings is created the same as an array for data types. For example,

```
class Program
    {
        static void Main(string[] args) {
            string[] fruits = { "Apple", "Mango", "Pineapple" };
            // print array elements
            foreach(string fruit in fruits)
            {
                Console.WriteLine(fruit);
            }
            Console.ReadKey();
        }
    }
```

Output:
Apple
Mango
Pineapple

Other Methods of C# string

1. Format()

The Format() method returns a formatted string based on the argument passed.

```
using System;
class Program
    {
        static void Main(string[] args) {
            string name = "John";
            string food = "Apple";
            string msg = string.Format("{0} eats {1}", name,food);
            Console.WriteLine(msg);
            Console.ReadKey();
        }
    }
```

Output: John eats Apple

Here,

```
string msg = string.Format("{0} eats {1}", name,food);
```

{0} is replaced by the first object passed in the method i.e. name

{1} is replaced by the second object passed in the method i.e. food

2. Spilt() method

The Split() method breaks up a string at the specified separator and returns its substrings. The default separator is space " ".

```
using System;
class Program
    {
        static void Main(string[] args) {
            string data = "Welcome to CSE";
            string[] list = data.Split();
            foreach (string s in list)
            {
                Console.WriteLine(s);
            }
            Console.ReadKey();
        }
 }
```

```
Output:
Welcome
to
CSE
```

We can also use any separator to split the string, for example the following string contains $

```
string data = "Wel$come$to$CSE";
string[] list = data.Split('$');
```

The output will be
Wel
come
to
CSE

3. Substring() method

The syntax of the string Substring() method is:

Substring(int startIndex, int length)

startIndex - the beginning index of the substring
length - (optional) - length of the substring

Example Code:

```
string data = "WelcomeToCSE";

//to extract substring 'come'
Console.WriteLine(data.Substring(3,4));

Output: come
```

4. ToCharArray() method

The **ToCharArray()** method copies the characters in the string to a character array.

Example Code:

```
string data = "Welcome";
char[] items = data.ToCharArray();
foreach (char c in items)
{
  Console.WriteLine(c);
}
```

Output:-

W
e
l
c
o
m
e

5. ToUpper(), ToLower(), Trim() Methods

The ToUpper() method converts all characters in the string to uppercase.
The ToLower() method converts all characters in the string to lowercase.
The Trim() method removes the space in front and rear side of the string

Example code:

```
string s1 = "HELLO";
Console.WriteLine(s1.ToLower());

string s2 = "hello";
Console.WriteLine(s2.ToUpper());
```

```
string s3 = "    John    ";
Console.WriteLine(s3.Trim());
```

Output
hello
HELLO
John

6. EndsWith(), StartsWith() method

- The String **EndsWith()** method checks whether the string ends with the specified string or not.
- The String **StartsWith()** method checks whether the string starts with the specified string or not.

Example code:

```
string data = "sample.pdf";
if (data.EndsWith(".pdf"))
    Console.WriteLine("PDF File");
else
    Console.WriteLine("Not a PDF file");

string[] fruits = { "Apple", "Mango", "Grapes", "Pear" };
foreach (string fruit in fruits)
{
  if(fruit.StartsWith("M"))
    Console.WriteLine(fruit);
}
```

Output:
PDF File
Mango

7. Join() Method

The **Join()** method joins the elements of an array using a specified separator.

Example:

```
string[] text = { "Apple", "Mango", "Orange" };
Console.WriteLine(String.Join(" ", text));
```

Output:
Apple, Mango, Orange

8. Contains() Method

The Contains() method checks whether the specified string is present in the string or not.

Example:

```
string text = "This is an important message";
```

```
if (text.Contains("important"))
    Console.WriteLine("Yes");
else
    Console.WriteLine("No");
```

Output:
Yes

7. IndexOf() Method

The String IndexOf() method returns the index of the first occurrence of the specified character/substring within the string.

Example

```
string str = "Ice cream";
// returns index of substring cream
int result =  str.IndexOf("cream");
Console.WriteLine(result);
```

Output: 4

8. LastIndexOf()

The LastIndexOf() method returns the index position of the last occurrence of a specified character or string within the given string.

Example

```
string str = "Ice cream Ice cream";
int result = str.LastIndexOf("cream");
Console.WriteLine(result);
```

Output: 14

9. PadLeft()

The String PadLeft() method returns a new string of a specified length in which the beginning of the current string is padded with spaces or with a specified character.

For Example:

```
string str = "John";
Console.WriteLine(str.PadLeft(10));
Console.WriteLine(str.PadLeft(10, '*'));
```

Output:
John
******John

10. PadRight()

The String **PadRight()** method returns a new string of a specified length in which the end of the current string is padded with spaces or with a specified character.

```
string str = "John";
Console.WriteLine(str.PadRight(10));
Console.WriteLine(str.PadRight(10, '*'));
```

Output:
John
John******