## C# string definition

A string is a sequence of characters. In C#, a string is a sequence of Unicode characters. It is a data type which stores a sequence of data values, usually bytes, in which elements usually stand for characters according to a character encoding. When a string appears literally in the source code, it is known as a string literal.

STRING REVERSE BY INBUILD FUNCTION

1. **class** Program
2. {
3. **static** **void** Main(**string**[] args)
4. {
5. **string** s = **string**.Empty;
6. s = Console.ReadLine();
7. **char**[] arr = s.ToCharArray();
8. Array.Reverse(arr);
9. Console.WriteLine(arr);
10. Console.ReadLine();
11. }
12. }

STRING REVERSE BY LOOPING

1. **class** Program
2. {
3. **static** **void** Main(**string**[] args)
4. {
5. **string** s = **string**.Empty;
6. s = Console.ReadLine();
7. **char**[] arr = s.ToCharArray();
8. **char**[] arr1 = **new** **char**[s.Length];
9. **string** s1 = **string**.Empty;
10. **for** (**int** i = s.Length - 1; i >= 0; i--)
11. {
12. s1 += arr[i];
13. }
14. Console.WriteLine(s1);
15. Console.ReadLine();
16. }
17. }

PALINDROME STRING

1. **class** Program
2. {
3. **static** **void** Main(**string**[] args)
4. {
5. **string** s = **string**.Empty;
6. s = Console.ReadLine();
7. **char**[] arr = s.ToCharArray();
8. **char**[] arr1 = **new** **char**[s.Length];
9. **string** s1 = **string**.Empty;
10. **for** (**int** i = s.Length - 1; i >= 0; i--)
11. {
12. s1 += arr[i];
13. }
14. **if** (s1 == s)
15. {
16. Console.WriteLine("Palindrome String");
17. Console.ReadLine();
18. }
19. **else**
20. {
21. Console.WriteLine("Not Palindrome String");
22. Console.ReadLine();
23. }
24. }
25. }

enter a string and count the number of capital letters, small letters, digits, spaces and special characters.

1. **class** Program
2. {
3. **static** **void** Main(**string**[] args)
4. {
5. **string** s = **string**.Empty;
6. s = Console.ReadLine();
7. **int** number = 0;
8. **int** capitalLetter = 0;
9. **int** smallletter = 0;
10. **int** space = 0;
11. **int** SpecialSymbol = 0;
12. **char**[] a = s.ToCharArray();
13. **for** (**int** i = 0; i < a.Length; i++)
14. {
15. **if** (a[i] >= 48 && a[i] <= 58)
16. {
17. number += 1;
18. }
19. **else** **if** (a[i] >= 65 && a[i] <= 90)
20. {
21. capitalLetter += 1;
22. }
23. **else** **if** (a[i] >= 97 && a[i] <= 122)
24. {
25. smallletter += 1;
26. }
27. **else** **if** (a[i] == 32)
28. {
29. space += 1;
30. }
31. **else**
32. {
33. SpecialSymbol += 1;
34. }
35. }
36. Console.Write("Capital Letter              " + capitalLetter);
37. Console.Write("\nsmall letter              " + smallletter);
38. Console.Write("\nnumber are                " + number);
39. Console.Write("\nTotal Spaces              " + space);
40. Console.Write("\nTotal Special Symbol are  " + SpecialSymbol);
41. Console.ReadLine();
42. }
43. }

String.compare(string,string2,[opyional[true|false])  
  
This function returns 0 if both strings are equal.  
  
It returns -1 if string2 is greater than string1 and returns 1 if string1 is greater than string2.  
It takes an optional parameter that is true or false. If we pass true than it ignores case for the comparison.

1. **string** s1 = "BIBEK";
2. **string** s2 = "BIBEK";
3. **int** n = **string**.Compare(s1, s2, **true**);
4. //int n = s1.CompareTo(s2);
5. Console.Write(n);
6. Console.ReadLine();
7. **string** s1 = "bIBEK";
8. **string** s2 = "BIBEK";
9. **int** n = **string**.Compare(s1, s2, **false**);
10. //int n = s1.CompareTo(s2);
11. Console.Write(n);
12. Console.ReadLine();

USE OF CONTINS IN STRING

1. **string** s1 = "The quick brown fox jumps over the lazy dog";
2. **string** s2 = "lox";
3. **bool** b;
4. b = s1.Contains(s2);
5. Console.Write(b);
6. Console.ReadLine();

**StartsWith**

static void Main()

{

// The input string.

string input = "http://site.com/test.html";

// See if input matches one of these starts.

if (input.**StartsWith**("http://www.site.com") ||

input.**StartsWith**("http://site.com"))

{

// Write to the screen.

Console.WriteLine(true);

}

}

**Foreach, StartsWith.**Next, we see that foreach can be used with StartsWith. Here we test elements in a string array against the input string, returning true if there is a match

static void Main()

{

// The input string.

string input = "http://site.com/test.html";

// The possible matches.

string[] m = new string[]

{

"http://www.site.com",

<http://site.com>,

<http://BIBEKANAND.com>,

};

// Loop through each possible match.

foreach (string s in m)

{

if (input.**StartsWith**(s))

{

// Will match second possibility.

Console.WriteLine(s);

return;

}

}

}

**EndsWith.** This tests the last parts of strings. It finds strings that have a certain ending sequence of characters. EndsWith is a simple way to test for ending substrings.

**First:**The EndsWith method, like its counterpart the StartsWith method, has three overloaded method signatures

static void Main()

{

// The input string.

string input = "http://site.com";

// Test these endings.

string[] arr = new string[]

{

".net",

".com",

".org"

};

// Loop through and test each string.

foreach (string s in arr)

{

if (input.**EndsWith**(s))

{

Console.WriteLine(s);

return;

}

}

}

**substring,** we extract a fragment of an existing string. A start and length (both ints) describe this view. Like an eagle's view, Substring() considers only a part.

static void Main()

{

string input = "OneTwoThree";

// Get first three characters.

string sub = input.**Substring**(0, 3);

Console.WriteLine("Substring: {0}", sub);

}

**One parameter.**This Substring overload receives just the start index int. The second parameter is considered the largest possible, meaning the substring ends at the last char.

**Program:**The program describes logic that takes all the characters in the input string excluding the first three.

static void Main()

{

string input = "OneTwoThree";

string sub = input.**Substring**(3);

Console.WriteLine("Substring: {0}", sub);

}

**Middle chars.**Here we take several characters in the middle of a string and place them into a new string. To take a middle substring, pass 2 arguments to Substring.

static void Main()

{

string input = "OneTwoThree";

string sub = input.**Substring**(3, 3);

Console.WriteLine("Substring: {0}", sub);

}

**Avoid chars.** Here we eliminate the last few chars in a string. This example eliminates the last five characters from the input string. It returns a new string without them.

static void Main()

{

string input = "OneTwoThree";

string sub = input.**Substring**(0, input.Length - 5);

Console.WriteLine("Substring: {0}", sub);

}

**Exceptions**are raised when Substring() is called with incorrect arguments. Exceptions can be scary, but they are trying to help you. This example triggers the ArgumentOutOfRangeException

static void Main()

{

string input = "OneTwoThree";

try

{

string sub = input.**Substring**(-1);

}

catch (Exception ex)

{

Console.WriteLine(ex);

}

try

{

string sub = input.**Substring**(0, 100);

}

catch (Exception ex)

{

Console.WriteLine(ex);

}

}

**One character.**It is possible to take a one-character substring. But if we simply use the string indexer to get a character, we will have better performance.

static void Main()

{

string value = "cat";

// ... In many programs, we can use a char instead of Substring.

Console.WriteLine(value[0]);

Console.WriteLine(value.**Substring**(0, 1));

}

SPLIT FUNCTION

**In a string too** we often find parts. These are separated with a delimiter. We can split lines and words from a string based on chars, strings or newlines.

static void Main()

{

string data = "bibeka nanda panigrahi";

// Split string on spaces (this will separate all the words).

string[] words = data.**Split**(' ');

**foreach** (string word in words)

{

Console.WriteLine("WORD: " + word);

}

}

static void Main()

{

// The directory from Windows.

const string dir = @"C:\Users\bibek\Documents\RubyonRail\test";

// Split on directory separator.

string[] parts = dir.**Split**('\\');

foreach (string part in parts)

{

Console.WriteLine(part);

}

}

**StringSplitOptions.**This affects the behavior of Split. The two values of StringSplitOptions (None and RemoveEmptyEntries) are integers (enums) that tell Split how to work

**First call:**In the first call to Split, these fields are put into the result array. These elements equal string.Empty.

**Second call:**We specify StringSplitOptions.RemoveEmptyEntries. The two empty fields are not in the result array.

static void Main()

{

// Input string contain separators.

string value1 = "bibek,nanda,panigrahi,,,programmer";

char[] delimiter1 = new char[] { ',' }; // <-- Split on these

// ... Use StringSplitOptions.None.

string[] array1 = value1.**Split**(delimiter1, StringSplitOptions.None);

foreach (string entry in array1)

{

Console.WriteLine(entry);

}

// ... Use StringSplitOptions.RemoveEmptyEntries.

string[] array2 = value1.**Split**(delimiter1, StringSplitOptions.RemoveEmptyEntries);

Console.WriteLine();

foreach (string entry in array2)

{

Console.WriteLine(entry);

}

}

**IndexOf.**With the IndexOf method, we can search for parts of a string, iterating through chars.

We use IndexOf to see if a string contains a word. We test the string for a substring "dog." We test the result of IndexOf against the special constant -1.

**Example:**IndexOf returns the location of the string "dog." It is not equal to -1. So the line is written to the console window.

static void Main()

{

// The input string.

const string value = "Your dog is cute.";

// Test with IndexOf method.

if (value.**IndexOf**("dog") != -1)

{

Console.WriteLine("string contains dog!");

}

}

**ubstring.**We can use IndexOf with the Substring method. Here we get the first substring that begins with a certain pattern or character.

static void Main()

{

// Input.

const string s = "I have a cat";

// Location of the letter c.

int i = s.**IndexOf**('c');

// Remainder of string starting at c.

string d = s.**Substring**(i);

Console.WriteLine(d);

}

**Skip start.**Often strings have leading characters that we know cannot contain the searched-for value. We can skip these chars. This will give a performance boost—less searching is needed.

using System;

class Program

{

static void Main()

{

string value = ":100,200";

// Skip the first character with a startIndex of 1.

int comma = value.**IndexOf**(',', 1);

Console.WriteLine(comma);

}

}

**Output**

4

The string.Join method combines many strings into one. It receives two arguments

sing System;

class Program

{

static void Main()

{

string[] arr = { "one", "two", "three" };

// "string" can be lowercase.

Console.WriteLine(**string.Join**(",", arr));

// ... "String" can be uppercase.

Console.WriteLine(**String.Join**(",", arr));

}

}

**Output**

one,two,three

one,two,three

**C# program that joins HTML strings**

using System;

class Program

{

static void Main()

{

// Problem: combine these words into lines in HTML

string[] dinosaurs = new string[] { "Aeolosaurus",

"Deinonychus", "Jaxartosaurus", "Segnosaurus" };

// Solution: join with break tag.

string html = **string.Join**("<br/>\r\n", dinosaurs);

Console.WriteLine(html);

}

}

**Output**

Aeolosaurus<br/>

Deinonychus<br/>

Jaxartosaurus<br/>

Segnosaurus

Creating a String Object

You 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
* using System;
* namespace StringApplication {
* class Program {
* static void Main(string[] args) {
* //from string literal and string concatenation
* string fname, lname;
* fname = "bibekananda";
* lname = "panigrahi";
* char []letters= { 'H', 'e', 'l', 'l','o' };
* string [] sarray={ "Hello", "From", "black", "knight" };
* string fullname = fname + lname;
* Console.WriteLine("Full Name: {0}", fullname);
* //by using string constructor { 'H', 'e', 'l', 'l','o' };
* string greetings = new string(letters);
* Console.WriteLine("Greetings: {0}", greetings);
* //methods returning string { "Hello", "From", " black ", " knight " };
* string message = String.Join(" ", sarray);
* Console.WriteLine("Message: {0}", message);
* //formatting method to convert a value
* DateTime waiting = new DateTime(2012, 10, 10, 17, 58, 1);
* string chat = String.Format("Message sent at {0:t} on {0:D}", waiting);
* Console.WriteLine("Message: {0}", chat);
* }
* }
* }