1. using **System**.**Text**;
2. namespace string\_function
3. {
4. class **Program**
5. {
6. static void **Main**(string[] args)
7. {
8. string firstname;
9. string lastname;

12. firstname = "Steven Clark";
13. lastname = "Clark";

16. **Console**.**WriteLine**(firstname.**Clone**());
17. // Make String Clone
18. **Console**.**WriteLine**(firstname.**CompareTo**(lastname));
19. //Compare two string value and returns 0 for true and
20. 1 for false
22. **Console**.**WriteLine**(firstname.**Contains**("ven")); //Check whether specified value exists or not in string
24. **Console**.**WriteLine**(firstname.**EndsWith**("n")); //Check whether specified value is the last character of string
25. **Console**.**WriteLine**(firstname.**Equals**(lastname));
26. //Compare two string and returns true and false

29. **Console**.**WriteLine**(firstname.**GetHashCode**());
30. //Returns HashCode of String
32. **Console**.**WriteLine**(firstname.**GetType**());
33. //Returns type of string
35. **Console**.**WriteLine**(firstname.**GetTypeCode**());
36. //Returns type of string
38. **Console**.**WriteLine**(firstname.**IndexOf**("e")); //Returns the first index position of specified value
39. the first index position of specified value
41. **Console**.**WriteLine**(firstname.**ToLower**());
42. //Covert string into lower case
44. **Console**.**WriteLine**(firstname.**ToUpper**());
45. //Convert string into Upper case
47. **Console**.**WriteLine**(firstname.**Insert**(0, "Hello")); //Insert substring into string
49. **Console**.**WriteLine**(firstname.**IsNormalized**());
50. //Check Whether string is in Unicode normalization
51. from C

54. **Console**.**WriteLine**(firstname.**LastIndexOf**("e")); //Returns the last index position of specified value
56. **Console**.**WriteLine**(firstname.**Length**);
57. //Returns the Length of String
59. **Console**.**WriteLine**(firstname.**Remove**(5));
60. //Deletes all the characters from begining to specified index.
62. **Console**.**WriteLine**(firstname.**Replace**('e','i')); // Replace the character
64. string[] split = firstname.**Split**(new char[] { 'e' }); //Split the string based on specified value

67. **Console**.**WriteLine**(split[0]);
68. **Console**.**WriteLine**(split[1]);
69. **Console**.**WriteLine**(split[2]);
71. **Console**.**WriteLine**(firstname.**StartsWith**("S")); //Check wheter first character of string is same as specified value
73. **Console**.**WriteLine**(firstname.**Substring**(2,5));
74. //Returns substring
76. **Console**.**WriteLine**(firstname.**ToCharArray**());
77. //Converts an string into char array.
79. **Console**.**WriteLine**(firstname.**Trim**());
80. //It removes starting and ending white spaces from
81. string.
83. }
84. }
85. }

**Working with Characters**

The solution is to work with characters instead of strings as much as possible. The char object in C# is a value type, which means all char variables are stored in the stack. Furthermore, since a string is a collection of characters, converting between chars and strings is very simple.

To convert a string to a char array, use the ToCharArray() .NET function:

string myStr = “hello world”;

char[] myStrChars = myStr.ToCharArray();

To convert a char array back to a string, simply create a new instance of a string:

char[] myChars = { ‘h’, ‘e’, ‘l’, ‘l’, ‘o’, ‘ ‘, ‘w’, ‘o’, ‘r’, ‘l’, ‘d’ };

string myStr = new string(myChars);

Writing efficient string functions thus boils down to working with char arrays. However you might remember that arrays are stored in the heap. Thus there isn’t much difference between working with a string and a character array in terms of performance if we end up handling arrays in the same way as strings.

Yet this does not mean working with array is not faster. For one thing, we can make use of dynamic arrays such as List (or ArrayList in .NET Framework 1.1) to make our array management as efficient as possible.

**Example Function**

Let's write a very simple string function and compare the difference between using strings and char arrays. The function will capitalize all the vowels in a string (working with the English alphabet), and make all other characters lowercase.

Using just strings:

public string CapitalizeVowels(string input)

{

    if (string.IsNullOrEmpty(input)) //since a string is a class object, it could be null

        return string.Empty;

    else

    {

        string output = string.Empty;

        for (int i = 0; i < input.Length; i++)

        {

            if (input[i] == 'a' || input[i] == 'e' ||

                input[i] == 'i' || input[i] == 'o' ||

                input[i] == 'u')

                output += input[i].ToString().ToUpper(); //Vowel

            else

                output += input[i].ToString().ToLower(); //Not vowel

        }

        return output;

    }

}

Using character arrays:

public string CapitalizeVowels(string input)

{

    if (string.IsNullOrEmpty(input)) //since a string is a class object, it could be null

        return string.Empty;

    else

    {

        char[] charArray = input.ToCharArray();

        for (int i = 0; i < charArray.Length; i++)

        {

            if (charArray[i] == 'a' || charArray[i] == 'e' ||

                charArray[i] == 'i' || charArray[i] == 'o' ||

                charArray[i] == 'u')

                charArray[i] = char.ToUpper(charArray[i]); //Vowel

            else

                charArray[i] = char.ToLower(charArray[i]); //Not vowel

        }

        return new string(charArray);

    }

}

Both functions will produce the exact same results given the same input data. We can perform some basic benchmarks to compare the performance of each function. For example, the string-based function took an average of 2181ms to process the string “hello world” 1,000,000 times while the array-based function only took 448ms (measured on my computer).

**C# program that tests for digits**

using System;

class Program

{

static void Main()

{

string test = "Abc,123";

foreach (char value in test)

{

bool digit = **char.IsDigit**(value);

Console.Write(value);

Console.Write(' ');

Console.WriteLine(digit);

}

}

/// <summary>

/// Returns whether the char is a digit char.

/// Taken from inside the char.IsDigit method.

/// </summary>

public static bool **IsCharDigit**(char c)

{

return ((c >= '0') && (c <= '9'));

}

}

**Output**

A False

b False

c False

, False

1 True

2 True

3 True