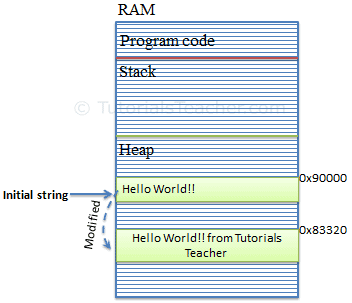
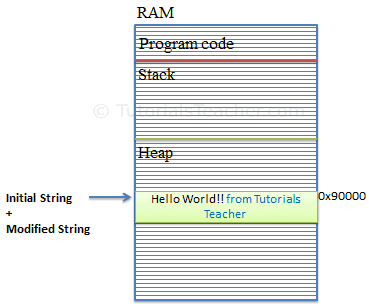
# C# - StringBuilder

In C#, the string type is immutable. It means a string cannot be changed once created. For example, a new string, "Hello World!" will occupy a memory space on the heap. Now, by changing the initial string "Hello World!" to "Hello World! from Tutorials Teacher" will create a new string object on the memory heap instead of modifying an original string at the same memory address. This behavior would hinder the performance if the original string changed multiple times by replacing, appending, removing, or inserting new strings in the original string.



To solve this problem, C# introduced the StringBuilder in the [System.Text](https://docs.microsoft.com/en-us/dotnet/api/system.text" \t "_blank) namespace. The StringBuilder doesn't create a new object in the memory but dynamically expands memory to accommodate the modified string.



Creating a StringBuilder Object

You can create an object of the StringBuilder class using the new keyword and passing an initial string. The following example demonstrates creating StringBuilder objects.

Example: StringBuilder

using System.Text; // include at the top

StringBuilder sb = new StringBuilder(); //string will be appended later

//or

StringBuilder sb = new StringBuilder("Hello World!");

Optionally, you can also specify the maximum capacity of the StringBuilder object using overloaded constructors, as shown below.

Example: StringBuilder

StringBuilder sb = new StringBuilder(50); //string will be appended later

//or

StringBuilder sb = new StringBuilder("Hello World!", 50);

Above, C# allocates a maximum of 50 spaces sequentially on the memory heap. This capacity will automatically be doubled once it reaches the specified capacity. You can also use the capacity or length property to set or retrieve the StringBuilder object's capacity.

You can iterate the using [for loop](https://www.tutorialsteacher.com/csharp/csharp-for-loop) to get or set a character at the specified index.

Example: StringBuilder Iteration

StringBuilder sb = new StringBuilder("Hello World!");

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

Console.Write(sb[i]); // output: Hello World!

## Retrieve String from StringBuilder

The StringBuilder is not the string. Use the ToString() method to retrieve a string from the StringBuilder object.

Example: Retrieve String from StringBuilder

StringBuilder sb = new StringBuilder("Hello World!");

var greet = sb.ToString(); //returns "Hello World!"

## Add/Append String to StringBuilder

Use the Append() method to append a string at the end of the current StringBuilder object. If a StringBuilder does not contain any string yet, it will add it. The AppendLine() method append a string with the newline character at the end.

Example: Adding or Appending Strings in StringBuilder

StringBuilder sb = new StringBuilder();

sb.Append("Hello ");

sb.AppendLine("World!");

sb.AppendLine("Hello C#");

Console.WriteLine(sb);

## Append Formated String to StringBuilder

Use the AppendFormat() method to format an input string into the specified format and append it.

Example: AppendFormat()

StringBuilder sbAmout = new StringBuilder("Your total amount is ");

sbAmout.AppendFormat("{0:C} ", 25);

Console.WriteLine(sbAmout);//output: Your total amount is $ 25.00

## Insert String into StringBuilder

Use the Insert() method inserts a string at the specified index in the StringBuilder object.

Example: Insert()

StringBuilder sb = new StringBuilder("Hello World!");

sb.Insert(5," C#");

Console.WriteLine(sb); //output: Hello C# World!

## Remove String in StringBuilder

Use the Remove() method to remove a string from the specified index and up to the specified length.

Example: Remove()

StringBuilder sb = new StringBuilder("Hello World!",50);

sb.Remove(6, 7);

Console.WriteLine(sb); //output: Hello

## Replace String in StringBuilder

Use the Replace() method to replace all the specified string occurrences with the specified replacement string.

Example: Replace()

StringBuilder sb = new StringBuilder("Hello World!");

sb.Replace("World", "C#");

Console.WriteLine(sb);//output: Hello C#!

Points to Remember :

1. StringBuilder is mutable.
2. StringBuilder performs faster than string when appending multiple string values.
3. Use StringBuilder when you need to append more than three or four strings.
4. Use the Append() method to add or append strings to the StringBuilder object.
5. Use the ToString() method to retrieve a string from the StringBuilder object.

Input & output in C# is based on streams. A Stream is an abstract base class of all streams. A stream is an abstraction of a sequence of bytes, such as a file, an input/output device, an inter-process communication pipe, or a TCP/IP socket.

## C# StringReader

StringReader reads text data from strings. It can read data synchronously or asynchronously. The reading operation is stream-based.

## StringReader ReadToEnd

The ReadToEnd method reads all characters from the current position to the end of the string and returns them as a single string.

using System;

using System.IO;

using System.Text;

var sb = new StringBuilder();

sb.AppendLine("There is a hawk in the sky.");

sb.AppendLine("The sun is shining.");

sb.AppendLine("The flowers are blossoming.");

using var reader = new StringReader(sb.ToString());

string text = reader.ReadToEnd();

Console.WriteLine(text);

## The example builds a string with StringBuilder and then reads the text with StringReader's ReadToEnd.

## StringReader ReadLine

The ReadLine method reads a line of characters from the current string and returns the data as a string.

using System;

using System.IO;

var text = @"The Battle of Thermopylae was fought between an alliance

of Greek city-states, led by King Leonidas of Sparta, and the Persian Empire of

Xerxes I over the course of three days, during the second Persian invasion of Greece.";

using var sr = new StringReader(text);

int count = 0;

string line;

while ((line = sr.ReadLine()) != null)

{

count++;

Console.WriteLine("Line {0}: {1}", count, line);

}

The ReadLine method returns the next line from the current string, or null if the end of the string is reached.

------------------------------------------------------------

**StringReader Read**

The Read method reads the next character from the input string and advances the character position by one character.

**Program.cs**

using System;

using System.IO;

var text = "There is my new book.";

using var reader = new StringReader(text);

int count = 0;

char mychar = 'h';

int n;

while ((n = reader.Read()) != -1)

{

char c = (char) n;

if (c.Equals(mychar))

{

count++;

}

}

Console.WriteLine($"There are {count} '{mychar}' characters in the string")

The Read method returns the next character from the underlying string, or -1 if no more characters are available.