

# C# StringBuilder with Examples

In c#, **StringBuilder** is a [class](#) that is useful to represent a mutable [string](#) of characters and it is an object of **System.Text** [namespace](#).

Like [string in c#](#), we can use a **StringBuilder** to create [variables](#) to hold any kind of text which is a sequential collection of characters based on our requirements.

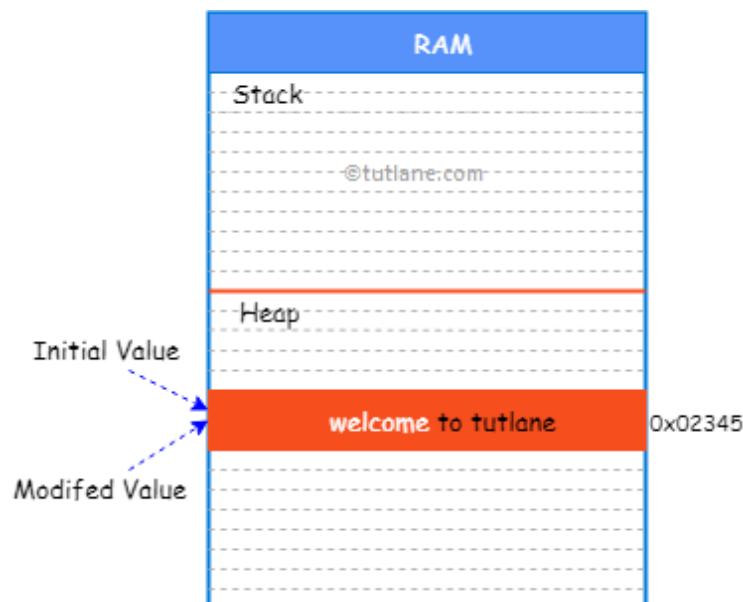
In c#, both [string](#) and **StringBuilder** will represent a sequence of characters and perform the same kind of operations but the only difference is [strings](#) are **immutable** and **StringBuilder** is **mutable**.

Generally, in c# the [string](#) object cannot be modified once it created. If any changes made to the string object like add or modify an existing value, then it will simply discard the old instance in memory and create a new instance to hold the new value. In case, if we are doing repeated modifications on the string object, then it will affect the performance of application. To know more about strings, check [strings in c# with examples](#).

To solve this problem, c# introduced an alternative called **StringBuilder**, which is a **mutable** string [class](#). **Mutability** means once an instance of the class is created, then the same instance will be used to perform any operations like inserting, appending, removing or replacing the characters instead of creating a new instance for every time.

In c#, the **StringBuilder** is a dynamic object which will expand a memory dynamically to accommodate the modifications of [string](#) instead of creating a new instance in the memory.

Following is the pictorial representation of memory allocation for **StringBuilder** object in c# programming language.



## C# StringBuilder Declaration and Initialization

As discussed, the **StringBuilder** [class](#) is an object of **System.Text** [namespace](#) so to use **StringBuilder** in our application, we need to import the **System.Text** [namespace](#).

In c#, the **StringBuilder** declaration and initialization will be the same as [class](#). The following are the different ways of declaring and initializing a **stringbuilder** in c# programming language.

```
StringBuilder sb = new StringBuilder();
```

```
//or
```

```
StringBuilder sb = new StringBuilder("Welcome to Tutlane");
```

If you observe the above code snippet, we created an instance of the **StringBuilder** [class](#) by defining our [variable](#) with overloaded [constructor](#) methods.

## C# StringBuilder Capacity

As discussed, the **StringBuilder** is a dynamic object which will expand dynamically to accommodate the number of characters based on the string modifications. We can also specify an initial capacity of characters, the **StringBuilder** can hold by passing an int value using one of the overloaded [constructors](#) or by using **StringBuilder Capacity** property.

For example, we created a **StringBuilder** by specifying the capacity of **25** characters and appending a string whose length is greater than the capacity of **25** characters. In this case, the new space will be allocated automatically and the capacity of **StringBuilder** will be doubled.

Following is the example of specifying the initial capacity of characters the **StringBuilder** can hold in **c#** programming language.

```
StringBuilder sb = new StringBuilder(25);

//or

StringBuilder sb = new StringBuilder("Welcome to Tutlane",25);

//or

sb.Capacity = 25;
```

In **c#**, whenever the defined capacity of **StringBuilder** is lesser than the appended [string](#) value, then the current capacity of **StringBuilder** automatically will increase to match the appended [string](#) value.

The default capacity of **StringBuilder** is **16** characters and its maximum capacity is more than **12 billion** characters.

## C# StringBuilder Methods

The following table lists the important methods of **StringBuilder** which we can use to modify the contents of **StringBuilder**.

Method	Description
StringBuilder.Append	This method will append the given <a href="#">string</a> value to the end of the current <b>StringBuilder</b> .
StringBuilder.AppendFormat	It will replace a format specifier passed in a <a href="#">string</a> with formatted text.
StringBuilder.Insert	It inserts a <a href="#">string</a> at the specified index of current <b>StringBuilder</b> .

Method	Description
StringBuilder.Remove	It removes a specified number of characters from the current StringBuilder.
StringBuilder.Replace	It replaces a specified character at a specified index.

## C# StringBuilder Append Method

The **Append** method is used to add or append a [string](#) objects at the end of the [string](#) represented by the StringBuilder.

Following is the example of initializing a StringBuilder with some text and appending a required text at the end of the [string](#) object.

```
StringBuilder sb = new StringBuilder("Suresh");
sb.Append(", Rohini");
sb.Append(", Trishika");
Console.WriteLine(sb);
// Output: Suresh, Rohini, Trishika
```

## C# StringBuilder AppendFormat Method

The **AppendFormat** method is used to add or append a [string](#) objects by formatting into specified format at the end of [string](#) represented by the StringBuilder.

Following is the example of initializing a StringBuilder with some text and appending a formatted text at the end of the [string](#) object.

```
int amount = 146;
StringBuilder sb = new StringBuilder("Total");
sb.AppendFormat(": {0:c}", amount);
Console.WriteLine(sb);
// Output is Total: $146.00
```

## C# StringBuilder Insert Method

The **Insert** method is used to insert a string at the specified index position of current StringBuilder object.

Following is the example of initializing a StringBuilder with some text and inserting a string at the specified index position of StringBuilder object.

```
StringBuilder sb = new StringBuilder("Welcome Tutlane");
sb.Insert(8, "to ");
Console.WriteLine(sb);
// Output: Welcome to Tutlane
```

## C# StringBuilder Remove Method

The **Remove** method is used to remove a specified number of characters from the current StringBuilder object, starting from the specified index position.

Following is the example of removing a specified number of characters from the StringBuilder object, starting from the specified index position.

```
StringBuilder sb = new StringBuilder("Welcome to Tutlane");
sb.Remove(8, 3);
Console.WriteLine(sb);
// Output: Welcome Tutlane
```

## C# StringBuilder Replace Method

The **Replace** method is used to replace all occurrences of specified string characters in current StringBuilder object with a specified replacement string characters.

Following is the example of replacing a specified number of characters from the StringBuilder object, with specified replace characters.

```
StringBuilder sb = new StringBuilder("Welcome to Tutlane");
sb.Replace("Tutlane", "C#");
Console.WriteLine(sb);
// Output: Welcome to C#
```

In c#, the StringBuilder is having another method called **AppendLine()** which is used to add a newline at the end of string.

## C# StringBuilder Example

Following is the example of using **StringBuilder** to insert or append or replace or to remove a particular string text in c# programming language.

```
using System;
using System.Text;

namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
            StringBuilder sb = new StringBuilder("Suresh");
            sb.Append(", Rohini");
            sb.Append(", Trishika");
            sb.AppendLine();
            sb.Append("Welcome to Tutlane");
            Console.WriteLine(sb);

            StringBuilder sb1 = new StringBuilder("Welcome World");
            sb1.Insert(8, "to Tutlane ");
            Console.WriteLine("Insert String: " + sb1);
        }
    }
}
```

```

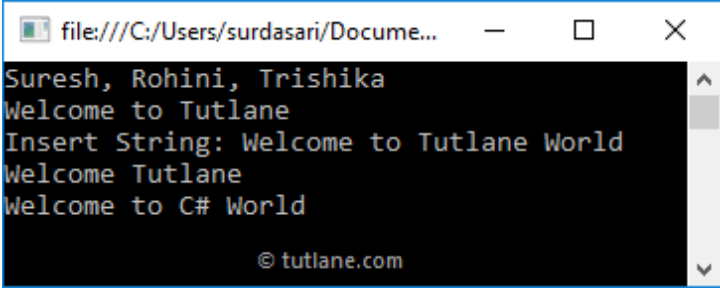
        StringBuilder sb2 = new StringBuilder("Welcome to Tutlane");
        sb2.Remove(8, 3);
        Console.WriteLine(sb2);

        StringBuilder sb3 = new StringBuilder("Welcome to Tutlane World");
        sb3.Replace("Tutlane", "C#");
        Console.WriteLine(sb3);
        Console.ReadLine();
    }
}
}

```

If you observe above code, to use StringBuilder in our application we imported a **System.Text** [namespace](#) and used a different methods of StringBuilder to make required modifications based on our requirements.

When you execute the above c# program, you will get the result as shown below.



The screenshot shows a console window with the following output:

```

Suresh, Rohini, Trishika
Welcome to Tutlane
Insert String: Welcome to Tutlane World
Welcome Tutlane
Welcome to C# World
© tutlane.com

```

This is how we can use StringBuilder in our applications to make the required modifications to the string objects based on our requirements.

## C# Convert StringBuilder to String

In c#, you can convert a StringBuilder object to a string by calling by **StringBuilder.ToString()** method.

Following is the example of converting a StringBuilder object to a string using the **ToString()** method in c# programming language.

```

using System;
using System.Text;

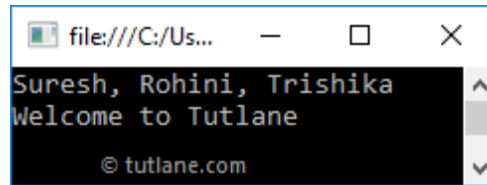
namespace Tutlane
{
    class Program
    {
        static void Main(string[] args)
        {
            StringBuilder sb = new StringBuilder("Suresh");
            sb.Append(", Rohini");
            sb.Append(", Trishika");
            sb.AppendLine();
            sb.Append("Welcome to Tutlane");
            Console.WriteLine(sb.ToString());
            Console.ReadLine();
        }
    }
}

```

```
}  
}  
}
```

If you observe above code, we are converting `StringBuilder` object (**sb**) to string object using **sb.ToString()** method.

When you execute the above c# program, you will get the result as shown below.



```
file:///C:/Us...  
Suresh, Rohini, Trishika  
Welcome to Tutlane  
© tutlane.com
```

This is how we can convert `StringBuilder` object to string based on our requirements.

In c#, the **StringBuilder** will offer better performance than **string** but we should not automatically replace a **string** with **StringBuilder** whenever we want to manipulate strings.

## Use StringBuilder in C#

The following are the important points which we need to remember while using a `StringBuilder` in c# programming language.

- We need to consider using `StringBuilder` only when we are unknown about the number of changes that will make to a [string](#) at design time otherwise `StringBuilder` will offer a negligible or no performance improvement over a [string](#).
- When we are performing a fixed number of concatenation operations on a [string](#), then it's better to avoid using `StringBuilder` because it will offer negligible performance.
- In another case when we expect our application will make a significant number of changes to a [string](#), then we need to use `StringBuilder` instead of a [string](#).