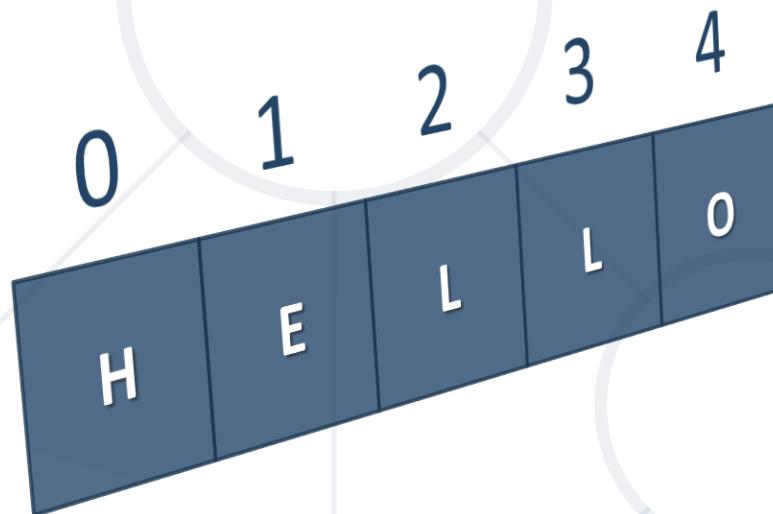


Strings and Text Processing

Processing and Manipulating Text
Using the .NET String Class



SoftUni Team

Technical Trainers



SoftUni



Software University
<https://softuni.bg>

Table of Contents

1. Strings

2. Manipulating Strings

- Concatenating, Searching, Substring
- Splitting, Replacing

3. Building and Modifying Strings

- Using `StringBuilder` class
- Why concatenation is a slow operation?



Strings

What is String?



What is String?

- **Strings** are sequences of characters (texts)
- The string data type in C#
 - Declared by the **string** keyword
 - Maps to **System.String** .NET data type
- Strings are enclosed in quotes
- Concatenated using the "+" operator

```
string s = "Hello, C#";
```

```
string s = "Hello" + " " + "C#";
```



In C# Strings Are Immutable, Use Unicode

- Strings are **immutable** (read-only) sequences of characters
- Accessible by index (read-only)

```
string str = "Hello, C#";  
char ch = str[2]; // OK  
str[2] = 'a';    // Error!
```

- Strings use **Unicode** (can use most alphabets, e.g., Arabic)

```
string greeting = "你好"; // (Lí-hó) Taiwanese
```



Initializing a String

- Initializing from a string literal

```
string str = "Hello, C#";
```

- Reading a **string** from the console

```
string name = Console.ReadLine();
Console.WriteLine("Hi, " + name);
```

- Converting a **string** from and to a **char array**

```
string str = new string(new char[] {'s','t','t'});
char[] charArr = str.ToCharArray();
// ['s', 't', 'r']
```





Manipulating Strings

Concatenating

- Use the **+** or the ****+=**** operators

```
string text = "Hello" + ", " + "world!";
// "Hello, world!"
```

```
string text = "Hello, ";
text += "John"; // "Hello, John"
```

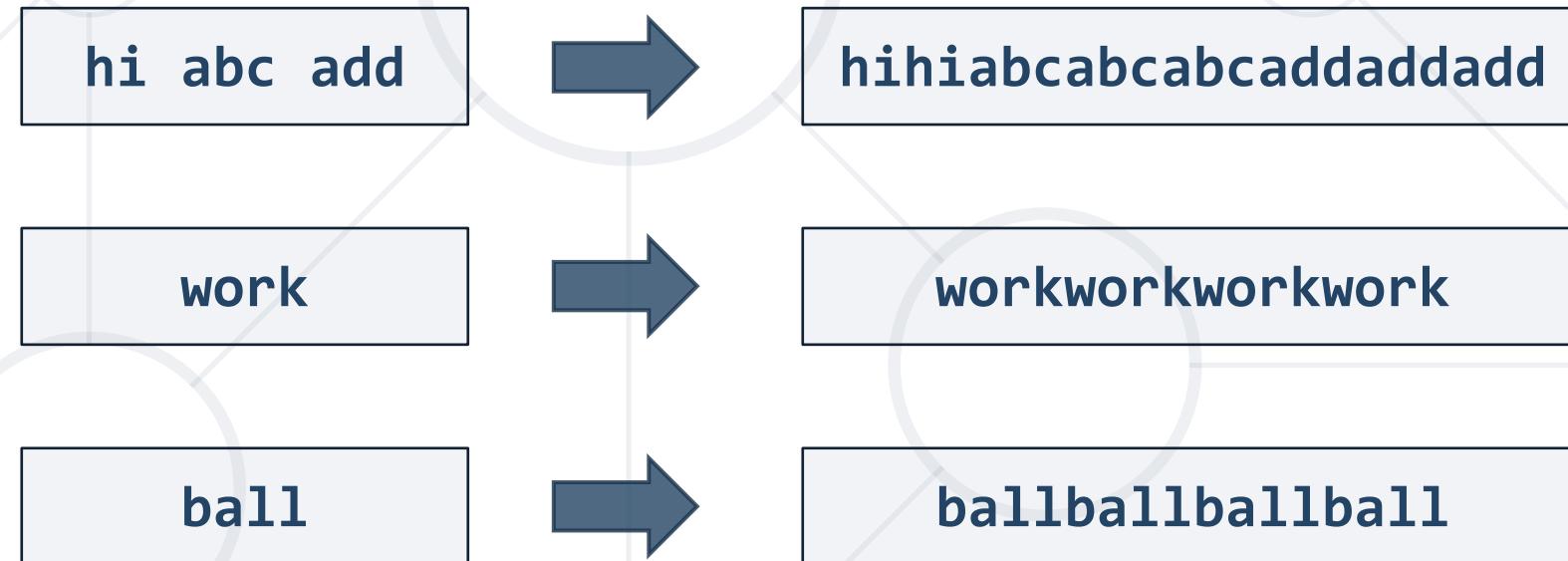
- Use the **Concat()** method

```
string greet = "Hello, ";
string name = "John";
string result = string.Concat(greet, name);
Console.WriteLine(result); // "Hello, John"
```



Problem: Repeat Strings

- Read an array from strings
- Repeat each word **n** times, where **n** is the length of the word



Check your solution here: <https://alpha.judge.softuni.org/contests/text-processing-lab/1216/practice#0>

Solution: Repeat Strings

```
string[] words = Console.ReadLine().Split();
string result = "";
foreach (string word in words)
{
    int repeatTimes = word.Length;
    for (int i = 0; i < repeatTimes; i++)
        result += word;
}
Console.WriteLine(result);
```

Check your solution here: <https://alpha.judge.softuni.org/contests/text-processing-lab/1216/practice#0>

- IndexOf() – returns the first match index or -1

```
string fruits = "banana, apple, kiwi, banana, apple";
Console.WriteLine(fruits.IndexOf("banana")); // 0
Console.WriteLine(fruits.IndexOf("orange")); // -1
```

- LastIndexOf() – finds the last occurrence

```
string fruits = "banana, apple, kiwi, banana, apple";
Console.WriteLine(fruits.LastIndexOf("banana")); // 21
Console.WriteLine(fruits.LastIndexOf("orange")); // -1
```

- Contains() - Check whether one string contains other string

```
string text = "I love fruits.";  
Console.WriteLine(text.Contains("fruits")); // True  
Console.WriteLine(text.Contains("banana")); // False
```

Substring

- **Substring(int startIndex, int length)**

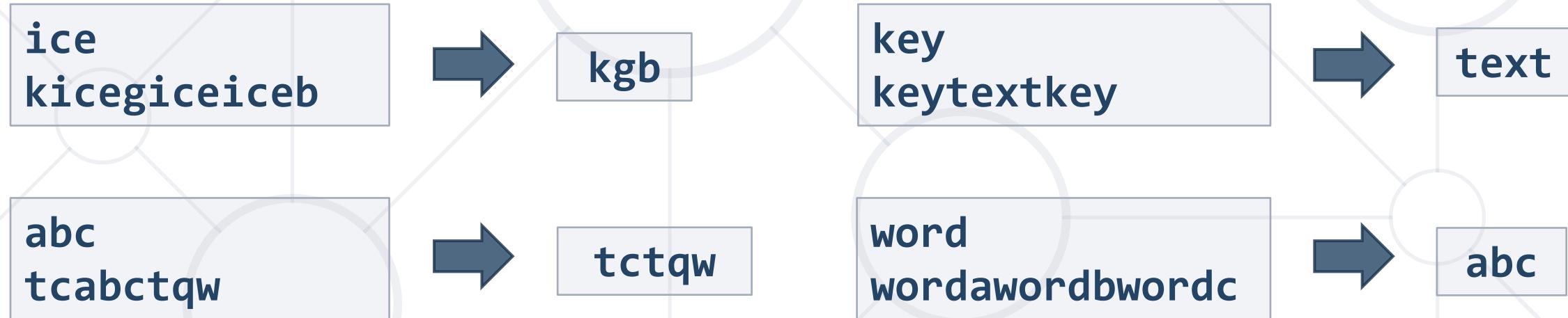
```
string card = "10C";
string power = card.Substring(0, 2);
Console.WriteLine(power); // 10
```

- **Substring(int startIndex)**

```
string text = "My name is John";
string extractWord = text.Substring(11);
Console.WriteLine(extractWord); // John
```

Problem: Substring

- You are given a **text** and a **remove word**
- Remove all substrings that are equal to the remove word



Check your solution here: <https://alpha.judge.softuni.org/contests/text-processing-lab/1216/practice#2>

Solution: Substring

```
string key = Console.ReadLine();
string text = Console.ReadLine();

int index = text.IndexOf(key);

while (index != -1)
{
    text = text.Remove(index, key.Length);
    index = text.IndexOf(key);
}

Console.WriteLine(text);
```

Splitting

- Split() a string by given **separator**

```
string text = "Hello, john@softuni.bg, you have been using  
john@softuni.bg in your registration";  
  
string[] words = text.Split(", ");  
  
// words[]:  
// "Hello"  
// "john@softuni.bg"  
// "you have been using john@softuni.bg in your registration"
```

- **Split()** can be used with multiple separators

```
char[] separators = new char[] { ' ', ',', '.', '•' };  
string text = "Hello, I am John.>";  
string[] words = text.Split(separators);  
// "Hello", "", "I", "am", "John", ""
```

- Using **StringSplitOptions.RemoveEmptyEntries** to remove empty array elements from the array returned

```
char[] separators = new char[] { ' ', ',', '.', '\n' };

string text = "Hello, I am John.";

string[] words = text
    .Split(separators,
StringSplitOptions.RemoveEmptyEntries);

// "Hello", "I", "am", "John"
```

Replacing

- **Replace(match, replacement)** – replaces all occurrences
 - The result is a new **string** (strings are immutable)

```
string text = "Hello, john@softuni.bg, you have been using john@  
softuni.bg in your registration.";  
  
string replacedText = text  
    .Replace("john@softuni.bg", "john@softuni.com");  
  
Console.WriteLine(replacedText);  
  
// Output:  
  
// Hello, john@softuni.com, you have been using john@softuni.com  
in your registration.
```

Problem: Text Filter

- You are given a text and a string of banned words
 - Replace all banned words in the text with asterisks

Linux, Windows

It is not Linux, it is GNU/Linux. Linux is merely the kernel, while GNU adds the functionality...



It is not ****, it is GNU/****. **** is merely the kernel, while GNU adds the functionality...

Solution: Text Filter

```
string[] banWords = Console.ReadLine()  
    .Split(...); // TODO: add separators  
string text = Console.ReadLine();  
foreach (var banWord in banWords)  
{  
    if (text.Contains(banWord))  
    {  
        text = text.Replace(banWord,  
            new string('*', banWord.Length));  
    }  
}  
Console.WriteLine(text);
```

Contains(...) checks
if the string contains
another string

Replace a word with a sequence
of asterisks of the same length



Building and Modifying Strings

Using the `StringBuilder` Class

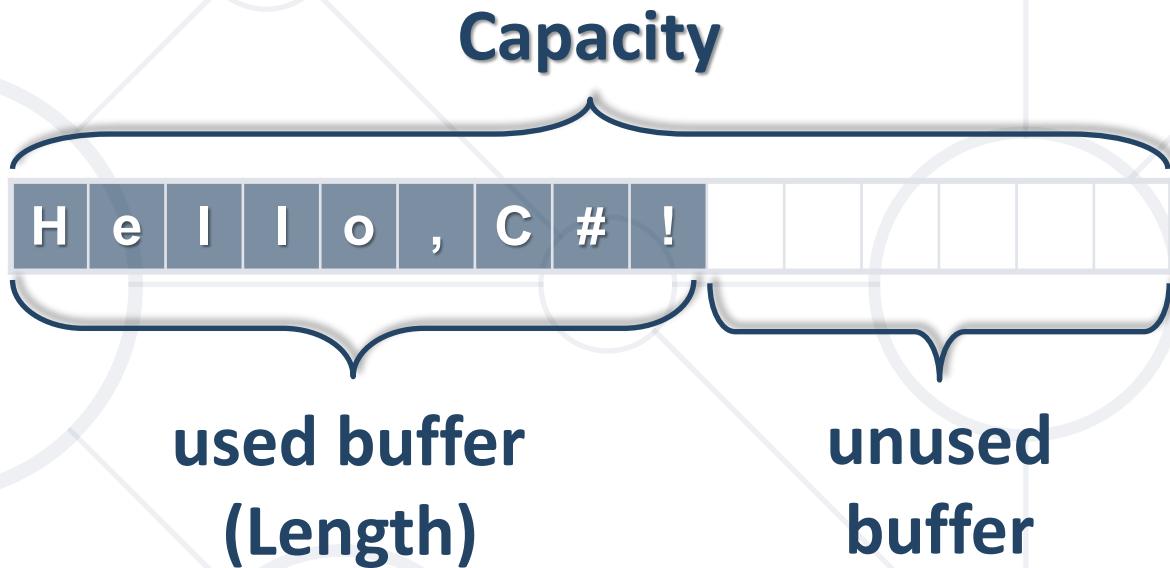
StringBuilder: How It Works?



StringBuilder:

Length = 9

Capacity = 15



- **StringBuilder** keeps a buffer space, allocated in advance
 - Do not allocate memory for most operations → performance

Using StringBuilder Class

- Use the StringBuilder to build / modify strings

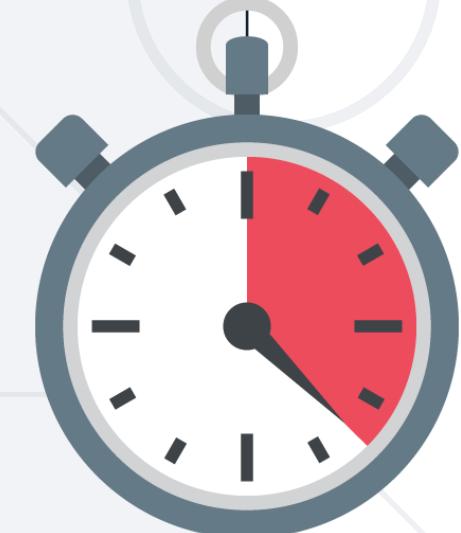
```
StringBuilder sb = new StringBuilder();
sb.Append("Hello, ");
sb.Append("John! ");
sb.Append("I sent you an email.");
Console.WriteLine(sb);
// Hello, John! I sent you an email.
```

use **System.Text**

Concatenation vs StringBuilder

- Concatenating strings is a **slow** operation because each iteration **creates a new string**

```
Stopwatch sw = new Stopwatch();
sw.Start();
string text = "";
for (int i = 0; i < 200000; i++)
    text += i;
sw.Stop();
Console.WriteLine(sw.ElapsedMilliseconds); // 73625
```



Concatenation vs StringBuilder

■ Using **StringBuilder**

```
Stopwatch sw = new Stopwatch();
sw.Start();
StringBuilder text = new StringBuilder();
for (int i = 0; i < 200000; i++)
    text.Append(i);
sw.Stop();
Console.WriteLine(sw.ElapsedMilliseconds); // 16
```



StringBuilder Methods

- Append(...) – add text or a string representation of an object to the end of a string

```
StringBuilder sb = new StringBuilder();
sb.Append("Hello Peter, how are you?");
```

- Length – holds the length of the string in the buffer

```
sb.Append("Hello Peter, how are you?");
Console.WriteLine(sb.Length); // 32
```

- Clear(...) – removes all characters

StringBuilder Methods

- **[int index]** – returns the char on current index

```
StringBuilder sb= new StringBuilder();
sb.Append("Hello Peter, how are you?");
Console.WriteLine(sb[1]); // e
```

- **Insert(int index, string str)** – inserts a string at the specified character position

```
sb.Insert(11, " Ivanov");
Console.WriteLine(sb); // Hello Peter Ivanov, how are you?
```

StringBuilder Methods

- **Replace(string oldValue, string newValue)** – replaces all occurrences of a specified string with another specified string

```
sb.Append("Hello Peter, how are you?");  
sb.Replace("Peter", "George");
```

- **ToString()** – converts the value of this instance to a String

```
string text = sb.ToString();  
Console.WriteLine(text);  
// Hello George, how are you?
```

- **Strings are immutable sequences of Unicode characters**
- **String processing methods**
 - **Concat(), IndexOf(), Contains(), Substring(), Split(), Replace(), ...**
- **StringBuilder efficiently builds / modifies strings**

