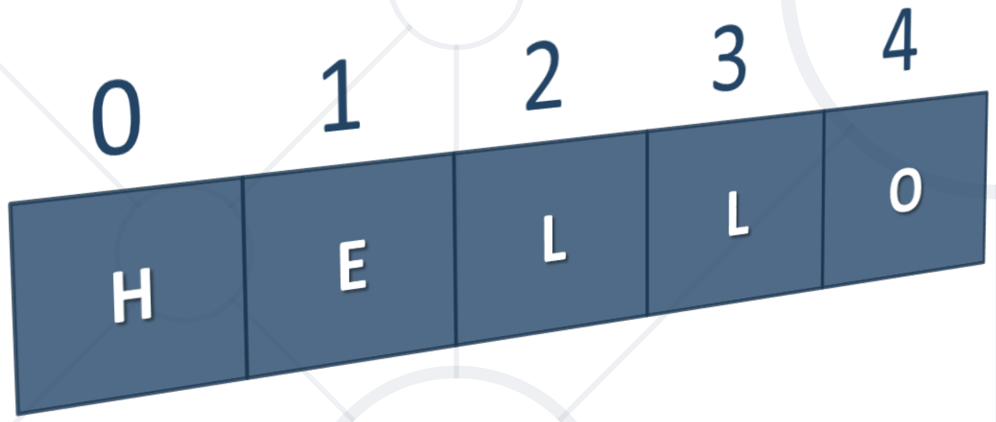
Strings and Text Processing



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

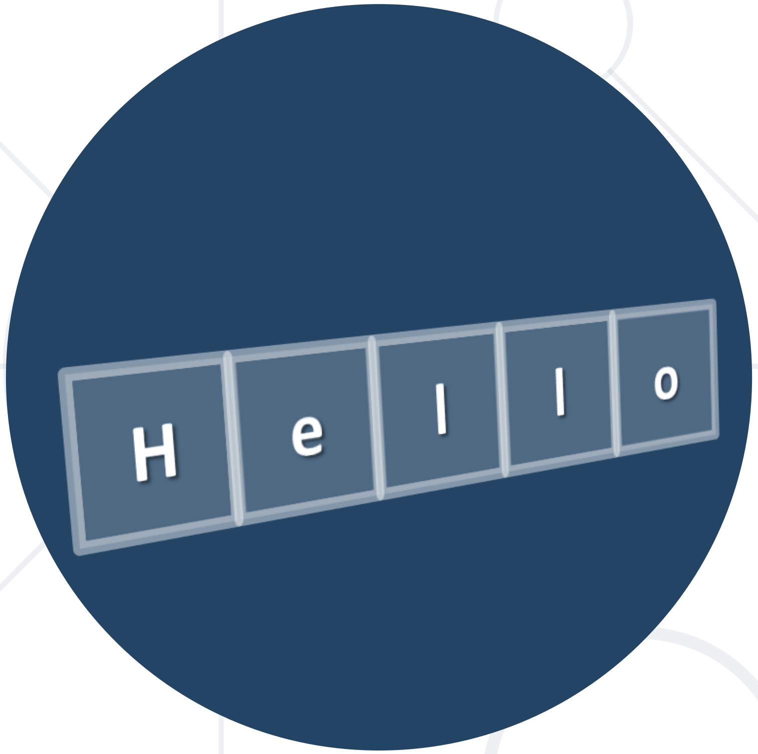
https://softuni.bg

Have a Question?

sli.do

#prgm-for-qa

Table of Contents



Strings

What Is a String?

What Is a String?

- Strings are **sequences** of characters
- The string data type in C#
  - Declared by the `string` keyword
- Strings are enclosed in double quotes:

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

Strings Are Immutable

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

```
string str = "Hello, C#";
char ch = str[2]; // l
```

- 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 = "str";
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"
```

Joining Strings

- string.Join("", …) concatenates strings

```
string t = string.Join("", "con", "ca", "ten", "ate");
// "concatenate"
```

- Or an array / list of strings
  - Useful for repeating a string

```
string s = "abc";
string[] arr = new string[3];
for (int i = 0; i < arr.Length; i++) { arr[i] = s; }
string repeated = string.Join("", arr); // "abcabcabc"
```

Problem: Repeat Strings

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

`hi abc add`	`hihiabcabcabcaddaddadd`
`work`	`workworkworkwork`
`ball`	`ballballballball`

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);
```

Searching

- **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
```

Searching

- Contains() - checks whether one string contains another

```
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

```
ice
kicegiciceib
```

```
kgb
```

```
key
keytextkey
```

```
text
```

```
abc
tcabctqw
```

```
tctqw
```

```
word
wordawordbwordc
```

```
abc
```

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

- Split by multiple separators

```
char[] separators = new char[] { ' ', ',', '.' };
string text = "Hello, I am John.";
string[] words = text.Split(separators);
// "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.org, you have been
using john@softuni.org in your registration.";

string replacedText = text
      .Replace("john@softuni.org", "john@softuni.com");

Console.WriteLine(replacedText);

// 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);
```

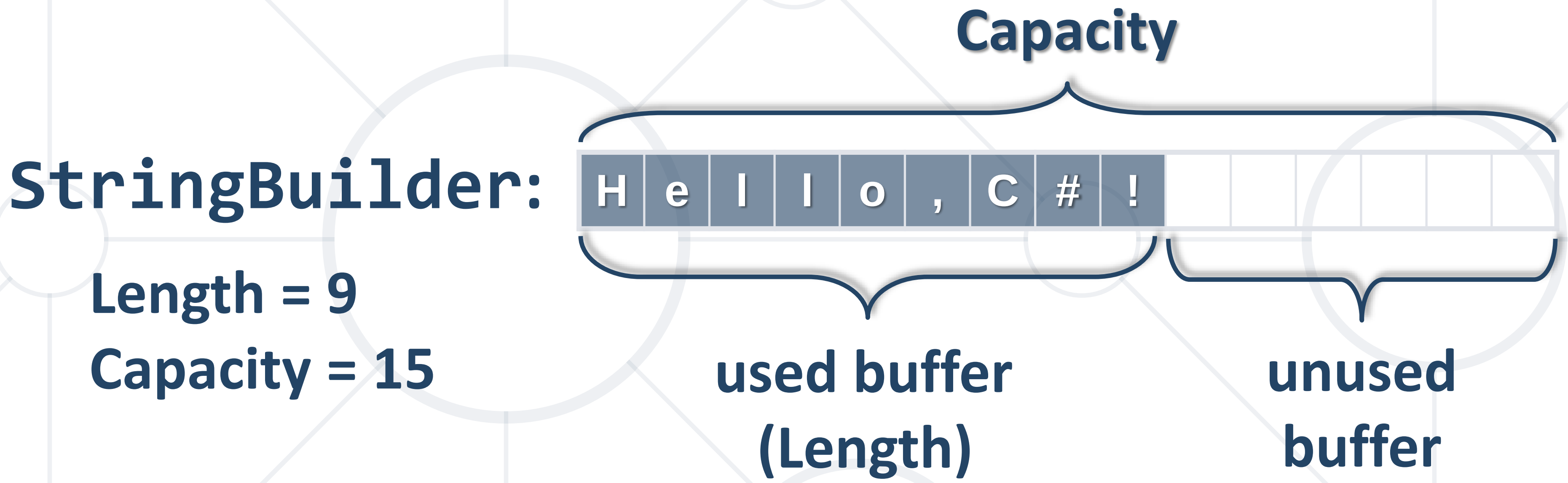
Contains(…) checks if the string contains another string

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** 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

```csharp
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?
```

Summary

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

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

SOFTWARE GROUP

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License