Telerik Academy

Strings and Text Processing

Processing and Manipulating Text Information

C# Advanced

Telerik Software Academy

https://telerikacademy.com

Follow us

Telerik Academy

Table of Contents

Telerik Academy

Table of Contents

Telerik Academy

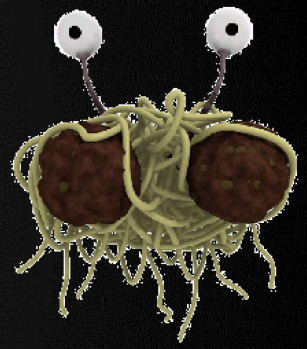
What Is String?

Sequences of Characters

Telerik Academy

What Is String?

- Represented by the `string` data type in C#
  - `System.String`
- Strings are sequences of characters
- Each character is a Unicode symbol
- *Example*:



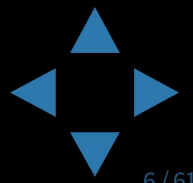
```
string s = "Hello, C#";
```

0	1	2	3	4	5	6	7	8
H	e	l	l	o	,		C	#

Telerik Academy

The System.String Class

- Strings are represented by `System.String` objects in .NET Framework
  - String objects contain an immutable (read-only) sequence of characters
  - Strings use Unicode to support multiple languages and alphabets
- Strings are stored in the dynamic memory (managed heap)
- `System.String` is reference type



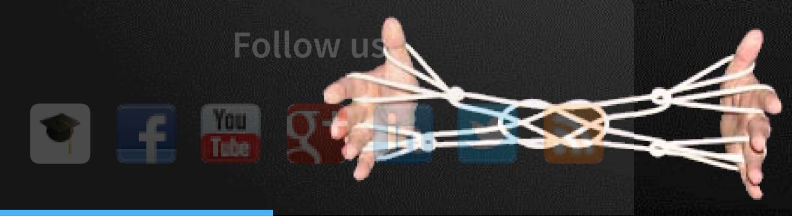
Telerik Academy

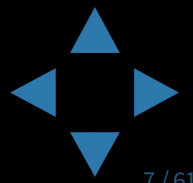
The System.String Class

- String objects are like arrays of characters (`char[]`)
  - Have fixed length (`String.Length`)
  - Elements can be accessed directly by index
    - The index is in the range `[0...Length-1]`

```
string s = "Hello!";
int len = s.Length; // len = 6char ch = s[1]; // ch = 'e'
```

index	0	1	2	3	4	5
str[index]	H	e	l	l	o	!



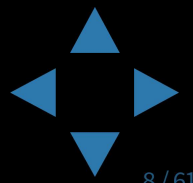


Telerik Academy

Strings: Examples

- *Example*: Printing the characters of a string with a for loop

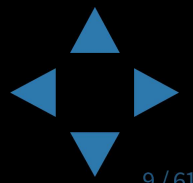
```
static void Main()
{
  string greeting = "Stand up, stand up, Balkan Superman.";
  Console.WriteLine("greeting = \"{0}\"", greeting);
  Console.WriteLine("greeting.Length = {0}", greeting.Length);

  for (int i = 0; i < greeting.Length; i++)
  {
    Console.WriteLine("greeting[{0}] = {1}", i, greeting[i]);
  }
}
```



Telerik Academy

Printing the Characters of a String

Demo



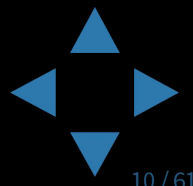
Telerik Academy

Declaring Strings

- Several ways of declaring string variables:
  - Using the C# keyword `string`
  - Using the .NET's fully qualified class name `System.String`

```
string str1;
System.String str2;
String str3;
```

  - The above three declarations are equivalent



Telerik Academy

Creating Strings

- Before initializing a string variable has `null` value
- Strings can be initialized by:
  - Assigning a string literal to the string variable
  - Assigning the value of another string variable
  - Assigning the result of operation of type string

Telerik Academy

Creating Strings

- Uninitialized variables has value of `null`
  - When their are local variables, they cannot be used

```
string s; // s is equal to null
```

- Assigning with string literal

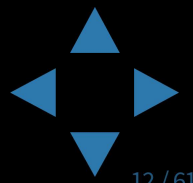
```
string s = "I am a string literal!";
```

- Assigning from another string variable

```
string s2 = s;
```

- Assigning from the result of string operation

```
string s = 42.ToString();
```

Telerik Academy

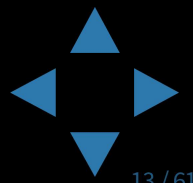
Reading and Printing Strings

- Reading strings from the console
  - Use the method `Console.ReadLine()`

```
string s = Console.ReadLine();
```

- Printing strings to the console
  - Use the methods `Write()` and `WriteLine()`

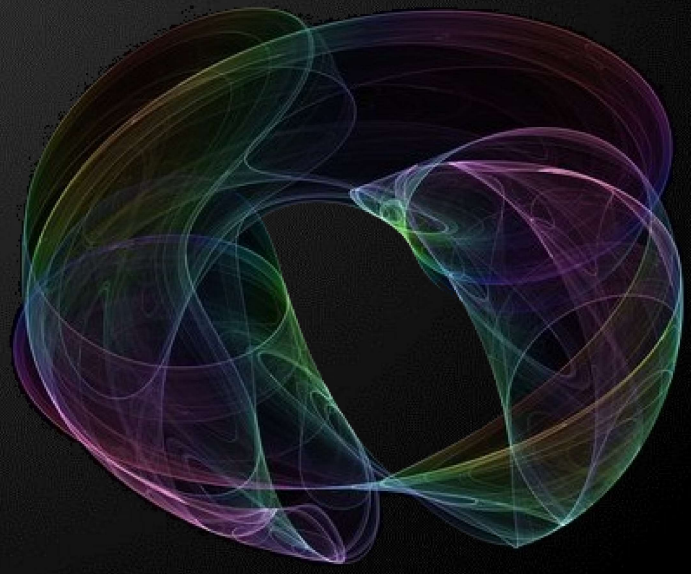
```
Console.Write("Please enter your name: ");
string name = Console.ReadLine();
Console.Write("Hello, {0}! ", name);
Console.WriteLine("Welcome to our party!");
```



Telerik Academy

Reading and Printing Strings

Demo



Telerik Academy

Manipulating Strings

Comparing, Concatenating, Searching, Extracting Substrings, Splitting

Telerik Academy

Comparing Strings

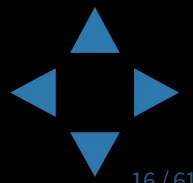
- Several ways to compare two strings:
  - Dictionary-based string comparison
  - Case-insensitive

```
string.Compare(str1, str2, true);
```

  - Case-sensitive

```
string.Compare(str1, str2, false);
```

Result	When
0	str1 is equal to str2
< 0	str1 is is before str2 lexicographically
> 0	str1 is is after str2 lexicographically



Telerik Academy

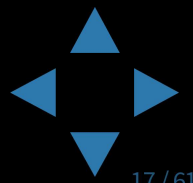
Comparing Strings

- Equality checking by operator ==
  - Always returns `true` or `false`
  - Performs case-sensitive compare

```
bool areStringsEqual = (str1 == str2);
```

- Using the case-sensitive Equals() method
  - The same effect like the operator ==

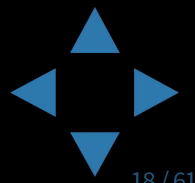
```
bool areStringsEqual = str1.Equals(str2));
```



Comparing Strings: *Example*

- Finding the first string in a lexicographical order from a given list of strings:

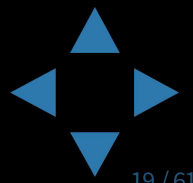
```
string[] towns = {"Sofia", "Varna", "Burgas", "Plovdiv",
                  "Pleven",  "Rousse", "Yambol"};

string bestTown = towns[0];
foreach (string town in towns)
{
   if (String.Compare(town, bestTown) < 0)
   {
      bestTown = town;
   }
}

Console.WriteLine("First town: {0}", bestTown);
```

Telerik Academy

Comparing Strings

Demo



Telerik Academy

Concatenating Strings

- There are two ways to combine strings:
  - Using the `Concat()` method

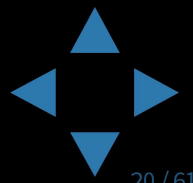
```
string str = String.Concat(str1, str2);
```

- Using the + or the += operators

```
string str = str1 + str2 + str3;
string str += str1;
```

- Any object can be appended to a `string`

```
string name = "Peter";
int age = 22;
string s = name + " " + age; // → "Peter 22"
```

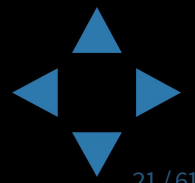



Telerik Academy

Concatenating Strings *Example*

- *Example:* concatenating strings

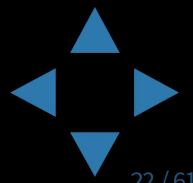
```
string firstName = "Telerik";
string lastName = "Academy";

string fullName = firstName + " " + lastName;
Console.WriteLine(fullName);
// Telerik Academy

int age = 5;

string nameAndAge = "Name: " + fullName + "\nAge: " + age;
Console.WriteLine(nameAndAge);
// Name: Telerik Academy
// Age: 5
```



Telerik Academy

Concatenating String

Demo



Telerik Academy

Searching in Strings

- Finding a character or substring within given string
  - Case sensitive
  - First occurrence

```
string.IndexOf(string str)
```

- First occurrence starting at given position

```
string.IndexOf(string str, int startIndex)
```

- Last occurrence

```
string.LastIndexOf(string str)
```

Telerik Academy

Searching in Strings
Example

- *Example:* searching in strings

```
string str = "C# Programming Course";
int index = str.IndexOf("C#"); // index = 0
index = str.IndexOf("Course"); // index = 15
index = str.IndexOf("COURSE"); // index = -1
// IndexOf is case-sensetive. -1 means not found
index = str.IndexOf("ram"); // index = 7
index = str.IndexOf("r"); // index = 4
index = str.IndexOf("r", 5); // index = 7
index = str.IndexOf("r", 8); // index = 18
```


Telerik Academy

Searching in Strings

Demo

Telerik Academy

Extracting Substrings

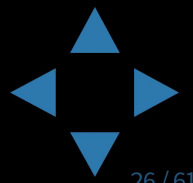
- Extracting substrings

```
string.Substring(int startIndex, int length)
```

```
string filename = @"C:\Pics\Rila2009.jpg";
string name = filename.Substring(8, 8);
// name is Rila2009
```

```
str.Substring(int startIndex)`
```

```
string filename = @"C:\Pics\Summer2009.jpg";
int index = filename.LastIndexOf(@"\");
string nameAndExtension = filename.Substring(index + 1);
// nameAndExtension is Summer2009.jpg
```



Telerik Academy

Extracting from Strings

Demo

Telerik Academy

Splitting Strings

- To split a string by given separator(s) use the following method:

```
string[] string.Split(params char[])
```

- *Example* splitting by comma:

```
string listOfBeers = "Amstel, Zagorka, Tuborg, Becks.";
string[] beers = listOfBeers.Split(' ', ',', '.');

Console.WriteLine("Available beers are:");
foreach (string beer in beers)
{
    Console.WriteLine(beer);
}
```


Telerik Academy

Splitting Strings

Demo

Telerik Academy

Other String Operations

Replacing, Removing, etc..

Telerik Academy

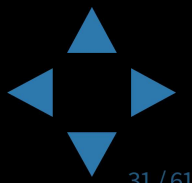
Replacing and Deleting Substrings

- `string.Replace(string, string)`
  - Replaces all occurrences of the first string with the second string
  - The result is new string (strings are immutable)

```
string cocktail = "Vodka + Martini + Cherry";
cocktail.Replace("+", "and"); // Vodka and Martini and Cherry
```

- `string.Remove(index, length)`
  - Deletes part of a string and produces new string as result

```
string price = "$ 1234567";
string lowPrice = price.Remove(2, 3); // $ 4567
```

Telerik Academy

Deleting and Replacing Strings

Demo

Telerik Academy Changing Character Casing

- Using method `string.ToLower()`

```
string alpha = "aBcDeFg";
string lowerAlpha = alpha.ToLower();
Console.WriteLine(lowerAlpha); // abcdefg
```

- Using method `string.ToUpper()`

```
string alpha = "aBcDeFg";
string upperAlpha = alpha.ToUpper();
Console.WriteLine(upperAlpha); // ABCDEFG
```

Telerik Academy

Changing Character Casing

Demo

Telerik Academy

Trimming White Space

- Using `string.Trim()`

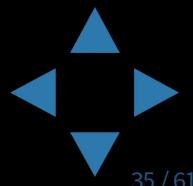
```
string s = "    example of white space    ";
string clean = s.Trim();
Console.WriteLine("\"{0}\", clean); // "example of white space"
```

- Using `string.Trim(chars)`

```
string s = " \t\nHello!!! \n";
string clean = s.Trim(' ', ',' ,'!', '\n','\t');
Console.WriteLine(clean); // Hello
```

- Using `string.TrimStart()` and `string.TrimEnd()`

```
string s = "   CSharp   ";
string clean = s.TrimStart();
Console.WriteLine("\"{0}\", clean); // "CSharp   "
```



Telerik Academy

Trimming Whitespace

Demo

Telerik Academy

Building and Modifying Strings

Using the StringBuilder Class

Telerik Academy

How are Strings Constructed?

- Strings are immutable!
  - Methods like `Concat()`, `Replace()`, `Trim()`, ... return new strings, do not modify the old one
- Do not use + for strings in a loop!
  - It runs very, very inefficiently!
  - *Example:*

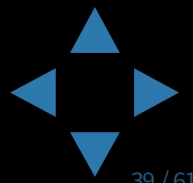
```
public static string DuplicateChar(char ch, int count)
{
    string result = "";
    for (int i = 0; i < count; i++)
    {
        result += ch;
    }

    return result;
}
```

Very bad practice. Avoid this!

Telerik Academy

Slow Character Duplication

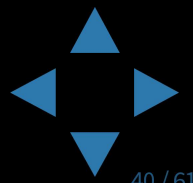
Demo



Telerik Academy

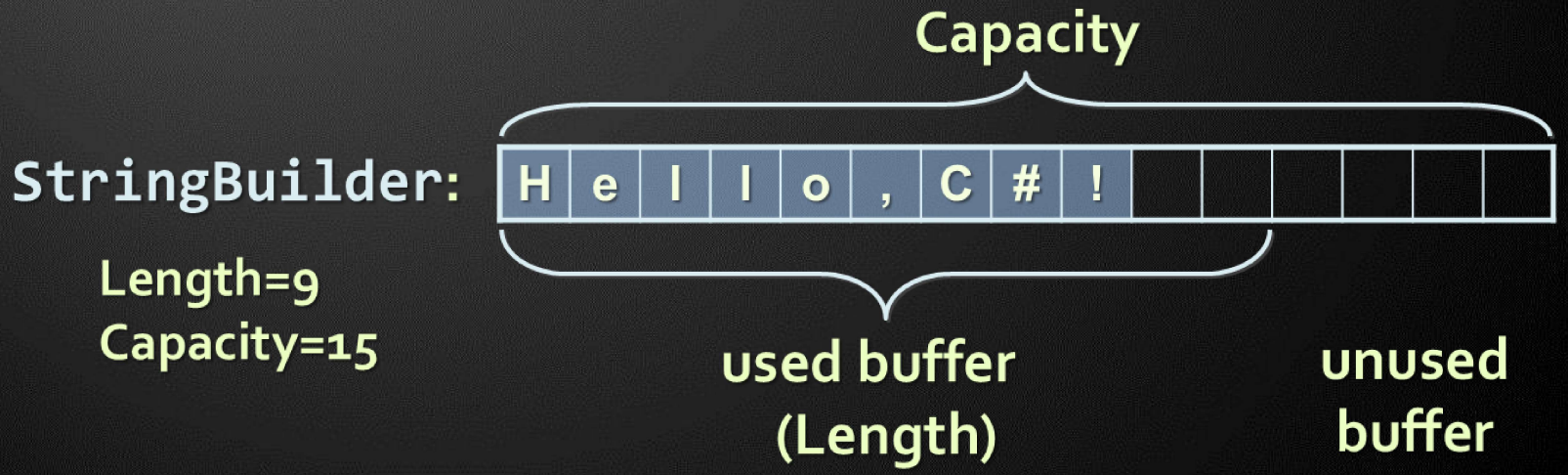
Introducing StringBuilder

Faster way to generate strings



Introducing StringBuilder

- `StringBuilder` keeps a buffer memory, allocated in advance
  - Most operations use the buffer memory and do not allocate new objects



Concatenating with StringBuilder?

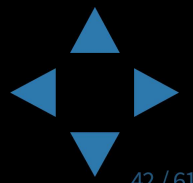
- Consider the following string concatenation:

```
string result = str1 + str2;
```

- It is equivalent to this code:

```
StringBuilder sb = new StringBuilder();
sb.Append(str1);
sb.Append(str2);
string result = sb.ToString();
```

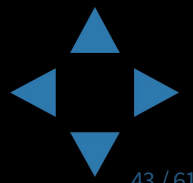
- Several new objects are created and left to the garbage collector for deallocation
  - What happens when using + in a loop?



Telerik Academy

The StringBuilder Class

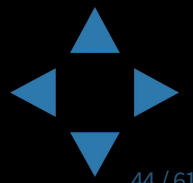
- `StringBuilder(int capacity)` constructor allocates in advance buffer of given size
  - By default 16 characters are allocated
- `string.Capacity` holds the currently allocated space (in characters)
- `this[int index]` (indexer in C#) gives access to the char value at given position
- `string.Length` holds the length of the string in the buffer



Telerik Academy

The StringBuilder Class

- `StringBuilder.Append(…)` appends a string or another object after the last character in the buffer
- `StringBuilder.Remove(int startIndex, int length)` removes the characters in given range
- `StringBuilder.Insert(int index, string str)` inserts given string (or object) at given position
- `StringBuilder.Replace(string oldStr, string newStr)` replaces all occurrences of a substring
- `StringBuilder.ToString()` converts `theStringBuilder` to a `string`

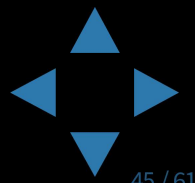


Telerik Academy

Changing the Contents of a `string` with a `StringBuilder`

- Use the `System.Text.StringBuilder` class for modifiable strings of characters:
  - *Example* Reversing a `string`:

```
public static string ReverseString(string s)
{
StringBuilder sb = new StringBuilder();

for (int i = s.Length-1; i >= 0; i--)
{
   sb.Append(s[i]);
}

return sb.ToString();
}
```

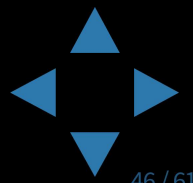



Telerik Academy

Appending characters to `string` with `StringBuilder`

- Use `StringBuilder` if you need to keep adding characters to a string
  - *Example* Extracting all capital letters from a `string`:

```
public static string ExtractCapitals(string text)
{
  StringBuilder result = new StringBuilder();

  foreach(char character in text)
  {
    if (Char.IsUpper(character))
    {
      result.Append(character);
    }
  }

  return result.ToString();
}
```

Telerik Academy

Using StringBuilder

Demo

Telerik Academy

Formatting Strings

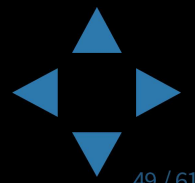
Using `ToString()` and `String.Format()`

Telerik Academy

Method ToString()

- All classes in C# have public virtual method ToString()
  - Returns a human-readable, culture-sensitive string representing the object
    - If implemented correctly
  - Most .NET Framework types have own implementation of ToString()
    - int, float, bool, DateTime
- *Example:*

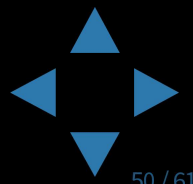
```
int number = 5;
string s = "The number is " + number.ToString();
Console.WriteLine(s); // The number is 5
```

Telerik Academy Method ToString(format)

- We can apply specific formatting when converting objects to string
  - ToString(formatString) method

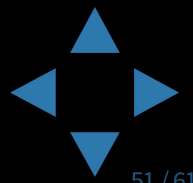
```
int number = 42;
string s = number.ToString("D5"); // 00042

s = number.ToString("X"); // 2A

// Considering the default culture is bulgarian
s = number.ToString("C"); // 42,00 лв

double d = 0.375;
s = d.ToString("P2"); // 37,50 %
```

Telerik Academy

Formatting Strings

- The formatting strings are different for the different types
- Some formatting strings for numbers:
  - D – number (for integer types)
  - C – currency (according to current culture)
  - E – number in exponential notation
  - P – percentage
  - X – hexadecimal number
  - F – fixed point (for real numbers)

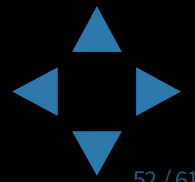


Telerik Academy

Method String.Format()

- String.Format(template, params...):
  - Applies templates for formatting strings
    - Placeholders are used for dynamic text
    - The same as Console.WriteLine(…)
- Example:

```
string template = "If I were {0}, I would {1}.";
string sentence1 =
    String.Format(template, "developer", "know C#");
Console.WriteLine(sentence1);
// If I were developer, I would know C#.

string sentence2 =
    String.Format(template, "elephant", "weigh 4500 kg");
Console.WriteLine(sentence2);
// If I were elephant, I would weigh 4500 kg.
```

Telerik Academy

Composite Formatting

- The placeholders in the composite formatting strings are specified as follows:

```
{index[,alignment][:formatString]}
```

- *Examples*:

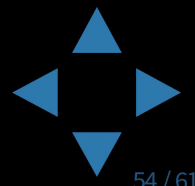
```
double d = 0.375;
s = String.Format("{0,10:F5}", d);
// s = "   0,37500"

int number = 42;
Console.WriteLine("Dec {0:D} = Hex {1:X}", number, number);
// Dec 42 = Hex 2A
```


Telerik Academy

Formatting Dates

- Dates have their own formatting strings
  - d, dd – day (with/without leading zero)
  - M, MM – month
  - yy, yyyy – year (2 or 4 digits)
  - h, HH, m, mm, s, ss – hour, minute, second

```
DateTime now = DateTime.Now;
Console.WriteLine ("Now is {0:d.MM.yyyy HH:mm:ss}", now);
// Now is 31.11.2009 11:30:32
```



Telerik Academy

Cultures

- Cultures in .NET specify formatting / parsing settings specific to country / region / language
- Printing the current culture:

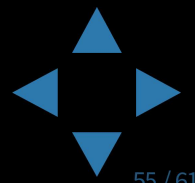
```
Console.WriteLine(System.Threading.Thread.CurrentThread.CurrentCultur
```

- Changing the current culture:

```
System.Threading.Thread.CurrentThread.CurrentCulture =
      new CultureInfo("en-CA");
```

- Culture-sensitive `ToString()`:

```
CultureInfo culture = new CultureInfo("fr-CA");
string s = number.ToString("C", culture); // 42,00 $
```



Telerik Academy

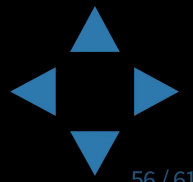
Parsing Numbers and Dates

- Parsing numbers and dates is culture-sensitive
- Parsing a real number using "." as separator:

```
string str = "3.14";
Thread.CurrentThread.CurrentCulture = CultureInfo.InvariantCulture;
float f = float.Parse(str); // f = 3.14
```

- Parsing a date in specific format:

```
string dateStr = "25.07.2011";
DateTime date = DateTime.ParseExact(dateStr,
        "dd.MM.yyyy", CultureInfo.InvariantCulture);
```

Telerik Academy

Formatting Strings

Demo

Telerik Academy

Summary

- Strings are immutable sequences of characters (instances of `System.String`)
  - Declared by the keyword `string` in C#
  - Can be initialized by string literals
- Most important string processing members are:
  - `IndexOf(str)`
  - `LastIndexOf(str)`
  - `Compare(str1, str2)`
  - `Substring(startIndex,length)`
  - `Remove(startIndex, length)`
  - `Replace(oldStr, newStr)`
  - `Length`
  - `this[]`
  - `ToLower()`
  - `ToUpper()`
  - `Trim()`

Telerik Academy

Summary

- Objects can be converted to strings and can be formatted in different styles (using `ToString()` method)
- Strings can be constructed by using placeholders and formatting strings (`String.Format(…)`)