



# Strings and Text Processing

Processing and Manipulating Text Information

---

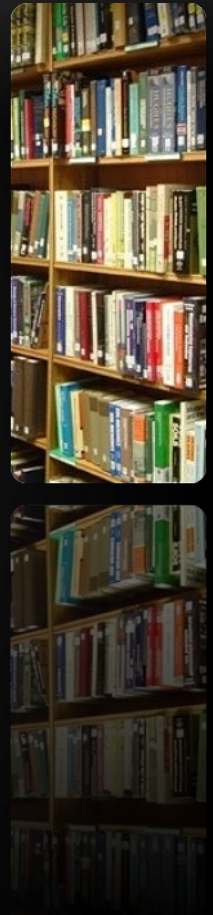
**Svetlin Nakov**

Telerik Corporation

[www.telerik.com](http://www.telerik.com)



1. What is String?
2. Creating and Using Strings
  - ♦ Declaring, Creating, Reading and Printing
3. Manipulating Strings
  - ♦ Comparing, Concatenating, Searching, Extracting Substrings, Splitting
4. Other String Operations
  - ♦ Replacing Substrings, Deleting Substrings, Changing Character Casing, Trimming



1. Building and Modifying Strings
  - ♦ Using StringBuilder Class
2. Formatting Strings





# What Is String?

# What Is String?

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

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



s → 

H	e	l	l	o	,		C	#
---	---	---	---	---	---	--	---	---

# 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 in to support multiple languages and alphabets
- ◆ Strings are stored in the dynamic memory (managed heap)
- ◆ `System.String` is reference type

# The System.String Class (2)

- ◆ 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 = 6  
char ch = s[1]; // ch = 'e'
```

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





# Strings – First Example



```
static void Main()
{
    string s =
        "Stand up, stand up, Balkan Superman.";
    Console.WriteLine("s = \"{0}\"", s);
    Console.WriteLine("s.Length = {0}", s.Length);
    for (int i = 0; i < s.Length; i++)
    {
        Console.WriteLine("s[{0}] = {1}", i, s[i]);
    }
}
```



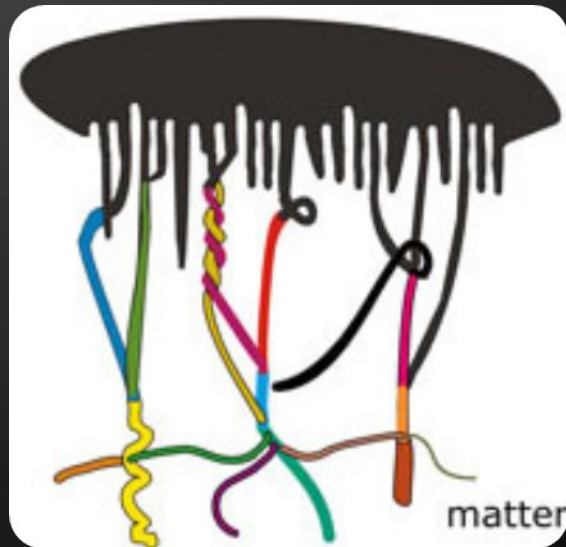
# Strings – First Example

Live Demo



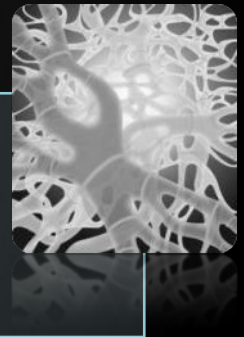
# Creating and Using Strings

Declaring, Creating, Reading and Printing



- ◆ There are two 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

- ◆ 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



# Creating Strings (2)

- ◆ Not initialized variables has value of null

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

- ◆ Assigning a 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();
```

# Reading and Printing Strings

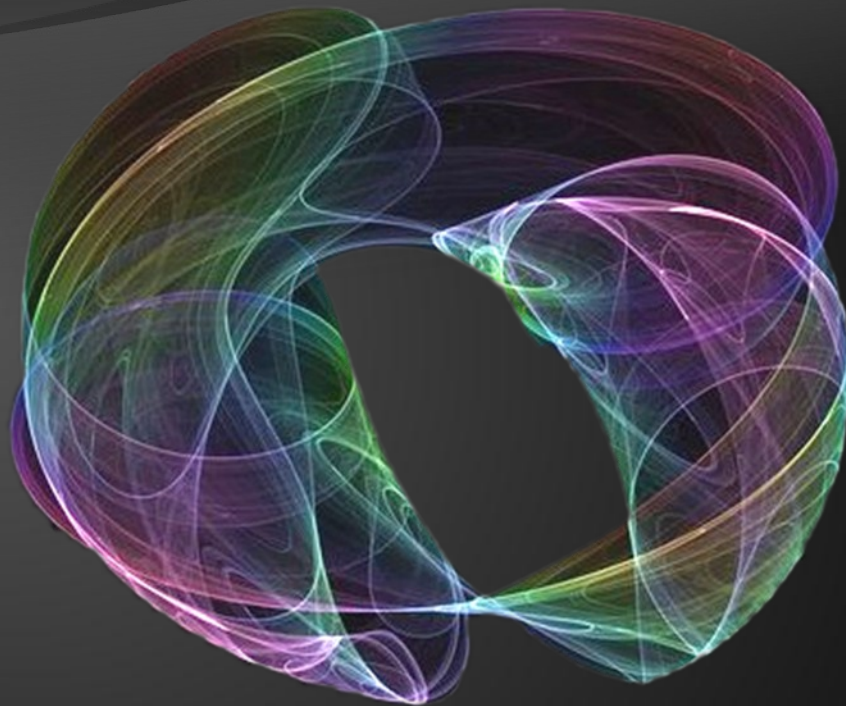
- ◆ Reading strings from the console
  - ◆ Use the method `Console.ReadLine()`

```
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!");
```





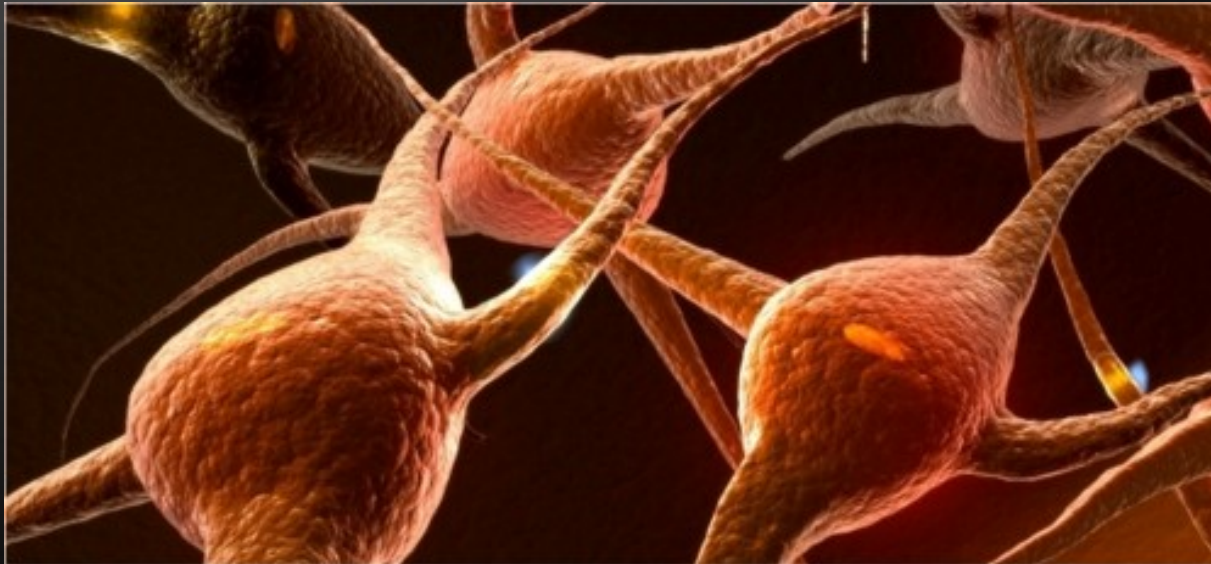
# Reading and Printing Strings

Live Demo



# Manipulating Strings

Comparing, Concatenating, Searching,  
Extracting Substrings, Splitting



- ◆ A number of ways exist to compare two strings:
  - ◆ Dictionary-based string comparison
    - ◆ Case-insensitive

```
int result = string.Compare(str1, str2, true);  
// result == 0 if str1 equals str2  
// result < 0 if str1 is before str2  
// result > 0 if str1 is after str2
```

- ◆ Case-sensitive

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

# Comparing Strings (2)

- ◆ Equality checking by operator ==
  - ◆ Performs case-sensitive compare

```
if (str1 == str2)
{
    ...
}
```

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

```
if (str1.Equals(str2))
{
    ...
}
```

# Comparing Strings – Example

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

```
string[] towns = {"Sofia", "Varna", "Plovdiv",  
                  "Pleven", "Bourgas", "Rousse", "Yambol"};  
string firstTown = towns[0];  
for (int i=1; i<towns.Length; i++)  
{  
    string currentTown = towns[i];  
    if (String.Compare(currentTown, firstTown) < 0)  
    {  
        firstTown = currentTown;  
    }  
}  
Console.WriteLine("First town: {0}", firstTown);
```

# Comparing Strings

Live Demo



# 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 string

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



# Concatenating Strings – Example

```
string firstName = "Svetlin";  
string lastName = "Nakov";  
  
string fullName = firstName + " " + lastName;  
Console.WriteLine(fullName);  
// Svetlin Nakov  
  
int age = 25;  
  
string nameAndAge =  
    "Name: " + fullName +  
    "\nAge: " + age;  
Console.WriteLine(nameAndAge);  
// Name: Svetlin Nakov  
// Age: 25
```



# Concatenating Strings

Live Demo



# Searching in Strings

- ◆ Finding a character or substring within given string

- ◆ First occurrence

```
IndexOf(string str)
```

- ◆ First occurrence starting at given position

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

- ◆ Last occurrence

```
LastIndexOf(string)
```



# Searching in Strings – Example

```
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-sensitive. -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
```

index =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...
s[index] =	C	#		P	r	o	g	r	a	m	m	i	n	g	...

# Searching in Strings

Live Demo



## ◆ Extracting substrings

- ◆ `str.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";  
string nameAndExtension = filename.Substring(8);  
// nameAndExtension is Summer2009.jpg
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C	:	\	P	i	c	s	\	R	i	l	a	2	0	0	5	.	j	p	g

# Extracting Substrings

Live Demo



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

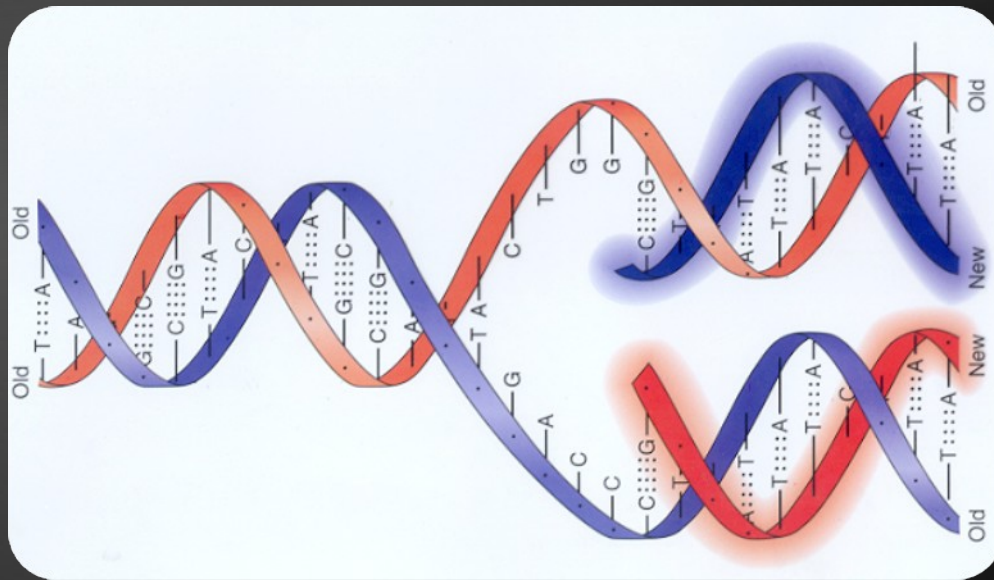
```
string[] Split(params char[])
```

- ◆ Example:

```
string listOfBeers =  
    "Amstel, Zagorka, Tuborg, Becks.";   
string[] beers =  
    listOfBeers.Split(' ', ',', '.');  
Console.WriteLine("Available beers are:");  
foreach (string beer in beers)  
{  
    Console.WriteLine(beer);  
}
```





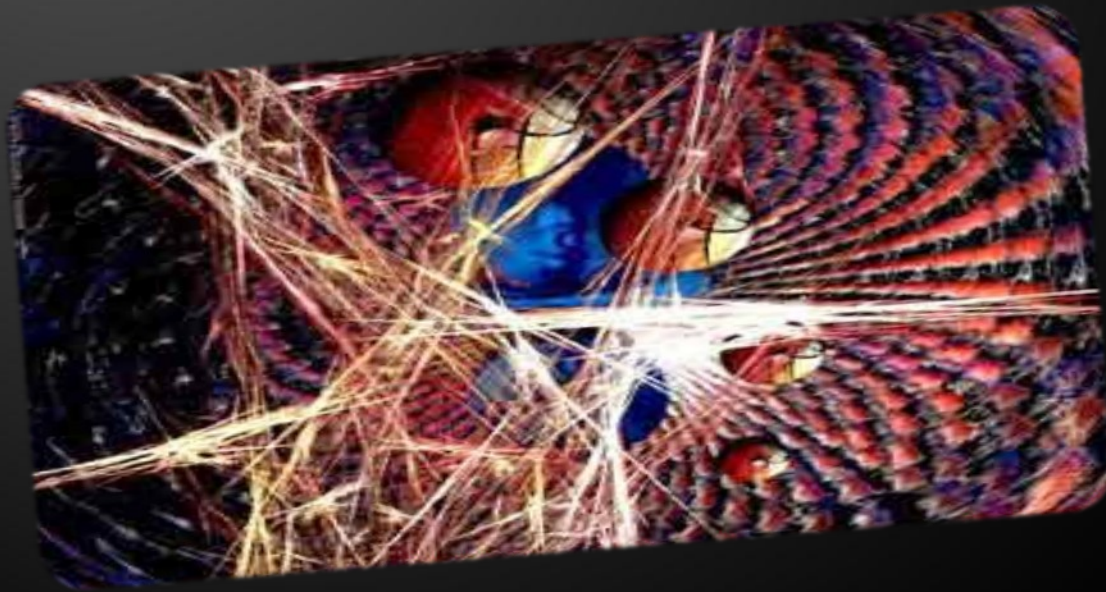


# Splitting Strings

Live Demo

# Other String Operations

Replacing Substrings, Deleting Substrings,  
Changing Character Casing, Trimming



# Replacing and Deleting Substrings

- ◆ `Replace(string, string)` – replaces all occurrences of given string with another
  - ◆ The result is new string (strings are immutable)

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

- ◆ `Remove(index, length)` – deletes part of a string and produces new string as result

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

# Changing Character Casing

## ◆ Using method ToLower()

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

## ◆ Using method ToUpper()

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



# Trimming White Space

- ◆ Using method `Trim()`

```
string s = "    example of white space    ";  
string clean = s.Trim();  
Console.WriteLine(clean);
```

- ◆ Using method `Trim(chars)`

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

- ◆ Using `TrimStart()` and `TrimEnd()`

```
string s = "    C#    ";  
string clean = s.TrimStart(); // clean = "C#    "
```





# Other String Operations

Live Demo

# Building and Modifying Strings

Using StringBuilder Class





# Constructing Strings

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

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

**Very bad  
practice. Avoid  
this!**



# Slow Building Strings with +

Live Demo

# Changing the Contents of a String

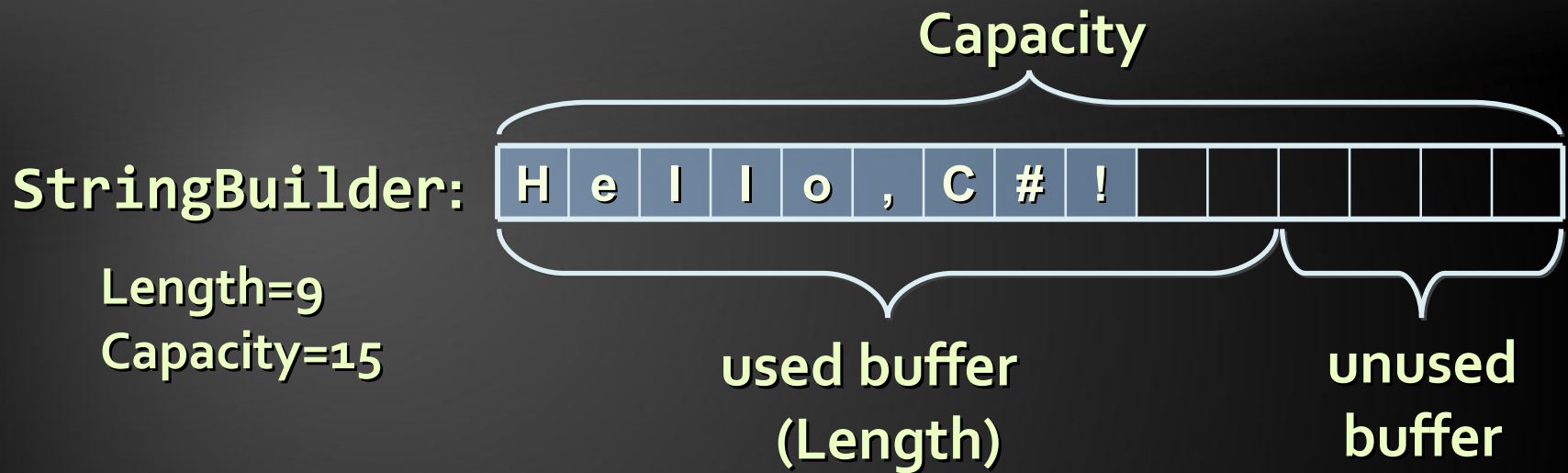
## – StringBuilder

- ◆ Use the `System.Text.StringBuilder` class for modifiable strings of characters:

```
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();
}
```

- ◆ Use `StringBuilder` if you need to keep adding characters to a string

# The StringBuilder Class



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

## The StringBuilder Class (2)

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

# The StringBuilder Class (3)

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

# StringBuilder – Example

- ◆ Extracting all capital letters from a string

```
public static string ExtractCapitals(string s)
{
    StringBuilder result = new StringBuilder();
    for (int i = 0; i < s.Length; i++)
    {
        if (Char.IsUpper(s[i]))
        {
            result.Append(s[i]);
        }
    }
    return result.ToString();
}
```





# How the + Operator Does String Concatenations?

- ◆ Consider 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();
```

- ◆ Actually several new objects are created and leaved to the garbage collector
  - ◆ What happens when using + in a loop?

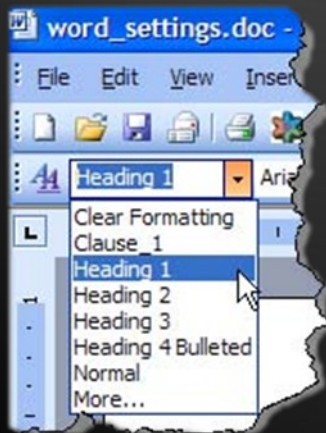


# Using StringBuilder

Live Demo

# Formatting Strings

Using ToString() and String.Format()



# Method ToString()

- ◆ All classes have public virtual method ToString()
  - ◆ Returns a human-readable, culture-sensitive string representing the object
  - ◆ Most .NET Framework types have own implementation of ToString()
    - ◆ int, float, bool, DateTime

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

# 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  
  
// Consider the default culture is Bulgarian  
s = number.ToString("C"); // 42,00 лв  
  
double d = 0.375;  
s = d.ToString("P2"); // 37,50 %
```

- ◆ 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)



# Method String.Format()

- ◆ Applies templates for formatting strings
  - ◆ Placeholders are used for dynamic text
  - ◆ Like Console.WriteLine(...)

```
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.
```

- ◆ 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
```

- ◆ 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
```

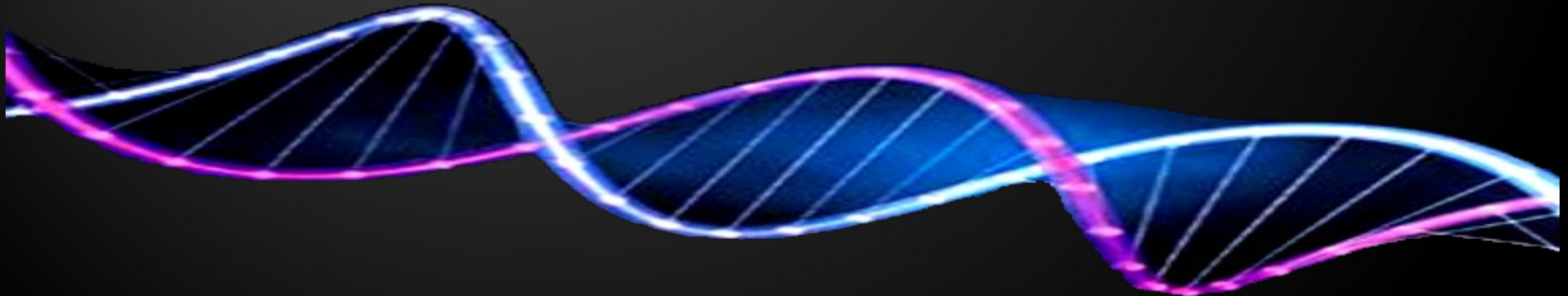
# Formatting Strings

Live Demo



- ◆ 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:
  - ◆ `Length`, `this[]`, `Compare(str1, str2)`, `IndexOf(str)`, `LastIndexOf(str)`, `Substring(startIndex, length)`, `Replace(oldStr, newStr)`, `Remove(startIndex, length)`, `ToLower()`, `ToUpper()`, `Trim()`

- ◆ 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(...)`)





# Strings and Text Processing



Questions?

1. Describe the strings in C#. What is typical for the `string` data type? Describe the most important methods of the `String` class.
2. Write a program that reads a string, reverses it and prints it on the console. Example: "sample" → "elpmas".
3. Write a program to check if in a given expression the brackets are put correctly. Example of correct expression:  $((a+b)/5-d)$ . Example of incorrect expression:  $)(a+b))$ .

1. Write a program that finds how many times a substring is contained in a given text (perform case insensitive search).

Example: The target substring is "in". The text is as follows:

```
We are living in an yellow submarine. We don't  
have anything else. Inside the submarine is very  
tight. So we are drinking all the day. We will  
move out of it in 5 days.
```

The result is: 9.

1. You are given a text. Write a program that changes the text in all regions surrounded by the tags `<upcase>` and `</upcase>` to uppercase. The tags cannot be nested. Example:

```
We are living in a <upcase>yellow  
submarine</upcase>. We don't have  
<upcase>anything</upcase> else.
```

**The expected result:**

```
We are living in a YELLOW SUBMARINE. We don't have  
ANYTHING else.
```

1. Write a program that reads from the console a string of maximum 20 characters. If the length of the string is less than 20, the rest of the characters should be filled with '\*'. Print the result string into the console.
2. Write a program that encodes and decodes a string using given encryption key (cipher). The key consists of a sequence of characters. The encoding/decoding is done by performing XOR (exclusive or) operation over the first letter of the string with the first of the key, the second – with the second, etc. When the last key character is reached, the next is the first.

1. Write a program that extracts from a given text all sentences containing given word.

**Example:** The word is "in". The text is:

```
We are living in a yellow submarine. We don't have  
anything else. Inside the submarine is very tight.  
So we are drinking all the day. We will move out  
of it in 5 days.
```

**The expected result is:**

```
We are living in a yellow submarine.  
We will move out of it in 5 days.
```

**Consider that the sentences are separated by "." and the words – by non-letter symbols.**



1. We are given a string containing a list of forbidden words and a text containing some of these words. Write a program that replaces the forbidden words with asterisks. Example:

```
Microsoft announced its next generation PHP  
compiler today. It is based on .NET Framework 4.0  
and is implemented as a dynamic language in CLR.
```

**Words: "PHP, CLR, Microsoft"**

**The expected result:**

```
***** announced its next generation ***  
compiler today. It is based on .NET Framework 4.0  
and is implemented as a dynamic language in ***.
```

1. Write a program that converts a string to a sequence of C# Unicode character literals. Use format strings. Sample input:

```
Hi!
```

Expected output:

```
\u0048\u0069\u0021
```

5. Write a program that reads a number and prints it as a decimal number, hexadecimal number, percentage and in scientific notation. Format the output aligned right in 15 symbols.

1. Write a program that parses an URL address given in the format:

```
[protocol]://[server]/[resource]
```

and extracts from it the [protocol], [server] and [resource] elements. For example from the URL `http://www.devgb.org/forum/index.php` the following information should be extracted:

```
[protocol] = "http"
```

```
[server] = "www.devgb.org"
```

```
[resource] = "/forum/index.php"
```

1. Write a program that reverses the words in given sentence.

Example: "C# is not C++, not PHP and not Delphi!"  
→ "Delphi not and PHP, not C++ not is C#!".

3. A dictionary is stored as a sequence of text lines containing words and their explanations. Write a program that enters a word and translates it by using the dictionary. Sample dictionary:

```
.NET - platform for applications from Microsoft  
CLR - managed execution environment for .NET  
namespace - hierarchical organization of classes
```

1. Write a program that replaces in a HTML document given as string all the tags `<a href="...">...</a>` with corresponding tags `[URL=...].../URL]`. Sample HTML fragment:

```
<p>Please visit <a href="http://academy.telerik.com">our site</a> to choose a training course. Also visit <a href="www.devbg.org">our forum</a> to discuss the courses.</p>
```



```
<p>Please visit [URL=http://academy.telerik.com]our site[/URL] to choose a training course. Also visit [URL=www.devbg.org]our forum[/URL] to discuss the courses.</p>
```

1. Write a program that reads two dates in the format: `day.month.year` and calculates the number of days between them. Example:

```
Enter the first date: 27.02.2006  
Enter the second date: 3.03.2004  
Distance: 4 days
```

4. Write a program that reads a date and time given in the format: `day.month.year`  
`hour:minute:second` and prints the date and time after 6 hours and 30 minutes (in the same format).



1. Write a program for extracting all the email addresses from given text. All substrings that match the format `<identifier>@<host>...<domain>` should be recognized as emails.
2. Write a program that extracts from a given text all the dates that match the format `DD.MM.YYYY`. Display them in the standard date format for Canada.
3. Write a program that extracts from a given text all palindromes, e.g. "ABBA", "1ama1", "exe".

1. Write a program that reads a string from the console and prints all different letters in the string along with information how many times each letter is found.
2. Write a program that reads a string from the console and lists all different words in the string along with information how many times each word is found.
3. Write a program that reads a string from the console and replaces all series of consecutive identical letters with a single one. Example: "aaaaabbbbbbcdddeeeedssaa" → "abcdedsa".

1. Write a program that reads a list of words, separated by spaces and prints the list in an alphabetical order.
2. Write a program that extracts from given HTML file its title (if available), and its body text without the HTML tags. Example:

```
<html>
  <head><title>News</title></head>
  <body><p><a href="http://academy.telerik.com">Telerik
    Academy</a>aims to provide free real-world practical
    training for young people who want to turn into
    skillful .NET software engineers.</p></body>
</html>
```