Strings in C#

Școala Informală de IT

Cluj-Napoca

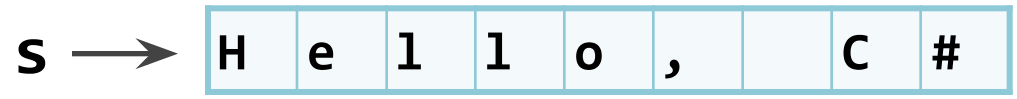
Agenda

- Strings
  - What is String?
  - Creating and Using Strings
  - Manipulating Strings
  - Other String Operations
  - Building and Modifying Strings
  - Formatting Strings
  - String Interpolation
  - Cultures and Culture-Sensitive Formatting
  - Parsing Numbers and Dates

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#";
```



Școala informală de IT

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

Școala informală de IT

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

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

Strings

```
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]);
    }
}
```

Școala
informală
de IT

Creating and Using Strings

Declaring, Creating, Reading and Printing

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

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

Creating Strings

- 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();
```

Școala informală de IT

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!");
```

Școala
informală
de IT

Manipulating Strings

Comparing, Concatenating, Searching, Extracting Substrings, Splitting

Comparing Strings

- Several ways 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

- 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

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

```
string[] towns = {"Cluj-Napoca", "Brasov", "Iasi",
   "Bucuresti", "Oradea", "Timisoara", "Alba Iulia"};
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);
```

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

Concatenating Strings

```
string firstName = "John";
string lastName = "Smith";

string fullName = firstName + " " + lastName;
Console.WriteLine(fullName);
// John Smith

int age = 25;

string nameAge = "Name: " + fullName + "\nAge: " + age;
Console.WriteLine(nameAge);
// Name: John Smith
// Age: 25
```

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)
```

Școala informală de IT

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-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	...

Extracting Substrings

- **Extracting substrings**
  - `str.Substring(int startIndex, int len)`

```
string filename = @"C:\Pics\Image1.jpg";
string name = filename.Substring(8, 6);
// name is Image1
```

  - `str.Substring(int startIndex)`

```
string filename = @"C:\Pics\Summer2015.jpg";
string nameAndExtension = filename.Substring(8);
// nameAndExtension is Summer2015.jpg
```

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

Splitting Strings

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

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

- Example:

```
string listOfBeers = "Ursus, Ciuc, Tuborg, Becks.";
string[] beers = listOfBeers.Split(' ', ',', '.');
Console.WriteLine("Available beers are:");

foreach (string beer in beers)
{
    Console.WriteLine(beer);
}
```

Școala
informală
de IT

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);
```

Școala informală de IT

Trimming White Space

- Using `Trim()`

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

- Using `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#   "
```

Școala
informală
de IT

Building and Modifying Strings

Using the `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;
}
```

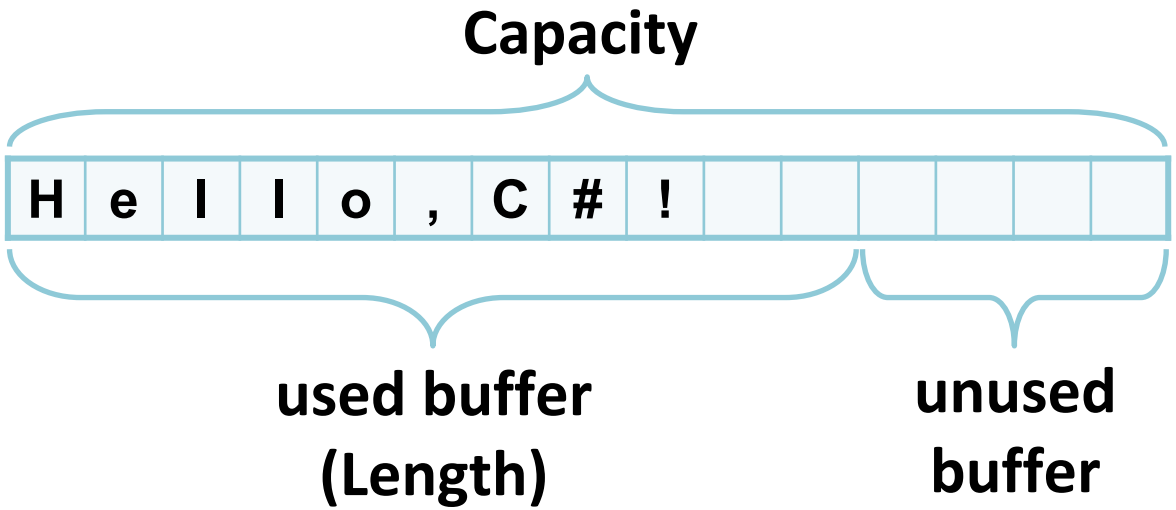
Very bad practice. Avoid this!

StringBuilder: How It Works?

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

`StringBuilder`:

Length=9
Capacity=15



Școala informală de IT

How the + Performs String Concatenations?

- 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?

Școala informală de IT

StringBuilder Class

- `StringBuilder(int capacity)` constructor allocates in advance buffer of 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

Școala informală de IT

StringBuilder Class

- `Append(...)` appends a 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
- `ToString()` converts the `StringBuilder` to `String`

Școala informală de IT

Changing a String with 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

Școala
informală
de IT

Formatting Strings

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

Școala informală de IT

Method ToString()

- All classes in C# 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 Romanian
s = number.ToString("C"); // 42,00 RON

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

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)

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 expression1 = String.Format(template,
    "dev", "know C#");
Console.WriteLine(expression1);
// If I were developer, I would know C#.
string expression2 = String.Format(template,
    "elephant", "weigh 4500 kg");
Console.WriteLine(expression2);
// If I were elephant, I would weigh 4500 kg.
```

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
```

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
```

Cultures

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

```
Console.WriteLine(System.Threading.
  Thread.CurrentThread.CurrentCulture);
```

- 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 $
```

Școala informală de IT

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);
```

Școala
informală
de IT

String Interpolation

String Interpolation

- $ sign marks an interpolated string which is much more fluent than template strings
- Syntax:

```
$"<text> {<interpolated-expression> [,<field-width>]
[:<format-string>] } <text> ..."
```

- field-width = number of characters in the field.
- format-string = type of object being formatted. Ex. DateTime value is "D" or "d".

Școala informală de IT

String Interpolation - Example

```
string student = "John";
string age = 25;
string str = $"Student {student} is {age} yrs old";
// Student John is 25 yrs old
```

```
string student = @"John "the best"";
string age = 25;
string str = $@"Student {student} is {age} yrs old";
// Student John "the best" is 25 yrs old
```

Școala informală de IT

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

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(...)`)

Resources

- http://csharp.net-informations.com/
- Telerik