Computer programs work with data. Spreadsheets, text editors, calculators or chat clients. Tools to work with various data types are essential part of a modern computer language. A data typeis a set of values, and the allowable operations on those values. The two fundamental data types in C# are value types and reference types.Strings are one of them.

**1.Strings**

String is a data type representing textual data in computer programs. A string is a sequence of characters. In C#, a string is a sequence of unicode characters. It is a data type which stores a sequence of data values, usually bytes, in which elements usually stand for characters according to a character encoding. When a string appears literally in source code, it is known as a string literal.Strings are enclosed by double quotes.Since strings are very important in every programming language. Here is a small example. Strings are represented by System.String objects in .NET Framework . Strings are stored in the dynamic memory (managed heap).System.String is reference type .Have fixed length (String.Length). Elements can be accessed directly by index. The index is in the range [0...Length-1] .

**2.Operations on strings**

## Multiline strings

## It is possible to create a multiline string in C#.

using System;

public class CSharpApp

{

static void Main()

{

string multiString = @"I cheated myself

like I knew I would

I told ya, I was trouble

you know that I'm no good";

Console.WriteLine(multiString);

}

## Comparing strings

We can compare two strings with a (==) operator.

using System;

public class CSharpApp

{

static void Main()

{

Console.WriteLine("12" == "12");

Console.WriteLine("17" == "9");

Console.WriteLine("aa" == "ab");

}

}

## Case-insensitive

## *int result = string.Compare(str1, str2, true);*

## *// r Case-insensitive esult == 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);*

## *Concatenating 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

## 

**3. Strings methods**

## Searching in Strings - finding a character or substring within given string q.

## First occurrence

## *IndexOf(string str)*

## Last occurrence

## *LastIndexOf(string)*

## Extracting Substrings

## *str.Substring(int startIndex, int length)*

## *string name = filename.Substring(8, 8);*

## Splitting Strings

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

## *string[] Split(params char[])*

## *string listOfBeers ="Amstel, Zagorka, Tuborg, Becks.";*

## *string[] beers = listOfBeers.Split(' ', ',', '.');*

## *Console.WriteLine("Available beers are:");*

## *foreach (string beer in beers)*

## *{*

## *Console.WriteLine(beer);*

## *}*

## 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*

## 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 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# "*

* **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 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.

* **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

* **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);*

## 4.StringBuilder

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

## 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.

## 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.

## *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();*

## *}*

#### 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 .