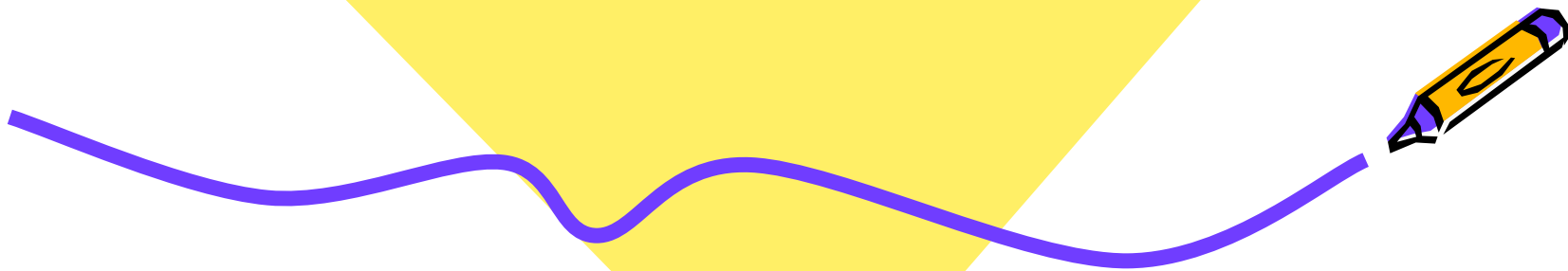
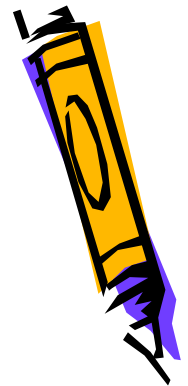


Лекция 6. Строингове, Масиви, Колекции.



Стрингове - `string` class

- В C# стрингове се обявяват с ключовата дума `string` (`System.String`).
- Този тип се отнася към примитивните, референтни типове.

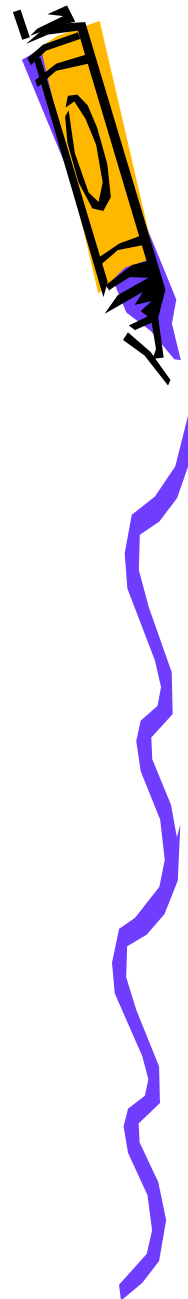


1. Стрингове - Обявяване

- Обекти на клас `System.String` се обявяват както всички обекти на простите типове:
 - с явна или отложена инициализация,
 - с явно или неявно извикване на конструктора на класа.
- Най-често конструктор явно не се извиква, а инициализацията е със символна константа:

```
string a = "12345";//12345
```

```
string b = "";
```



Стрингове - Обявяване

- Но при клас **String** има достатъчно много конструктори, които позволяват конструиране на стринг от:

- символ, повторен зададено количество пъти:

```
string sssss = new string('s',5); //sssss
```

- масив от символи char[]:

```
char[] yes = "Yes".ToCharArray();
```

//масив с име „yes“ от символи

- чрез използване на части от масиви от символи.

```
char[] yes = { 'Y', 'e', 's' }; //друг вариант
```

```
string b = new string(yes); // yes
```

```
string c = new string(yes, 0, 2); // ye
```



Стрингове - Обявяване

- Не е допустимо обаче следното:

```
//string s1 = new string("s1");
```

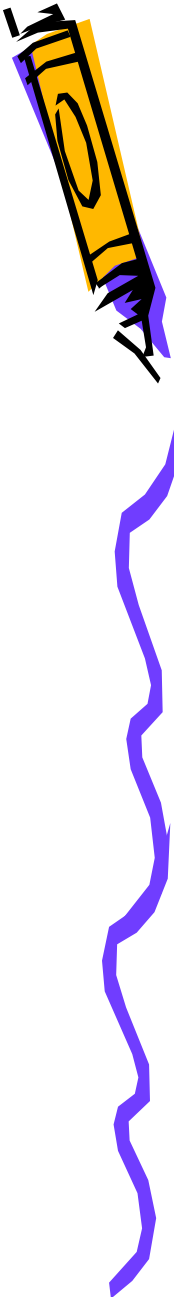
```
//string s2 = new string();
```

 !!!Тоест, не е допустимо извикване на конструктор без параметри, а също извикване на конструктор, с аргумент - символна константа.

- Не е допустимо също следното:

```
string a="12345";    //a[1] = 4;
```

- Причината е, че класът string е неизменяем клас



Стрингове - Обявяване

- В C# съществува понятието неизменяем (immutable) клас. Такъв неизменяем клас е именно клас string за C# (съответно клас System.String за Net.framework). Нито един от методите на този клас не изменя стойността на обект от тип стринг.
- Разбира се някои методи създават усещането, че връщат като резултат - нова стойност на обекта от тип стринг (подаден като параметър от извикващия метод), но на практика те връщат в качеството на резултат - нов стринг (не променят по никакав начин стринга на извикващия метод).
- Това е причината поради която например, ако извикаме метод ToUpper() за един string, той не променя оригиналния стринг, а създава и връща нов обект от тип string, който е upper case представяне на оригиналния обект.



Стрингове - Обявяване

- Тази невъзможност за изменение важи не само за методите. Аналогично, ако се работи със стринг като масив, е разрешено само четене на отделните му символи, но не и тяхната промяна.
- Ето защо операторът за присвояване, който се опитва да измени символ от стринг, е недопустим:

```
string a="12345";    //a[1] = 4;
```



2. Операции над стрингове

- присвояване (=);

string a="12345"; string b=a;

- проверка за еквивалентност (==) и (!=):

Console.WriteLine(a==b);//True

- конкатенация на стрингове (+): **string c=a+b;**

- Независимо, че има и метод Concat, тоест:

String.Concat(mFirstName," ", mLastName);

- извличане на индекс ([]):

char k=a[1]; char y="test"[2];//s

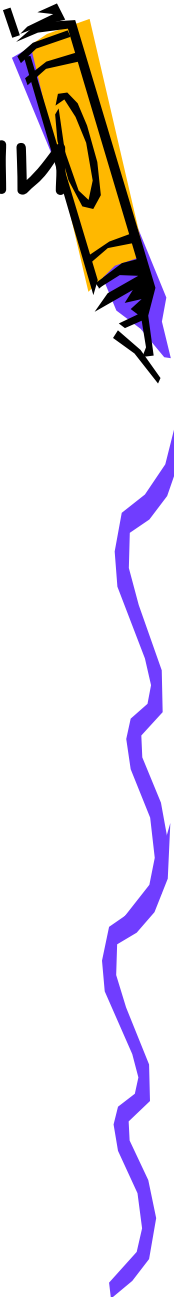
- !!! За разлика от другите референтни типове, при проверка за еквивалентност (**Equals**), се проверява стойността, а не референцията.



3. Символни константи

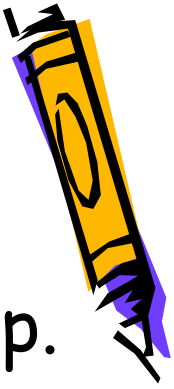
В C# съществуват два вида символни константи:

- **обикновени** – символна последователност в кавички;
- **@-константи**, задават се като обикновени символни константи, предшествани от знак @.



Обикновени константи

- В тези константи някои символи се интерпретират по особен начин, например символ за табулация, преход на нов ред и др.
- За всички такива цели се използва комбинация от символи, започващи с "\" – обратна наклонена черта:
- "\\n" – преход на нов ред
- "\\t" – табулация
- "\\\" – \"
- "\\\"" – кавички
- "\\xNNNN" – задава символ, определен от шестнадесетичния код NNNN.



- При обикновените константи често възниква неудобство например, при задаване на константа, определяща път към файл - налага се всяка \ да се удвоява.
- Това е една от причините за поява на *@-константи*. В @-константите всички символи се третират така както са написани. Единствен проблем при тях е - как да зададем символ кавички? Решението - удвояване на символа.



Обикновени и @ константи - пример

```
string s1 = "\\x50"; //P
```

```
string s2 = @"\\x50""; //\\x50"
```

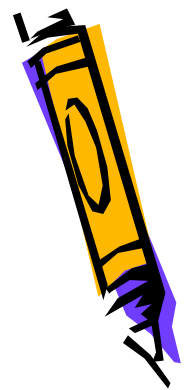
```
s1 = "c:\\c#book\\ch5\\chapter5.doc";  
//c:\\c#book\\ch5\\chapter5.doc
```

```
s2 = @"c:\\c#book\\ch5\\chapter5.doc";  
//c:\\c#book\\ch5\\chapter5.doc
```

```
Console.WriteLine(s1 == s2); //True
```

```
s1= "\\\"A\\\""; //\"A"
```

```
s2=@"\"A\""; //\"A"
```



Статични методи и свойства на клас String

Описание

Метод

Empty

Връща празен стринг. Свойство със статус read only.
`string t = string.Empty;`

```
if(t==string.Empty)
    Console.WriteLine("ura");//ura
```

Compare

За сравнение на 2 низа. Пример:

```
string m="Hello Students";
Console.WriteLine(string.Compare(m, "Hello"));//1
// 0 или -1
```

Concat

Конкатенация на стрингове:

```
string str1 = "abc";
string str2 = "xyz";
string d =string.Concat(str1,str2);
Console.WriteLine("d = '{0}'", d);//'abcxyz'
```

Copy

Създаване на копие на стринг:

```
string str1 = "abc";
string str2 = "xyz";
str2 = String.Copy(str1) //"abc"
```

Format

Изпълнява форматиране в съответствие със зададени спецификации на формата. Виж отдолу: извън таблицата

Join

Конкатенация на масив от стрингове в един стринг. При конкатенации между елементите на масива се поставят разделители.

```
string[] k = { "Elena", "Kirilova", "Ivanova" };  
            Console.WriteLine(string.Join(" ", k));  
// Elena Kirilova Ivanova
```

Split

Стрингът се разделя на части, индикирани от сепаратор, например „,“. Пример:

```
string txt = "printing, easier searches, security.";  
string[] SimpleSentences; // масив от стрингове  
SimpleSentences = txt.Split(','); // разделител  
// SimpleSentences = txt.Split(null); разделител  
шпация
```

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

Метод Format

- Метод Format в нашите примери се среща многократно. Всеки път, когато се изпълнява извеждане на резултати на конзолата неявно се извиква метод Format.
- Да разгледаме печат на конзолата:

```
Console.WriteLine("s1={0}, s2={1}", s1,s2);
```

Общият синтаксис, специфициращ формат, е:

```
{N [,M [:<кодове_за_форматиране>]]}
```

- На практика във фигурните скоби на показания пример се съдържа формата, който в дадения случай е съвсем прост, съдържа 1 параметър N - само индекса, тоест номера на обекта от списъка, който е задължителен и започва от 0.
- Вторият параметър M, определя минималната ширина на полето за извеждане.



Метод Format

```
Console.WriteLine("s1={0,20}, s2={1,-20}", s1,s2);
```

- //20 - подравнение в дясно, ширина 20 символа
- //-20 - подравнение в ляво
- Третият параметър задава кода на форматиране. Например:
- код C (Currency) - параметъра трябва да се форматира като валута с отчет на националните особености, тоест зависи от текущата култура.
- Код P (Percent) - форматиране в проценти.
- Код за дати: d - MM/dd/yyyy, D - Dd MMMM yyyy, f - Dd MMMM yyyy HH:mm, F - Dd MMMM yyyy HH:mm:ss, t - къс формат за време, T - дълъг формат за време, hh - код за време.


```
Console.WriteLine("{0:hh}", DateTime.Now);
```

//05, т.е тоест 5 часа

```
Console.WriteLine(DateTime.Now);
```

//11/6/2007 05:50:07 PM





```
string value1 = "TU-Varna";  
double value2 = 10000.00;  
DateTime value3 = new DateTime(2015,11,1);  
string result =
```

```
string.Format("{0}: {1:0} students -  
{2:yyyy} year",value1,value2, value3);
```

```
Console.WriteLine(result);
```

```
//TU-Varna: 10000 students - 2015 year
```

```
//Вторият аргумент, т.е {1:0} е  
форматиран като цяло число
```



```
double ratio = 0.73;
```

```
string result2 = string.Format("string =  
{0:0.0%}", ratio);
```

```
Console.WriteLine(result2); //73%
```

// {0:0.0%} – първият аргумент е
форматиран като реално число, с една
позиция след точката и %

Може и така:

```
string result2 = string.Format("string =  
{0:P}", ratio); //73.00%
```



// В шестнадцатичен формат.

```
int number = 10995;
```

```
Console.WriteLine("{0:x}", number); //2af3
```

```
Console.WriteLine("{0:x8}", number); //00002af3
```

```
Console.WriteLine("{0:X}", number); //2AF3
```

```
Console.WriteLine("{0:X8}", number); //00002AF3
```

```
int money = 1000;
```

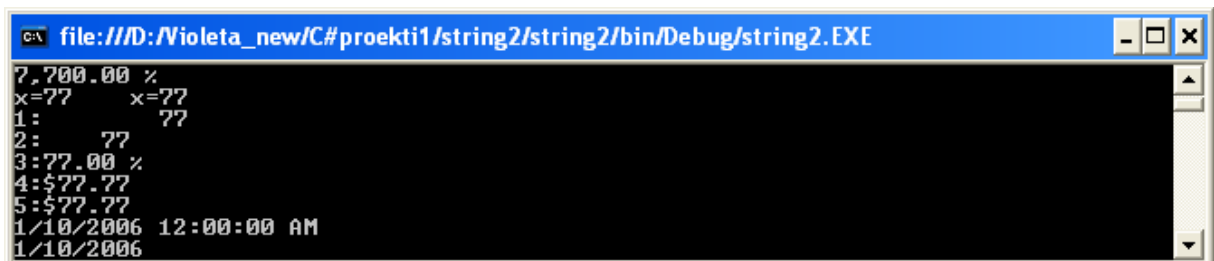
```
string result3 = string.Format("{0:0.00$}", money);
```

```
Console.WriteLine(result3); //1000.00$
```

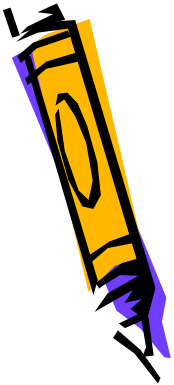


Пример:

```
public static void TestFormat()
{ //метод Format
    int x = 77;
    string t = x.ToString("p");//7,700.00 %
    Console.WriteLine(t);
    string s = string.Format("x={0}", x);//x=77    x=77
    Console.WriteLine(s + "\tx={0}", x);
    s = string.Format("1:{0,10}", x);          //1:      10
    Console.WriteLine(s);
    s = string.Format("2:{0,6:#####}", x); //2:      77
    Console.WriteLine(s);
    s = string.Format("3:{0:P} ", 0.77);      //3:77.00%
    Console.WriteLine(s);
    s = string.Format("4:{0,4:C} ", 77.77);   //4:$77.77
    Console.WriteLine(s);
    //Национални особености
    System.Globalization.CultureInfo ci =
        new System.Globalization.CultureInfo("en-US");
    s = string.Format(ci, "5:{0,4:C} ", 77.77); //5:77.77
    Console.WriteLine(s);
    DateTime d = new DateTime(2006,1,10);
    Console.WriteLine(d);          //1/10/2006 12:00:00 AM
    string f = d.ToString("d");//1/10/2006
    Console.WriteLine(f);          //1/10/2006
} //TestFormat
```



```
C:\ file:///D:/Violeta_new/C#proekti1/string2/string2/bin/Debug/string2.EXE
7.700.00 %
x=77    x=77
1:      10
2:      77
3:77.00 %
4:$77.77
5:77.77
1/10/2006 12:00:00 AM
1/10/2006
```

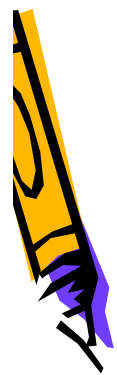


Динамични методи на клас String

- Клас String наследява методите на клас Object.
- Клас String наследява, и следователно реализира 4 интерфейса: IComparable, ICloneable, IConvertible, IEnumerable.
- Сега ще разгледаме най-характерните методи за работа със стрингове, позволяващи вмъкване, отстраняване, замяна, търсене на подстринг в стринг.
- Трябва да се помни, че клас string е неизменчив. Ето защо Replace, Insert и другите методи представляват сами по себе си функции, връщащи нов стринг в качеството на резултат, като не променят по никакав начин стринга на извикващия метод.



Метод	Описание
Insert	Вмъква подстринг, в зададена позиция: <pre>string d = "Bulgaria"; Console.WriteLine(d.Insert(1, "123")); //B123ulgaria</pre>
Remove	Премахва подстринг, при зададена позиция: <pre>string d = "Bulgaria"; Console.WriteLine(d.Remove(1, 3)); //Baria</pre>
Replace	Заменя символ с друг: <pre>string d = "Bulgaria"; Console.WriteLine(d.Replace('u', 'a')); //Balgaria</pre>
Substring	Отделя подстринг от зададена позиция: <pre>string d = "Bulgaria"; Console.WriteLine(d.Substring(0, 3)); //Bul</pre>
ToLower, ToUpper	<pre>string d = "Bulgaria"; Console.WriteLine(d.ToLower()); //bulgaria Console.WriteLine(d.ToUpper()); //BULGARIA</pre>
IndexOf, IndexOfAny, LastIndexOf, LastIndexOfAny	Определя индекса на първото, последното или някакво срещане на даден подстринг или символ; <pre>string d = "Bulgaria "; Console.WriteLine(d.IndexOf('a')); //4 Console.WriteLine(d.LastIndexOf('a')); //7 Console.WriteLine(d.IndexOfAny(new char[] { 'a', 'l' })); //2 Console.WriteLine(d.LastIndexOfAny(new char[] { 'a', 'l' })); //7</pre>



Parse метод

```
string my = "12345";  
int a = int.Parse(my);  
string my2 = "12345,345";  
//a = int.Parse(my2); !!!Error  
a = int.Parse  
(my2, System.Globalization.NumberStyles.  
AllowThousands);
```

StartsWith,
EndsWith

връща true или false, в зависимост от това, започва или завършва стрингът със зададен подстринг

```
string d = "Bulgaria    ";  
Console.WriteLine(d.StartsWith("Bul"));  
//True
```

PadLeft,
PadRight

Подравнява в ляво, или в дясно

```
string d = "Bulgaria";  
Console.WriteLine(d.PadLeft(10, '-'));  
//--Bulgaria
```

Trim,
TrimStart,
TrimEnd

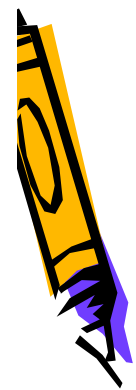
Обратно на Pad. Премахва празни позиции:

```
string d = "          Bulgaria    ";  
Console.WriteLine(d.Trim());
```

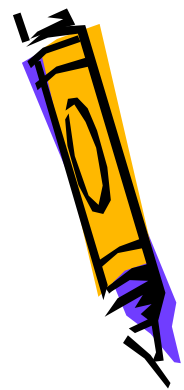
ToCharArray

Преобразуване на стринга в масив от символи:

```
string d = "Bulgaria    ";  
char [] ss=d.ToCharArray();
```

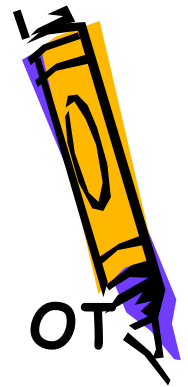


Свойства	Описание
Length	<p>връща броя на символите в даден стринг</p> <pre>string my = "12345"; Console.WriteLine(my.Length);/5</pre>
Char[int]	<p>Връща символа на зададена позиция</p> <pre>string my = "12345"; Console.WriteLine(my[3]);//4</pre>
String. Compare(my1,my2)	<p>Сравнява 2 низа: 0 – за равенство, 1 (my1>my2) и -1 (my1<my2)</p> <pre>string my1 = "12345"; string my2 = "123"; Console.WriteLine(String.Compare(my1,my2));/ /1</pre>
String. CompareOrdinal(my1,my2))	<p>Сравнява 2 обекта от тип стринг, чрез изчисляване на числовите стойности на съответните Char обекти на всеки стринг; Ако в резултат се получи положително число, то my1>my2; Ако се получи отрицателно число, то my1<my2. Ако се получи 0, то my1=my2. Методът не отчита културата.</p> <pre>string my1 = "12345"; string my2 = "123"; Console.WriteLine(String.CompareOrdinal(my1, my2));//52</pre>
String. Equals(my1,my2)	<p>Ако е обратно – резултат (-52).</p> <pre>string my1 = "123"; string my2 = "12345"; Console.WriteLine(String.Equals(my1,my2));// False</pre>
my1.Equals(my2)	<pre>string my1 = "123"; string my2 = "12345"; Console.WriteLine(my1.Equals(my2));//False</pre>
if (my1==my2)... if (my1!=my2)...	<pre>string my1 = "123"; string my2 = "12345"; Console.WriteLine(my1==my2);//False</pre>



Клас `StringBuilder`

- Клас `System.String` не разрешава изменение на съществуващите обекти от този тип.
- Символният клас `StringBuilder` позволява да се компенсира този недостатък. Това е изменяем клас и може да се намери в именованото пространство `System.Text` (`System.Text.StringBuilder`).



Конструктори на StringBuilder

- Обекти на този клас се създават чрез явно извикване на конструктора на класа (без параметри):

```
using System.Text; //Много важно е!!!
```

```
... StringBuilder s3 = new StringBuilder();
```

- Създаване на обект от тип StringBuilder, и инициализация:

```
StringBuilder s1 =new StringBuilder("ABC"),  
s2 =new StringBuilder("CDE");
```

- Капацитет: 16, 32, 64, 128 ... Int32.MaxValue

```
Console.WriteLine(s2.Capacity); //16 символа
```

Макс.капацитет=Int32.MaxValue, т.е 2147483647.



Конструктори на `StringBuilder`

- `StringBuilder` има и конструктор, на който може да се предадат две групи параметри:
 - Първата група - задава се стринга или подстринга, с чието значение ще бъде инициализиран създавания обект от клас *StringBuilder*.
 - Втората група от параметри позволява да се зададе капацитет на обекта - обем памет, за дадения екземпляр на клас *StringBuilder*. Ако сме задали по-малко от дължината на стринга, то автоматично се отпуска допълнително капацитет.

`public StringBuilder (string str, int cap)`. Параметър `str` задава стринга за инициализация, `cap` - *капацитет на обекта*;

`StringBuilder M = new StringBuilder("Hello", 5);`

//специфицира се максимален брой 5 символа.

`Console.WriteLine(M.Capacity);`*//5 символа*

`StringBuilder M2 = new StringBuilder("Hello", 2);`

//специфицира се максимален брой 5 символа.

//но автоматично се увеличава капацитета до 5 символа.

`Console.WriteLine(M2.Capacity);`*//5 символа*



Конструктори на StringBuilder

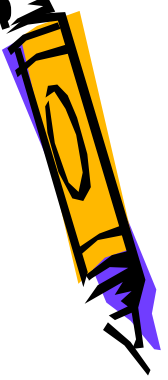
- StringBuilder има и конструктор, на който може да се предадат 4 групи параметри:
 - str - стринг,
 - start - начална позиция,
 - len - дължина на стринга,
 - cap - капацитет на обекта.

```
public StringBuilder (string str, int start, int  
len, int cap).
```

```
StringBuilder M22 = new  
StringBuilder("Hello", 1, 3, 25);
```

```
Console.WriteLine(M22.Capacity); //25
```

```
//M22="ell"
```



Операции над StringBuilder

Използват се същите операции, които се използват и за класа String:

- присвояване (=);
- проверка за еквивалентност (==) и (!=);
- извличане на индекс ([]).

Но операция конкатенация (+) не се използва за клас StringBuilder, нейната роля се изграе от метод Append.

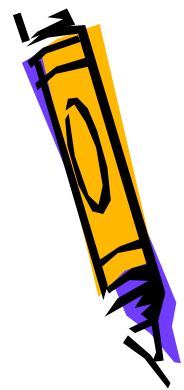
```
s2 = new StringBuilder("CDE");
```

```
StringBuilder s3 = new StringBuilder();
```

```
s3 = s1.Append(s2); //!!!не s3 = s1+s2;
```

- Със стринг от този клас може да се работи като с масив , но тук вече можем не само да четем отделен символ, но и да го променяме за разлика от клас String, тоест:

```
StringBuilder s = new StringBuilder("Zenon"); s[0]='L';
```



ОСНОВНИ МЕТОДИ НА `StringBuilder`

- Тук има значително по-малко методи отколкото при *String*:

`public StringBuilder Append (<обект>).`

- Чрез метода **Append** се конкатенират стрингове.

```
StringBuilder a = new StringBuilder("11111");  
string b = "22222";  
Console.WriteLine(a.Append(b)); //1111122222  
StringBuilder d = new StringBuilder("333");  
Console.WriteLine(a.Append(d)); //1111122222333
```

`public StringBuilder Insert (int location, <обект>).`

- Чрез метода **Insert** се вмъква стринг, в позицията, указана чрез. Метод `Append` е частен случай на `Insert`:

```
StringBuilder a = new StringBuilder("The formula is ");  
a.Insert(0, "2*2=5 - ");  
Console.WriteLine(a);           //2*2=5 - The formula is
```



- `public StringBuilder Remove (int start, int len).` Премахва подстринг с дължина `len`, започвайки от позиция `start`;

```
StringBuilder a = new StringBuilder("abcde");
```

```
Console.WriteLine(a.Remove(2,2)); //abe
```

- `public StringBuilder Replace (string str1, string str2).` Заменя се стринг `str1` със стринг `str2`;

```
StringBuilder a = new StringBuilder("abcde");
```

```
Console.WriteLine(a.Replace('a', 'A')); //Abcde
```

```
Console.WriteLine(a.Replace("ab", "AA")); //AAcde
```

- `public StringBuilder AppendFormat (string format, Object arg0).` Долепя се форматиран стринг към края на `StringBuilder` обект.

```
StringBuilder sb = new StringBuilder();
```

```
int MyInt = 25;
```

```
StringBuilder MyStringBuilder = new StringBuilder("Your total is ");
```

```
MyStringBuilder.AppendFormat("{0:C} ", MyInt);
```

```
Console.WriteLine(MyStringBuilder); // Your total is $25.00
```

//Долепя `MyInt`, форматиран като валута, след обект

`MyStringBuilder`

