

Laborator 7

Șiruri de caractere

Pentru reprezentarea șirurilor de caractere, în limbajul C#, tipul de date utilizat este clasa **System.String** (sau aliasul **string**). Se definesc două tipuri de șiruri:

- regulate
- de tip „**Verbatim**”

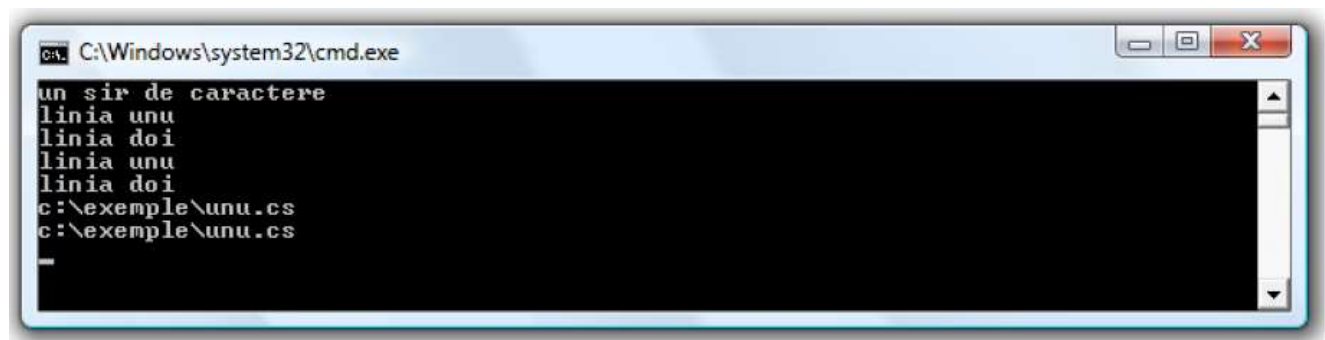
Tipul regulat conține între ghilimele zero sau mai multe caractere, inclusiv secvențe escape.

```
string a = "Acesta este un sir de caractere";  
string b = "";  
string nume = "Gigel";
```

Limbajul C# introduce, pe lângă șirurile regulate și cele de tip **verbatim**. În cazul în care folosim multe secvențe escape, putem utiliza șirurile **verbatim**. Aceste șiruri se folosesc în special în cazul în care dorim să facem referiri la fișiere, la prelucrarea lor, la regiștri. Un astfel de șir începe cu simbolul „@” înaintea ghilimelelor de început.

Exemplu:

```
using System;  
  
namespace SiruriDeCaractere  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            string a = "un sir de caractere";  
            string b = "linia unu \nlinia doi";  
            string c = @"linia unu  
linia doi";  
            string d = "c:\\exemple\\unu.cs";  
            string e = @"c:\exemple\unu.cs";  
            Console.WriteLine(a);  
            Console.WriteLine(b);  
            Console.WriteLine(c);  
            Console.WriteLine(d);  
            Console.WriteLine(e);  
            Console.ReadLine();  
        }  
    }  
}
```



```
C:\Windows\system32\cmd.exe  
un sir de caractere  
linia unu  
linia doi  
linia unu  
linia doi  
c:\exemple\unu.cs  
c:\exemple\unu.cs  
-
```

Secvențele escape permit reprezentarea caracterelor care nu au reprezentare grafică precum și reprezentarea unor caractere speciale: backslash, caracterul apostrof, etc.

Secvență escape	Efect
\'	apostrof
\"	ghilimele
\\	backslash
\0	null
\a	alarmă
\b	backspace
\f	form feed – pagină nouă
\n	new line – linie nouă
\r	carriage return – început de rând

\t	horizontal tab – tab orizontal
\u	caracter unicode
\v	vertical tab – tab vertical
\x	caracter hexazecimal

Concatenarea șirurilor de caractere

Pentru a concatena șiruri de caractere folosim operatorul “+”

Exemplu:

```
string a = "Invat " + "limbajul " + "C#";
//a este "Invat limbajul C#"
```

Compararea șirurilor de caractere

Pentru a compara două șiruri de caractere vom utiliza operatorii “==” și “!=”.

Definiție: două șiruri se consideră **egale** dacă sunt amândouă **null**, sau dacă amândouă au aceeași lungime și pe fiecare poziție au caractere respectiv identice. În caz contrar șirurile se consideră **diferite**.

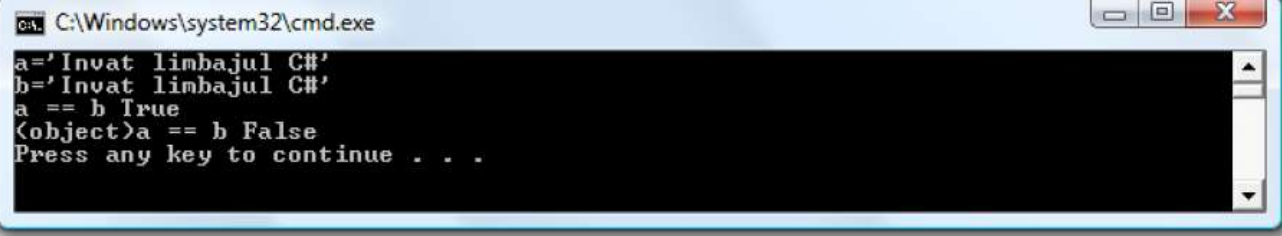
Exemplul 50: Exemplul următor demonstrează că operatorul “==” este definit pentru a compara valoarea obiectelor **string** și nu referința lor

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Exemplul_50
{
    class Program
    {
        static void Main(string[] args)
        {
            string a = "Invat limbajul C#";
            string b = "Invat " + "limbajul ";
            b += "C#";
            Console.WriteLine("a='{0}'", a);
            Console.WriteLine("b='{0}'", b);
            Console.WriteLine("a == b {0}", a == b);
            Console.WriteLine("(object)a == b {0}", (object)a == b);
        }
    }
}

```



```

C:\Windows\system32\cmd.exe
a='Invat limbajul C#'
b='Invat limbajul C#'
a == b True
(object)a == b False
Press any key to continue . . .

```

Funcții importante pentru șiruri

Clasa **String** pune la dispoziția utilizatorului mai multe metode și proprietăți care permit prelucrarea șirurilor de caractere. Dintre acestea amintim:

- metode de comparare:

- **Compare**
- **CompareOrdinal**
- **CompareTo**

- metode pentru căutare:

- **EndsWith**
- **StartsWith**
- **IndexOf**
- **LastIndexOf**

- metode care permit modificarea șirului curent prin obținerea unui nou șir:

- **Concat**
- **CopyTo**
- **Insert**
- **Join**
- **PadLeft**
- **PadRight**

- Remove
- Replace
- Split
- Substring
- ToLower
- ToUpper
- Trim
- TrimEnd
- TrimStart

Proprietatea Length am folosit-o pe parcursul acestei lucrări și, după cum știm returnează un întreg care reprezintă lungimea (numărul de caractere) șirului.

Tabelul de mai jos prezintă câteva dintre funcțiile (metodele) clasei String

Funcția (metodă a clasei Strig)	Descrierea
<code>string Concat(string u, string v)</code>	returnează un nou șir obținut prin concatenarea șirurilor <code>u</code> și <code>v</code>
<code>int IndexOf(char c)</code>	returnează indicele primei apariții a caracterului <code>c</code> în șir
<code>int IndexOf(string s)</code>	returnează indicele primei apariții a subșirului <code>s</code>
<code>string Insert(int a, string s)</code>	returnează un nou șir obținut din cel inițial prin inserarea în șirul inițial, începând cu poziția <code>a</code> , a șirului <code>s</code>
<code>string Remove(int a, int b)</code>	returnează un nou șir obținut din cel inițial prin eliminarea, începând cu poziția <code>a</code> , pe o lungime de <code>b</code> caractere
<code>string Replace(string u, string v)</code>	returnează un nou șir obținut din cel inițial prin înlocuirea subșirului <code>u</code> cu șirul <code>v</code>
<code>string Split(char[] c)</code>	împarte un șir în funcție de delimitatorii <code>c</code>
<code>string Substring(int index)</code>	returnează un nou șir care este un subșir al șirului inițial începând cu indicele <code>index</code>
<code>string Substring(int a, int b)</code>	returnează un nou șir care este un subșir al șirului inițial, începând de pe poziția <code>a</code> , pe lungimea <code>b</code> caractere
<code>string ToLower()</code>	returnează un nou șir obținut din cel inițial prin convertirea tuturor caracterelor la minuscule
<code>string ToUpper()</code>	returnează un nou șir obținut din cel inițial prin convertirea tuturor caracterelor la majuscule
<code>string Trim()</code>	returnează un nou șir obținut din cel inițial prin ștergerea spațiilor goale de la începutul și sfârșitul șirului inițial
<code>string TrimEnd()</code>	returnează un nou șir obținut din cel inițial prin ștergerea spațiilor goale de la sfârșitul șirului inițial
<code>string TrimStart()</code>	returnează un nou șir obținut din cel inițial prin ștergerea spațiilor goale de la începutul șirului inițial

Exemplul 51: Exemplificăm aplicarea funcțiilor de mai sus:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Exemplul_51
{
    class Program
    {
        static void Main(string[] args)
        {
            string a = "Invat limbajul ";
            string b = "C#";
            string c;
            Console.WriteLine("a = '{0}'", a);
            Console.WriteLine("b = '{0}'", b);

```

```

            c = string.Concat(a, b);
            Console.WriteLine("string.Concat(a, b) = \"{0}\"", c);
            Console.WriteLine("a.IndexOf(\"v\") = {0}",
Convert.ToString(a.IndexOf("v")));
            Console.WriteLine("a.IndexOf(\"mba\") = {0}",
Convert.ToString(a.IndexOf("mba")));
            Console.WriteLine("a.Insert(6, \"de zor \") = {0}", a.Insert(6,
"de zor "));
            Console.WriteLine("a.Remove(5, 7) = {0}", a.Remove(5, 7));
            Console.WriteLine("a.Replace(\"limbajul \", \"la informatica.\")
= {0}", a.Replace("limbajul ", "la informatica."));
            Console.WriteLine("a.Substring(6) = {0}", a.Substring(6));
            Console.WriteLine("a.Substring(10, 3) = {0}", a.Substring(10,
3));
            Console.WriteLine("a.ToLower() = {0}", a.ToLower());
            Console.WriteLine("a.ToUpper() = {0}", a.ToUpper());
            string d = "    Ana are mere.    ";
            Console.WriteLine("d = {0}", d);
            Console.WriteLine("d.Trim() = {0}", d.Trim());
            Console.WriteLine("d.TrimStart() = {0}", d.TrimStart());
        }
    }
}

```

```

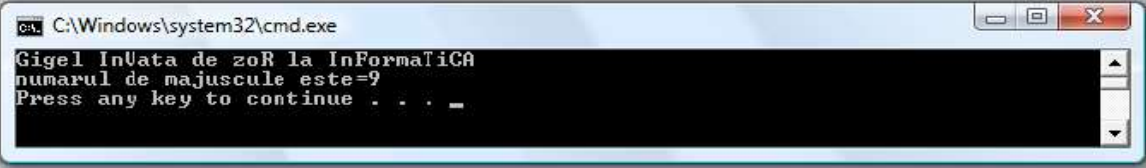
C:\Windows\system32\cmd.exe
a = 'Invat limbajul '
b = 'C#'
string.Concat(a, b) = "Invat limbajul C#"
a.IndexOf("v") = 2
a.IndexOf("mba") = 8
a.Insert(6, "de zor ") = Invat de zor limbajul
a.Remove(5, 7) = Invatul
a.Replace("limbajul ", "la informatica.") = Invat la informatica.
a.Substring(6) = limbajul
a.Substring(10, 3) = aju
a.ToLower() = invat limbajul
a.ToUpper() = INVAT LIMBAJUL
d =     Ana are mere.
d.Trim() = Ana are nere.
d.TrimStart() = Ana are mere.
Press any key to continue . . .

```

Exemplul 52: Programul următor contorizează majusculele dintr-un text.

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Exemplul_52
{
    class Majuscule
    {
        static void Main()
        {
            int i, nrm = 0;
            string text = System.Console.ReadLine();
```


```
            for (i = 0; i < text.Length; i++)
            { if (text[i] >= 'A' && text[i] <= 'Z') nrm++; }
            System.Console.WriteLine("numarul de majuscule este=" + nrm);
        }
    }
}
```



```
C:\Windows\system32\cmd.exe
Gigel InUata de zoR la InFormaTiCa
numarul de majuscule este=9
Press any key to continue . . .
```

Exemplul 53: Să se verifice dacă cuvintele s1 și s2 citite de la tastatură au aceeași *textură*. Două cuvinte au aceeași textură dacă au aceeași lungime și toate caracterele corespondente au același tip. Nu se face distincție între litere mari, litere mici. Ex : *acum* și *elev* au aceeași textură (vocală consoană vocală consoană)

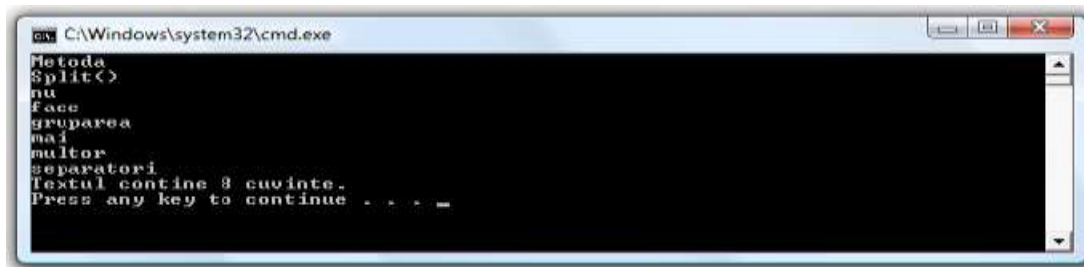
```
using System;
namespace Exemplul_53
{
    class Program
    {
        private static bool strchr(string p, char p_2)
        {
            for (int i = 0; i < p.Length; i++)
                if (p[i] == p_2) return true;
            return false;
        }
        static void Main()
        {
            String s1 = Console.ReadLine();
            String s2 = Console.ReadLine();
            String v = string.Copy("aeiouAEIOU");
            bool textura = true;
            int i;
            if (s1.Length != s2.Length) textura = false;
            else
            {
                for (i = 0; i < s1.Length; i++)
                    if (strchr(v, s1[i]) && !strchr(v, s2[i]) || !strchr(v,
s1[i]) && strchr(v, s2[i]))
                        textura = false;
            }
            if (textura) Console.WriteLine("Au aceeasi textura");
            else Console.WriteLine("Nu au aceeasi textura");
        }
    }
}
```



```
C:\Windows\system32\cmd.exe
acum
EleU
Au aceeasi textura
Press any key to continue . . .
```


Exemplul 54: Folosind metoda Split, să se numere cuvintele unui text știind că acestea sunt separate printr-un singur separator din mulțimea { ' ', ',', ';' }.

```
using System;
namespace Exemplul_54
{
    class Program
    {
        static void Main(string[] args)
        {
            String s = "Metoda Split() nu face gruparea mai multor separatori";
            char[] x = { ' ', ',', ';' };
            String[] cuvant = s.Split(x);
            int nrcuv = 0;
            for (int i = 0; i < cuvant.Length; i++)
            {
                Console.WriteLine(cuvant[i]);
                nrcuv++;
            }
            Console.WriteLine("Textul contine {0} cuvinte.", nrcuv);
        }
    }
}
```

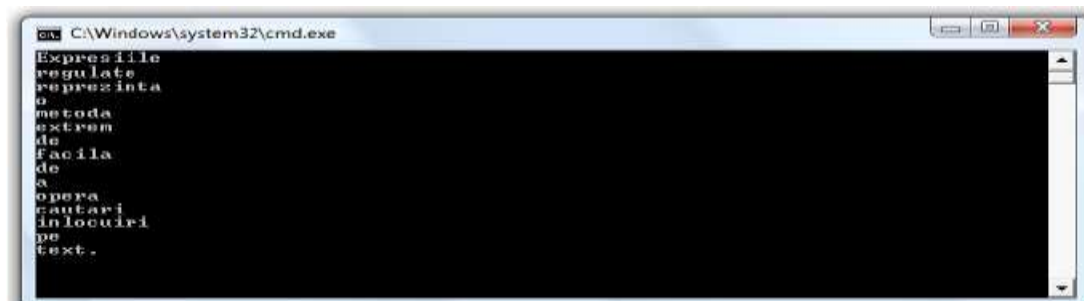


```
C:\Windows\system32\cmd.exe
Metoda
Split()
nu
face
gruparea
mai
multor
separatori
Textul contine 9 cuvinte.
Press any key to continue . . . _
```

Metoda Split() nu face gruparea mai multor separatori, lucru care ar fi de dorit. Pentru aceasta se folosesc **expresii regulate**. Expresiile regulate reprezintă o metodă extrem de utilă pentru a opera căutări/inlocuiri pe text.

Exemplul 55:

```
using System;
using System.Text.RegularExpressions;
namespace Exemplul_55
{
    class Program
    {
        static void Main(string[] args)
        {
            String s = "Expresiile regulate , reprezinta o metoda extrem de facila de a opera cautari, inlocuiri pe text. ";
            //separator: virgula, spatiu sau punct si virgula
            //unul sau mai multe, orice combinatie
            Regex regex = new Regex("[, ;]+");
            String[] cuvant = regex.Split(s);
            for (int i = 0; i < cuvant.Length; i++)
            {
                Console.WriteLine(cuvant[i]);
            }
            Console.ReadKey();
        }
    }
}
```



```
C:\Windows\system32\cmd.exe
Expresiile
regulate
reprezinta
o
metoda
extrem
de
facila
de
a
opera
cautari
inlocuiri
pe
text.

```