

V-191

Limbaajul C# pentru începători
Noțiuni de bază

Liviu Negrescu, Lavinia Negrescu

1. CONSTRUCȚIILE DE BAZĂ ALE LIMBAJULUI C#

- 1.1. Nume
- 1.2. Tipuri predefinite
- 1.3. Literali
 - 1.3.1. Literali numerici
 - 1.3.1.1. Literali întregi
 - 1.3.1.2. Literali numerici oarecare
 - 1.3.2. Literali caracter
 - 1.3.3. Literali șir (șir de caractere)
 - 1.3.4. Literali logici sau Booleeni
- 1.4. Variabile
- 1.5. Tablou
 - 1.5.1. Inițializarea tablourilor
 - 1.5.2. Referirea la tablouri
- 1.6. Programarea obiectuală
 - 1.6.1. Date membru
 - 1.6.2. Metode
 - 1.6.3. Spațiu de nume
- 1.7. Comentarii

2. REALIZAREA PROGRAMELOR EXECUTABILE ȘI LANSAREA LOR ÎN EXECUȚIE

- 2.1. Program care afișează un șir de caractere
- 2.2. Noțiuni introductive despre calculatoarele electronice
 - 2.2.1. Memorii
 - 2.2.2. Folder
 - 2.2.3. Fișiere
 - 2.2.4. Sistem de operare
 - 2.2.4.1. Mouse
 - 2.2.4.2. Lansarea în execuție a programelor
 - 2.2.4.3. Sistemul de operare MS-DOS
 - 2.2.5. Ferestre
 - 2.2.5.1. Meniuri
 - 2.2.5.2. Alte componente ale ferestrelor
- 2.3. Lansarea în execuție a mediului pe programare Microsoft Visual Studio.NET
- 2.4. Editarea programelor în limbaajul C# utilizând aplicația Notepad și păstrarea lor pe discurile magnetice

- 2.4.1. Crearea directorului de lucru
- 2.4.2. Editarea cu ajutorul aplicației Notepad
- 2.5. Crearea programelor executabile cu ajutorul compilatorului cu linie de comandă și lansarea lor în execuție
- Exerciții
- 2.6. Optimizări în scrierea programelor sursă
- Exerciții

3. ȘIRURI DE CARACTERE

- 3.1. Din nou despre șiruri de caractere
- Exerciții
- 3.2. Parametrii liniei de comandă
- Exerciții

4. CREAREA ȘI LANSAREA ÎN EXECUȚIE A PROGRAMELOR C# SUB MEDIUL DE PROGRAMARE MICROSOFT VISUAL STUDIO.NET

- 4.1. Lansarea mediului integrat de dezvoltare a programelor C#
- 4.2. Crearea proiectelor
 - 4.2.1. Fereastra mediului
- 4.3. Editarea aplicațiilor, compilarea și lansarea lor în execuție sub mediul integrat de dezvoltare
- 4.4. Crearea imaginilor executabile sub mediul integrat de dezvoltare a programelor C#
- 4.5. Lansarea în execuție a aplicațiilor, cu parametri în linia de comandă, sub mediul integrat de dezvoltare
- Exerciții

5. INTRĂRI-IEȘIRI STANDARD

- 5.1. Ieșiri standard
- Exerciții
- 5.2. Intrări standard
- Exerciții

3. ȘIRURI DE CARACTERE

În acest capitol revenim la șirurile de caractere pentru a preciza unele aspecte privind definirea și utilizarea acestora. În prima parte a capitolului indicăm modul în care se pot defini și gestiona șirurile de caractere. Astfel, un șir de caractere se poate reprezenta printr-un literal șir (vezi paragraful 1.3.3.), dar mai există și alte posibilități cum ar fi de exemplu definirea unui tablou ale cărui elemente sunt de tip *char* și care au ca valori caracterele din compunerea șirului respectiv sau o altă posibilitate este de a instanția obiecte ale clasei *string* în care putem păstra șiruri de caractere.

În partea a doua a acestui capitol se arată că programele C# pot fi lansate în execuție utilizând parametri care sunt interpretați ca obiecte ale clasei *string*.

3.1. Din nou despre șiruri de caractere

Prin *șir de caractere* înțelegem o succesiune de zero sau mai multe caractere oarecare incluse între ghilimele.

În paragraful 1.3.3. s-au introdus șirurile de caractere. Cu această ocazie s-a indicat faptul că într-un șir de caractere putem utiliza *secvențe escape* (secvențe de forma `\ ...`) pentru caractere *neimprimabile* (amintim că sunt *imprimabile* caracterele care au codul ASCII în intervalul [32,126]). O excepție de la această regulă o reprezintă caracterele *ghilimele* și caracterul *backslash*. Deși aceste caractere sunt imprimabile ele pot fi componente ale unui șir de caractere numai dacă le reprezentăm prin *secvențele lor escape*. Aceasta este normal să fie așa deoarece ghilimele se utilizează pentru a defini caracterele ce aparțin unui șir de caractere. Deci, pentru a include caracterul ghilimele într-un șir de caractere, vom utiliza *secvența escape*:

\"

Caracterul *backslash* se poate reprezenta într-un șir de caractere dublându-l:

\\

Șirurile de caractere se pot păstra în *tablouri* ale căror elemente sunt de tip *char*.

Știm că un tablou unidimensional se declară astfel:

tip [] nume;

unde *nume* este numele tabloului, iar *tip* este tipul comun elementelor tabloului. În cazul tablourilor destinate pentru a păstra șiruri de caractere, *tip* va fi cuvântul rezervat *char*:

char [] sir;

În paragraful 1.5.1. s-a arătat că la declararea tablourilor, acestea se pot și inițializa. Tablourile ale căror elemente sunt de tip *char* se pot inițializa prin caractere incluse între acolade și separate prin virgulă

Exemplu

```
1. char [ ] sir = { '+', '-', '(', ')', '*', '/' }
```

Tabloul *sir* declarat în acest fel are 6 elemente, fiecare este de tip *char* și au următoarele valori inițiale:

```
sir[0] – codul ASCII al caracterului +  
sir[1] – codul ASCII al caracterului -  
sir[2] – codul ASCII al caracterului (  
sir[3] – codul ASCII al caracterului )  
sir[4] – codul ASCII al caracterului *  
sir[5] – codul ASCII al caracterului /
```

Șirurile de caractere se pot păstra și în obiecte ale clasei *String*. Clasa *String* se află în spațiul de nume *System*. Un obiect al clasei *String* se poate declara sub forma:

```
System.String numeobiect;
```

Evident, în cazul în care este prezentă directiva *using System* putem să declarăm mai simplu obiectul respectiv, astfel:

```
String numeobiect;
```

Există și un sinonim pentru *System.String* și anume, cuvântul *string*. Astfel, indiferent dacă este prezentă sau nu directiva *using System*; putem întotdeauna să utilizăm cuvântul *string* pentru a declara obiecte ale clasei *String*, astfel:

```
string numeobiect;
```

Aici, *string* nu înseamnă altceva decât clasa *String* din spațiul de nume *System*.

În cele ce urmează vom accepta faptul că *string* este și el un *nume de tip* și anume același *tip* care este definit de clasa *String* din spațiul de nume *System*.

Obiectele clasei *string* pot fi inițializate la declararea lor. În acest scop putem folosi o declarație de forma:

```
string nume="...";
```

Prin această declarație, *nume* este un obiect al clasei *string* și el conține șirul de caractere situat la dreapta caracterului =.

De exemplu declarația:

```
string sir="abc";
```

declară pe *sir* ca obiect al clasei *string* și acesta este inițializat cu șirul de caractere "abc".

Un obiect al clasei *string* poate fi parametru al metodei *WriteLine*, deci apelul de mai jos este corect:

```
Console.WriteLine(sir);
```

În urma acestui apel se va afișa pe ecranul monitorului caracterele *abc* din compunerea șirului păstrat în obiectul *sir*.

Știm că același efect se obține dacă am fi apelat metoda *WriteLine* astfel:

```
Console.WriteLine("abc");
```

În fine, metoda *WriteLine* se poate apela având ca parametru și un tablou ale cărui elemente sunt de tip *char*. Același efect se obține dacă considerăm declarația:

```
char []tab = { 'a', 'b', 'c' };
```

și apoi apelul metodei *WriteLine* având ca parametru pe *tab*:

```
Console.WriteLine(tab);
```

În limbajul C# putem defini tablouri ale căror elemente să fie obiecte ale clasei *string*. Un astfel de tablou îl vom declara printr-o declarație de forma:

```
string []nume;
```

unde *nume* este numele tabloului care se declară. De asemenea, la declararea unui tablou ale cărui elemente sunt de tip *string* se pot indica și valori pentru inițializarea elementelor tabloului. În acest scop, vom utiliza o declarație de forma:

```
string []nume = {sir1, sir2, ..., sirn};
```

unde *sir1*, *sir2*, . . . , *sir_n* sunt șiruri de caractere.