1. INSTRUCȚIUNI

Limbajul C# posedă un set bogat de instrucțiuni care permit transcrierea reprezentărilor diferiților algoritmi în limbajul respectiv.

Printre altele, limbajul C# conține *instrucțiuni* adecvate pentru a transcrie structurile proprii *programării structurate.*

În afară de acestea, limbajul C# conține şi alte instrucțiuni care facilitează transcrierea algoritmilor în limbajul C#.

Structurile secvențiale se pot descrie folosind instrucțiunea *vidă,* instrucțiunea *expresie* şi instrucțiunea *bloc.*

Structura alternativă se descrie cu ajutorul instrucțiunii *if.*

Structura repetitivă condiționată anterior se descrie cu ajutorul instrucțiunilor *while, for* şi *foreach.*

Structura repetitivă condiționată posterior se descrie cu ajutorul instrucțiunii *do-while.*

Structura selectivă se descrie cu ajutorul instrucțiunii *switch.*

Limbajul C# posedă şi alte instrucțiuni şi anume instrucțiunile *break* şi *continue* care nu sunt proprii programării structurate, dar sunt utile, de exemplu pentru a termina forțat o instrucțiune repetitivă (ciclică). O altă instrucțiune utilă este instrucțiunea *return,* necesară pentru a reveni dintr-o metodă, altfel decât la terminarea acesteia. Cu ajutorul ei se returnează valoarea determinată de metodă.

În acest capitol vom descrie structura acestor instrucțiuni. De asemenea, capitolul conține exerciții în care se vor utiliza instrucțiunile limbajului C#.

1.1. Instrucțiunea *vidă*

Instrucțiunea *vidă* se reduce la caracterul punct şi virgulă:

```
;
```

Ea *nu are nici un efect.*

Instrucțiunea vidă îşi găseşte utilizări în acele puncte ale aplicației în care este necesară o instrucțiune dar, de exemplu, din cauza compactărilor care au avut loc, în punctul respectiv nu trebuie să se execute nimic.

Instrucțiunea vidă este componentă a *structurilor secvențiale.*

1.2. Instrucțiunea *expresie*

Instrucțiunea *expresie* se compune dintr-o *expresie* urmată de caracterul punct şi virgulă:

(1) expresie;

Are ca efect evaluarea expresiei *expresie* din compunerea sa.

O astfel de instrucțiune se utilizează ca şi componentă a *structurilor secvențiale.*

Expresia din compunerea unei *instrucțiuni expresie* poate fi o *atribuire,* adică aceasta poate fi de forma:

(2) partestângă = expresie;

Despre o astfel de instrucțiune vom spune că este o *instrucțiune de atribuire.* Cu alte cuvinte, o *instrucțiune de atribuire* este un caz particular de *instrucțiune expresie* şi anume aceasta are loc când expresia care formează instrucțiunea *expresie* este o expresie de atribuire.

Un alt caz particular întâlnit frecvent, este acela când instrucțiunea expresie este de forma:

(3) **...**numemetodă(lista parametrilor efectivi);

adică situația în care *expresie* din compunerea *instrucțiunii expresie* este un *apel de metodă.* În acest caz, vom spune că *instrucțiunea expresie* este o *instrucțiune de apel de metodă* sau mai simplu *instrucțiune de apel.*

Până în prezent am apelat frecvent metoda *WriteLine* a clasei *Console* din spațiul de nume *System.* De asemenea, am apelat şi alte metode pantru a realiza operații de intrări/ieşiri standard şi anume: *Write, Read* şi *ReadLine*.

Apelurile acestor metode s-au realizat utilizând *instrucțiuni de apel.*

În continuare vom folosi frecvent şi instrucțiuni de atribuire.

Exemple

1.
```
int x=10;
 int y;
 y=x+7;
```

Ultima construcție este o *instrucțiune expresie* şi anume, chiar o *instrucțiune de atribuire.*

2.
```
Console.WriteLine("Sir afisat cu WriteLine");
```

Apelul metodei *WriteLine* se realizează printr-o *instrucțiune expresie.* Ea este o *instrucțiune de apel.*

3.
```
int x=10;
x++;
```

Ultima construcție este o *instrucțiune expresie.* Ea are același efect cu *instrucțiunea de atribuire:*

```
x=x+1;
```

4.
```
int x=10;
int y;
y=x++;
```

Ultima construcție este o *instrucțiune de atribuire.* Ea are același efect cu secvența de *instrucțiuni expresie:*

```
y=x;
x++;
```

deci, variabilei *y* i se atribuie valoarea 10 (valoarea neincrementată a lui *x*), iar *x* se incrementează și devine egal cu 11.

5.
```
int x=10;
int y;
y=++x;
```

Ultima construcție este o *instrucțiune de atribuire.* Efectul ei este echivalent cu secvența:

```
x++;
y=x;
```

deci, atât variabila *x* cât și variabila *y* devin egale cu 11.

Exerciții

1.1. Să se scrie o aplicație care afișează valoarea expresiei:

$$(x*x*x + 4*x*x - 7x - 3)/(x*x + 4*x - 7), \quad \text{pentru } x = 10.$$

Proiectul aplicației va avea numele *iexpr1*.

Valoarea expresiei se poate afișa utilizând expresia respectivă drept parametru al metodei *WriteLine*:

```
...WriteLine(x*x*x + 4*x*x - 7x - 3)/(x*x + 4*x - 7);
```

Având în vedere că expresia de mai sus poate fi pusă sub forma:

```
((x*x + 4*x - 7)*x - 3)/(x*x + 4*x - 7);
```

valoarea ei poate fi evaluată mai simplu dacă se atribuie valoarea numitorului unei variabilei, de exemplu *y:*

```
y=x*x + 4*x - 7;
```

Apoi apelăm metoda *WriteLine* astfel:

```
...WriteLine((y*x - 3)/y);
```

V-193
Limbajul C# pentru începători
Instrucțiunile limbajului C#
Liviu Negrescu, Lavinia Negrescu

1. INSTRUCȚIUNI