1. EXPRESII

O *expresie* se compune din *operanzi* şi *operatori.* În cea mai simplă formă, o expresie se compune dintr-un singur *operand.*

Evident, nu orice combinație de *operanzi* şi *operatori* va forma o *expresie.*

O *expresie* are întotdeauna o *valoare.*

Valoarea unei expresii reduse la un *operand* va fi chiar valoarea acelui *operand.*

O *valoare* aparține întotdeuna unui *tip* . De aceea şi *expresiile* au un *tip* şi anume acesta este tipul valorilor lor.

Deci, putem spune că o expresie are întotdeauna o *valoare* şi un *tip* care este tipul valorii sale.

De obicei, *tipul* unei expresii *rămâne acelaşi* pe parcursul execuției aplicației care conține expresia respectivă. Totuşi, pot exista şi expresii a căror tip să *difere* pe parcursul execuției aplicației care le conțin.

La *compilarea* unei aplicații se pot determina numai *tipurile expresiilor* care *nu se modifică pe parcursul execuției aplicației* care le conțin.

Pentru expresiile ale căror *tipuri pot diferi pe parcursul execuției aplicațiilor* care le conțin, *tipurile* acestora *se pot determina* numai la *execuția aplicațiilor* respective.

La începutul capitolului ne vom opri puțin asupra tipurilor limbajului C#, apoi ne vom ocupa de *operanzii* expresiilor, iar în continuare vom trece în revistă *operatorii* care sunt destul de numeroşi. Aşa cum o să vedem în continuare, aceştia pot fi împărțiți în mai multe categorii.

Din matematică se ştie că operatorilor li se atribuie anumite *priorități* care au importanță la evaluarea expresiilor care conțin mai mulți operatori. *Prioritățile* impun ordinea de execuție a operatorilor din compunerea unei expresii. De asemenea, există şi reguli de *asociere* a operatorilor de aceiaşi prioritate, când aceştia sunt întâlniți într-o aceiaşi expresie.

1.1. Despre tipuri

În limbajul C# se disting mai multe categorii de tipuri şi anume:

- tipuri *valoare;*
- tipuri *referinţă;*
- tipuri *pointeri.*

Datele ale căror tipuri sunt *tipuri valoare,* memorează *valori,* aşa cum le spune şi numele.

Datele ale căror tipuri sunt *tipuri referinţă,* păstrează *referinţe* la diferite date, cum ar fi obiecte, tablouri, funcţii etc.

Pointerii sunt date care au ca valori *adrese.* Ei au fost introduşi în limbajul C şi au o mare utilizare în limbajele C şi C++. Au fost excluşi din limbajul Java, deoarece s-a considerat că ei sunt surse de erori în programare. În limbajul C# ei sunt prezenţi şi pot fi utilizaţi într-o manieră restrictivă. Astfel, putem utiliza pointeri în C# numai în aşa numite contexte *unsafe* (nesigure). Contextele din C# sunt în general *sigure* (contexte *safe*), în sensul că ele sunt *verificabile* la execuţie.

Un context *unsafe* poate fi introdus în C# printr-o construcţie de forma:

```
unsafe
{
        ...//aici se pot folosi date de tipuri pointeri
}
```

Un *tip valoare* poate fi convertit spre un *tip referinţă* şi invers.

Tipurile valoare se pot împărţi în:

- tip *structurat (Struct type);*
- tip *enumerare (Enumeration type).*

Tipurile *structurate* conţin *tipurile definite de utilizator* folosind construcţia *struct* şi tipurile *predefinite* numite şi *primitive.*

Tipurile *primitive* pot fi şi ele împărţite în: tipuri *numerice,* tipul *char* (pentru lucrul cu caractere) şi tipul *bool* (tipul datelor logice).

Tipurile *numerice* conţin *tipurile integrale* (*Integral types*) (pentru date întregi), *tipurile flotante (Floating-point types) (*pentru date flotante, adică numere oarecare) şi tipul *decimal* (pentru numere de până la 28-29 cifre).

Tipurile *primitive* se precizează cu ajutorul unor cuvinte cheie care au fost indicate în paragraful 1.2., din volumul 2.

În ordine alfabetică, acestea sunt:

bool, byte, char, decimal, double, float, int, long, sbyte, short, uint, ulong, ushort.

Tipurile *flotante* sunt definite prin cuvintele cheie *double* şi *float,* iar tipul *logic* este definit prin cuvântul cheie *bool.*

Tipul *decimal* este definit prin cuvântul cheie *decimal.* Celelalte cuvinte cheie definesc *tipuri integrale* (tipuri de date ce au valori întregi).

Fiecărui tip primitiv îi corespunde o clasă în spațiul de nume *System.* De obicei, clasa are acelaşi nume cu numele tipului dar prima literă a numelui clasei este literă mare. Există şi câteva excepții pentru tipul *bool, float, int, long* etc.

Mai jos se listează corespondența dintre tipurile primitive şi clasele din spațiul *System* care le corespund.

Tip definit printr-un cuvânt cheie	Clasă din spațiul de nume System corespunzătoare
bool	Boolean
byte	Byte
char	Char
decimal	Decimal
double	Double
float	Single
int	Int32
long	Int64
sbyte	SByte
short	Int16
uint	UInt32
ulong	UInt64
ushort	UInt16

Acest tabel indică faptul că putem utiliza numele claselor în locul cuvintelor cheie ale tipurilor.

De exemplu, o declarație de forma:

```
int x = 123456789;
```

poate fi scrisă şi sub forma

```
System.Int32 x = 123456789;
```

sau dacă este prezentă directiva *using System;,* atunci putem scrie mai simplu:

```
Int32 x = 123456789;
```

Să ne amintim că o declarație de forma:

```
string s;
```

poate fi scrisă şi sub forma

```
System.String s;
```

sau mai simplu:

```
String s;
```

V-192
Limbajul C# pentru începători
Expresii
Liviu Negrescu, Lavinia Negrescu

1. EXPRESII
1.1. Despre tipuri
1.2. Operanzi
Exerciţii
1.3. Operatori
1.3.1. Operatorii aritmetici
Exerciţii
1.3.2. Operatorul paranteze rotunde
1.3.3. Operatorul punct
1.3.4. Despre conversii
Exerciţii
1.3.5. Operatorul de conversie explicită (Operatorul cast)
Exerciţii
1.3.6. Operatorii relaţionali
Exerciţii
1.3.7. Operatorii de egalitate
Exerciţii
1.3.8. Din nou despre operatorul paranteze rotunde
Exerciţii
1.3.9. Operatorii logici condiţionali şi operatorul logic de negaţie
Exerciţii
1.3.10. Operatori logici
Exerciţii
1.3.11. Operatorul de complementare faţă de 1
Exerciţii
1.3.12. Operatorii de deplasare
Exerciţii
1.3.13. Operatorul paranteze pătrate sau de indexare
Exerciţii
1.3.14. Operatorul dimensiune (sizeof)
Exerciţii
1.3.15. Operatorul de determinare a tipului (typeof)
Exerciţii
1.3.16. Operatorii pentru testarea (checked) şi netestarea (uncheked) depăşirilor la evaluarea expresiilor
Exerciţii
1.3.17. Operatorii de atribuire
Exerciţii