

SILVIA GRECU

LUCIA MIRON

MIRELA ȚIBU

MEMORATOR DE INFORMATICĂ

LIMBAJUL C++

Clasele 9-12

Ediția a II-a

Editura Paralela 45

Cuprins

I. Elemente de bază ale limbajului C++	7
II. Algoritmi elementari.....	17
III. Fişiere text.....	33
IV. Tablouri unidimensionale (Vectori).....	35
V. Tablouri bidimensionale (Matrice).....	43
VI. Şiruri de caractere	47
VII. Structuri de date neomogene	53
VIII. Subprograme (Funcţii)	57
IX. Funcţii recursive.....	63
X. Elemente de combinatorică.....	67
XI. Elemente de teoria grafurilor.....	73

I. ELEMENTE DE BAZĂ ALE LIMBAJULUI C++

Structura unui program C++

```
#include <iostream>
    //includerea altor librării necesare în
    program
using namespace std;
    //declarare variabile globale
    //declarare funcții utilizator
int main()
{
    //declarare variabile locale
    instrucțiuni
    return 0;
}
```

Citirea valorilor variabilelor de la tastatură / afișarea valorilor expresiilor pe ecran

```
cin>>var1>>var2>>...>>varn;
cout<<exp1<<' '<<exp2<<' '<<...<<expn;
```

Atribuirea

```
variabila = expresie;  
variabila <op> = expresie;  
    //unde op ∈ {+, -, *, /, %, >>, <<, &, |, ^}, este  
    echivalentă cu  
    //variabila = variabila <op> (expresie);
```

Instrucțiunea *if*

```
a) if (expresie)  
    instrucțiune_A;  
    else  
        instrucțiune_B;  
b) if (expresie)  
    instrucțiune_A;
```

Instrucțiunea *switch*

```
switch (expresie) {  
    case constantă_1: instrucțiuni_1  
                      break;  
    case constantă_2: instrucțiuni_2  
                      break;  
    .....  
    case constantă_n: instrucțiuni_n  
                      break;  
    default: instrucțiuni  
}
```

Instrucțiunea *while*

```
while (expresie)  
{  
    Instrucțiuni  
}
```

Instrucțiunea *do-while*

```
do {  
    Instrucțiuni  
} while (expresie);
```

Instrucțiunea *for*

```
for (expresie1; expresie2; expresie3)  
{  
    Instrucțiuni  
}
```

expresie1: expresie de inițializare;

expresie2: expresie de test;

expresie3: expresie de continuare.

Tipuri de date simple

Tip variabilă:	
<i>short int</i>	
Nr. octeți (bytes): 2	
Domeniul de valori: $[-2^{15}, 2^{15} - 1]$	
<i>unsigned short int</i>	
Nr. octeți (bytes): 2	
Domeniul de valori: $[0, 2^{16} - 1]$	
<i>int = long int</i>	
Nr. octeți (bytes): 4	
Domeniul de valori: $[-2^{31}, 2^{31} - 1]$	
<i>unsigned int = unsigned long int</i>	
Nr. octeți (bytes): 4	
Domeniul de valori: $[0, 2^{32} - 1]$	
<i>long long int</i>	
Nr. octeți (bytes): 8	
Domeniul de valori: $[-2^{63}, 2^{63} - 1]$	
<i>unsigned long long int</i>	
Nr. octeți (bytes): 8	
Domeniul de valori: $[0, 2^{64} - 1]$	
Operatori:	
Aritmetici: +, -, *, /, %	Pe biți: >>, <<, &, , ^, ~
Relaționali: <, <=, >=, >	De egalitate: ==, !=
Funcții specifice:	
<cmath.h>	
sqrt(x), $x \geq 0$, pentru $\sqrt{x}$	
abs(x) pentru $ x $	
pow(a,b) pentru a^b	

Tip variabilă:
<i>float</i>
Nr. octeți (bytes): 4 Domeniul de valori: $[-3.2 \cdot 10^{38}, 3.2 \cdot 10^{38}]$
<i>double</i>
Nr. octeți (bytes): 8 Domeniul de valori: $[-1.7 \cdot 10^{308}, 1.7 \cdot 10^{308}]$
Operatori:
Aritmetici: +, -, *, / Relaționali: <, <=, >=, > De egalitate: ==, !=
Funcții specifice:
<cmath.h> sqrt(x), $x \geq 0$, pentru $\sqrt{x}$ abs(x) pentru $ x $ pow(a,b) pentru a^b

Tip variabilă:
<i>char</i>
Nr. octeți (bytes): 1 Domeniul de valori: $[-3.2 \cdot 10^{38}, 3.2 \cdot 10^{38}]$
<i>unsigned char</i>
Nr. octeți (bytes): 1 Domeniul de valori:
Operatori:
Aritmetici: +, -, *, /, % Relaționali: <, <=, >=, > De egalitate: ==, !=

Funcții specifice:	
$\text{islower}(\text{car}) = \begin{cases} 1, & \text{dacă car e literă mică} \\ 0, & \text{altfel} \end{cases}$	
$\text{isupper}(\text{car}) = \begin{cases} 1, & \text{car e literă majusculă} \\ 0, & \text{altfel} \end{cases}$	
$\text{tolower}(\text{car}) = \text{car}$ în minusculă = $\text{car} + 'a' - 'A'$	
$\text{toupper}(\text{car}) = \text{car}$ în majusculă = $\text{car} - 'a' + 'A'$	

Operatori

1. Prioritate maximă:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
(), [], → .	Apel de funcție Expresie cu indici Selectorii de membru la structuri	St-Dr
2. Operatori unari:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
! ~, +, - ++, --, &, * sizeof(tip) (cast)	Negare logică Negare bit cu bit (complementare cu 1) Plus și minus unari Incrementare/decrementare (pre și post) Obținerea adresei/indirectare Dimensiune operand (în octeți) Conversie explicită de tip – cast	Dr-St

3. Operatori aritmetici multiplicativi:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
$*, /, \%$	Înmulțire/împărțire/restul împărțirii întregi	St-Dr
4. Adunare, scădere:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
$+, -$	Plus și minus binari	St-Dr
5. Deplasări:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
$<<, >>$	Deplasare stânga/dreapta pe biți	St-Dr
6. Relaționali:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
$<, <=, >, >=$	Mai mic/mai mic sau egal/Mai mare/ mai mare sau egal	St-Dr
7. Egalitate:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
$==, !=$	Egal/Diferit	St-Dr

8. 9. 10. Operatori logici pe biți:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
&, ^,	SI/SAU EXCLUSIV/SAU logic bit cu bit	St-Dr
11. 12. Operatori logici:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
&&,	SI/SAU logic	St-Dr
13. Operatori de atribuire compuși:		
<i>Operatori</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
= <op>=	• Atribuire simplă: <i>variabila = expresie;</i> • Atribuire multiplă: <i>variabila1 = variabila2 = ... = expresie;</i> • Atribuire compusă: <i>variabila <op> = expresie;</i> unde $op \in \{+, -, *, /, \%, >>, <<, \&, , \wedge\}$, este echivalentă cu <i>variabila = variabila <op> expresie;</i>	Dr-St
14. Operator condițional:		
<i>Operator</i>	<i>Semnificație</i>	<i>Asocia-tivitate</i>
?:	Operatorul condițional (ternar)	Dr-St

15. Virgula:		
<i>Operator</i>	<i>Semnificație</i>	<i>Asocia- tivitate</i>
,	Evaluare <i>expresie1, expresie2</i> ; Valoarea rezultatului este valoarea <i>expresie2</i> .	St-Dr

II. ALGORITMI ELEMENTARI

Algoritmi care prelucrează cifrele unui număr

Construirea oglinditului și verificarea lui n dacă este palindrom

```
copie  $\leftarrow n$   
oglindit  $\leftarrow 0$   
cât timp  $n \neq 0$  execută  
    cif  $\leftarrow n \% 10$   
    oglindit  $\leftarrow \text{oglindit} * 10 + \text{cif}$   
     $n \leftarrow [n/10]$   
■  
dacă copie=oglindit atunci  
    prelucrare_cifra(palindrom)  
■
```

Media aritmetică a cifrelor nenule

```
sum  $\leftarrow 0$   
nrcif  $\leftarrow 0$   
cât timp  $n \neq 0$  execută  
    cif  $\leftarrow n \% 10$   
    dacă cif > 0 atunci  
        sum  $\leftarrow \text{sum} + \text{cif}$   
        nrcif  $\leftarrow \text{nrcif} + 1$   
    ■  
     $n \leftarrow [n/10]$   
■
```

```
└dacă  $nrcif > 0$  atunci  
    medie  $\leftarrow$  sum/nrcif  
■
```

Cifra maximă din n

```
cifmax  $\leftarrow$  0  
└cât timp  $n \neq 0$  execută  
    cif  $\leftarrow$   $n \% 10$   
    └dacă  $cif > cifmax$   
        atunci  
            cifmax  $\leftarrow$  cif  
    ■  
    n  $\leftarrow$   $[n/10]$   
■  
prelucrare(cifmax)
```

Eliminarea cifrelor impare din n

```
nrNou  $\leftarrow$  0  
p  $\leftarrow$  1  
└cât timp  $n \neq 0$  execută  
    cif  $\leftarrow$   $n \% 10$   
    └dacă  $cif \% 2 = 0$  atunci  
        nrNou  $\leftarrow$  nrNou + cif * p  
        p  $\leftarrow$  p * 10  
    ■  
    n  $\leftarrow$   $[n/10]$   
■  
prelucrare(nrNou)
```

Dublarea aparițiilor cifrelor pare

```
nrNou  $\leftarrow$  0
p  $\leftarrow$  1
cât timp  $n \neq 0$  execută
    cif  $\leftarrow$   $n \% 10$ 
    dacă  $cif \% 2 = 0$  atunci
        nrNou  $\leftarrow$  nrNou + cif * p
        p  $\leftarrow$  p * 10
    ■
    nrNou  $\leftarrow$  nrNou + cif * p
    p  $\leftarrow$  p * 10
    n  $\leftarrow$   $\lfloor n/10 \rfloor$ 
■
prelucrare(nrNou)
```

Numărarea cifrelor pare din n

```
nrPare  $\leftarrow$  0
repetă
    cif  $\leftarrow$   $n \% 10$ 
    dacă  $cif \% 2 = 0$ 
        atunci
            nrPare  $\leftarrow$  nrPare + 1
    ■
    n  $\leftarrow$   $\lfloor n/10 \rfloor$ 
până când  $n = 0$ 
prelucrare(nrPare)
```

Cifra de control a unui număr n

Cifra de control a unui număr n se obține calculând suma cifrelor lui n , apoi repetând procesul cu cifrele sumei obținute anterior până când se obține un număr format dintr-o singură cifră, numită cifră de control. De exemplu, pentru $n = 7912$ se obțin pe rând sumele $7 + 9 + 1 + 2 = 19$, $1 + 9 = 10$, $1 + 0 = 1$, iar 1 este cifra de control a lui 7912.

```

┌cât timp  $n > 9$  execută
│  sumcif  $\leftarrow 0$ 
│  ┌cât timp  $n \neq 0$  atunci
│  │  cif  $\leftarrow n \% 10$ 
│  │  sumcif  $\leftarrow$  sumcif + cif
│  │   $n \leftarrow [n/10]$ 
│  │  ■
│  └n  $\leftarrow$  sumcif
│  ■
prelucrare(n)
```

Algoritmul eficient ca timp de execuție se bazează pe observația că cifra de control a unui număr este periodică și respectă relația:

$$cifControl(n) = \begin{cases} 0, & \text{dacă } n = 0 \\ 9, & \text{dacă } n \% 9 = 0 \text{ și } n \neq 0 \\ n \% 9, & \text{altfel} \end{cases}$$

Divizibilitate. Algoritmi care prelucrează divizorii proprii/improprii/primi ai unui număr

Fie n un număr natural. Definim mulțimile:

Divizorii **proprii** ai lui n : $d \in \left\{ 2, 3, 4, \dots, \left\lfloor \frac{n}{2} \right\rfloor \right\}, n \div d$.

Divizorii **primi** ai lui n : $d \in \{2, 3, 5, 7, \dots, \lfloor \sqrt{n} \rfloor\}, n \div d$.

Divizorii **improprii** ai lui n : $d \in \{1, n\}$.

Divizorii proprii ai lui n

```
p ← 1
pentru d ← 2, [n/2], 1 execută
    dacă n % d = 0 atunci
        prelucrare (d)
```

Divizorii proprii ai lui n - optimizat

```
p ← 1
pentru d ← 2, [√n]-1, 1 execută
    dacă n % d = 0 atunci
        prelucrare (d)
        prelucrare (n/d)
dacă d * d = n atunci
    prelucrare (d)
```