

# Programarea calculatoarelor

## #7

C++

Structura unui program C++.

Baze de numeratie

Adrian Runceanu

[www.runceanu.ro/adrian](http://www.runceanu.ro/adrian)

# ***Curs 7***

# Capitolul 6

## 6.1. Preprocesare

## 6.2. Structura unui program C++

## 6.3. Baze de numerație

## 6.4. Conversia din baza 2 în bazele 8 și 16 și invers

### 6.4.1. Conversia din baza 2 în baza 8 și invers

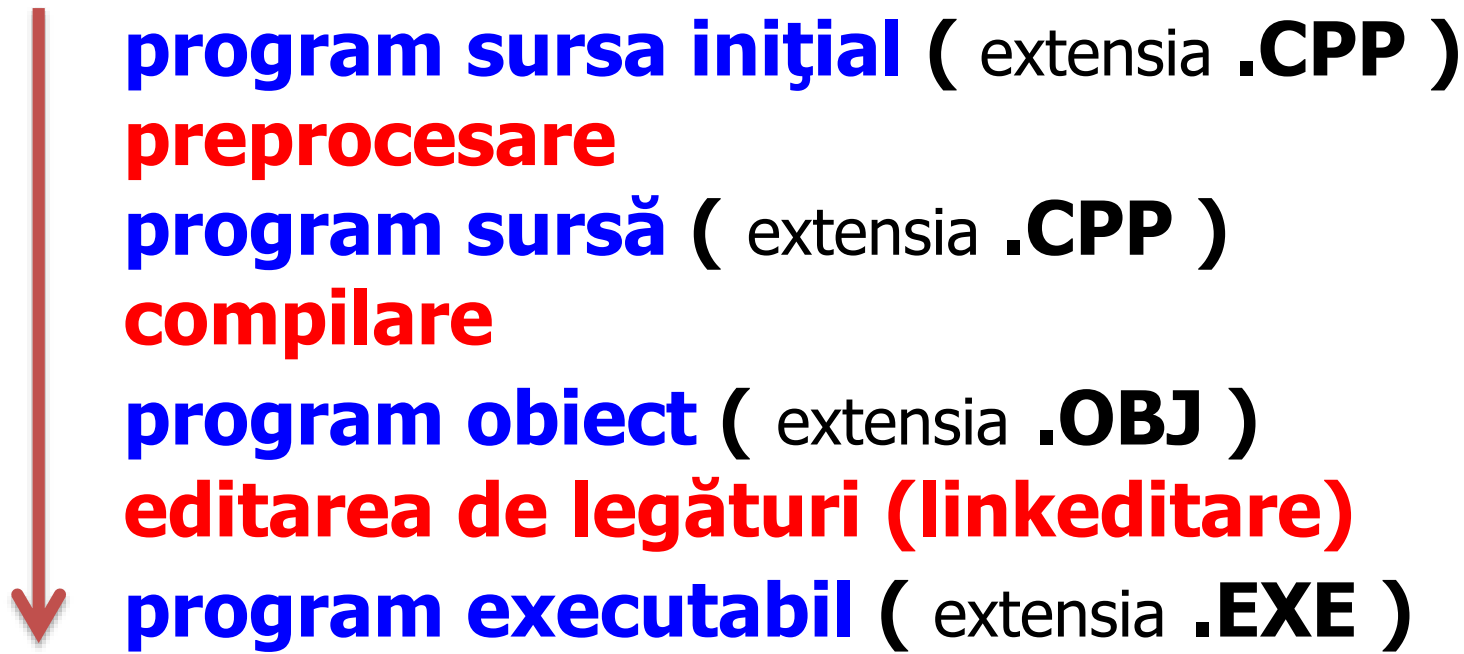
### 6.4.2. Conversia din baza 2 în baza 16 și invers

### 6.4.3. Operații aritmetice în binar, octal, hexazecimal

## 6.5. Probleme propuse spre rezolvare

## 6.1. Preprocesare

Schematic, parcursul unui program sursa scris în C++, până la executarea lui este următorul:



## 6.1. Preprocesare

În C++, după construirea unui program sursă, *se pot evalua anumite valori* dacă se utilizează preprocesorul.

**Activitatea preprocesorului** se împarte în:

- a) includere de fișiere standard / utilizator
- b) definire constante simbolice
- c) definire macro-uri(macroinstrucțiuni)
- d) compilare condiționată

## 6.1. Preprocesare

a) **Includerea de fișiere standard/utilizator** se face pentru a putea:

- utiliza **funcțiile predefinite** ale limbajului C++ care se afla în **fișiere standard** numite **header**-e (au extensia **.h**)
- sau pentru a putea utiliza *funcții proprii aflate în fișiere utilizator*

## 6.1. Preprocesare

Astfel, pentru a putea utiliza funcțiile standard de intrare / ieșire **cin** / **cout**, trebuie scris la începutul unui program C++:

**#include <iostream.h>** - fișier standard

iar pentru a utiliza fișiere utilizator:

**#include "nume fisier"** – fișier utilizator

Observație:

Includerea fișierelor se execută numai pe timpul compilării

## 6.1. Preprocesare

### b) Definire constante simbolice

Pentru a mări portabilitatea programelor C++, se pot folosi *constante*.

O **constantă** este un nume pe care compilatorul C++ îl asociază unei valori care nu se modifică.

Pentru aceasta se utilizează directiva ***#define***.



## 6.1. Preprocesare

```
#define  nume_constanta  text
```

- Asociază numelui **nume\_constanta** textul denumit **text** care se poate prelungi pe mai linii.
- Aceasta substituție a numelui este valabilă în tot fișierul până la întâlnirea unei directive de compilare **#undef nume**.

## 6.1. Preprocesare

**Exemplu:**

```
#define  NRLINII          30
#define  NRCOLOANE       20
#define  DIMENSIUNE      NRLINII*NRCOLOANE
int t[NRLINII][NRCOLOANE];
double a[DIMENSIUNE];
```

Ce va întâlni compilatorul după preprocesor?

```
int t[30][20];
double a[30*20];
```

Observație: nu se vor face calculele pentru că este vorba doar de informație la nivel de **text**

## 6.1. Preprocesare

Dacă se vor utiliza **macroinstrucțiuni** sau **constante simbolice** în programele C++, atunci acestea trebuie să aibă **nume sugestive și scrise cu litere mari** pentru a putea ajuta programatorii care citesc codul sursă să facă diferența ușor între constante și variabile.

Exemplu: Putem defini următoarele constante:

```
#define TRUE 1
```

```
#define FALSE 0
```

```
#define PI 3.1415
```

```
#define PROGRAMATOR "Pop Ion"
```

## 6.1. Preprocesare

### c) Definirea de macroinstrucțiuni

**Un macrou (o macroinstrucțiune)** este o definiție în care funcționează reguli de substituție de text.

```
#define nume(param_form1, param_form2, . . . ,param_formn) text
```

Apelul macroului se face astfel:

```
nume(param_actual1, param_actual2,...,param_actualn)
```

## 6.1. Preprocesare

Unde:

- **nume** reprezintă numele macroului
- **param\_form<sub>1</sub>, param\_form<sub>2</sub>, . . . , param\_form<sub>n</sub>**  
parametrii formali ai macroului
- iar **text** este textul în care se vor face înlocuirile
- iar **param\_actual<sub>1</sub>, param\_actual<sub>2</sub>, . . . , param\_actual<sub>n</sub>** sunt parametrii cu care se apelează macroul în funcția principală

## 6.1. Preprocesare

Funcționarea macroului este următoarea:

1. în textul dat *se înlocuiesc parametrii formali cu parametrii actuali*
2. iar apoi *textul obținut va substitui apelul macroului.*

**Exemplu:**

```
#define MAX(a, b) ( (a) < (b) ? (b) : (a) )
```

```
#define MIN(a, b) ( (a) > (b) ? (b) : (a) )
```

```
#define SGN(x) ((x) > 0 ? 1 : ( (x) == 0 ? 0 : (-1)))
```

## 6.1. Preprocesare

Am definit câteva macrouri care pot fi utile în multe programe C++, și anume:

- **MAX** și **MIN** determină *maximul*, respectiv *minimul* a două numere de orice tip
- iar **SGN** determină *signatura* unui număr dat

a, b parametri  
formali

## 6.1. Preprocesare

Următorul  
program C++  
va folosi  
aceste două  
macro-uri:

```
#include<iostream.h>
#define MAX(a, b)  ( (a) < (b) ? (b) : (a) )
#define MIN(a, b)  ( (a) > (b) ? (b) : (a) )

int main(void)
{
    cout<<"Maximul valorilor 10.0 si 25.0  
este " <<MAX(10.0, 25.0)<<endl;
    cout<<"Minimul valorilor 3.4 si 3.1  
este " <<MIN(3.4, 3.1)<<endl;
}
```

parametri actuali



# Capitolul 6

**6.1. Preprocesare**

**6.2. Structura unui program C++**

**6.3. Baze de numerație**

**6.4. Conversia din baza 2 în bazele 8 și 16 și invers**

**6.4.1. Conversia din baza 2 în baza 8 și invers**

**6.4.2. Conversia din baza 2 în baza 16 și invers**

**6.4.3. Operații aritmetice în binar, octal,  
hexazecimal**

**6.5. Probleme propuse spre rezolvare**

## 6.2. Structura unui program C++

```
/*-- nume.cpp -- comentariu inițial --*/
```

```
#include <iostream.h>
```

```
#include <math.h>
```

```
... /*-- alte directive include --*/
```

```
[ declarații și definiții globale ]
```

```
int main(void)
```

```
{
```

```
    [ declarații locale ]
```

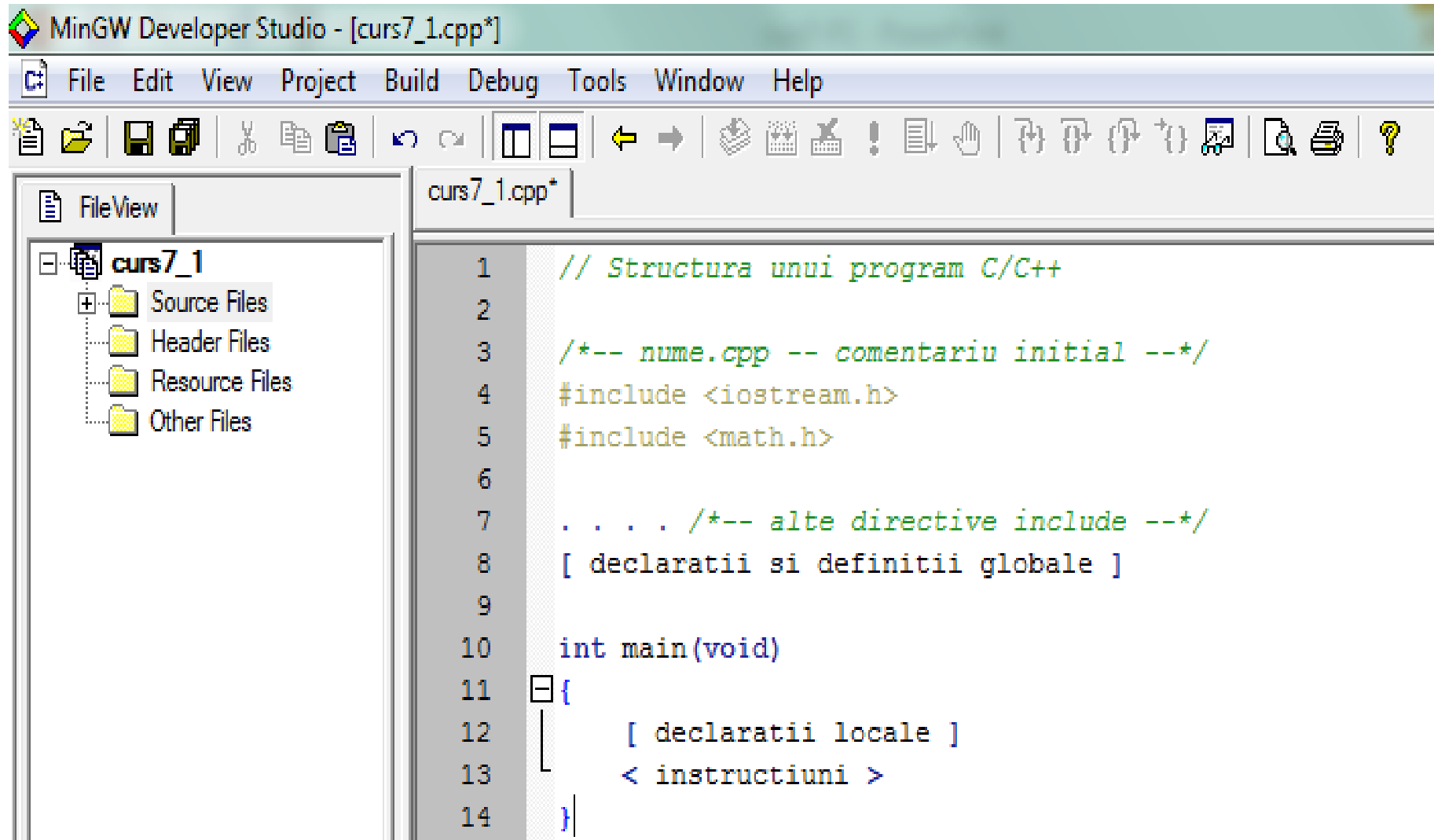
```
    < instrucțiuni >
```

```
}
```



optional

## 6.2. Structura unui program C++



# Capitolul 6

**6.1. Preprocesare**

**6.2. Structura unui program C++**

**6.3. Baze de numerație**

**6.4. Conversia din baza 2 în bazele 8 și 16 și invers**

**6.4.1. Conversia din baza 2 în baza 8 și invers**

**6.4.2. Conversia din baza 2 în baza 16 și invers**

**6.4.3. Operații aritmetice în binar, octal,  
hexazecimal**

**6.5. Probleme propuse spre rezolvare**

## 6.3. Baze de numerație

### Baza de numerație 2 (sistemul *BINAR*)

Ca în orice bază de numerație, cifrele folosite în reprezentarea numerelor sunt cuprinse în intervalul:

*[0, baza-1]*

Rezultă că în baza 2 avem o reprezentare a numerelor folosind doar cifrele **0** și **1**.

Fiecare dintre cifrele semnificative ale unei baze de numerație poartă denumirea de *digit*.

În baza 2 deoarece sunt doar *doi* digiți posibili aceștia au preluat denumirea de *binary digit (bit)*.

De aici proveniența cuvântului **bit**.

## 6.3. Baze de numerație

În baza *10* aceste cifre sunt *0...9*.

Alte baze de numerație folosite în legătură cu sistemul binar sunt: *4, 8 (octal) și 16 (hexazecimal)*.

Pentru baza *16* cifrele de reprezentare sunt:

*0 . . 9 și A . . F*

S-a convenit folosirea primelor litere ale alfabetului, cu semnificația:

*A ține locul lui 10*

*B lui 11*

*C lui 12*

*D lui 13*

*E lui 14*

*F lui 15*

## 6.3. Baze de numerație

În calculatoarele actuale baza de numerație este **2**.

Au existat încercări de creare a unor calculatoare în bază **10**, dar nu s-au putut ridica la performanțele calculatoarelor binare.

S-a păstrat astfel **sistemul binar** ca standard pentru calculatoarele digitale.

## 6.3. Baze de numerație

Să luăm un exemplu de număr în baza **2**:

**0110 1101**

Ce înseamnă acesta?

Cum poate fi interpretat astfel încât să poată fi înțeles de către noi (adică tradus în baza 10)?

La aceste întrebări se răspunde plecând de la **regula de reprezentare** în **orice** bază de numerație (pozițională): *fiecărei poziții în număr îi corespunde o putere a acelei baze de numerație.*



## 6.3. Baze de numerație

Astfel:

- primei poziții din dreapta îi corespunde puterea 0 a lui 2,
- următoarei poziții îi corespunde puterea 1 a lui 2,
- iar ultimei poziții (prima din stânga) îi corespunde puterea 7 a lui 2.

Atunci putem *genera valoarea acestui număr în baza 10 - plecând de la dreapta spre stânga* - astfel:

$$\begin{aligned} 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5 + 1 \cdot 2^6 + 0 \cdot 2^7 = \\ = 1 + 4 + 8 + 32 + 64 = 109_{10} \end{aligned}$$

## 6.3. Baze de numerație

La fel procedăm cu un număr în **baza 10** când dorim să-i aflăm valoarea.

Dar, în **baza 10** interpretăm natural și aproape instantaneu orice număr, care ne sugerează și o puternică semnificație "cantitativă" (adică **mărimea** celui număr).

**Exemplu:**

$$578_{10} = 8 \cdot 10^0 + 7 \cdot 10^1 + 5 \cdot 10^2 = 8 + 70 + 500$$

Știm imediat că avem de-a face cu '*cinci sute șaptezeci și opt*' și că acesta se situează cam la jumătatea intervalului  $[0-1000]$ .

## 6.3. Baze de numerație

Notatii:

- ✓ *Bitul* se notează cu **b**.
- ✓ Combinația de *8 biți* succesivi se numește **Byte (octet)** și este reprezentat prin litera **B**

## 6.3. Baze de numerație

Într-un număr în baza 2 sunt importante două poziții:

Prima din dreapta - care poartă denumirea de **Least Significant bit (LSb)**

Prima din stânga - care poartă denumirea de **Most Significant bit (MSb)**

Poziția **MSb** are de obicei *rolul de semn* al numărului:

*0* are semnificația de *plus*

*1* are semnificația de *minus*

# Capitolul 6

**6.1. Preprocesare**

**6.2. Structura unui program C++**

**6.3. Baze de numerație**

**6.4. Conversia din baza 2 în bazele 8 și 16 și invers**

**6.4.1. Conversia din baza 2 în baza 8 și invers**

**6.4.2. Conversia din baza 2 în baza 16 și invers**

**6.4.3. Operații aritmetice în binar, octal,  
hexazecimal**

**6.5. Probleme propuse spre rezolvare**

## 6.4.1. Conversia din baza 2 în baza 8 și invers

Știm că sistemele de numerație **octal** (baza 8) și **hexazecimal** (baza 16) au particularitatea de a folosi ca **bază un număr (8 sau 16)** care rezultă din **ridicarea la puterea a 3-a sau a 4-a a cifrei 2**, astfel între cele trei sisteme de numerație se pot stabili compatibilități directe.

## 6.4.1. Conversia din baza 2 în baza 8 și invers

Astfel, conversia **octal-binar** și **binar-octal** pornește de la faptul că orice cifră octală se poate reprezenta prin 3 cifre binare:

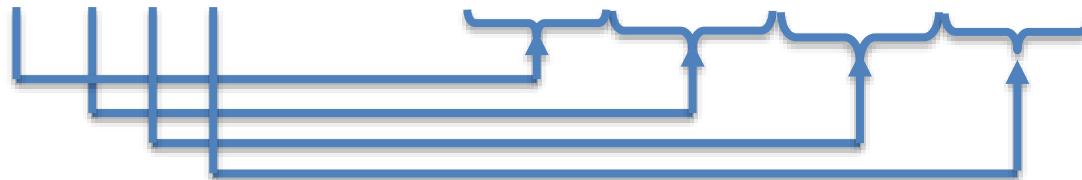
<b>0 = 000</b>	<b>4 = 100</b>
<b>1 = 001</b>	<b>5 = 101</b>
<b>2 = 010</b>	<b>6 = 110</b>
<b>3 = 011</b>	<b>7 = 111</b>

## 6.4.1. Conversia din baza 2 în baza 8 și invers

Dacă se consideră un număr octal, pentru conversia în binar se va scrie *fiecare cifră octală prin 3 cifre binare*.

Exemplu :

$$(3\ 4\ 7,\ 5)_8 = (011\ 100\ 111,\ 101)_2$$





## 6.4.1. Conversia din baza 2 în baza 8 și invers

Dacă se consideră un număr binar, pentru conversia în octal *se vor grupa câte 3 cifre binare pornind de la poziția virgulei spre stânga pentru partea întreagă, respectiv dreapta pentru partea fracționară, găsind corespondentul în octal.*

Exemplu:

$$\begin{array}{cccccc}
 ( & 001 & 110 & 011 & 101 & , & 101 & 100 & )_2 = (1635,54)_8 \\
 \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \\
 1 & 6 & 3 & 5 & 5 & & 5 & 4 & 
 \end{array}$$

# Capitolul 6

**6.1. Preprocesare**

**6.2. Structura unui program C++**

**6.3. Baze de numerație**

**6.4. Conversia din baza 2 în bazele 8 și 16 și invers**

**6.4.1. Conversia din baza 2 în baza 8 și invers**

**6.4.2. Conversia din baza 2 în baza 16 și invers**

**6.4.3. Operații aritmetice în binar, octal,  
hexazecimal**

**6.5. Probleme propuse spre rezolvare**

## 6.4.2. Conversia din baza 2 în baza 16 și invers

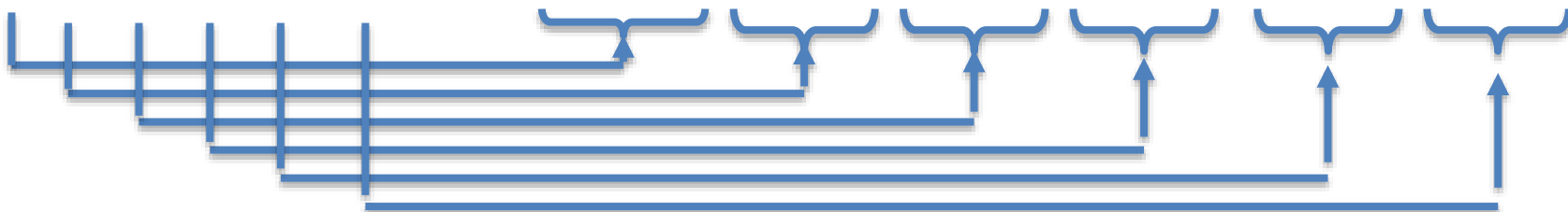
La fel se poate proceda și în cazul conversiei **binar - hexazecimal**, orice cifra hexazecimală putându-se reprezenta prin patru cifre binare:

<b>0 = 0000</b>	<b>4 = 0100</b>	<b>8 = 1000</b>	<b>C = 1100</b>
<b>1 = 0001</b>	<b>5 = 0101</b>	<b>9 = 1001</b>	<b>D = 1101</b>
<b>2 = 0010</b>	<b>6 = 0110</b>	<b>A = 1010</b>	<b>E = 1110</b>
<b>3 = 0011</b>	<b>7 = 0111</b>	<b>B = 1011</b>	<b>F = 1111</b>

## 6.4.2. Conversia din baza 2 în baza 16 și invers

Exemple:

$$(5 \text{ A F } 4, 3 \text{ E})_{16} = (0101 \ 1010 \ 1111 \ 0100, 0011 \ 1110)_2$$



## 6.4.2. Conversia din baza 2 în baza 16 și invers

$$( \underbrace{10}_{2} \underbrace{0110}_{6} \underbrace{1011}_{B} \underbrace{1100}_{C}, \underbrace{0011}_{3} \underbrace{1101}_{D} \underbrace{11}_{C} )_2 = ( 26BC, 3DC )_{16}$$

# Capitolul 6

**6.1. Preprocesare**

**6.2. Structura unui program C++**

**6.3. Baze de numerație**

**6.4. Conversia din baza 2 în bazele 8 și 16 și invers**

**6.4.1. Conversia din baza 2 în baza 8 și invers**

**6.4.2. Conversia din baza 2 în baza 16 și invers**

**6.4.3. Operații aritmetice în binar, octal,  
hexazecimal**

**6.5. Probleme propuse spre rezolvare**

## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

*a) Operații aritmetice în binar:*

adunare	înmulțire	scădere
$0 + 0 = 0$	$0 \times 0 = 0$	$0 - 0 = 0$
$0 + 1 = 1$	$0 \times 1 = 0$	$1 - 0 = 1$
$1 + 0 = 1$	$1 \times 0 = 0$	$1 - 1 = 0$
$1 + 1 = 10$	$1 \times 1 = 1$	$0 - 1 = 1^*$

Unde ‘\*’ semnifică un împrumut de la poziția imediat următoare a descăzutului, care pentru poziția curentă înseamnă 2 (deci se interpretează  $2 - 1 = 1$ ).

## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

*Exemple de operații în binar:*

$$\begin{array}{r} 11101101,101 + \\ 1011010,001 \\ \hline 101000111,110 \end{array}$$

$$\begin{array}{r} 1000101,110 - \\ 111010,011 \\ \hline 1011,011 \end{array}$$

$$\begin{array}{r} 110,11 \times \\ 10,11 \\ \hline 11011 \\ 11011 \\ 00000 \\ 11011 \\ \hline 10010,1001 \end{array}$$



## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

*b) Operații aritmetice în octal:*

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

### *Exemple de operații în octal:*

Se ține seama de următoarele reguli:

- La **adunare** și **înmulțire** rezultatul va fi constituit din **restul împărțirii sumei** sau **produsului la bază**, *câtul constituind transportul pentru poziția următoare*
- La **scădere**, **un împrumut** de la poziția următoare a numărului înseamnă **adunarea la descăzutul poziției curente, a bazei de numerație**

## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

$$\begin{array}{r} 1475,367 + \\ 562,51 \\ \hline 2260,077 \end{array}$$

$$\begin{array}{r} 34022,56 - \\ 1234,25 \\ \hline 32566,31 \end{array}$$

$$\begin{array}{r} 357,26 \times \\ 3,7 \\ \hline 321332 \\ 131602 \\ \hline 1637,352 \end{array}$$

## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

*c) Operații aritmetice în hexazecimal:*

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

## 6.4.3. Operații aritmetice în binar, octal, hexazecimal

*Exemple de operații în hexazecimal:*

<b>AF59C +</b>	<b>F000 –</b>
<b>D8E2</b>	<b>1</b>
<hr/>	<hr/>
<b>BCE7E</b>	<b>EFFF</b>
	<b>5DA2 ×</b>
	<b>B8</b>
	<hr/>
	<b>2ED10</b>
	<b>405F6</b>
	<hr/>
	<b>434C70</b>

# Capitolul 6

**6.1. Preprocesare**

**6.2. Structura unui program C++**

**6.3. Baze de numerație**

**6.4. Conversia din baza 2 în bazele 8 și 16 și invers**

**6.4.1. Conversia din baza 2 în baza 8 și invers**

**6.4.2. Conversia din baza 2 în baza 16 și invers**

**6.4.3. Operații aritmetice în binar, octal,  
hexazecimal**

**6.5. Probleme propuse spre rezolvare**

## 6.5. Probleme propuse spre rezolvare:

1. Să se convertească din sistemul **binar** în sistemul **octal** numerele reprezentate prin:

$$(101,101)_2 = ?_8$$

$$(111000111,101)_2 = ?_8$$

$$(10110,1101)_2 = ?_8$$

2. Să se convertească din sistemul **binar** în sistemul **hexazecimal** numerele reprezentate prin:

$$(110010,11011)_2 = ?_{16}$$

$$(111000111,101)_2 = ?_{16}$$

$$(10111111101)_2 = ?_{16}$$



## 6.5. Probleme propuse spre rezolvare:

3. Să se convertească din sistemul **octal** în sistemul **binar** numerele reprezentate prin:

$$(173,236)_8 = ?_2$$

$$(153)_8 = ?_2$$

4. Să se convertească din sistemul **hexazecimal** în sistemul **binar** numerele reprezentate prin:

$$(43,AC)_{16} = ?_2$$

$$(1C8,B)_{16} = ?_2$$

## 6.5. Probleme rezolvate

1) Se introduce un număr natural cu maxim 9 cifre. Să se determine și să se afișeze numărul de cifre, cea mai mare cifră, cea mai mică cifră și suma tuturor cifrelor acestui număr.

Exemplu:

Date de intrare: 24356103

Date de ieșire:

Numarul de cifre 8

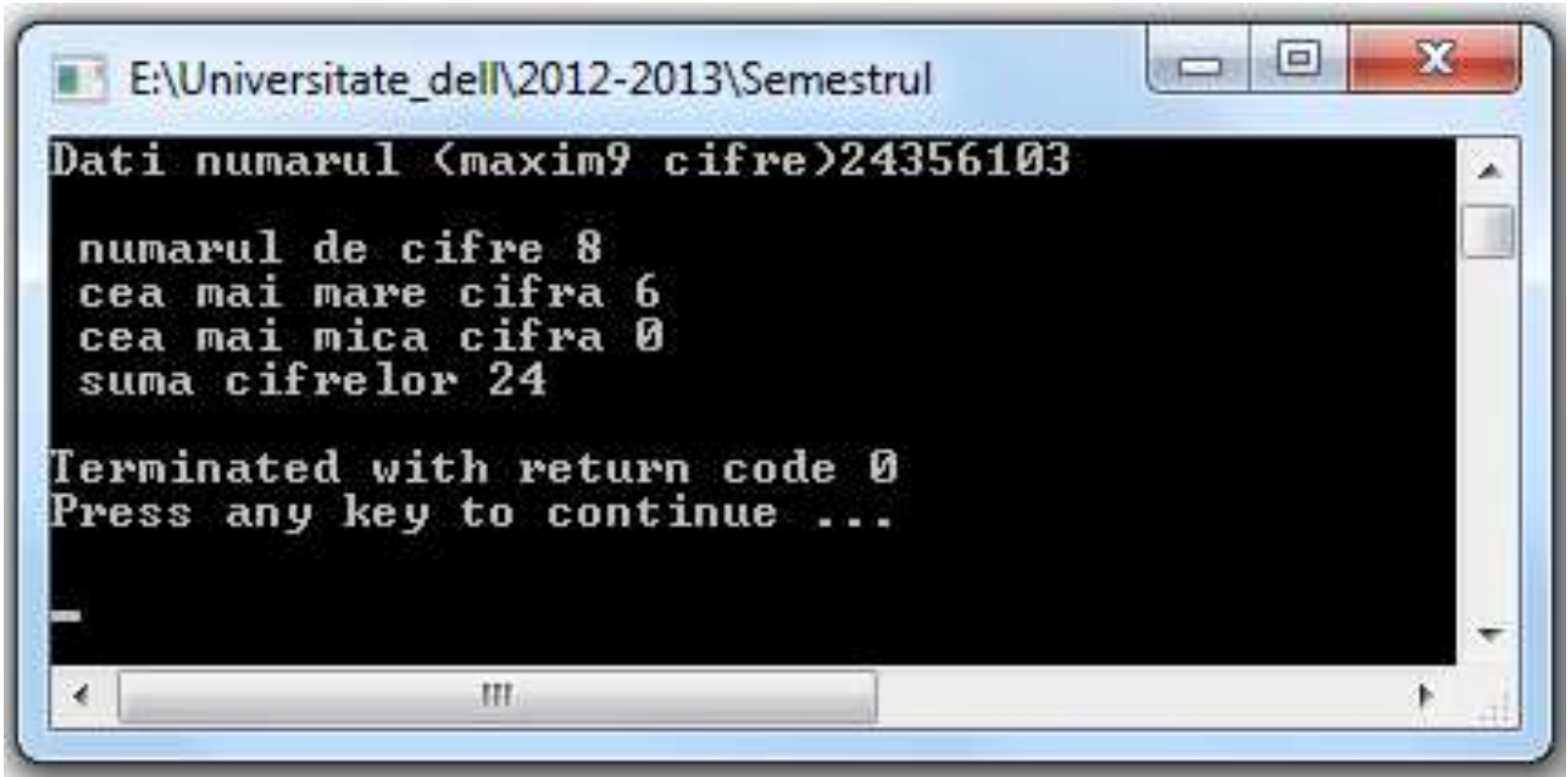
Cea mai mare cifra 6

Cea mai mica cifra 0

Suma cifrelor 24

```
#include<iostream.h>
int main()
{
    long int n;
    int nr_cifre=0;
    int min=100;
    int max=-100;
    int suma=0;
    int cifra;
    cout<<"Dati numarul
(maxim 9 cifre) ";
    cin>>n;
```

```
while(n!=0)
{
    cifra=n%10;
    nr_cifre++;
    if(cif>max) max=cifra;
    if(cif<min) min=cifra;
    suma=suma+cifra;
    n=n/10;
}
cout<<"\n numarul de cifre
"<<nr_cifre;
cout<<"\n cea mai mare cifra "<<max;
cout<<"\n cea mai mica cifra "<<min;
cout<<"\n suma cifrelor "<<suma;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "E:\Universitate\_de\2012-2013\Semestrul". The window has standard minimize, maximize, and close buttons. The command prompt displays the following text:

```
Dati numarul <maxim 9 cifre>24356103  
  
  numarul de cifre 8  
  cea mai mare cifra 6  
  cea mai mica cifra 0  
  suma cifrelor 24  
  
Terminated with return code 0  
Press any key to continue ...
```

The text is displayed in a monospaced font on a black background. A horizontal scrollbar is visible at the bottom of the window.

## 6.5. Probleme rezolvate

2) Dat un număr întreg de maxim 9 cifre, să se afișeze numărul de apariții al fiecărei cifre.

### Exemplu:

Date de intrare 364901211

Date de ieșire:

0 apare de 1 ori

1 apare de 3 ori

2 apare de 1 ori

3 apare de 1 ori

4 apare de 1 ori

5 apare de 0 ori

6 apare de 1 ori

7 apare de 0 ori

8 apare de 0 ori

9 apare de 1 ori

```

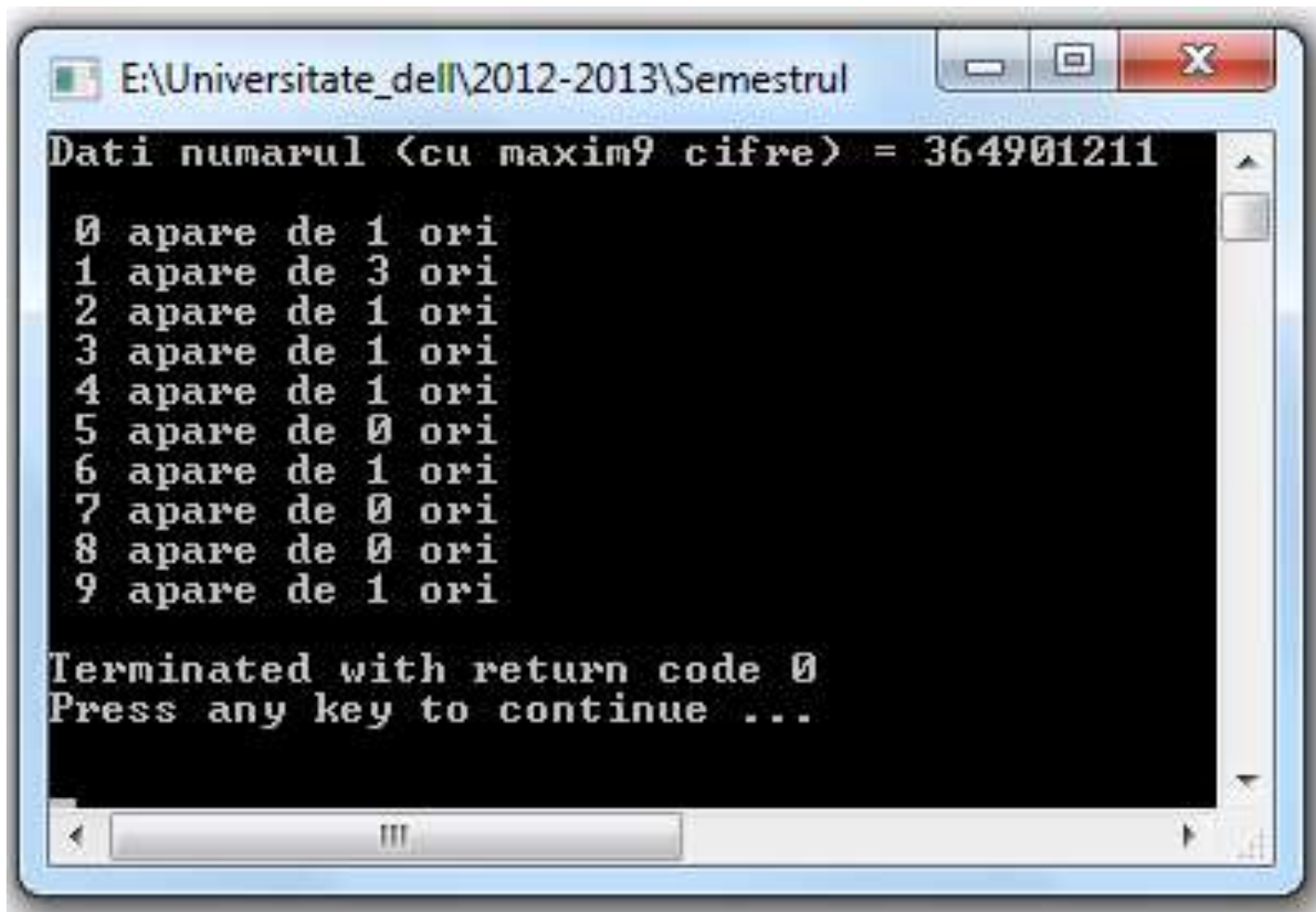
#include<iostream.h>
int main()
{
    long int n;
    int n0,n1,n2,n3,n4,n5,n6,n7,n8,n9;
    n0=n1=n2=n3=n4=n5=n6=n7=n8=n9=0;
    cout<<"Dati numarul (cu maxim 9 cifre) = ";
    cin>>n;
    while(n!=0) {
        switch(n%10){
            case 0: n0++;break;
            case 1: n1++;break;
            case 2: n2++;break;
            case 3: n3++;break;
            case 4: n4++;break;
            case 5: n5++;break;
            case 6: n6++;break;
            case 7: n7++;break;
            case 8: n8++;break;
            case 9: n9++;break;
        }
        n=n/10;
    }
}

```

```

cout<<"\n 0 apare de "<<n0<<" ori";
cout<<"\n 1 apare de "<<n1<<" ori";
cout<<"\n 2 apare de "<<n2<<" ori";
cout<<"\n 3 apare de "<<n3<<" ori";
cout<<"\n 4 apare de "<<n4<<" ori";
cout<<"\n 5 apare de "<<n5<<" ori";
cout<<"\n 6 apare de "<<n6<<" ori";
cout<<"\n 7 apare de "<<n7<<" ori";
cout<<"\n 8 apare de "<<n8<<" ori";
cout<<"\n 9 apare de "<<n9<<" ori";
}

```



The screenshot shows a Windows command prompt window with the title bar "E:\Universitate\_dell\2012-2013\Semestrul". The command prompt displays the following text:

```
Dati numarul <cu maxim9 cifre> = 364901211
```

Cifra	apare de	ori
0	1	ori
1	3	ori
2	1	ori
3	1	ori
4	1	ori
5	0	ori
6	1	ori
7	0	ori
8	0	ori
9	1	ori

Terminated with return code 0  
Press any key to continue ...

# Întrebări?