

CURS 01 – FP

Mediul de dezvoltare **Code::Blocks**:

<https://www.codeblocks.org/downloads/binaries/>

Varianta care conține și compilator:

codeblocks-20.03mingw-setup.exe

Limbajul C a fost creat în anii '70 (Brian Kernighan și Dennis Ritchie).

Limbajul C este un limbaj de nivel înalt.

Limbajul C este un limbaj compilat.

Tipuri de date primitive/simple/nestructurate:

1. void = tip vid (fără tip)

2. numerice:

a. tipuri întregi

- cu semn: char, short int, int, long int, long long int
- fără semn: unsigned + tip de date cu semn

b. tipuri reale ("cu virgulă") -> diferă prin precizie (standardul IEEE 754)

Tipuri de date întregi (pe b biți):

a. fără semn => toți cei b biți sunt biți de valoare

Exemplu: unsigned char

unsigned char => 1 octet fără semn => 8 biți =>

- minim = $00000000_{(2)} = 0$
- maxim = $11111111_{(2)} = 1*2^0 + 1*2^1 + \dots + 1*2^7 = 1 + 2^1 + \dots + 2^7 = 2^8 - 1 = 255 = 2^b - 1$

Progresie geometrică: fiecare termen b_1, b_2, b_3, \dots se obține înmulțind termenul precedent cu o constantă numită rație q !

$$S_n = \frac{b_1(q^n - 1)}{q - 1}$$

b. cu semn => primul bit este bit de semn (0 = pozitiv și 1 = negativ) și b-1 biți de valoare

Exemplu: char

char => 1 octet cu semn => 1 bit de semn și 7 biți de valoare =>

- minim = $-2^{b-1} = -128$
- maxim = $2^{b-1} - 1 = 127$

Tipuri de date derivate/compuse/structurate:

- tablouri (unidimensionale, bidimensionale etc.)
- șiruri de caractere
- enumerări
- structuri
- uniuni
- pointeri

Variabile

Variabilele trebuie declarate înainte de a fi utilizate și, eventual, inițializate (altfel, vor conține valori aleatorii/reziduale)!

Exemple:

```
int a, b = 20, c;
```

`float x;`

O variabilă constă din:

- nume
- tip de date al valorii memorate/stocate
- valoarea curentă (la un moment dat)
- adresa de memorie unde este stocată

Constante

Se declară, de obicei, global și numele sunt formate din litere mari.

Exemplu:

`const int MAX = 10;`

Operatorii limbajului C

Expresie = operanzi, operatori și paranteze rotunde

Exemplu:

Expresia $s = a + b + c$ conține operanzii a , b și c , precum și operatorii $=$ și $+$.

Un operator se caracterizează prin:

- **aritatea** = numărul de operanzi (unari, binari și ternari)
- **prioritatea** = ordinea de efectuare a operațiilor
Exemplu: $2 + 3 * 4 = 2 + 12 = 14$
- **asociativitatea** = ordinea efectuării operațiilor în cazul unor operatori cu priorități egale
Exemplu: $2 + 3 + 4 = (2 + 3) + 4 = 5 + 4 = 9$ (asociativitatea de la stânga spre dreapta)

1. operatorii aritmetici: +, -, /, %

+5, -7 => + și - sunt operatori unari

5+7, x-10 => + și - sunt operatori binari

a/b = împărțire "cu virgulă", dacă cel puțin un operand este de tip real

a/b = "câtul" împărțirii întregi, dacă ambii operanzi sunt de tip întreg, astfel: se calculează $|a|/|b|$ (unde $|a|$ = modul sau valoare absolută) și se adaugă semnul conform regulii semnelor

Exemplu:

```
int a = 7, b = 10;
```

```
float x;
```

```
x = (a+b)/2;
```

Variabila x va conține valoarea 8.000000!!!!

```
int a = 7, b = 10;
```

```
float x;
```

```
x = (a+b)/2.0;
```

Variabila x va conține valoarea 8.500000!!!!

```
int a = 7, b = 2;
```

```
float x;
```

```
x = (float)a/b; //=>x = 3.500000
```

Conversie explicită folosind operatorul cast: (tip de date temporar)!

Variante corecte:

```
x = (float)a/b;
```

```
x = a/(float)b;
```

```
x = (float)a/(float)b;
```

Varianta incorectă:

$x = (\text{float})(a/b); // \Rightarrow x = 3.000000$ deoarece se efectuează prima dată $a/b = 3$

$a\%b$ = se poate folosi doar pentru operanzi de tip întreg și furnizează **valoarea modulo a împărțirii lui a la b (ci nu restul!!!)**, conform următoarelor formule:

$$a = (a/b)*b + a\%b \Rightarrow a\%b = a - (a/b)*b$$

Observație:

Restul r al împărțirii lui a la b este cuprins între 0 și $|b|-1$!

Exemplu:

$$-7 / 2 = -3$$

$$-7 \% 2 = -1$$

$$\text{Explicație: } -7 = (-7/2)*2 + (-7\%2) \Rightarrow -7 = (-3)*2 + (-7\%2) \Rightarrow -7\%2 = -1$$

Corect (matematic) ar fi:

$$-7 / 2 = -4$$

$$-7 \% 2 = 1$$

Soluție: dacă $r < 0$, atunci $r = r + |b|$

Testarea parității unui număr

GREȘIT	CORECT
<pre>int a = -11; if(a % 2 == 1) printf("Impar"); else printf("Par");</pre>	<pre>int a = -11; if(a % 2 != 0) printf("Impar"); else printf("Par");</pre>

2. operatorii relaționali: <, >, <=, >=, == (egalitate), != (diferit)

Orice valoare egală cu 0 se asimilează cu False, iar orice valoare nenulă se asimilează cu True!

Orice expresie care conține operatorii relaționali va avea fie valoarea 0 (fals), fie valoarea 1 (adevărat) după evaluare.

Exemplu:

```
int a = 11, b = -20, r;
```

```
//variabila r ia valoarea de adevăr a expresiei a <= b
```

```
r = a <= b;
```

Variabila r va conține valoarea 0.

Testarea apartenenței lui b la intervalul [a, c]

GREȘIT	CORECT
<pre>int a = 110, b = 20, c = 200; //conditia se evalueaza de la stanga la dreapta, astfel: //1. 110 <= 20 => 0 //2. 0 <= 20 => 1 if (a <= b <= c) printf("%d este cuprins intre %d si %d\n", b, a, c); else printf("%d NU este cuprins intre %d si %d\n", b, a, c);</pre>	<pre>int a = 110, b = 20, c = 200; if (a <= b && b <= c) printf("%d este cuprins intre %d si %d\n", b, a, c); else printf("%d NU este cuprins intre %d si %d\n", b, a, c);</pre>

3. operatorii logici: ! (negare = not), && (și = and), || (sau = or)

NU există true/false în limbajul C, ci valorile nule corespunzătoare tipurilor de date primitive sunt considerate "false" (0 și 0.0), iar orice altă valoare se consideră "true" (1, -7, 213, 3.14).

Orice expresie care conține operatorii logici va avea fie valoarea 0 (fals), fie valoarea 1 (adevărat) după evaluare.

!	"false"	"true"
	"true"	"false"

&&	"false"	"true"
"false"	"false"	"false"
"true"	"false"	"true"

	"false"	"true"
"false"	"false"	"true"
"true"	"true"	"true"

Operatorii logici se evaluează prin scurtcircuitare:

a) într-un șir de && evaluarea se oprește la prima valoare de 0

a) într-un șir de || evaluarea se oprește la prima valoare de 1

Exemplu:

int a = -100;

if(a > 0 && b > 10)...

Nu se mai evaluează deoarece a > 0 este 0,
deci rezultatul este 0 indiferent de b > 10

4. operatorul de conversie explicită (cast): (tip temporar de date)

5. operatorul sizeof: furnizează dimensiunea în octeți a zonei de memorie pe care o ocupă o variabilă sau este necesară pentru un tip de date.

Exemple:

sizeof(char) = 1 (obligatoriu conform standardului limbajului C)

sizeof(int) = 4 (de obicei)

int a = 123;

printf("%d\n", sizeof(a)); //se va afișa valoarea 4 (de obicei)

6. operatorul de atribuire: =

←
variabilă = expresie;

Are asociativitate de la dreapta spre stânga!!!

Tipul de date al rezultatului expresiei trebuie să fie compatibil cu tipul de date al variabilei, altfel se vor realiza conversii implicite (posibil să apară valori incorecte)!

```
int x;  
  
// 17 / 2.0 = 8.5  
x = 17 / 2.0;  
  
//va afișa valoarea 8 din cauza conversiei implicite!!!  
printf("x = %d\n", x);
```

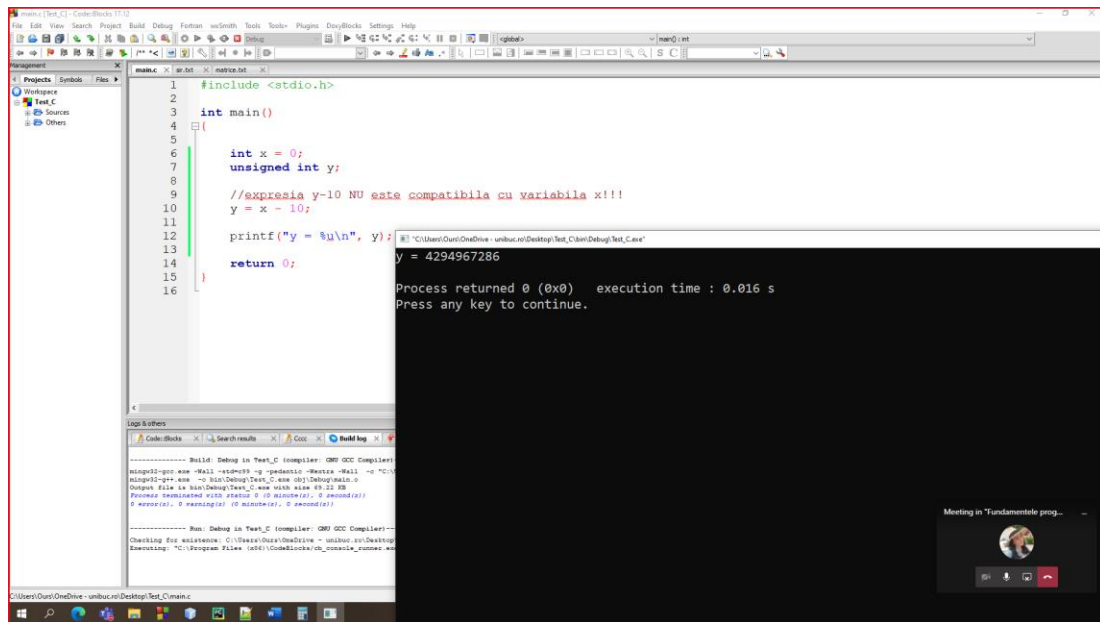
Se permit atribuiri multiple: $x = y = z = 0$;

NU confundați operatorul de atribuire (=) cu operatorul de testare a egalității (==)!!!

Exemplu:

```
int a = 0;  
  
//if(0), indiferent de valoarea initiala a lui a!!!  
if(a = 0)  
    printf("Valoare nula");  
else  
    printf("Valoare nenula");
```

Va afișa "Valoare nenula"!!!



Explicație:

unsigned int: 0, 1, 2,...,4294967293, 4294967294, 4294967295

unsigned int y = -10;

y = 4294967286

01. 4294967295

02. 4294967294

03. 4294967293

04. 4294967292

05. 4294967291

06. 4294967290

07. 4294967289

08. 4294967288

09. 4294967287

10. 4294967286

7. operatorii de atribuire compusă

variabilă = variabilă *op* expresie \Leftrightarrow variabilă *op* = expresie
unde $op \in \{+, -, *, /, \%, \ll, \gg, \&, |, ^\}$.

Exemple:

$x = x + 1 \Leftrightarrow x \text{ += } 1$

$x = x + y + z \Leftrightarrow x \text{ += } y+z$

$y = y * 10 \Leftrightarrow y \text{ *= } 10$

8. operatorii de incrementare/decrementare (++ sau --)

Acești operatori sunt folosiți pentru a crește/scădea cu 1 valoarea unei variabile.

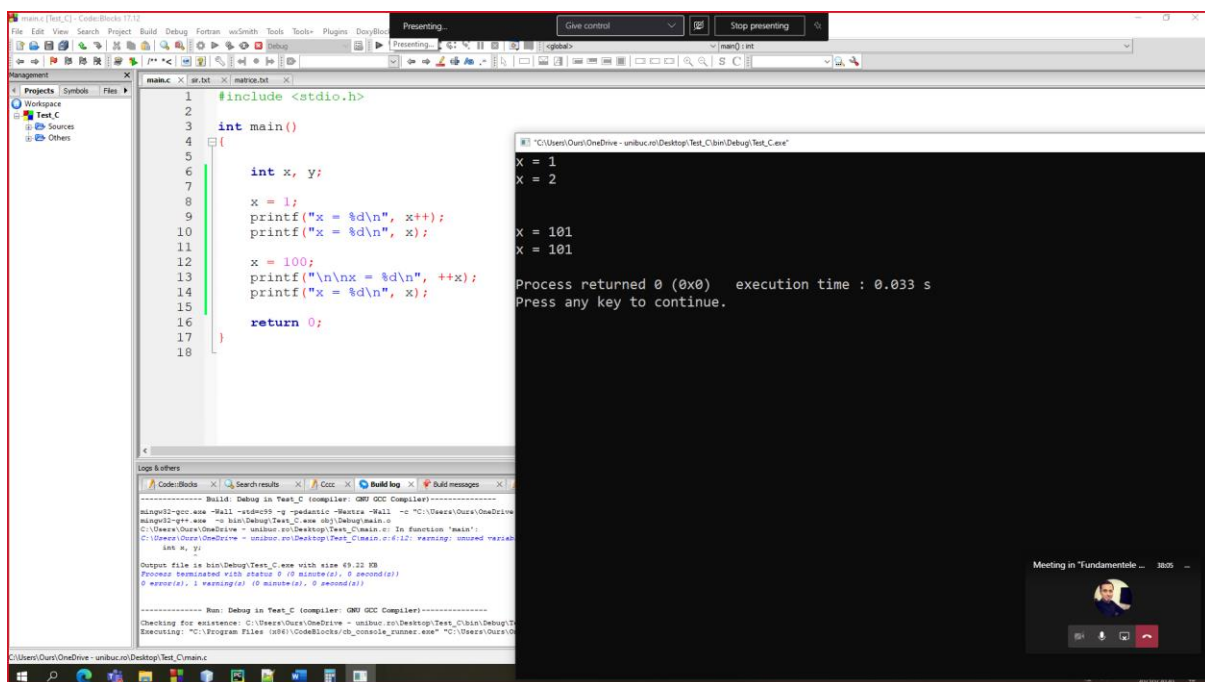
$x++ \Leftrightarrow x = x+1 \Leftrightarrow x += 1$
 $x-- \Leftrightarrow x = x-1 \Leftrightarrow x -= 1$

Atenție, $x++ \neq x+1!!!$ Expresia $x++$ conduce la modificare valorii variabilei x (crește cu 1), în timp ce expresia $x+1$ nu modifică valoarea lui $x!!!$

Operatorii de incrementare/decrementare au două forme:

1. **forma prefixată (++variabilă, --variabilă):** mai întâi se modifică valoarea variabilei x și apoi noua sa valoare este utilizată în cadrul expresiei respective;

2. **forma postfixată (variabilă++, variabilă--):** mai întâi se utilizează în expresie valoarea curentă a variabilei x , iar apoi aceasta se va modifica.



```
#include <stdio.h>

int main()
{
    int x, y;

    x = 1;
    printf("x = %d\n", x++);
    printf("x = %d\n", x);

    x = 100;
    printf("\n\nx = %d\n", ++x);
    printf("x = %d\n", x);

    return 0;
}
```

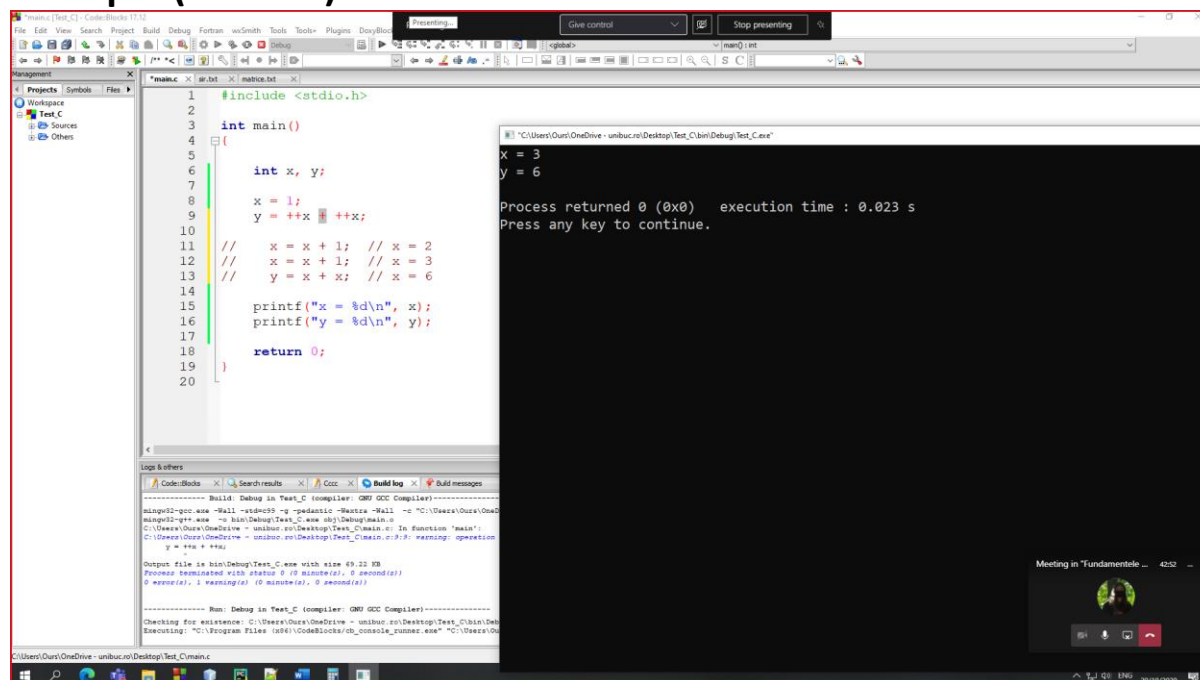
Output:

```
x = 1
x = 2

x = 101
x = 101

Process returned 0 (0x0)   execution time : 0.033 s
Press any key to continue.
```

Exemplu (interviu):



```
1 #include <stdio.h>
2
3 int main()
4 {
5     int x, y;
6     x = 1;
7     y = 6;
8     ++x;
9     ++y;
10    // x = x + 1; // x = 2
11    // y = y + 1; // y = 7
12    printf("x = %d\n", x);
13    printf("y = %d\n", y);
14    return 0;
15 }
```

Process returned 0 (0x0) execution time : 0.023 s
Press any key to continue.

9. operatorul condițional (?:)

expresie_logică ? expresie_1 : expresie_2

- Se evaluează expresia logică și, dacă aceasta este adevărată, se evaluează expresie_1 și operatorul furnizează valoarea obținută, altfel se evaluează expresie_2 și operatorul furnizează valoarea sa.
- Exemplu (maximul dintre două numere): $vmax = x > y ? x : y$;
- Este singurul operator ternar din limbajul C.

10. operatorul virgulă

- este folosit pentru a scrie mai multe expresii pe o singură linie
- se evaluează de la stânga la dreapta, valoarea sa fiind egală cu ultima valoare obținută

