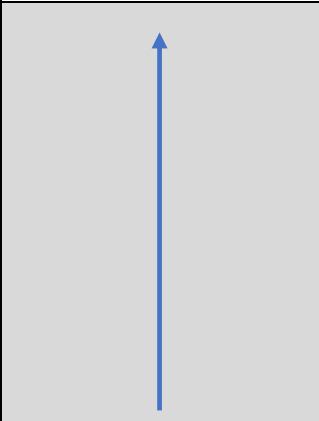


CURS 05 – PP

ZONELE DE MEMORIE ASOCIATE UNUI PROGRAM

CLASE DE MEMORARE

Adresa mare		Argumentele liniei de comandă și variabilele de mediu
	STACK	Variabilele locale Parametrii funcțiilor
	Memoria liberă	
	HEAP	Variabilele alocate dinamic
	Date neinițializate (BSS)	Variabilele statice locale și variabilele globale neinițializate
	Date inițializate (DATA)	Variabilele statice locale și variabilele globale inițializate
Adresa mică	Zona de text/cod	Instrucțiunile programului în cod mașină (read-only)

BSS = Block Started by Symbol

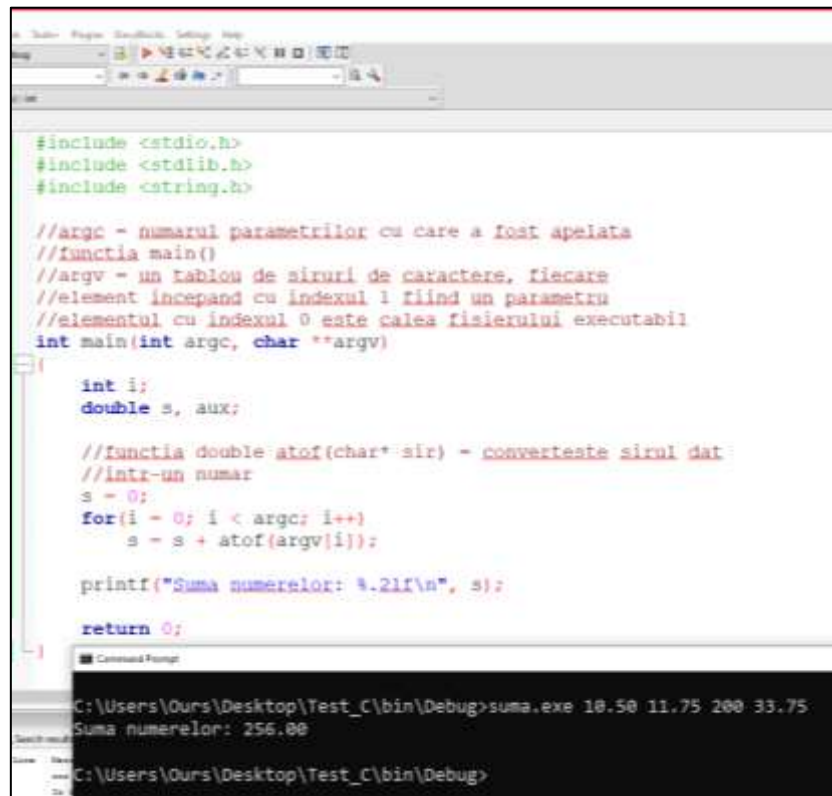
PARAMETRII FUNCȚIEI MAIN()

```
#include <stdio.h>

//argc = numarul parametrilor cu care a fost apelata
//functia main()
//argv = un tablou de siruri de caractere, fiecare
//element incepand cu indexul 1 fiind un parametru
//elementul cu indexul 0 este calea fisierului executabil
int main(int argc, char **argv)
{
    int i;

    printf("Numarul de parametri ai functiei main(): %d\n", argc);
    printf("Parametrii functiei main():\n");
    for(i = 0; i < argc; i++)
        printf("%s\n", argv[i]);

    return 0;
}
```



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//argc - numarul parametrilor cu care a fost apelata
//functia main()
//argv - un tablou de siruri de caractere, fiecare
//element incepand cu indexul 1 fiind un parametru
//elementul cu indexul 0 este calea fisierului executabil
int main(int argc, char **argv)
{
    int i;
    double s, aux;

    //functia double atof(char* sir) - converteste sirul dat
    //intr-un numar
    s = 0;
    for(i = 0; i < argc; i++)
        s = s + atof(argv[i]);

    printf("Suma numerelor: %.2lf\n", s);

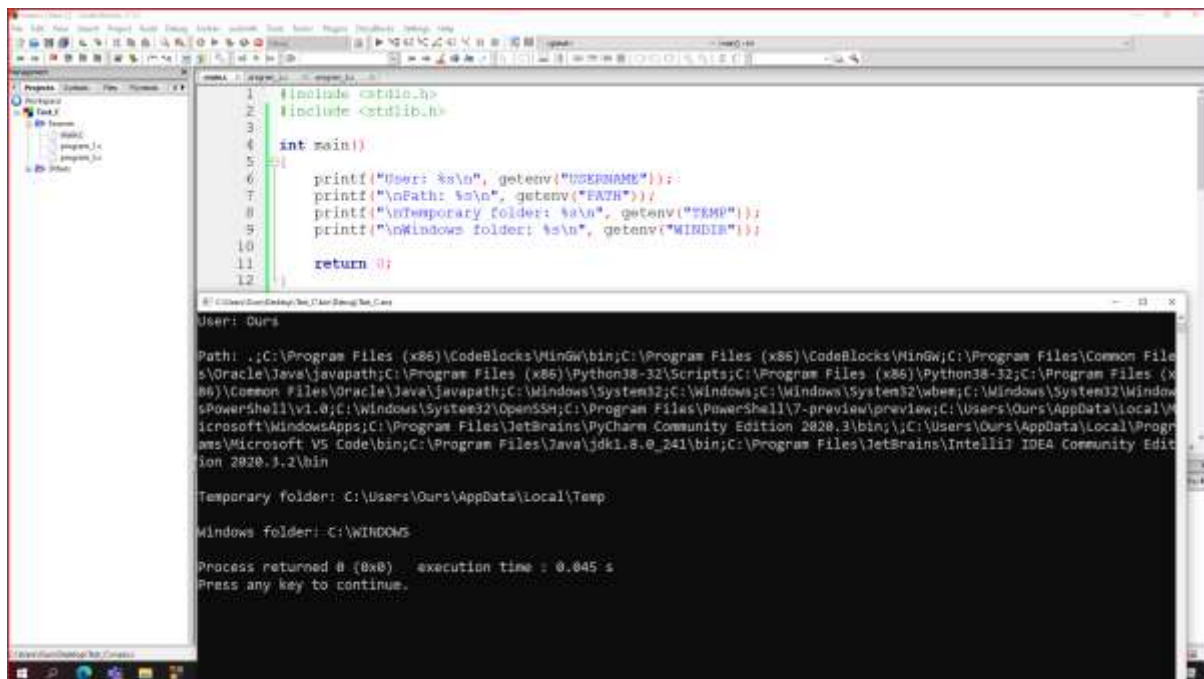
    return 0;
}
```

Command Prompt

C:\Users\Ours\Desktop\Test_C\bin\Debug>suma.exe 10.50 11.75 200 33.75
Suma numerelor: 256.00

C:\Users\Ours\Desktop\Test_C\bin\Debug>

VARIABLELE DE MEDIU (ENVIRONMENT VARIABLES)



```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("User: %s\n", getenv("USERNAME"));
    printf("\nPath: %s\n", getenv("PATH"));
    printf("\nTemporary folder: %s\n", getenv("TEMP"));
    printf("\nWindows folder: %s\n", getenv("WINDIR"));

    return 0;
}
```

Command Prompt

User: Ours

Path: .;C:\Program Files (x86)\CodeBlocks\MinGW\bin;C:\Program Files (x86)\CodeBlocks\MinGW\bin;C:\Program Files\Common File
s\Oracle\Java\javapath;C:\Program Files (x86)\Python38-32\Scripts;C:\Program Files (x86)\Python38-32;C:\Program Files (x
86)\Common Files\Oracle\Java\javapath;C:\Windows\System32;C:\Windows\System32\wbem;C:\Windows\System32\Window
sPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Program Files\PowerShell\7-preview\preview;C:\Users\Ours\AppData\Local\W
icrosoft\WindowsApps;C:\Program Files\JetBrains\PyCharm Community Edition 2020.3\bin;;C:\Users\Ours\AppData\Local\Progr
ams\Microsoft VS Code\bin;C:\Program Files\Java\jdk1.8.0_241\bin;C:\Program Files\JetBrains\IntelliJ IDEA Community Edit
ion 2020.3.2\bin

Temporary folder: C:\Users\Ours\AppData\Local\Temp

Windows folder: C:\WINDOWS

Process returned 0 (0x0) execution time : 0.045 s
Press any key to continue.

Clasele de memorare induc următoarele caracteristici unei variabile:

- a) zona de memorie în care va fi alocată;
- b) durata de viață;
- c) modalitatea de inițializare;
- d) domeniul de vizibilitate (variabilă locală / globală).

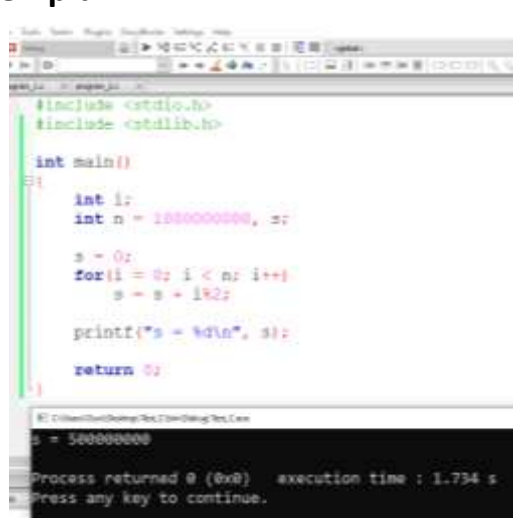
1. Clasa auto

- implicită pentru variabilele locale și parametrii funcțiilor;
- variabilele se alocă în zona de stivă (STACK);
- variabilele NU sunt inițializate automat (au valori reziduale);
- durată de viață a unei variabile este egală cu durata de viață a blocului în care a fost declarată;
- variabila este vizibilă doar în blocul în care a fost declarată.

2. Clasa register

- cere procesorului ca variabila respectivă să aibă acces rapid (alocare în regiștrii procesorului);
- dacă procesorul refuză să-i acorde variabilei acces rapid, atunci variabila respectivă rămâne în clasa auto;
- variabilele NU sunt inițializate automat (au valori reziduale);
- **variabilele din clasa register NU au adrese de memorie, deci nu pot fi declarate tablouri în această clasă de memorare;**
- durată de viață a unei variabile este egală cu durata de viață a blocului în care a fost declarată;
- variabila este vizibilă doar în blocul în care a fost declarată.

Exemplu:



```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i;
    int n = 100000000, s;

    s = 0;
    for(i = 0; i < n; i++)
        s = s + i*i;

    printf("s = %d\n", s);

    return 0;
}
```

s = 500000000

Process returned 0 (0x0) execution time : 1.734 s



```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    register int i;
    int n = 100000000, s;

    s = 0;
    for(i = 0; i < n; i++)
        s = s + i*i;

    printf("s = %d\n", s);

    return 0;
}
```

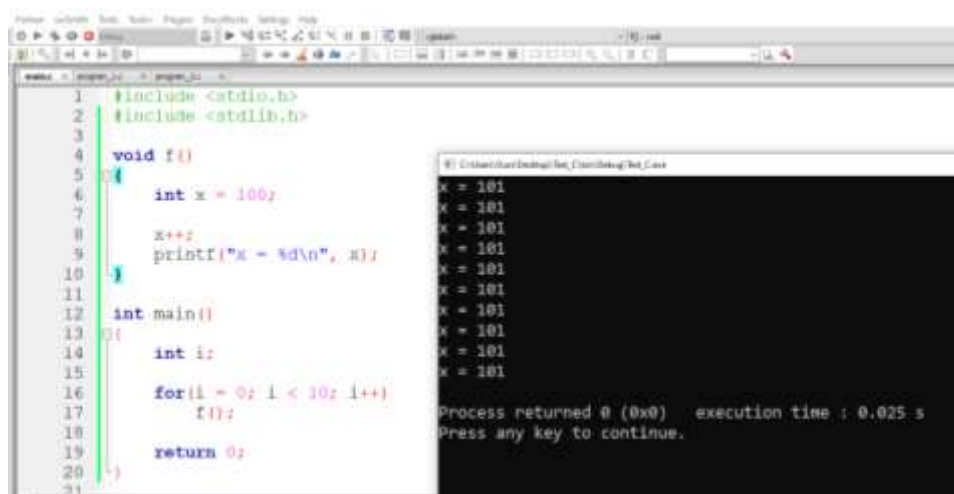
s = 500000000

Process returned 0 (0x0) execution time : 1.202 s

3. Clasa static

- o variabilă locală are aceeași adresă pe durata întregului program;
- variabilele se alocă în zona DATA dacă sunt inițializate explicit sau în zona BSS dacă nu sunt inițializate explicit;
- variabilele inițializate explicit sunt inițializate o singură dată, la începutul programului;
- variabilele neinițializate explicit sunt inițializate implicit cu valoarea 0, o singură dată, la începutul programului;
- durată de viață a unei variabile statice este egală cu durata de viață a programului;
- o variabilă locală statică este vizibilă doar în blocul în care a fost declarată;
- o variabilă globală statică este vizibilă doar în fișierul (programul) în care a fost declarată (variabila globală devine locală fișierului respectiv).

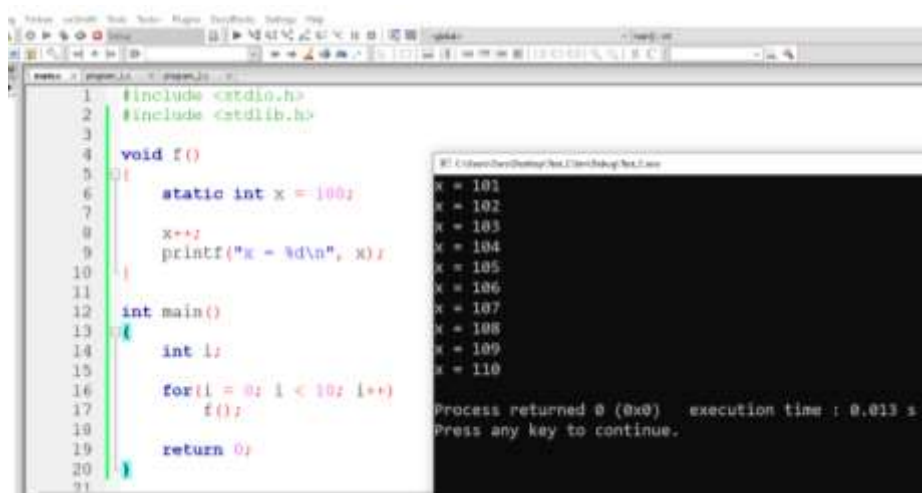
Exemple:



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void f()
5 {
6     int x = 100;
7
8     x++;
9     printf("x = %d\n", x);
10 }
11
12 int main()
13 {
14     int i;
15
16     for(i = 0; i < 10; i++)
17         f();
18
19     return 0;
20 }
```

Output:

```
x = 101
x = 101
x = 101
x = 101
x = 101
x = 101
x = 101
x = 101
x = 101
x = 101
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.
```



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void f()
5 {
6     static int x = 100;
7
8     x++;
9     printf("x = %d\n", x);
10 }
11
12 int main()
13 {
14     int i;
15
16     for(i = 0; i < 10; i++)
17         f();
18
19     return 0;
20 }
```

Output:

```
x = 101
x = 102
x = 103
x = 104
x = 105
x = 106
x = 107
x = 108
x = 109
x = 110
Process returned 0 (0x0)   execution time : 0.013 s
Press any key to continue.
```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void f()
5 {
6     static int x;
7
8     x++;
9     printf("x = %d\n", x);
10 }
11
12 int main()
13 {
14     int i;
15
16     for(i = 0; i < 10; i++)
17         f();
18
19     return 0;
20 }
21

```

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

4. Clasa extern

- permite accesarea unei variabile definite în alt fișier sursă (program) aflat în același director;
- o variabilă sau o funcție poate fi **declarată** în mai multe fișiere sursă, dar poate fi **definită** într-un singur fișier sursă.

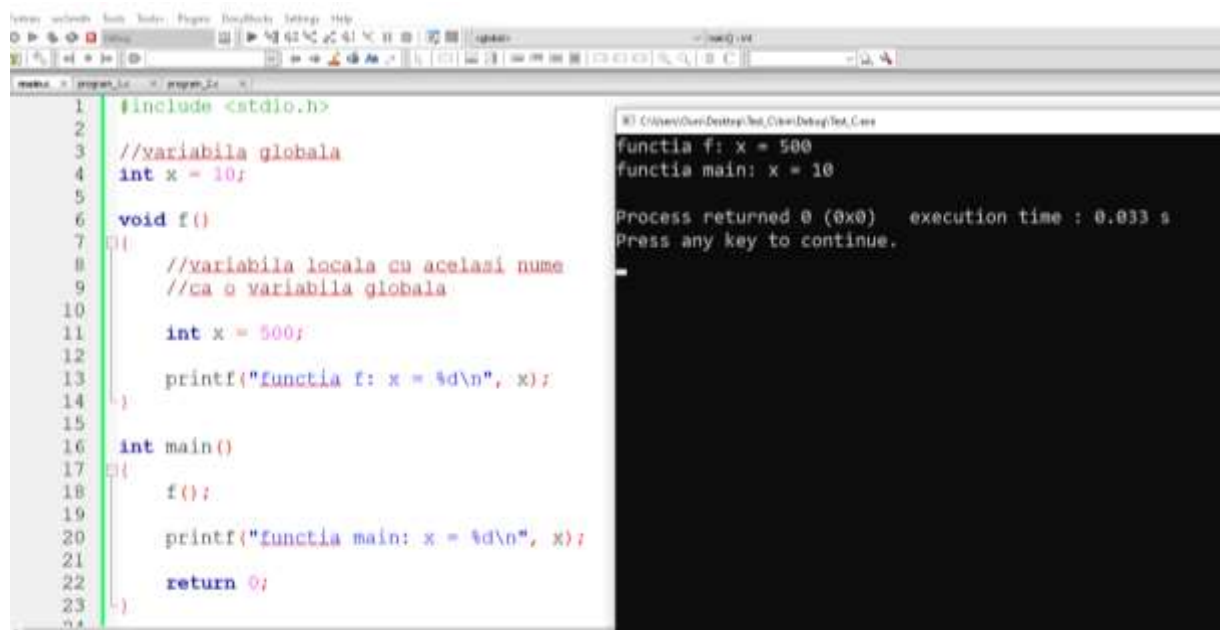
program_1.c	program_2.c
<pre> #include <stdlib.h> //definirea unei variabile x int x; //declararea unei functii //afisare care nu are corp si //este externa, deci trebuie //sa fie definita in alt fisier //din acelasi director extern void afisare(); //functia main trebuie sa fie //definita intr-un singur //fisier int main() { x = 5; afisare(); return 0; } </pre>	<pre> #include <stdio.h> //declararea unei variabile x, //deci trebuie sa fie definita //in alt fisier din acelasi //director extern int x; void afisare() { printf("x = %d\n", x); } </pre>

Observație: Dacă adăugăm variabilei `int x` modificatorul `static`, atunci ea va deveni locală fișierului sursă `program_1.c`, deci nu va mai fi vizibilă în `program_2.c` și va fi generată o eroare la compilare!!!

O **variabilă locală** este o variabilă definită în cadrul unui bloc de instrucțiuni și va fi considerată, în mod implicit, în clasa de memorare auto.

O **variabilă globală** este o variabilă definită în afara oricărei funcții, este vizibilă în toate funcțiile din fișierele sursă din directorul respectiv și are durata de viață egală cu durata întregului program.

Dacă într-un bloc de instrucțiuni sunt vizibile o variabilă locală și o variabilă globală cu același nume, atunci va fi luată în considerare variabila locală!!!



```
1 #include <stdio.h>
2
3 //variabila globala
4 int x = 10;
5
6 void f()
7 {
8     //variabila locala cu acelasi nume
9     //ca o variabila globala
10
11     int x = 500;
12
13     printf("functia f: x = %d\n", x);
14 }
15
16 int main()
17 {
18     f();
19
20     printf("functia main: x = %d\n", x);
21
22     return 0;
23 }
```

Output:

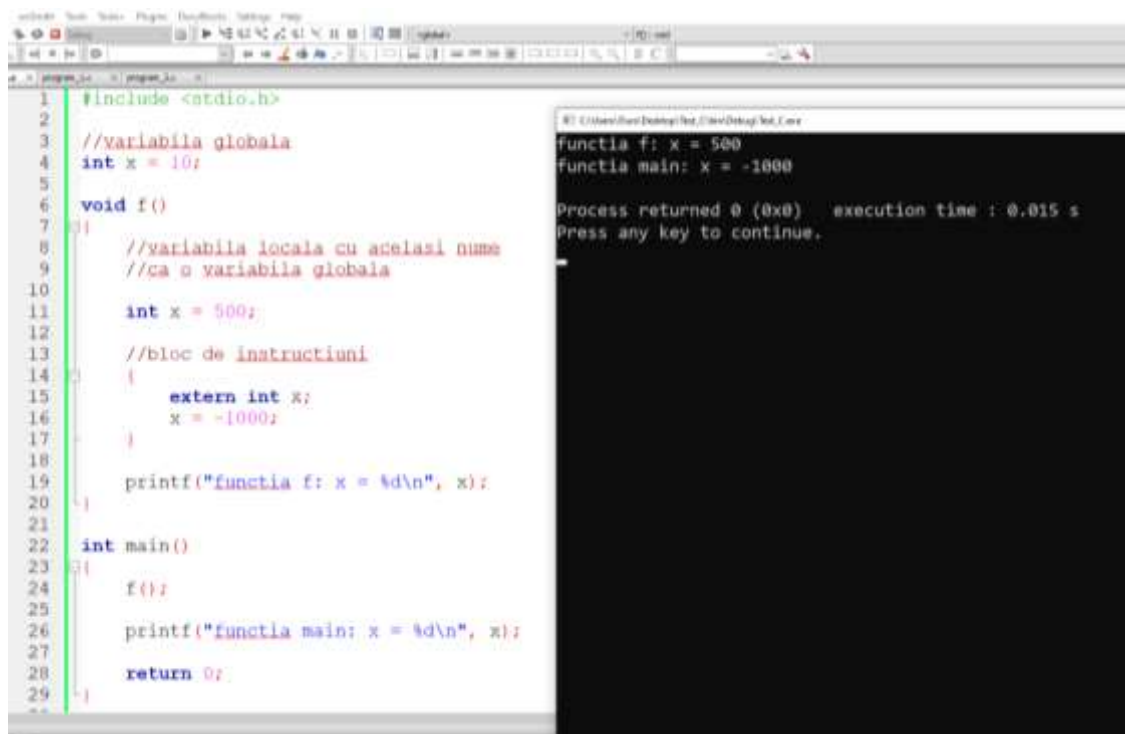
```
functia f: x = 500
functia main: x = 10

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

Întrebare de interviu:

Cum putem accesa o variabilă globală într-o funcție în care este definită și o variabilă locală cu același nume?

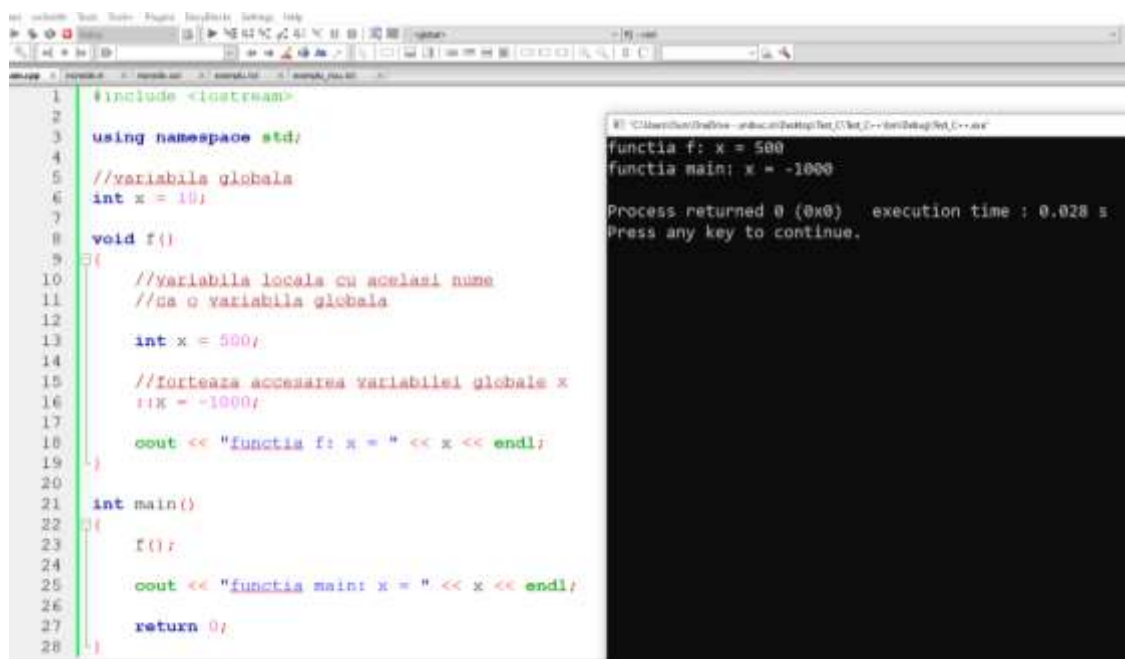
Limbajul C:



```
1 #include <stdio.h>
2
3 //variabila globala
4 int x = 10;
5
6 void f()
7 {
8     //variabila locala cu acelasi nume
9     //ca o variabila globala
10
11     int x = 500;
12
13     //bloc de instructiuni
14     {
15         extern int x;
16         x = -1000;
17     }
18
19     printf("functia f: x = %d\n", x);
20 }
21
22 int main()
23 {
24     f();
25
26     printf("functia main: x = %d\n", x);
27
28     return 0;
29 }
```

functia f: x = 500
functia main: x = -1000
Process returned 0 (0x0) execution time : 0.015 s
Press any key to continue.

Limbajul C++:



```
1 #include <iostream>
2
3 using namespace std;
4
5 //variabila globala
6 int x = 10;
7
8 void f()
9 {
10     //variabila locala cu acelasi nume
11     //ca o variabila globala
12
13     int x = 500;
14
15     //fortezza accesarea variabilei globale x
16     :X = -1000;
17
18     cout << "functia f: x = " << x << endl;
19 }
20
21 int main()
22 {
23     f();
24
25     cout << "functia main: x = " << x << endl;
26
27     return 0;
28 }
```

functia f: x = 500
functia main: x = -1000
Process returned 0 (0x0) execution time : 0.028 s
Press any key to continue.

ȘIRURI DE CARACTERE

Șir de caractere = o succesiune de caractere care se termină cu caracterul NUL ('\\0') = un tablou unidimensional cu elemente de tip char

Exemplu:

```
char s[101], t[21]; //trebuie alocat un element în plus
                    //pentru '\\0'
```

```
char s[] = "testare";
char s[] = {'t', 'e', 's', 't', 'a', 'r', 'e', '\\0'};
```

0	1	2	3	4	5	6	7
't'	'e'	's'	't'	'a'	'r'	'e'	'\\0'

1 caracter = 1 octet = codul ASCII al caracterului respectiv

Constante de tip char: 'A', 'b', '@', '?'

Constante de tip șir de caractere: "A", "bac", "@", "@gmail.com"

Observație: 'A' ≠ "A"

'A'	"A"	
65	0	1
1 octet	'A' -> 65	'\\0' -> 0
	2 octeți	

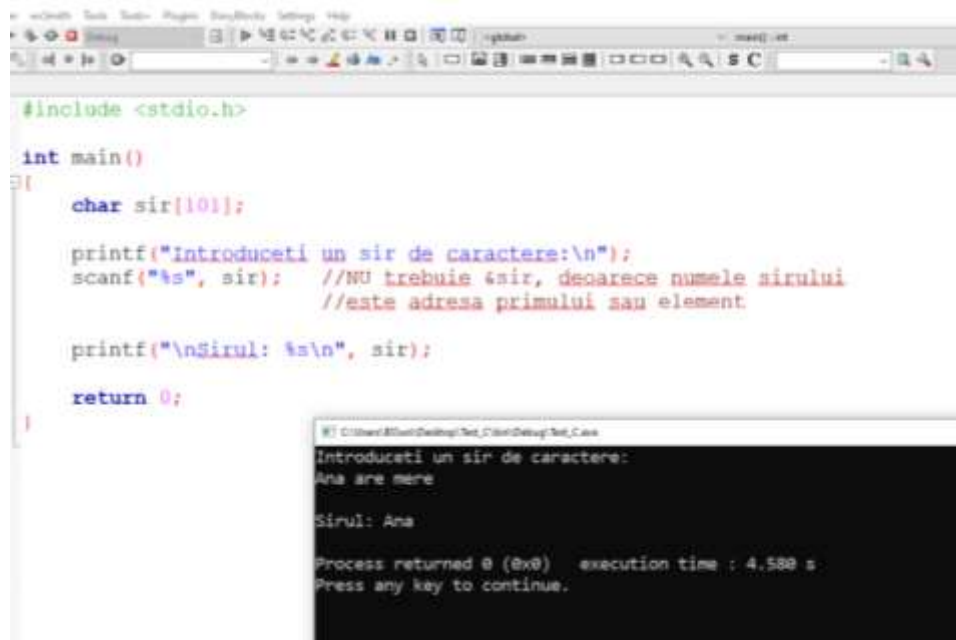
Observație: 'a' > 'A'

Transformarea unui caracter într-un șir:

```
char sir[2];
sir[0] = 'A';
sir[1] = '\\0';
```


Citirea șirurilor de caractere de la tastatură:

- folosind funcția `scanf` cu specificatorul `%s`, dar citirea se va opri la primul separator (de obicei, spațiu)



The screenshot shows a C program in a code editor and its execution in a terminal. The code defines a character array `sir` of size 101, prompts the user to enter a string, and uses `scanf("%s", sir)` to read the input. The terminal output shows the user entering "Ana are mere" and the program printing "Sirul: Ana".

```
#include <stdio.h>

int main()
{
    char sir[101];

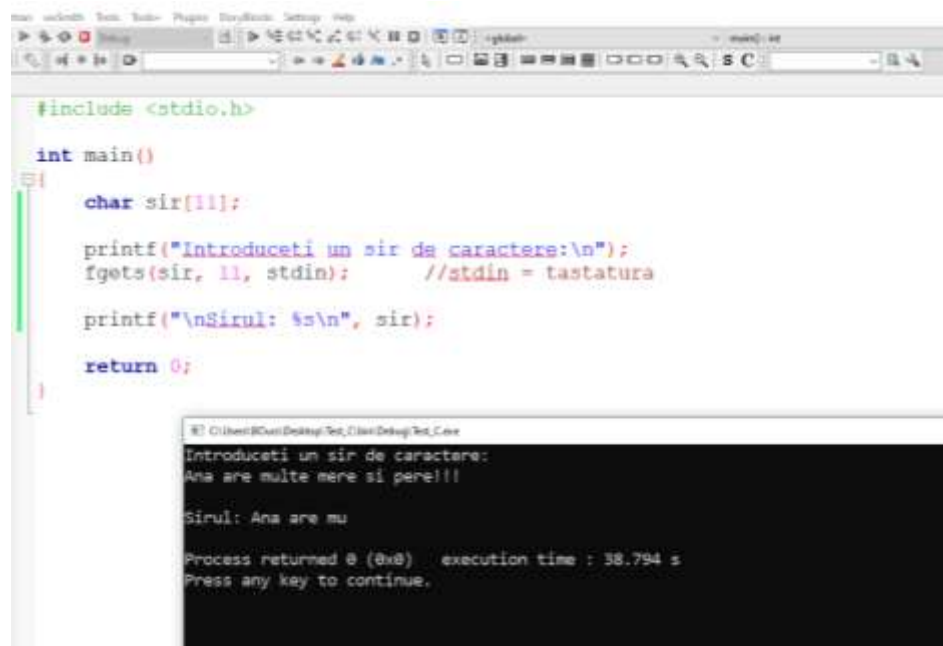
    printf("Introduceti un sir de caractere:\n");
    scanf("%s", sir); //NU trebuie %sir, deoarece numele sirului
                    //este adresa primului sau element

    printf("\nSirul: %s\n", sir);

    return 0;
}
```

Introduceti un sir de caractere:
Ana are mere
Sirul: Ana
Process returned 0 (0x0) execution time : 4.580 s
Press any key to continue.

- folosind funcția `fgets`:
`char* fgets(char *șir, int nr_max_caractere, stdin)`



The screenshot shows a C program in a code editor and its execution in a terminal. The code defines a character array `sir` of size 11, prompts the user to enter a string, and uses `fgets(sir, 11, stdin)` to read the input. The terminal output shows the user entering "Ana are multe mere si pere!!!" and the program printing "Sirul: Ana are mu".

```
#include <stdio.h>

int main()
{
    char sir[11];

    printf("Introduceti un sir de caractere:\n");
    fgets(sir, 11, stdin); //stdin = tastatura

    printf("\nSirul: %s\n", sir);

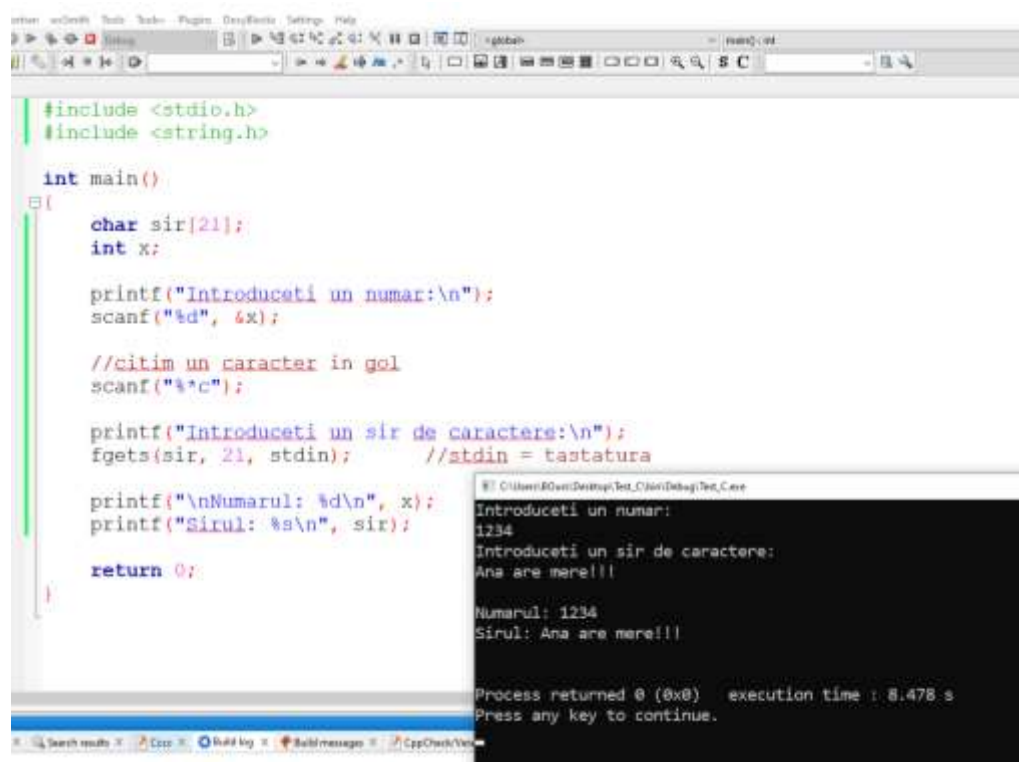
    return 0;
}
```

Introduceti un sir de caractere:
Ana are multe mere si pere!!!
Sirul: Ana are mu
Process returned 0 (0x0) execution time : 38.794 s
Press any key to continue.

Atenție, funcția `fgets` păstrează la sfârșitul șirului caracterul `'\n'` corespunzător tastei Enter dacă nu s-a atins lungimea maximă a șirului, deci trebuie să îl eliminăm noi!!!

```
if(sir[strlen(sir)-1] == '\n')
    sir[strlen(sir)-1] = '\0';
```

Atenție, în cazul în care citim un număr de la tastatură, caracterul `'\n'` corespunzător tastei Enter va rămâne în buffer-ul tastaturii, deci, dacă vom citi imediat un șir de caractere, acesta va fi format doar din acel caracter `'\n'` !!! Pentru a evita această problemă, vom efectua o citire în gol a unui caracter, după ce vom citi numărul!



The screenshot shows a C++ IDE with the following code in the editor:

```
#include <stdio.h>
#include <string.h>

int main()
{
    char sir[21];
    int x;

    printf("Introduceti un numar:\n");
    scanf("%d", &x);

    //citim un caracter in gol
    scanf("%c");

    printf("Introduceti un sir de caractere:\n");
    fgets(sir, 21, stdin); //stdin = tastatura

    printf("\nNumarul: %d\n", x);
    printf("Sirul: %s\n", sir);

    return 0;
}
```

The output window shows the following execution:

```
Introduceti un numar:
1234
Introduceti un sir de caractere:
Ana are mere!!!

Numarul: 1234
Sirul: Ana are mere!!!

Process returned 0 (0x0)   execution time : 8.478 s
Press any key to continue.
```

Aceeași problemă apare și dacă vrem să citim de la tastatură două caractere despărțite printr-un spațiu!!!

```
char a, b;
```

```
printf("Introduceti doua caractere:\n");
//citim in gol separatorul dintre cele doua caractere
scanf("%c%c", &a, &b); //scanf("%c %c", &a, &b);
printf("\nCaracterele: %c %c\n", a, b);
```