# Fundamentele programării

# Curs 5

# Funcţii în limbajul C

Elena BAUTU & Dorin-Mircea POPOVICI
{ebautu,dmpopovici}@univ-ovidius.ro

# Cuprins

- Ce este o functie?
  1)Declararea, implementarea si apelul unei functii
  2)Durata de viata v.s. domeniul de accesibilitate
- Transferul parametrilor
- Functii matematice si alte functii
- Functii recursive

# Ce este o functie?

# Ce este o funcţie în limbajul C

- O **funcţie** este o secvenţă de instrucţiuni care au fost grupate sub un singur nume.
  - poate să folosească parametri pentru a primi date de intrare
  - poate să producă un rezultat (denumit valoare de retur)
  - poate produce rezultate multiple, recuperabile prin intermediul parametrilor
- Avantaje:
  - încapsularea şi
  - reutilizarea secvenţelor de program
- Funcţiile pot fi:
  - definite de utilizator(i.e. programator) sau
  - standard (de biblioteca)

# Exemplu

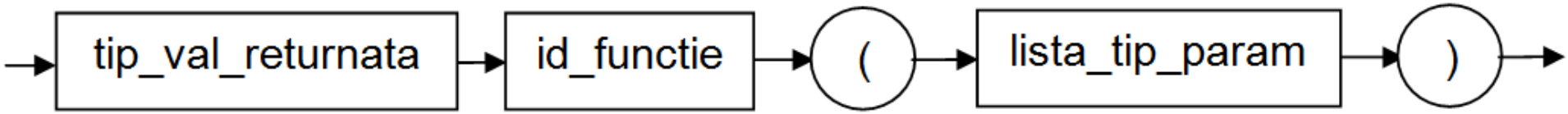
Cel mai ades vedem funcţia principală, main(), apelată la începutul execuţiei oricărui program C

- Exemplu:
  Scrieti o functie *putere* care primeste doi parametri de tip int ce reprezinta *baza* si *exponentul* si calculeaza si returneaza valoarea baza la puterea exponent. Scrieti apoi un program C care foloseste aceasta functie.

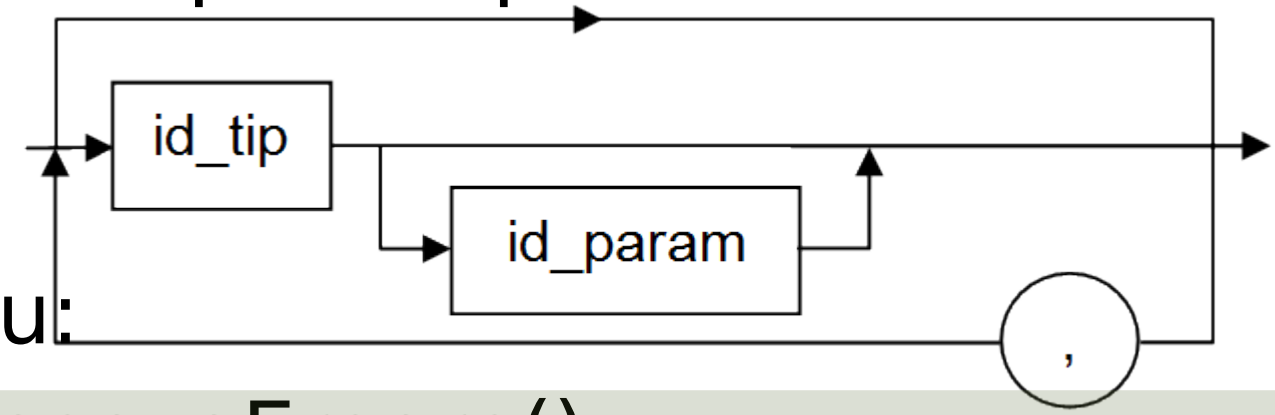
```
//declararea functiei

int main() {
  long int valoare = putere(5,2); //apelul functiei
  printf("5 la puterea 2 este %li\n", valoare);
}

//implementarea functiei
```

# Declararea unei functii

# Declararea unei functii = prototip




- unde lista tipurilor parametrilor este:




- Exemplu:
- void afiseazaEroare();
- int factorial(int);
- float diferenta(float a, float);

# Funcţii definite de utilizator

- funcţia trebuie declarata înainte de utilizare
  - **Declaraţia inseamna precizarea antetului**

- **Antet (prototip)**: precizeaza care sunt
  - **numele**,
  - **tipul parametrilor** şi
  - **tipul de retur** a funcţie

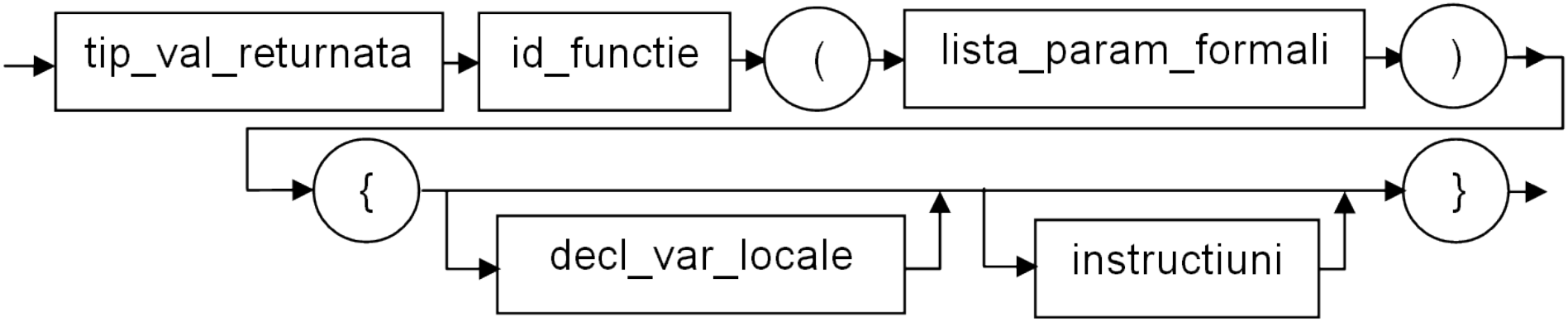
```
tipreturnat nume_functie(lista tipuri parametri formali)
```

- Exemplu:
  - `int suma(int, int);`
  - `long int putere(int, int);`
  - `double radical2(double);`
  - `double sinus(double);`
  - `double ariaCerc(double raza);`

**Apelul unei funcţii:**
nume_functie(lista parametri actuali)
Exemplu: `int s = suma(2, 3);`

# Implementarea unei functii

# Implementarea unei functii




- Exemple:
- void afiseazaEroare();
- ...
- void afiseazaEroare() {
- printf(“Eroare!\n”);
- }

**B – declaratii, ct, fctii, tipuri**

**D – implementarea functiilor declarate in zona B**

# Implementarea unei functii (cont)

```
int factorial(int);
   ...
int factorial(int n) {
   int f=1;
   for (int i=1;i<=n;i++)
       f*=i;
   return f;
}
```

# Implementarea unei functii (cont)

- float diferenta(float a, float b);
- ...
- float diferenta(float a,float b) {
- return a-b;
- }

# Restrictii

- O functie NU poate fi declarata in alta functie!
- O functie NU poate fi implementata in alta functie!

# Utilizarea unei functii

# Apelul unei functii

- Utilizare functie = apel de functie
- Apelul funcţiei se face prin scrierea numelui său şi valorile (**actuale** ale) parametrilor între paranteze rotunde şi separaţi prin virgule.
  - dacă funcţia nu are parametri la definitie, atunci la apel între paranteze nu se scrie nimic.

Exemplu

```
void afiseazaEroare(){
           printf(“Date eronate!!!”);
}
int main(){
int x; scanf(“%i”, &x);
 if(x<0)
  afiseazaEroare();
}
```

- De obicei, apelul de functie are loc in formarea unei expresii care permite in formarea sa operanzi de tipul valorii returnate de functie

- Exemple:

```
printf("Salut!\n");
x=sinus(3.14/3); //60 de grade
d=sqrt(dx*dx+dy*dy);
```

# Apelul unei functii (cont)

```
void afiseazaEroare();
int main() {
int n;
  printf("Introduceti un numar pozitiv:");
  scanf("%d",&n);
  if (n<0) { afiseazaEroare(); return -1; }
  //...
  return 1;
}
void afiseazaEroare() {
  printf(“Eroare!\n”);
}
```

Apelul functiei

**functia main()**

# Apelul unei functii (cont)

```
int factorial(int);
int a,f;
int main() {
  printf("Introduceti un numar pozitiv:");
  scanf("%d”,&a);
  f=factorial(a);
  printf("%d!=%d",a,f);
  return 1;
}
int factorial(int n) {
   int f=1;
   for (int i=1;i<=n;i++)
       f*=i;
   return f;
}
```

**B – declaratii, ct, fctii, tipuri**

a - Parametru efectiv

Apelul functiei

**C – functia main()**

n - Parametru formal

**D – implementarea functiilor declarate in zona B**

# Apelul unei functii (cont)

```
float diferenta(float a, float);
float x,y;
int main() {
   float dif;
   scanf("%f %f",&x,&y);
   dif=diferenta(x,y);
   printf("%f-%f=%f",x,y,dif);
   return 1;
}
float diferenta(float a,float b) {
   return a-b;
}
```

**B – declaratii, ct, fctii, tipuri**

x, y sunt Parametrii efectivi

a,b – sunt Parametrii formali

**C – functia main()**

**D – implementarea functiilor declarate in zona B**

# Exercitiu

- Scrieti o functie care verifica daca ultima cifra a unui numar este 0. Functia sa returneze 1 (cu semnificatia de true) daca daca ultima cifra este 0 si 0 (cu semnificatia de false) daca ultima cifra este diferita de 0.

```
int verifica (int n)
{



}
```

# Exercitiu - solutie

- Scrieti o functie care verifica daca ultima cifra a unui numar este 0. Functia sa returneze 1 (cu semnificatia de true) daca daca ultima cifra este 0 si 0 (cu semnificatia de false) daca ultima cifra este diferita de 0.

```
int verifica (int n)
{
  int este = 0;
  if (n%10 == 0) este = 1;
  return este;
}
```

# Scrieti o functie care calculeaza nr de zerouri cu care se sfarseste un numar.

```
int nrZero(int n); // declaratia functiei

int main()// programul principal
{
  int nr; scanf(“%d”, &nr);
  printf(“%d are %d cifre de 0 la coada”, nr, nrZero(nr)); // apelam functia
  return 0;
}

int nrZero(int n) // definirea (sau implementarea) functiei
{









}
```

# Scrieti o functie care calculeaza nr de zerouri cu care se sfarseste un numar.

```
int nrZero(int n); // declaratia functiei

int main()// programul principal
{
  int nr; scanf(“%d”, &nr);
  printf(“%d are %d cifre de 0 la coada”, nr, nrZero(nr)); // apelam functia
  return 0;
}

int nrZero(int n) // definirea (sau implementarea) functiei
{ int c, contor=0;
  while(n!=0){
     c=n%10;
     if(c==0)
         contor++;
     else
        break;
     n=n/10;
  }
  return contor;
}
```

# Durata de viata
# v.s.
# Domeniul de vizibilitate

# Durata de viata

- O variabila “traieste” (adica este alocata in memorie!!!) atata vreme cat **blocul de memorie** in care ea a fost declarata este activ.
  - Blocul de memorie poate fi delimitat de o pereche de acolade (**variabile locale**)
  - Blocul de memorie poate fi considerat fisierul in care variabila este declarata (**variabile globale**)
- Variabilele globale sunt accesibile in cadrul oricarei functii din program.
  - *Ele sunt **inaccesibile** daca exista in functie variabile locale cu acelasi nume*.
  - Ele pot fi manipulate din orice functie din program

# Domeniu de accesibilitate

- Variabila accesibila: variabila care poate fi identificata unic + durata de viata nu i s-a terminat

O variabila a carei durata de viata nu s-a incheiat este accesibila/vizibila daca poate fi unic identificata.

# Exemplu (cont)

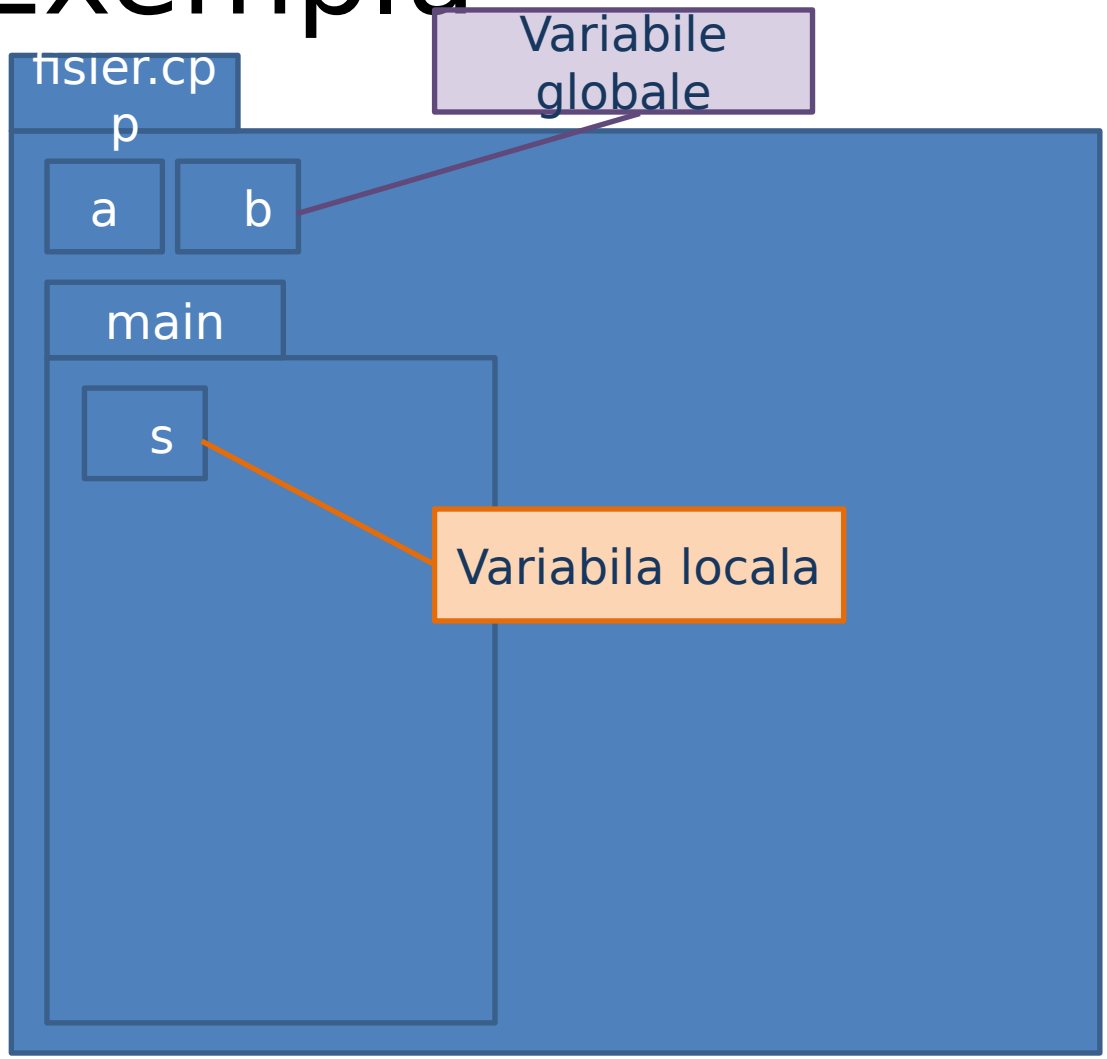
```
#include<stdio.h>
int a = 5, b = 5;
int suma();
int main(){
    printf("1 Global: a = %i, b = %i\n", a, b);
    int s;
    scanf("%i%i", &a, &b);
    printf("2 Global: a = %i, b = %i\n", a, b);

    s = suma(a, b);
    printf("Suma s este %i, iar a si b sunt %i si %i\n", s, a , b);
}

int suma(){
    int c = a+b;
    a = b = 0;
    return c;
}
```

# Exemplu (cont)

```
#include<stdio.h>
int a = 5, b = 5;
int suma();
int main(){
    printf("1 Global: a = %i, b = %i\n", a, b);
    int s;
    int a=10,b=10;
    printf("2 Local: a = %i, b = %i\n", a, b);
    scanf("%i%i", &a, &b);
    printf("3 Local: a = %i, b = %i\n", a, b);

    s = suma();
    printf("Suma s este %i, iar a si b sunt %i si %i\n", s, a , b);
}

int suma(){
    int x = a+b;
    a = b = 0;
    return x;
}
```

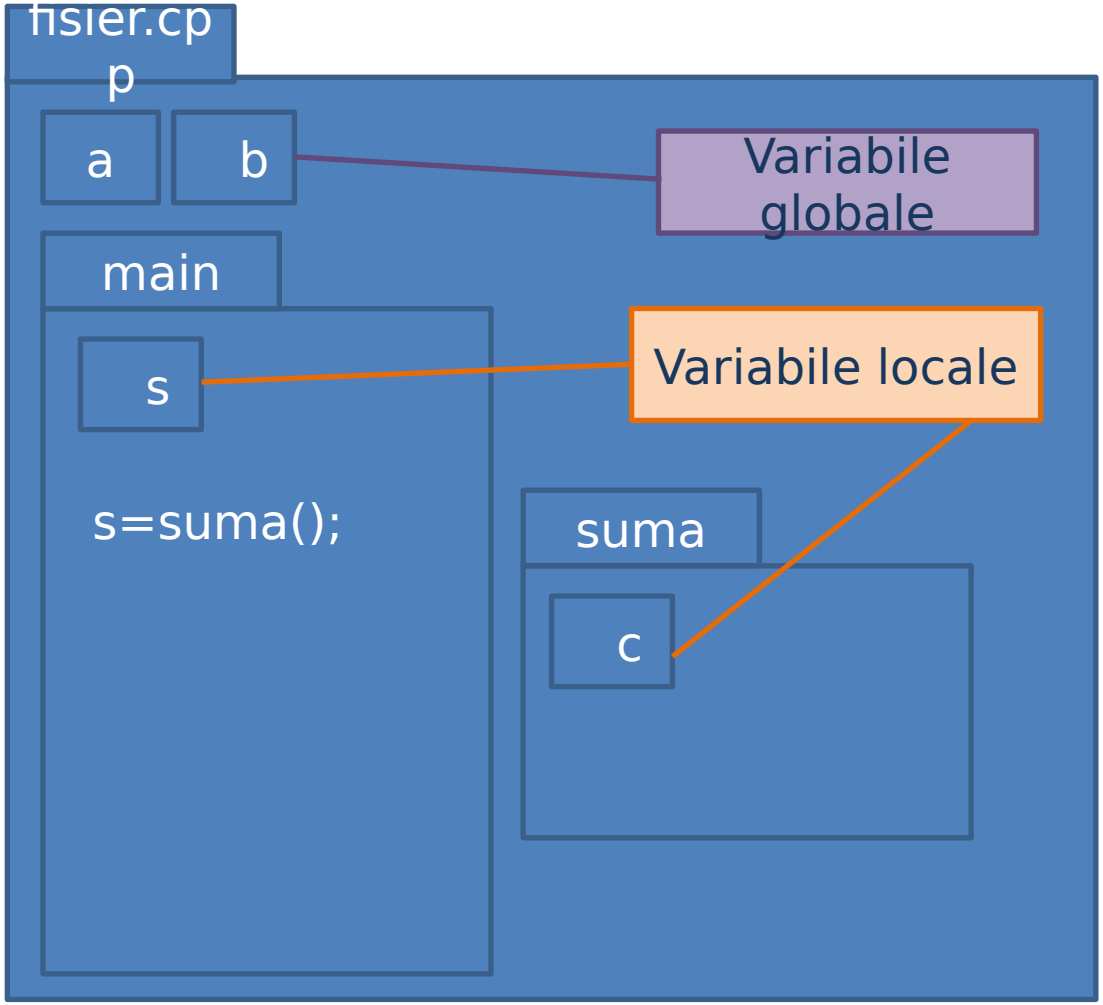
# Exemplu

```
int a,b;
Int main() {
  int s;
  -> a; // citim a
  -> b; // citim b
  s=a+b;
  <- s; // afisam s
}
```



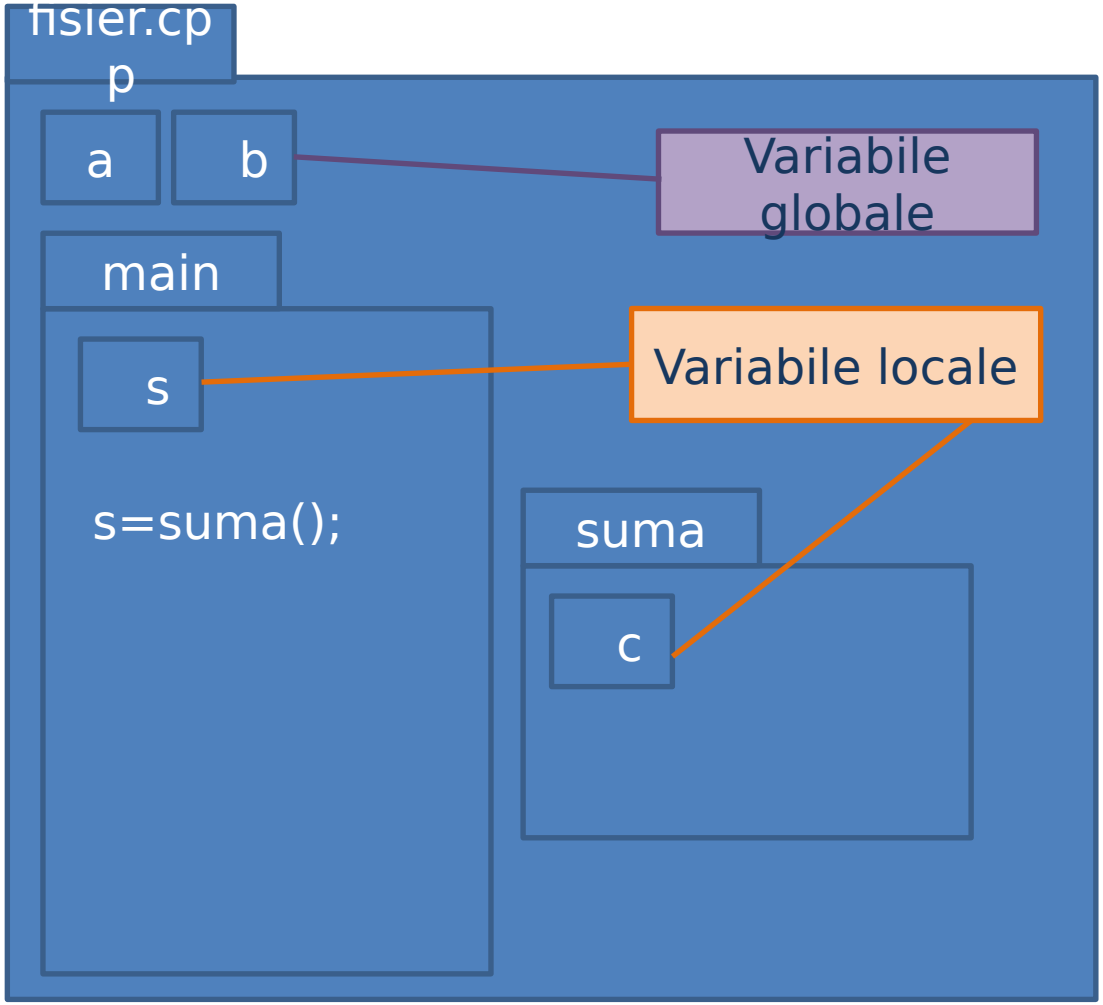
# Exemplu (cont)

```
int a,b;
int suma();
int main() {
  int s;
  -> a; // citim a
  -> b; // citim b
  s=suma();
  <- s; // afisam s
}
int suma() {
 int c=a+b;
return c;
}
```



# Exemplu (cont)

- Durata de viata (DV)
- DV(a)=DV(b)=fisier.exe (i.e. durata executiei aplicatiei)
- DV(s)=DV(main)=fisier.exe
- DV(c)=DV(suma) (i.e. durata apelului functiei)



# Tipul de date void

- Funcţiile care nu returnează nici un rezultat, precizează ca tip returnat **void**.
- Definiţia unei astfel de funcţii va fi

```
void nume_functie(lista parametri formali)
{
   Declaratii de variabile locale
   instructiuni
}
```

- Daca o functie nu primeste parametri, ea se defineste

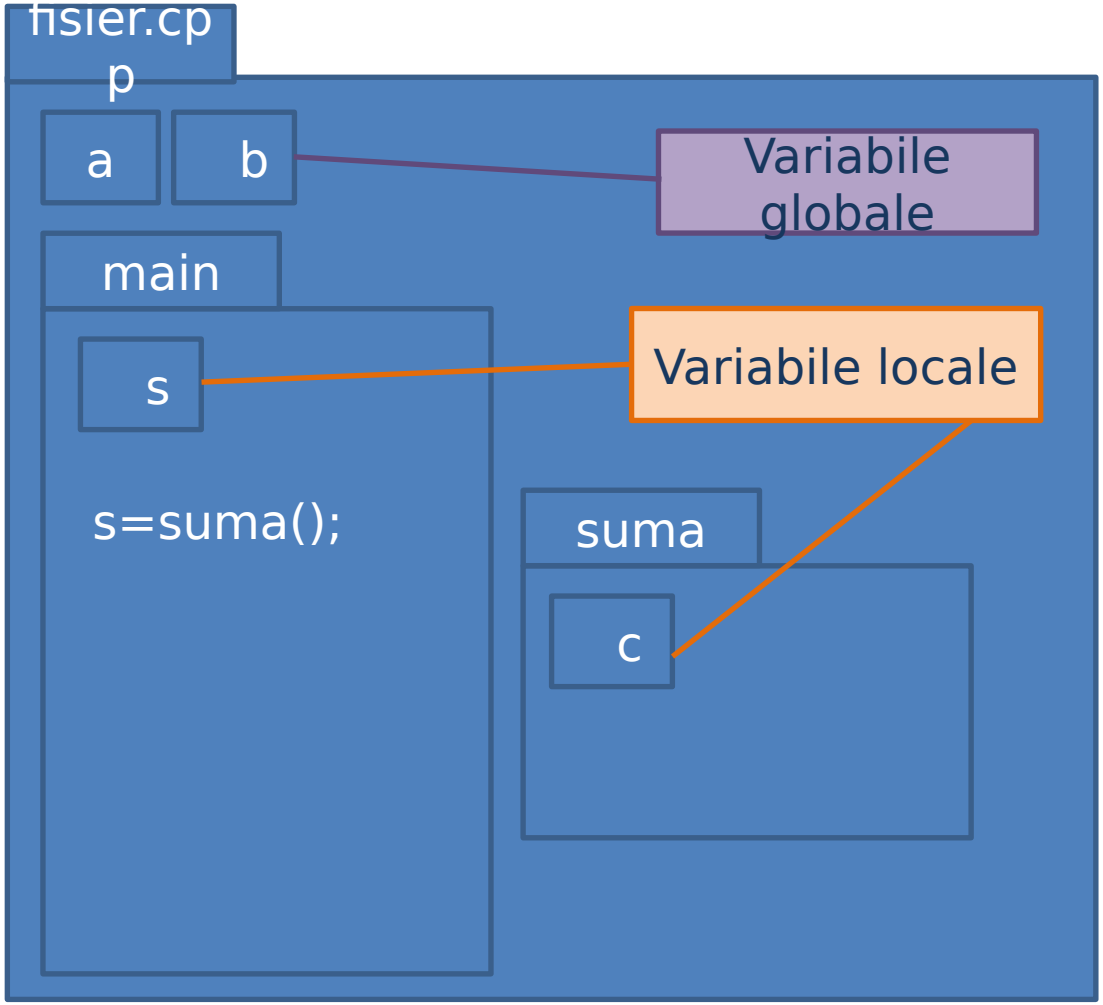
```
tip_returnat nume_functie(void) { …}
```

- sau

```
tip_returnat nume_functie () { … }
```

# Exemplu

- Domeniul de accesibilitate( DA)
- DA(a)=DB(b) =fisier.cpp (i.e. orice bloc, main+suma)
- DA(s)=main
- DA(c)=suma



# Exemplu (cont)

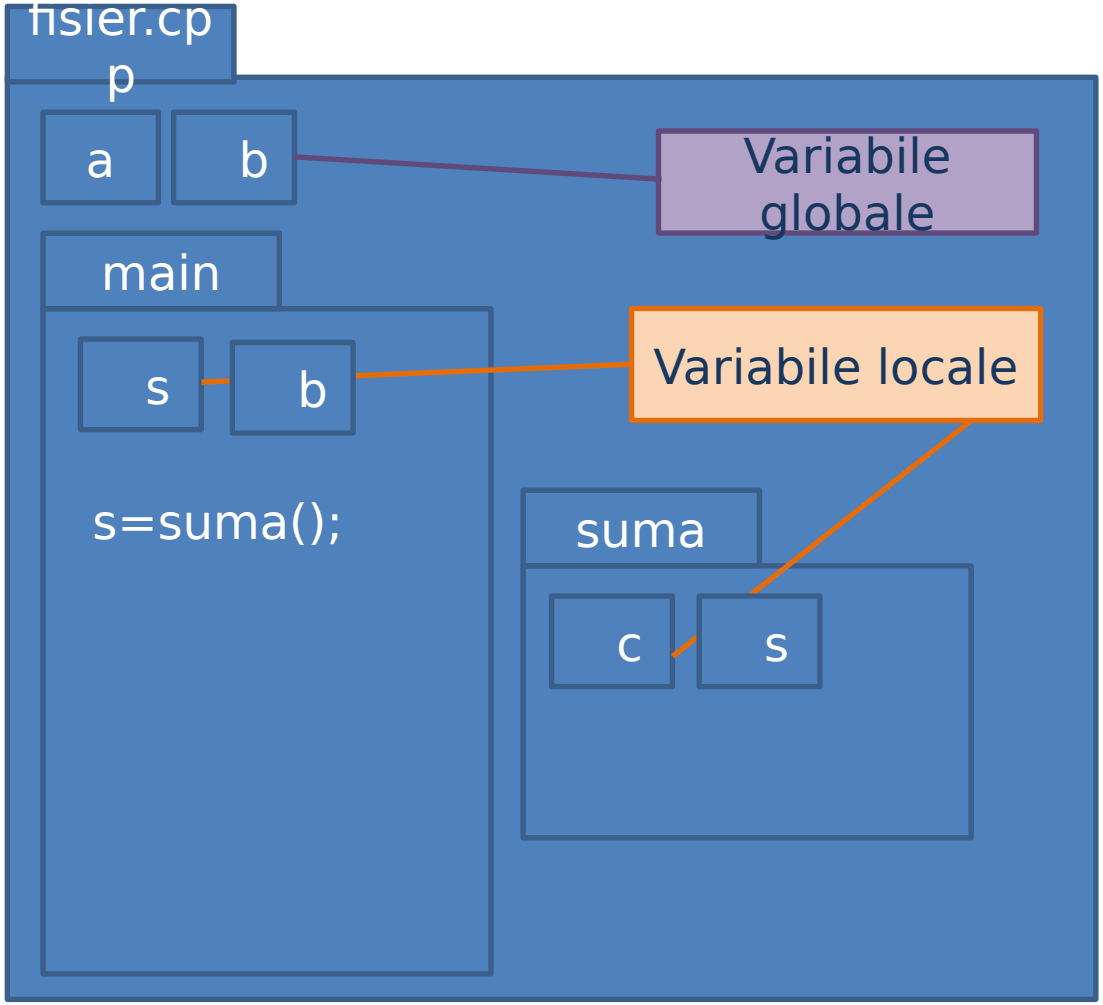
DV(a)=DV(b)

DA(a)=fisier

DA(b)=fisier-main

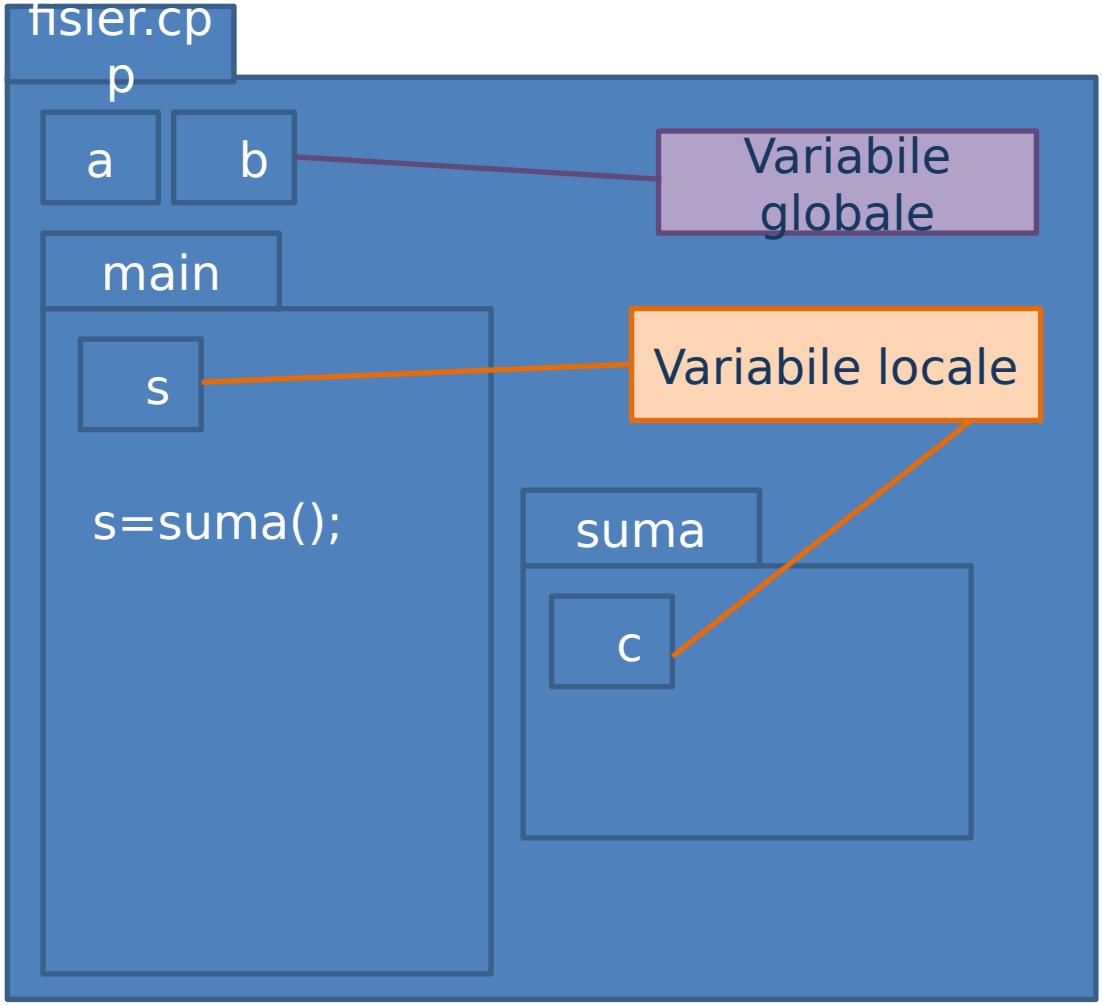
DA(s)=DA(b)=main

DA(c)=DA(s)=suma



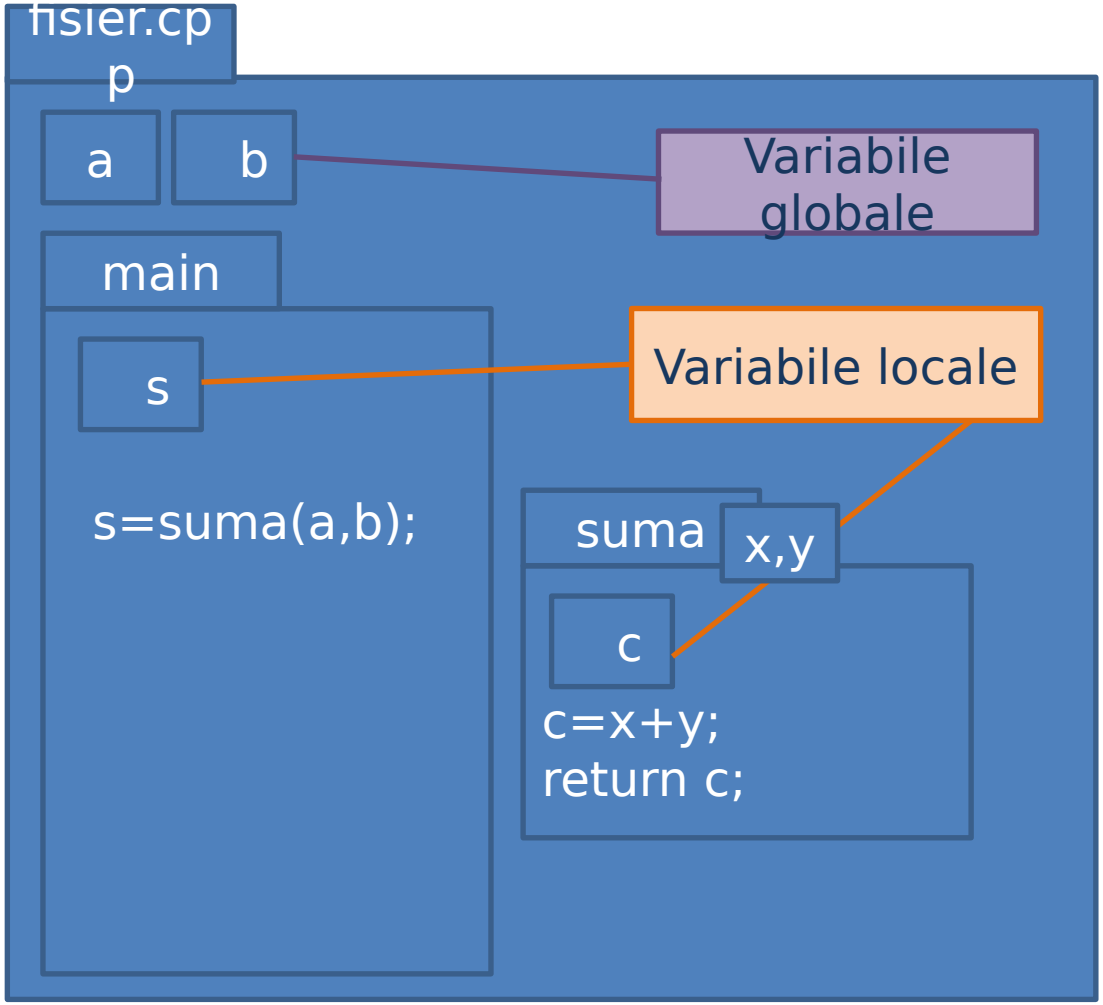
# Ce este in neregula aici?

```
int a,b;
int suma();
void main() {
  int s;
  -> a; // citim a
  -> b; // citim b
  s=suma();
  <- s; // afisam s
}
int suma() {
 int c=a+b;
 return c;
}
```



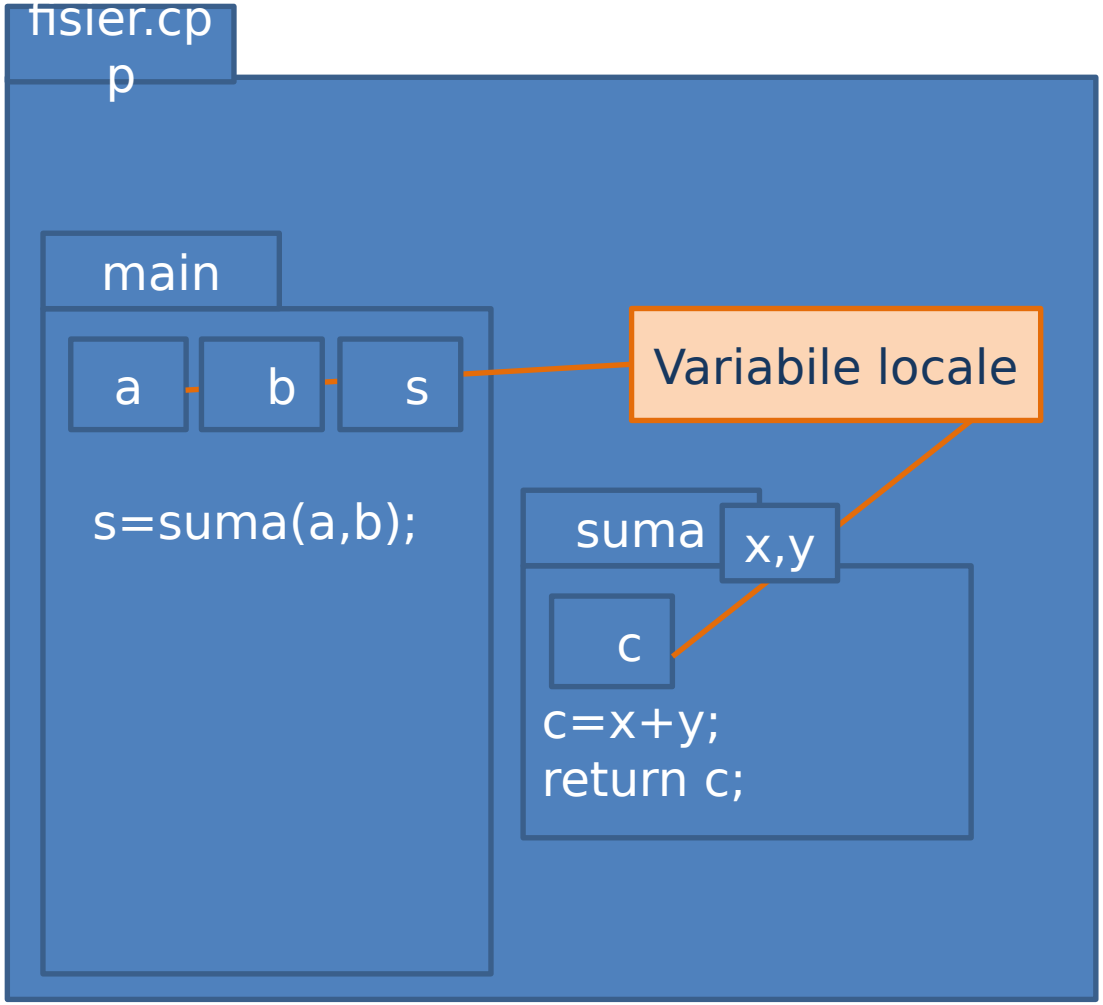
# Solutia corecta este ...

```
int a,b;
int suma(int,int);
void main() {
  int s;
  -> a; // citim a
  -> b; // citim b
  s=suma(a,b);
  <- s; // afisam s
}
int suma(int x,int y) {
 int c=x+y;
 return c;
 }
```



# … independenta de context!

```
int suma(int,int);
void main() {
  int a,b,s;
  -> a; // citim a
  -> b; // citim b
  s=suma(a,b);
  <- s; // afisam s
}
int suma(int x,int y)
   {
 int c=x+y;
 return c;
}
```



# Exercitiu

1. Scrieti o functie care returneaza void si are parametru void, care lucreaza cu o variabila globala de tip sir de caractere **s** de dimensiune **N** (**N** constanta simbolica) si care face urmatoarele operatii:
   a) citeşte un caracter **x** (declarat ca variabila locala in functie)
   b) citeşte un caracter **y** (declarat ca variabila locala in functie)
   c) înlocuieşte în **s** apariţiile lui **x** cu **y**.
   d) Exemplu:pentru s = Limbajul C are functii, x = C şi y = B
   e) se afişează: Limbajul B are functii.
2. Scrieti un program C care foloseste aceasta functie. Afisati sirul s inainte si dupa utilizarea functiei.

# Exercitiu - solutie

```
#define N 100
char s[N]="Limbaj";
void prelucrare();
int main(){
   printf(“Sirul este %s\n", s);
   prelucrare();
   printf("Rezultatul este %s\n", s);
}
void prelucrare(){
   char x, y;
   int i;
   printf("Introduceti doua caractere:\n");
   scanf("%c%c", &x, &y);
   for(i = 0; s[i] != 0; i++) {
       if(s[i] == x) s[i] = y;
   }
}
```

# Transferul parametrilor

# Transferul parametrilor

- În limbajul C, parametrii funcţiilor se transmit prin **valoare**.
  - funcţia apelată lucrează cu copii ale datelor de intrare şi nu poate să modifice variabilele din funcţia apelantă

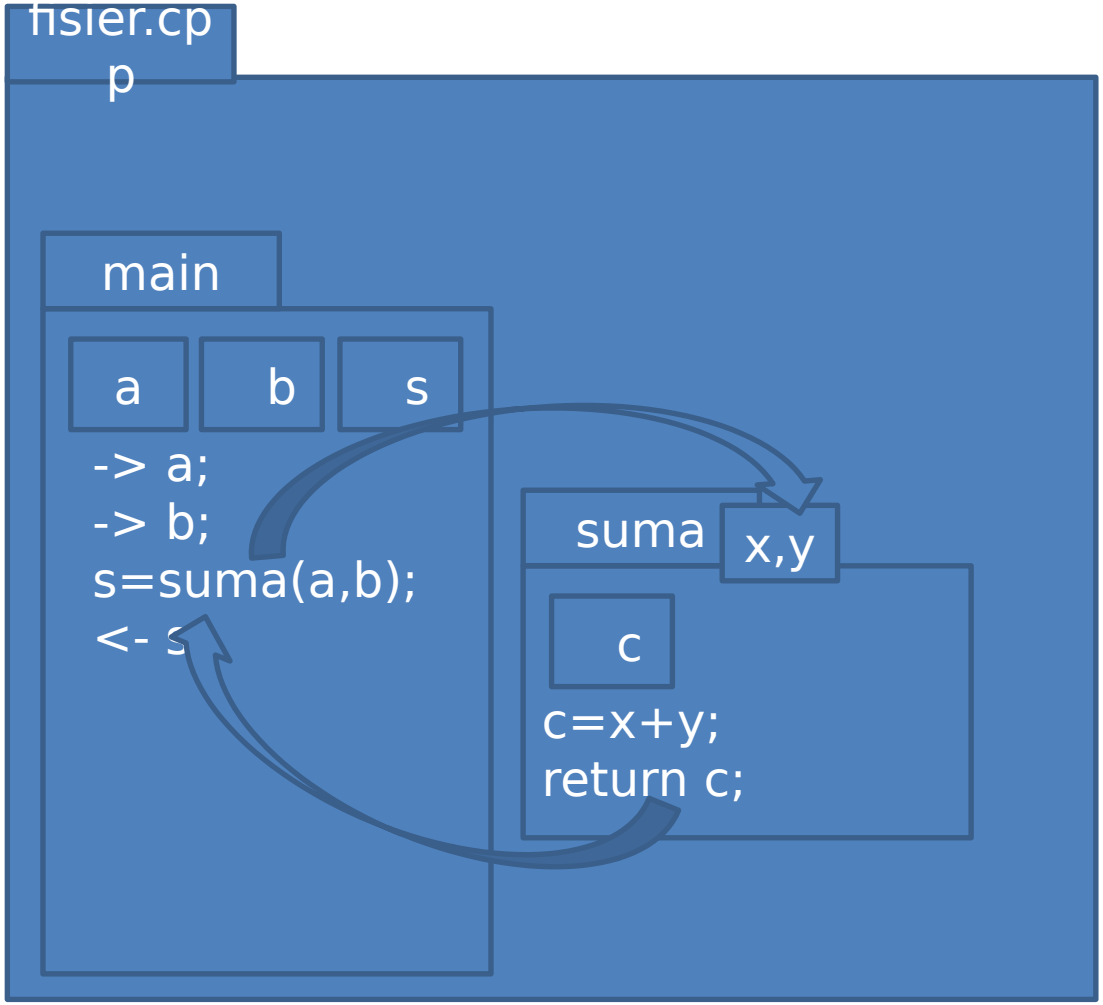
```
void test(int n) {
  n = 0;
}
int main(){
  int n = 5;
  printf("n este %i", n);
  test(n);
  printf("n este %i", n);
}
```

# Transferul parametrilor

```
#include<stdio.h>

int suma(int, int);

int main(){
    int s;
    int a=10,b=10;
    printf("Local: a = %i, b = %i\n", a, b);

    scanf("%i%i", &a, &b);
    printf("3 Local: a = %i, b = %i\n", a, b);

    s = suma(a,b);
    printf("Suma s este %i, iar a si b sunt %i si %i\n", s, a , b);
}

int suma(int a, int b){
    int x = a+b;
    a = b = 0;
    printf("\t in functia suma, a %i, b %i\n", a, b);
    return x;
}
```

# Transferul parametrilor = VALOARE

```
int suma(int,int);
void main() {
  int a,b,s;
  -> a; // citim a
  -> b; // citim b
  s=suma(a,b);
  <- s; // afisam s
}
int suma(int x,int y)
   {
 int c=x+y;
 return c;
}
```



# Exemplu

```
int suma (int, int);

int main(){
  int x = 5, y = 6;
  printf("x este %d, y este %d\n", x,y);
  int s = suma(x,y);
  printf("x este %d, y este %d\n", x,y);
  printf("Suma lor este %d\n.", s);
}

int suma (int a, int b) {
  b = a+b;
  a = 0;
  return b;
}
```

# Exercitiile 3.4

- Scrieti o functie care determina maximul dintre 2 numere intregi: max(a,b).
- Scrieti o functie care determina maximul dintre 3 numere intregi: max(a,b,c).
- Scrieti o functie care determina suma primelor n numere naturale: suma(n).
- Scrieti o functie care determina produsul primelor n numere naturale: produs(n).

# Exercitiile 3.5

- Scrieti o functie care determina suma numerelor naturale cuprinse intre n si m: sumadintre(n,m).
- Scrieti o functie care afiseaza coordonatele a n puncte echidistante situate pe un cerc de centru (x,y) si raza r.
- Fiind dat un numar t din [0,1], gasiti-i corespondentul u dintr-un interval [a,b], cu a si b numere reale.
- Aplicati apoi intervalul [a,b] in intervalul [c,d]. Si apoi in [d,c].

# Functii matematice si alte functii

# Biblioteca de functii matematice

- fişierul antet math.h
- conţine
  - constante
    - diverse valori remarcabile în matematică
  - funcţii standard
    - aritmetice
    - trigonometrice

# Constante

| Nume | Descriere | Nume | Descriere |
|---|---|---|---|
| M_E | $e$ | M_LOG2E | $\log_2 e$ |
| M_LOG10E | $\lg e$ | M_LN2 | $\ln 2$ |
| M_LN10 | $\ln 10$ | M_PI | $\pi$ |
| M_PI_2 | $\frac{\pi}{2}$ | M_PI_4 | $\frac{\pi}{4}$ |
| M_1_PI | $\frac{1}{\pi}$ | M_2_PI | $\frac{2}{\pi}$ |
| M_2_SQRTPI | $\frac{2}{\sqrt{\pi}}$ | M_SQRT1_2 | $\frac{1}{\sqrt{2}}$ |
| M_SQRT2 | $\sqrt{2}$ | | |

# Funcţii trigonometrice

- lucreaza cu radiani
- double sin (double x)
- double cos (double x)
- double asin (double x)
- double acos (double x)
- double tan (double x)
- double atan (double x)
- double atan2 (double x, double y)
- double cosh (double x)
- double sinh (double x)
- double tanh (double x)

# Funcţii putere şi logaritmice

- double pow (double x, double y)
- double sqrt (double x)
- double exp (double x)
- double log (double x)
- double log10 (double x)
- double log2 (double x)

# Alte funcţii

- rotunjire
  - double floor (double x)
  - double ceil (double x)
  - double round (double x)
  - double trunc (double x)
- valoare absolută
  - int abs (int x)
  - double fabs (double x)
- numere aleatoare
  - int rand ()

# Exercitiile 3.7

- un program C care implementează funcţia

$$f(x)=\begin{cases} xe^{1-x^2}, & x\in(-\infty,-1) \\ \sqrt[3]{x+1}\ln(1+x^2) & x\in[-1,1] \\ \sin(x)*sh(x) & x\in(1,\infty) \end{cases}$$

- un program C care calculează $e_n=\sum_{i=0}^{n}\frac{1}{i!}$

# Functii recursive

# Functii recursive

- Apelul unei functii se poate produce in orice functie, inclusiv functia in curs de implementare (recursivitate)
- Funcţie recursivă = funcţie care se foloseste pe ea insasi in definitia ei
  - Directă
    - în definiţia funcţiei f se apeleaza direct funcţia f
  - Indirectă
    - în definiţia funcţiei f se foloseşte o altă funcţie care foloseşte direct sau indirect funcţia f

- In order to understand  recursion, one must first understand recursion. Search for it on google.com!

# Exemplu

- factorial(n):
  - daca n = 0, factorial(n) este 1
  - daca n > 0, factorial(n) este n * factorial(n-1)

```
int factorial (int n)
{
  if (n == 0)
  return 1;
  return n * factorial (n-1);
}
```

# Funcţii recursive

- Unul sau mai multe **cazuri de baza**
  - valori de intrare pentru functie pentru care functia produce rezultat fara recurenta
- Unul sau mai multe **cazuri de recurenta**
  - valori de intrare pentru care programul intra in recursivitate (se autoapeleaza)
- Recursivitatea presupune executia repetata a unui modul
  - in cursul executiei lui (si nu la sfirsit, ca in cazul iteratiei), se verifica o conditie a carei nesatisfacere implica reluarea executiei modulului de la inceput, fara ca executia curenta sa se fi terminat.
  - In momentul satisfacerii conditiei se revine in ordine inversa din lantul de apeluri, reluandu-se si incheindu-se apelurile suspendate.
- La fiecare apel recursiv al unei functii se creaza copii locale ale parametrilor actuali si ale variabilelor locale, ceea ce poate duce la risipa de memorie.

# Exercitiile 3.8

1. Scrieti o functie care calculeaza al n-lea termen din sirul lui Fibonacci.
   - fib(0) = 1 si fib (1) = 1;
   - fib(n) = fib(n-1)+ fib(n-2), daca n>1
2. Folositi aceasta functie intr-un program C si aflati valorile ei pentru n de la 1 la 20.

# Exercitiile 3.9

1. Scrieţi o funcţie care calculează funcţia lui Ackerman

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1,A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

2. Folosiţi această funcţie într-un program C şi aflaţi valorile lui A(1,1), A(1,2), A(2,1), A(2,2), A(4,2). Atenţie. Lucraţi cu numere long integer!

# Exercitiul 3.10

- Implementati o functie recursiva care calculeaza suma primelor n numere naturale.

- Scrieti o functie care are ca parametru un numar intreg si afiseaza in ordine, de la stanga la dreapta, cifrele acestuia.
  - Daca ati reusit, incercati sa obtineti o varianta recursiva!