FUNCTII IN C++

FUNCȚIE = secvență de instrucțiuni, relativ independentă de program, care rezolvă o subproblemă.

Un program in C/C++ este un ansamblu de functii, în care functia `main` este apelata automat la lansarea in executie a acestuia.

Avantajele utilizării funcţiilor:

- Descompunerea problemei în subprobleme mai simple, mai uşor de rezolvat modular
- Reutilizarea codului sursă (apelul repetat al funcţiei în acelaşi program dar pentru date diferite)
- Creşterea claritatăţii codului sursă
- Uşurinţa depanării
- Posibilitatea de a distribui catre programatori diferiţi, unele module de program
- Dezvoltarea bibliotecilor de funcţii pentru anumite tipuri de prelucrări şi reutilizarea lor în aplicaţii complexe

Orice funcție utilizator trebuie declarată înainte de a fi utilizată.

- **Declararea unei funcţii.**

```
tip_rezultat nume_functie (lista_de_parametri_formali)
{     declarari date locale
      ......
      instructiuni
}
```

unde:

`tip_rezultat` - specifica tipul de date pe care il returneaza functia și poate fi:

- întreg (de ex: int, long, long long)
- real (de ex: float, double)
- char
- pointer (adresă de memorie)
- void (vid, adică niciun rezultat de unul dintre tipurile de mai sus)

`lista_de_parametri formali` - este o listă separată prin virgule, care conține nume de parametri și tipurile lor, declarati individual. O functie poate sa nu aiba parametri formali, caz in care lista lor este vida (parantezele sunt necesare, chiar daca nu exista parametri). Aceasta lista de parametri formali este interfata functiei cu toate modulele care o vor utiliza (apela).

Dacă rezultatul funcţiei NU este void, ea trebuie să conţină instrucţiuni de tip `return expresie;` prin care se va returna o valoare de tipul precizat.

Ex. `int test(int i, int k, float j)` /*antete corect de funcție*/
`int test(int i, k; float j)` /*antet gresit – lista incorecta de parametri formali*/

ATENȚIE!
NU se poate defini o funcție în interiorul altei funcții!!
NU se poate defini o functie intr-o alta functie, deci toate functiile au aceeasi sfera de influenta.

- **Apelul unei functii**

`nume_functie(lista_parametrilor_actuali)`

`lista_parametrilor_actuali` este formata din variabile, constante sau expresii compatibile ca tip cu parametrii formali, adica fiecarui parametru formal i se asociaza o variabila, o constanta sau o expresie compatibila ca tip. Daca lista de parametrii este vida, apelul se face specificand numele functiei urmat de paranteze rotunde.

- **Tipuri de parametri și modalități de transfer**

a) Parametri de intrare – transmiși prin valoare

- la declarare se precizează doar tipul (simplu sau pointer) și numele
- la apel: se copiază automat valoarea parametrului actual pe STACK (zona de memorie cu care lucrează funcțiile); orice modificare ar suporta această valoare în funcție, se va distruge doar copia, nu originalul. La revenirea din apelul funcției, valoarea inițială a parametrului este nemodificată.

Ex:

```
int sumcif(int n)
{ int s=0;
  while (n!=0)
      {
          s = s + n% 10;
          n = n / 10;
      }
return n;
}
.....
int n = 12345;
cout << sumcif(n) << ' ' << n;            //15 12345
```

b) Parametri de ieșire – transmiși prin referință

- la declarare se precizează tipul (simplu sau pointer) și numele precedat de simbolul &. Vectorii și matricele sunt implicit transmiși prin referință, nu au nevoie de &
- la apel: se salvează automat adresa de memorie (referința) parametrului actual pe STACK astfe încât, orice modificare ar suporta aceast parametru în funcție, va fi afectat automat originalul. La revenirea din apelul funcției, valoarea inițială a parametrului este definitiv modificată.

Ex:

```
void schimbă(int &a, int &b)
{ int aux;
  aux = a; a = b; b = aux;
}
.....
```

```
int x = 555, y = 222;
cout << x << ' ' << y;   //555 222
schimba(x, y);
cout << x << ' ' << y;   //222 555
```

Atentie!!

- Daca nu exista nici o instructiune return intr-o functie care nu este void, atunci valoarea returnata de functie este teoretic nedefinita.
- O instructiune de forma: `modific(x,y)=100; /*instructiune incorecta*/` este gresita. Compilatorul o va taxa ca eroare si nu va compila un program care contine asa ceva.

- **Domeniul de vizibilitate și valabilitate a datelor și identificatorilor**

Variabile locale	Variabile globale
- Se declară în interiorul funcțiilor - Se alocă pe STACK - NU sunt inițializate implicit cu valori nule - Sunt vizibile doar în funcția în care au fost declarate - Sunt valabile doar pe durata de execuţie a funcției	- Se declară în afara oricărei funcții - Se alocă în segmentul de date - Sunt inițializate implicit cu valori nule - Sunt vizibile în tot programul (în main şi în toate celelalte funcţii) - Sunt valabile pe toată durata de execuţie a programului

Regula de omonimie: O variabilă locală poate avea același nume cu o variabilă globală dar, e durata de execuție a funcției, variabila locală este prioritară.

Se poate face o analogie intre o functie si o piesa de teatru. Piesa este scrisa de autor o singura data, are personaje (`parametri formali`) si o descriere a cadrului in care ar trebui sa se desfasoare (`variabile locale`, declarate in interiorul functiei). Piesa de teatru devine astfel o resursa la dispozitia oricarui utilizator si asteapta sa fie pusa in scena (`apelata`) cu actorii unui anumit teatru (`parametrii efectivi, actuali`).

In general, la scrierea de programe in C++, functiile vor fi de trei tipuri:

- ✓ functii de calcul simplu - care sunt proiectate pentru a efectua operatii asupra argumentelor lor si a returna o valoare rezultata din aceste operatii (Ex. functie care calculeaza suma divizorilor lui n sau numarul acestora);
- ✓ functii de test care returnează valoarea de adevăr a unei validări (Ex. functia de verificare a primalitatii unui numar transmis ca parametru);
- ✓ functii care nu au o valoare returnata explicit si au rezultat `void`. In esenta, o astfel de functie este strict de procedura si nu returneaza o valoare (Ex. functie care schimba continutul a doua variabile, functie care ordoneaza un vector). Toate functiile care nu returneaza valori trebuie declarate ca returnand tipul void.