**Subprograme în C++**

Un subprogram este modul de program (de cod) parametrizabil, în urma căruia se poate întoarce o valoare, respectiv subprogramul poate fi capabil de a schimba valorile parametrilor săi.

Una dintre modalităţile de definire a subprogramelor constă în scrierea lor înainte de main, punând mai întâi antetul şi apoi conţinutul (codul).

În funcţie de faptul că subprogramul NU întoarce sau întoarce o valoare acesta poate fi de tip void sau de unul dintre tipurile simple de date (int, char, double şi variaţiuni ale acestora) sau poate fi şi de tip struct, însă NU poate fi tablou!!.

Iată sintaxa antetului:

tip\_rez\_întors nume\_subprogram(tip1 param1, tip2 param2 ...)

{ //de aici urmeaza corpul

codul subprogramului

}

Parametrii din această linie de definiţie se numesc **parametri formali**.

Parametrii din linia de apel (cea care cheamă funcţia să se execute) se numesc **parametri actuali** sau **parametri efectivi**.

La rândul lor, parametrii formali sunt de 2 feluri:

- cei a căror valoare NU se modifică la ieşirea din subprogram, chiar dacă îi modificăm în cadrul subprogramului. Aceşti parametrii se numesc "transmişi prin valoare". Parametrii prin valoare sunt toate variabilele simple (NE-vectori sau tablouri) care NU au & în faţă.

- cei a căror valoare se modifică şi în programul care a apelat funcţia, imediat ce îi modificăm în cadrul subprogramului. Aceşti parametrii se numesc "transmişi prin referinţă". Parametrii prin referinţă sunt orice vectori sau tablouri şi orice variabilă simplă în faţa căreia apare un &. (!! În faţa tablourilor NU putem pune & - pentru că primim eroare !!). La apel, pe poziţiile parametrilor transmişi prin referinţă trebuie să punem **variabile** care să fie **de exact acelaşi tip**.

- rezultatul întors de funcţie se stabileşte prin comanda

return valoare;

care printre altele iese IMEDIAT din funcţie, întrerupând orice repetitivă în curs de executare.

Acest mecanism ne permite **relaxarea** codului care în mod normal ar fi conţinut flag-uri, deoarece se poate merge pe tehnica:

- imediat ce o anumită dată NU convine, întoarcem 0.

- dacă firul execuţiei funcţiei supravieţuieşte până la final, la acest final putem lăsa un

return 1;

Ex: Una dintre funcţiile foarte utile este cea care verifică dacă un număr este sau nu prim, întorcând 0.

Iată codul său:

int is\_prime(int x)

{

if(x<=1 || x>2 && x%2==0) return 0;

for(int i=3;i\*i<=x;i+=2)

if(x%i==0) return 0;

return 1;

}

Ex: O funcţie care formează oglinditul unui număr într-un parametru:

void ogl(int x,int &y)

{

y=0;

do

{

y=y\*10+x%10;

x/=10;

}while(x);

}

Un exemplu de apel:

int q;

ogl(4132,q);

cout<<q;

O funcţie poate avea propriile sale variabile (declarate în funcţie), care se numesc variabile locale.

Variabilele locale sunt recunoscute DOAR în funcţia (blocul - chiar la nivel de "for") în care sunt declarate şi au avantajul că pot să aibă acelaşi nume cu alte variabile locale din alte funcţii sau din main, sau chiar cu alte variabile globale.

Variabilele locale iau naştere (se alocă) la intrarea în funcţie şi mor (se "dezalocă") la ieşirea din aceasta.