**Funcţii de tip procedural**

Sunt funcţii care pot returna 0 sau mai multe valori (prin parametri de tip adresă/referinţă sau prin variabile globale).

1. **Definirea funcţiei**

***void nume\_functie (lista\_parametri\_formali) // antetul funcţiei***

*{*

*// declaraţii variabile locale*

*// instrucţiuni*

*}*

unde:

* **tip\_rezultat = void (nicio valoare)**
* **Nu mai există instrucţiunea “return expresie\_rezultat”, deoarece funcţia nu mai întoarce valoare prin numele ei.**

**Obs.**

* ***lista\_parametri formali*** – conţine atât datele de intrare cât şi **datele de iesire** pentru subproblema rezolvată de funcţie şi este opţională. Numele parametrilor care reprezintă **date de ieşire/rezultate** vor fi precedate de operatorul **&**.

***Ex.*** *void f(int x, int y, int &z)*

1. **Apelul funcţiei**

Apelul funcţiei înseamnă activarea/executarea ei. Apelul se face prin:

***nume\_funcţie (lista\_parametri\_actuali/efectivi);***

Lista de parametri efectivi este formată din numele parametrilor separaţi prin virgulă.

**Apelul de funcţie procedurală este o instrucţiune** care se inserează acolo unde sunt necesare efectele produse de functie.

**Parametri efectivi/actuali trebuie să corespundă ca număr şi în ordinea definirii lor ca tip cu parametri formali din antetul funcţiei.**

**Ex.1** Să se scrie o funcţie care returnează **prin parametrul s** aria unui triunghi de laturi a, b, c (cu a, b, c – numere reale).

**void** arie (float a, float b, float c, **float &s**)

{ float p=(a+b+c)/2; // variabila locala

s=sqrt(p\*(p-a)\*(p-b)\*(p-c);

}

**Exemplu de apel:**

float a,b,c,s;

cin>>a>>b>>c;

**arie(a,b,c,s)**; // apel

cout<<s;

**Ex.2.** Să se scrie o funcţie care returnează media aritmetica si media geometrica a doua numere reale x si y.

**void** medii (float x, float y, **float &ma**, **float &mg**)

{ ma=(x+y)/2;

mg=sqrt(x\*y);

}

**Exemplu de apel:**

float x,y,m1,m2;

cin>>x>>y;

**medii(x,y,m1,m2)**; // apel

cout<<m1<<” ”<<m2;

**Ex. 3** Functii **fara parametri** pentru citire, afisare si sortare vector.

int v[100], n; // variabile globale

void citire()

{ int i;

cin>>n;

for (i=1; i<=n; i++) cin>>v[i];

}

void afisare()

{ int i;

for (i=1; i<=n; i++) cout<<v[i]<<” ”;

cout<<endl;

}

void sortare()

{ int i,j,aux;

for (i=1; i<n; i++)

for(j=i+1; j<=n;j++)

if(v[i]>v[j]) swap(v[i],v[j]);

}

int main()

{ citire(); afisare();

sortare(); afisare();

return 0;

}

**Obs. Parametri de tip tablou (vector/matrice) sunt transmişi implicit prin adresă/referinţă – ies modificaţi din funcţie.**

**Tema**1: acelasi funcţii cu parametri.

int a[100],b[100],n,m; // variabile globale

void citire(int &n, int v[100]) // n si v sunt rezultate

{ int i;

cin>>n;

for (i=1; i<=n; i++) cin>>v[i];

}

void afisare(int n, int v[100]) // n si v sunt date de intrare

{ int i;

for (i=1; i<=n; i++) cout<<v[i]<<” ”;

cout<<endl;

}

void sortare(int n, int v[100]) // n e data de intrare si v este data de iesire

{ int i,j,aux;

for (i=1; i<n; i++)

for(j=i+1; j<=n;j++)

if(v[i]>v[j]) swap(v[i],v[j]);

}

int main()

{ citire(n,a); citire(m,b)

sortare(n,a); sortare(m,b);

afisare(n,a); afisare(m,b);

return 0;

}

**Tema: 3119, 21, 906, 905, 800, 907, 1633, 910, 32, 17, 23, 20, 29, 912,30 - Pbinfo**