Funcții

- O bucată de cod identificabilă și apelabilă printr-un nume.
- Când se utilizează:
  - Vreau să împart un program mai mare în bucăți mai mici pe care să le testez/gestionez mai ușor;
  - Când trebuie să execut un set de operații de mai multe ori, în mai multe locuri din cod
- Exemple de funcții utilizare până în prezent:
  - min(a, b) – returnează minimul
  - sqrt(n) – radical din n
- Analizăm o funcție:
  - Sqrt(parametru număr) returnează (furnizează) o valoare
  - Double sqrt(double)
- Definirea unei funcții:
  Tip_data_returnat numeleFunctiei(**parametri**)
  { corpul funcției
  }
-
- **Tip_data_returnat** - poate fi orice tip predefinit (int, float, double, char) sau definite de utilizator
- **Parametri** – se separă prin virgulă, declară individual
- **Exemplu**:
- int suma(int a, int b) – o funcție care returnează un întreg și primește ca parametri două valori întregi

Exemple:

```
int fact(int n)
{
    int i, f=1;
    for(i=1; i<=n; i++)
        f = f * i;
    return f;
}
int modul(int x)
{
    if(x<0) return -x;
        else return x;
}
```

Returnarea valorilor prin parametri

Exemplu: determinarea numărului de cifre pare și impare dintr-un număr

```
// returneaza cate cifre impare si cate pare are n
void determ(int n, int &nrCifImp, int &nrCifPar)
{
    nrCifImp = nrCifPar = 0;
    do
    {
        if(n%2==0)nrCifPar++;
            else nrCifImp++;
        n/=10;
    }while(n!=0);
```

```
}

int main()
{
    int n, x, y;
    cin >> n;
    determ(n, x, y);
    cout << "Nr cif impare =" << x << " nr cif pare =" << y;

    return 0;
}
```

În funcția **determ** primul parametru a fost transmis prin valoare, iar următorii doi parametri prin referință.

Deoarece funcția nu returnează o valoare se pune **void** înaintea numelui funcției.

Transmiterea ca parametru a unui vector:

```
/// functia calculeaza suma valorilor dintr-un vector
int suma1(int *x, int n) /// vectorul e x, n-nr de elem
{
    int i, s=0;
    for(i=1; i<=n; i++) s+=x[i];
    return s;
}
int suma2(int x[], int n) /// vectorul e x, n-nr de elem
{
    int i, s=0;
    for(i=1; i<=n; i++) s+=x[i];
    return s;
}
int suma3(int x[100], int n) /// vectorul e x, n-nr de elem
{
    int i, s=0;
    for(i=1; i<=n; i++) s+=x[i];
    return s;
}
int main()
{
    int a[]={0,2,4,6,7,9}, n=5;
    cout << suma3(a,n);
    return 0;
}
```

Transmiterea ca parametru a unei matrice

```
/// am definit un tip de data numit matrice care este o matrice cu 1--
linii si 100 coloane
typedef int matrice[100][100];
matrice a, b, c; /// declar 3 matrici de tipul matrice
/// suma valorilor dintr-o matrice
int suma(matrice x, int n, int m) /// matricea x are n linii si m
coloane
{
    int i, j, s=0;
    for(i=1; i<=n; i++)
        for(j=1; j<=m; j++)
            s+=x[i][j];
```

```
    return s;
}
int suma1(int x[100][100], int n, int m) /// matricea x are n linii si
m coloane
{
    int i, j, s=0;
    for(i=1; i<=n; i++)
        for(j=1; j<=m; j++)
            s+=x[i][j];
    return s;
}
int suma2(int x[][100], int n, int m) /// matricea x are n linii si m
coloane
{
    int i, j, s=0;
    for(i=1; i<=n; i++)
        for(j=1; j<=m; j++)
            s+=x[i][j];
    return s;
}
/// citeste o matrice
void citire(matrice x, int &n, int &m)
{
    int i, j;
    cin >> n >> m;
    for(i=1; i<=n; i++)
        for(j=1; j<=m; j++) cin >> x[i][j];
}

int main()
{
    int  n, m;
    citire(a,n,m);
    cout << suma2(a,n,m);
    return 0;
}
```