

CURS 10 – FP

FUNCȚII (subprograme)

Funcție = o grupare de declarații și instrucțiuni care are un nume propriu, efectuează o prelucrare a unor date de intrare primite prin intermediul unor parametrii și returnează/furnizează un anumit rezultat, **dar care nu poate fi executată de sine-stătător, ci doar prin apelarea sa!**

Exemplu: Să se calculeze câți studenți au media maximă.

```
//funcția calculează...
float calcul_medie_maxima(float medii[100], int n)
{
    .....
}

//funcția calculează...
int numara_medii(float medii[100], int n, float medie)
{
    .....
}

int main()
{
    .....
    mmax = calcul_medie_maxima(medii, n);
    nmm = numara_medii(medii, n, mmax);
    .....
}
```

Rolurile funcțiilor: modularizarea și reutilizarea codului

Definirea unei funcții:

```
tip_de_date_returnat nume_functie(parametrii formali)    //antetul funcției
{
    variabile locale = variabile care pot fi utilizate doar în această funcție
    instrucțiuni

    return expresie;    //valoarea furnizată de funcție
}
```

Corpul funcției = conținutul funcției

Orice program scris în limbajul C conține cel puțin o funcție, respectiv funcția `int main()`. Funcția `main()` este prima apelată în momentul în care se execută programul respectiv.

Apelarea unei funcții = înlocuirea parametrilor formali cu parametri efectivi (valori)

```
#include <stdio.h>

int f(int x)
{
    return x+1;
}

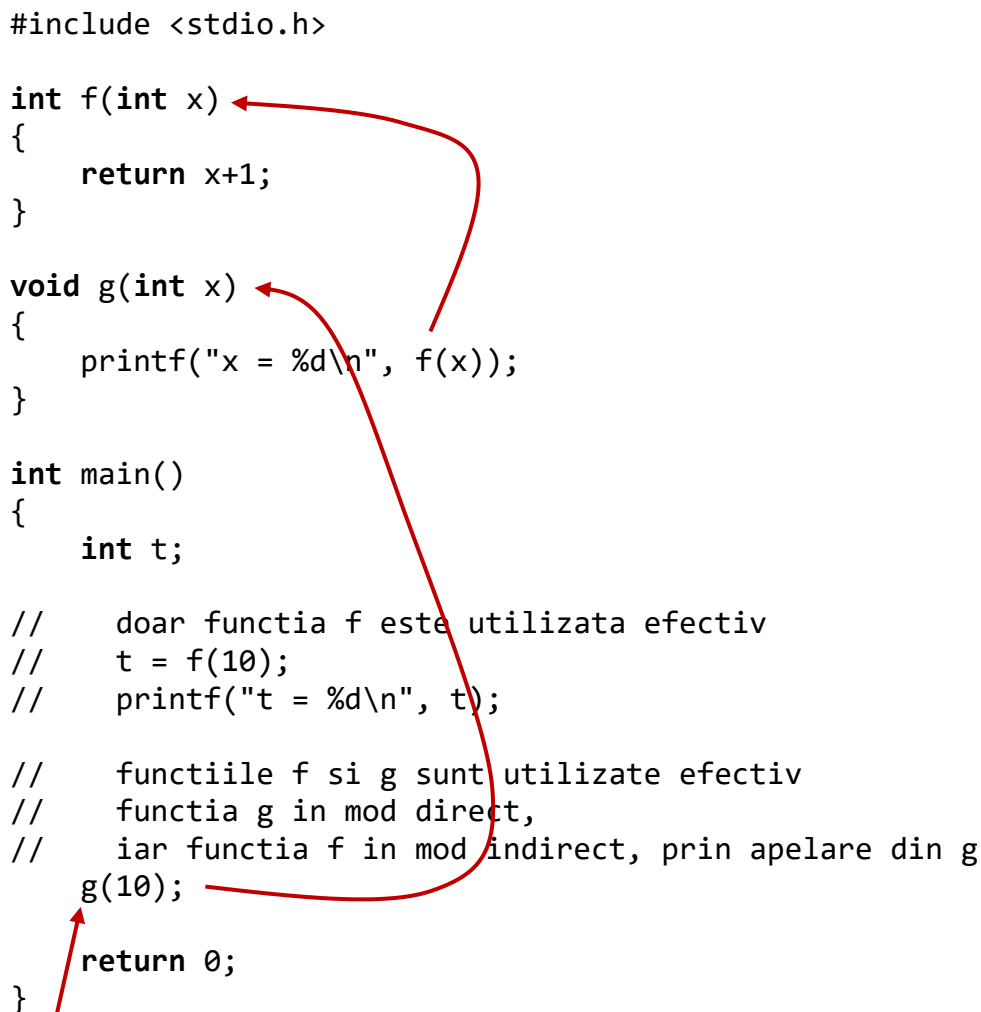
void g(int x)
{
    printf("x = %d\n", f(x));
}

int main()
{
    int t;

    // doar functia f este utilizata efectiv
    // t = f(10);
    // printf("t = %d\n", t);

    // functiile f si g sunt utilizate efectiv
    // functia g in mod direct,
    // iar functia f in mod indirect, prin apelare din g
    g(10);

    return 0;
}
```



O funcție este utilizată efectiv într-un program dacă există un lanț de apeluri care începe în funcția `main()` și se termină cu funcția respectivă!!!

Exemplu: `main()` -> `g(10)` -> `f(10)` => funcțiile `f` și `g` sunt ambele utilizate în program

Parametrii efectivi ai unei funcții trebuie să coincidă ca număr cu parametrii formali și să fie compatibili ca tip de date!

Valoarea returnată de o funcție (în urma unui apel) poate fi utilizată în orice context în care s-ar putea utiliza o variabilă de același tip cu tipul valorii returnate de funcție, mai puțin în stânga unei instrucțiuni de atribuire!

Rezultatul returnat de o funcție trebuie să fie de tip primitiv (momentan!!!).

Exemplu:

```
#include <stdio.h>

//functie care calculeaza suma a doua numere intregi
//numele funcției: suma
//adaug parametrii de intrare: suma(int x, int y)
//adaug tipul rezultatului: int suma(int x, int y)
//adaug corpul funcției: ...

//suma:ZxZ->Z, suma(x, y) = x + y    (definitie matematica)
//suma(5, 1) = 5 + 1 = 6
int suma(int x, int y)
{
    int s; //variabila locala

    s = x + y; //instructiune

    return s; //furnizarea rezultatului

    //forma scurta: return x + y;
}

int main()
{
    int a, b, s;

    //parametrii efectivi = constante
    s = suma(5, 1);
    printf("Suma: %d\n", s);

    //parametrii efectivi = constanta + variabila
    a = 10;
    s = suma(5, a);
    printf("Suma: %d\n", s);

    //parametrii efectivi = variabile
    a = 10;
    b = 7;
    s = suma(b, a);
    printf("Suma: %d\n", s);
}
```

```

//parametrii efectivi = variabila + expresie
a = 10;
b = 7;
s = suma(a, 2*a-b*b);
printf("Suma: %d\n", s);

//parametrii efectivi = expresii
a = 10;
b = 7;
s = suma(a+b, a*b);
printf("Suma: %d\n", s);

//parametrii efectivi = variabila + apel de functie (expresie!!!)
a = 10;
b = 7;
s = suma(b, suma(a,b));
printf("Suma: %d\n", s);

//utilizarea valorii furnizata de o functie
printf("\n")
printf("Suma: %d\n", suma(30, 50));

a = 10;
s = 2*suma(5, a) + 3;
printf("Suma: %d\n", s);

//gresit!!!
//suma(5, 7) = 10;
return 0;
}

```

O funcție poate să nu furnizeze niciun rezultat, caz în care tipul returnat este *void* și instrucțiunea *return expresie* nu se mai folosește în corpul funcției (eventual, se poate utiliza doar *return*; pentru a întrerupe executarea funcției).

În momentul executării unei instrucțiuni *return* într-o funcție, se termină executarea funcției!!!

Exemplu:

```

#include <stdio.h>

//functie care afiseaza numerele naturale
//cuprinse intre doua numere naturale a si b
void afisare(unsigned int a, unsigned int b)
{

```

```

    int i;

    if(a > b)
    {
        printf("\nNu exista niciun numar natural cuprins
                intre %u si %u!\n", a, b);
        // "blind return" = are rolul de a intrerupe executarea
        functiei
        return;
    }

    printf("\nNumerele naturale cuprinse intre %u si %u:\n", a, b);
    for(i = a; i <= b; i++)
        printf("%d ", i);
    printf("\n");
}

int main()
{
    afisare(20, 10);

    afisare(10, 20);

    return 0;
}

```

Exemplul inițial: Să se calculeze câți studenți au media maximă.

```

#include <stdio.h>
#include <stdlib.h>

//funcția calculează media maxima,
//adica valoarea maxima din tabloul medii cu n elemente
//tabloul medii ar putea fi declarat si prin float medii[100]
float calcul_medie_maxima(float medii[], int n)
{
    int i;
    float med_max;

    med_max = medii[0];
    for(i = 1; i < n; i++)
        if(medii[i] > med_max)
            med_max = medii[i];

    return med_max;
}

```

```

//funcția calculează cati studenti au o anumita medie
int numara_medii(float medii[], int n, float med)
{
    int i, nrm;

    nrm = 0;
    for(i = 0; i < n; i++)
        if(medii[i] == med)
            nrm++;
    return nrm;
}

int main()
{
    //nrs = numarul de studenti
    int nrs;

    //tabloul ms contine mediile a ns studenti
    float ms[100];

    //mmax = media maxima a unui student
    float medie_max;

    //nr_smm = numarul studentilor cu media maxima
    int nr_smm;

    int i;

    //citire date de intrare
    printf("Numar studenti: ");
    scanf("%d", &nrs);

    printf("\nMediile studentilor:\n");
    for(i = 0; i < nrs; i++)
    {
        do
        {
            printf("\tMedia studentului %d: ", i+1);
            scanf("%f", &ms[i]);
            if(ms[i] < 1 || ms[i] > 10)
                printf("\tMedia %.2f este incorecta (trebuie sa fie
                    intre 1 si 10)!\n", ms[i]);
        }
        while(ms[i] < 1 || ms[i] > 10);
    }

    medie_max = calcul_medie_maxima(ms, nrs);
    printf("\nMedia maxima: %.2f\n", medie_max);

    nr_smm = numara_medii(ms, nrs, medie_max);
    printf("\nNumarul studentilor cu media maxima: %d\n", nr_smm);

    return 0;
}

```