# TRANSFERUL PARAMETRILOR UNEI FUNCȚII

Funcțiile comunică între ele prin argumente (parametrii).

Există următoarele moduri de transfer (transmitere) a parametrilor către funcțiile apelate:

- **Transfer prin valoare;**
- **Transfer prin pointeri;**
- **Transfer prin referință.**

## TRANSFERUL PARAMETRILOR PRIN VALOARE

În exemplele anterioare, parametrii de la funcția apelantă la funcția apelată au fost transmişi prin valoare. De la programul apelant către funcția apelată, prin apel, se transmit valorile partametrilor efectivi, reali. Aceste valori vor fi atribuite, la apel, parametrilor formali. Deci procedeul de transmitere a parametrilor prin valoare constă în încărcarea valorii parametrilor efectivi în zona de memorie a parametrilor formali (în stivă). La apelul unei funcții, parametrii reali trebuie să corespundă - ca ordine şi tip - cu cei formali.

Limbajul C++ este numit limbajul apelului prin valoare, deoarece, de fiecare dată când o funcție transmite argumente unei funcții apelate, este transmisă, de fapt, o copie a parametrilor efectivi. În acest mod, dacă valoarea parametrilor formali (inițializați cu valorile parametrilor efectivi) se modifică în interiorul funcției apelate, valorile parametrilor efectivi din funcția apelantă nu vor fi afectate.

**Exemplu:**

```
void calcule(int x, int y)
{x=x+2;
y=y+3;
cout<<x<<" "<<y;
}
int main()
{
     int a=2,b=3;
     calcule(a,b);  //va afisa 4 si 6
     cout<<endl<<a<<" "<<b;  // va afisa 2 3, valorile pt a si b
raman neschimbate
return 0;}
```

## TRANSFERUL PARAMETRILOR PRIN POINTERI

În unele cazuri, parametrii transmişi unei funcții pot fi pointeri (variabile care conțin adrese). În aceste cazuri, parametrii formali ai funcției apelate vor fi inițializați cu valorile parametrilor efectivi, deci cu valorile unor adrese. Astfel, funcția apelată poate modifica conținutul locațiilor spre care pointează argumentele (pointerii).

**Exercițiu**: Să se citească 2 valori întregi şi să se interschimbe cele două valori. Se va folosi o funcție de interschimbare.

```
#include <iostream>
using namespace std;

void schimbă(int *x, int *y)
{
int aux;
aux=*x;
*x=* y;
*y=aux;
}
```

```
int main()
{
int a=2,b=3,*pa=&a,*pb=&b;
schimbă(&a,&b); //se poate apela si astfel: schimbă(pa,pb);
cout<<endl<<a<<" "<<b; // va afisa 3 2, valorile pt a si b raman
neschimbate
return 0;
}
```

Dacă parametrii funcției **schimbă** ar fi fost transmişi prin valoare, această funcție ar fi interschimbat copiile parametrilor formali, iar în funcția main modificările asupra parametrilor transmişi nu s-ar fi păstrat.

## TRANSFERUL PARAMETRILOR PRIN REFERINȚĂ

În acest mod de transmitere a parametrilor, unui parametru formal i se poate asocia (atribui) chiar obiectul parametrului efectiv. Astfel, parametrul efectiv poate fi modificat direct prin operațiile din corpul funcției apelate.

Exemplul devenit clasic pentru explicarea apelului prin referință este cel al funcției de permutare (interschimbare) a două variabile.

Parametri funcției schimb sunt transmişi prin valoare: parametrilor formali x, y li se atribuie (la apel) valorile parametrilor efectivi a, b. Funcția **schimb** permută valorile parametrilor formali x şi y, dar permutarea nu are efect asupra parametrilor efectivi a şi b.

Fie funcția **schimb** definită astfel:

```
void schimb (double x, double y)
{ double t=x; x=y; y=t; }

int main()
{ double a=4.7, b=9.7;
schimb(a, b);          // apel funcție
cout<<a<<" "<<b; //va afisa tot 4.7 si 9.7
return 0;
}
```

Pentru ca funcția de interschimbare să poată permuta valorile parametrilor efectivi, în limbajul C/C++ parametrii formali trebuie să fie referinte către valorile care trebuie interschimbate:

```
void schimb(double &x, double &y)
{ int t=x; x=y; y=t; }

void main()
{ double a=4.7, b=9.7;
 schimb(a,b);
cout<<a<<" "<<b; //afiseaza 9.7   4.7
}
```

În acest caz, x şi y sunt sinonime cu a şi b (nume diferite pentru aceleaşi grupuri de locații de memorie). Interschimbarea valorilor variabilelor de x şi y înseamnă interschimbarea valorilor variabilelor a şi b.

Comparând cele trei moduri de transmitere a parametrilor către o funcție, se poate observa:

1. La apelul prin valoare transferul datelor este unidirecțional, adică valorile se transferă numai de la funcția apelantă către cea apelată. La apelul prin referințătransferul datelor este bidirecțional, deoarece o modificare a parametrilor formali determină modificarea parametrilor efectivi, care sunt sinonime (au nume diferite, dar referă aceleaşi locații de memorie).
2. La transmiterea parametrilor prin valoare, ca parametrii efectivi pot apare expresii sau nume de variabile. La transmiterea parametrilor prin referință, ca parametri efectivi nu pot apare expresii, ci doar nume de variabile.(altfel eroare logica. Nu va fi semnalata si o eroare de sintaxa!) La transmiterea parametrilorprin pointeri, ca parametri efectivi pot aparea expresii de pointeri.
3. Transmiterea parametrilor unei funcții prin referință este specifică limbajului C++.

Fie functia urmatoare:

```
void calcule(int v[20],int n)
{for(int I=1;I<=n;I++)
       v[I]=2*I;
}

int main()
{int a[20],nr;
 nr=5;
for(int k=1;k<=nr;k++)
    a[k]=k;
calcule(a,nr);
for(k=1;k<=nr;k++)
    cout<<a[k]<<" "; //afiseaza 2 4 6 8 10
return 0;
}
```

La revenirea in main se pastreaza modificarile survenite in functia calcule asupra vectorului. Explicatie: numele unui vector este un poiner constant ceea ce inseamna ca atunci cand se transmit tablouri in functii, mecanismul de tranmitere a parametrilor va fi cel de transmitere prin pointeri.