Curs 4

1) Constructorul de copiere

```
CLASS idCLASA {
        PUBLIC:
            idClasa(const idClasa & obsursa);
        };
```
DEFINIRE

```
Complex C1(1,2);
Complx C2(C1); <=> Complx C2=C1;
```
se apelează constructorul de copiere

APELARE

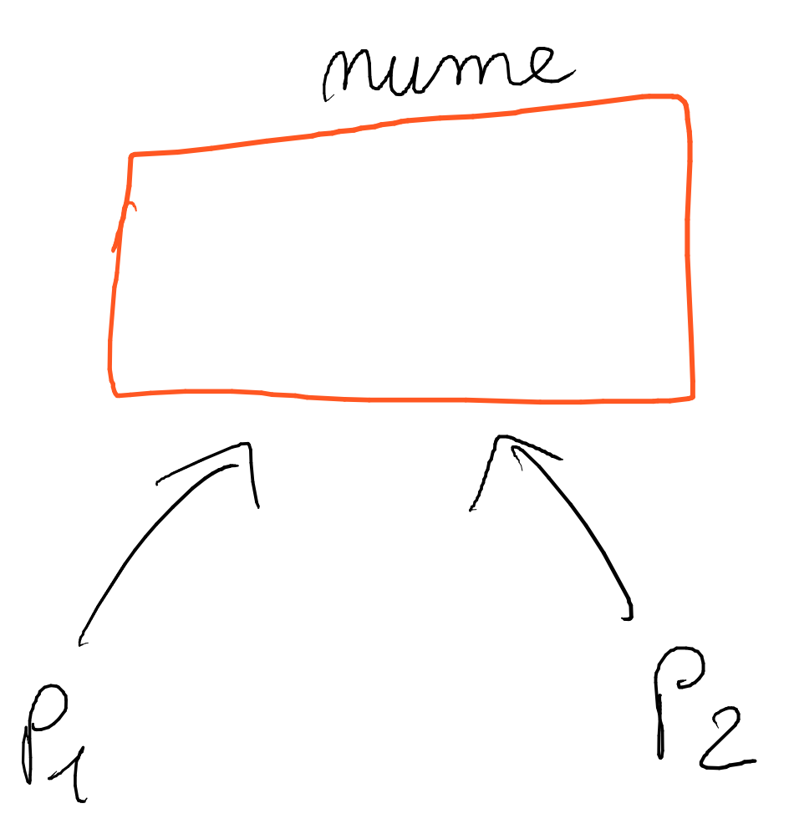
Constructorul de copiere poate fi generat de compilator sau defenit de programator.

Constructorul de copiere IMPLICIT: copie la nivel de bit a ob. sursă ->CC i

↳ Problemă! dacă obiectul conține date alocate dinamic => constructorul realizează BIT-WISE COPY fară să se aloce o nouă zonă de memorie

```
char *nume; char ~~nume~~[50];
                                        this->nume = NULL.
Persoană (char *nume, int varstă){
    this->nume = new char[strlen(nume)+1]; "\0"
    strcpy(this->nume, nume);
    this->varsta = varstă;
}
```
H E A P — CONSTRUCTOR

```
~Persoana(){ if(nume) delete nume;}
```
DESTRUCTOR

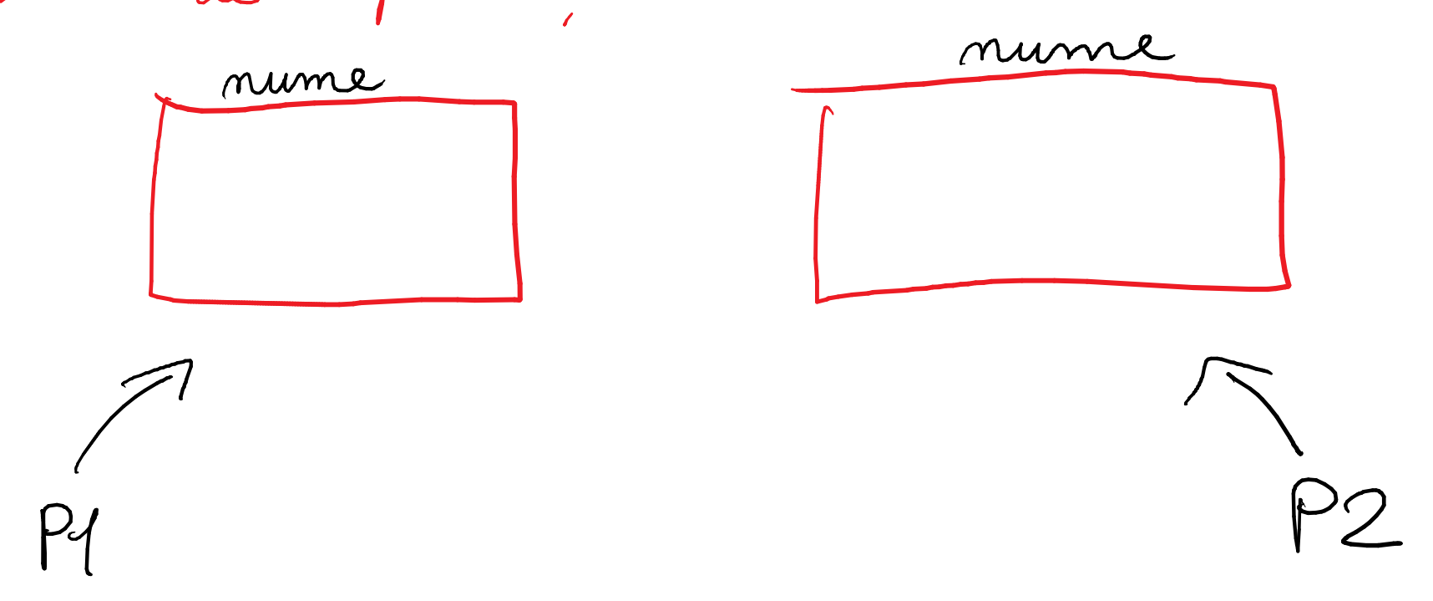


```
Persoana P1("Maria", 23);
Persoana P2 = P1; CCi
```

```
strcpy(P2.nume, "Dana");
cout<< P1.nume; ——> "Dana"
```

↳ Soluția! -> se oferă o emplementare a constructorului de copiere!

```
Persoana(const Persoana & obsursă){
    this->nume = new nume[strlen(obsursa.nume)+1];
    strcpy(this->nume, obsursă.nume);
    this->varsta = obsursa.varstă;
}
```



Persoana (const Persoana &obSursa) -> Prin valoare -> se apelează CC.
-> efect
pt -> o recursiv CC fără condiție de oprire (fără const)

2) Date și metode membre statice

Ex. class Persoana {
    char natonalitate[10]; -> Date membre statice
    char* nume;
    int varstă; -> Date membre de instanță
    Public:
        Persoană(char nume[ ], int varsta){...}
};
Persoană P1("Maria", 23)
Persoană P2("Daniel", 23);

Sintaxă:
static tip denumireDatemembru;
efect -> DenumireDateMembru se alocă o singură dată, în zona de memorie de date {la fel precum variabilele globale}
↳ valoare nula tip (int = 0; double = 0;)

Sintaxă definire date membru

```
class Produs {
        static double TVA;
};
double Produs:: TVA = 0,05;
```
efect -> Se alocă zonă de memorie

[tipData idClasa::DateMembruStatic = [valoare initială]]

Accesare dateMembru static (Publica)

```
Produs P1("Cola", 12);
Produs P2("Cafea", 6);
```

Afisare TVA -> cout << P1.TVA; ⇕ cout << P2.TVA;
-> Produs::TVA;

OBS: datele membre statice ce sunt partajate de toate obiectele

Sintaxă: static tip_returna membru(<list_arg>)

-> Nu primeste ca argument pointerul This

4) Separarea Codului clasei de implementarea sa

PAS 1: $\xrightarrow{Header}$ Persoana.h [Declararea Clasei]

```
class Persoana {
        date membru;
        Public:
            Persoana();
            void afisare;
};
```

PAS 2: $\xrightarrow{CPP}$ Persoana.cpp [Definire/implementare] Metodele din clasă (Definitie clasă)

tipReturnat idClasa::metodă (<list_arg>)
{...}

PAS 3: $\xrightarrow{main.cpp}$ #include "Persoana.h"