PROGRAMAREA ORIENTATĂ OBIECT C++

Conf.univ.dr. Ana Cristina DĂSCĂLESCU

Universitatea Titu Maiorescu

Constructorul de copiere

- Funcția principală: inițializarea unui obiect folosind datele membre ale unui obiect creat anterior.

- Sintaxa constructorului de copiere:

```
class IdClasa
{...
IdClasa (const IdClasa &ob);
...
};
```

Referința obiectului sursă

- Constructorul de copiere primește ca parametru o referința a obiectului sursă.

Exemplu de apel constructori de copiere:

```
Complex c1(1,2);//Constructor cu argumente
Complex c2(c1); // Constructor copiere
Complex c3 = c2; // Constructor copiere
```

- Constructorul de copiere poate fi generat de compilator sau definit de către programator!!!

➢**Observații:**

- dacă nu este definit un constructor de copiere al clasei, **compilatorul generează un constructor de copiere** care copiază datele membru cu membru din obiectul referinţă în obiectul nou creat (*bitwise copy -> shallow copy*).

```
Complex c3 = c2;
```

- dacă datele membre ale unui obiect sunt alocate dinamic, atunci programatorul definește explicit un constructor de copiere care alocă o zonă de memorie pentru obiectul destinație *(deep copy*).

```
class  Persoana {
        int varsta;
        public:
        char* nume;
        Persoana(char* sn, int v)
          {
            nume=new char[strlen(sn)+1];
           strcpy(nume, sn);
            varsta=v;
          }
       void afisare()
          {cout<<nume<<“ ”<<varsta;}
};
```

```
int main()
{
   Persoana p1(“Popa”, 23);
   Persoana p2=p1;
   strcpy(p2.nume,”Matei”);
   p1.afisare();
   p2.afisare();
}
```

Apel constructor de copiere implicit

```
Matei 23
Matei 23
```

- **Constructorul de copiere implicit nu realizează alocarea dinamică pentru datele membre ale obiectului destinație!!!!**

```
 Persoana(const Persoana &ob){
                    nume=new char[strlen(ob.nume)+1];
                    strcpy(nume, ob.nume);
                    varsta=ob.varsta;}
int main()
{
    Persoana p1("Popa", 23);
    Persoana p2=p1;
    strcpy(p2.nume,"Matei");
    p1.afisare();
    p2.afisare();
}
```

→

Popa 23
Matei 23

Constructori de copiere - Utilizare

- Definirea unui obiect inițializat cu valorile datelor membre ale unui obiect creat anterior.
- Apelul unei funcţii care are ca argument un obiect transmis prin valoare.
- O funcție returnează un obiect prin valoare.

Transmiterea unui argument al unei funcții prin valoare

```
int func(int x)
{
    int i=3
    x = x+i;
    return x;
}

int main()
{
     int a = 7;
     cout<<func(a);
     cout<<a;

     return 0;
}
```

Call stack

valoare returnată x = 10 0x08804
variabila locala i 0x08804
Se copiază valoarea 7 0x07804
return address 0x07800
7 0x06008

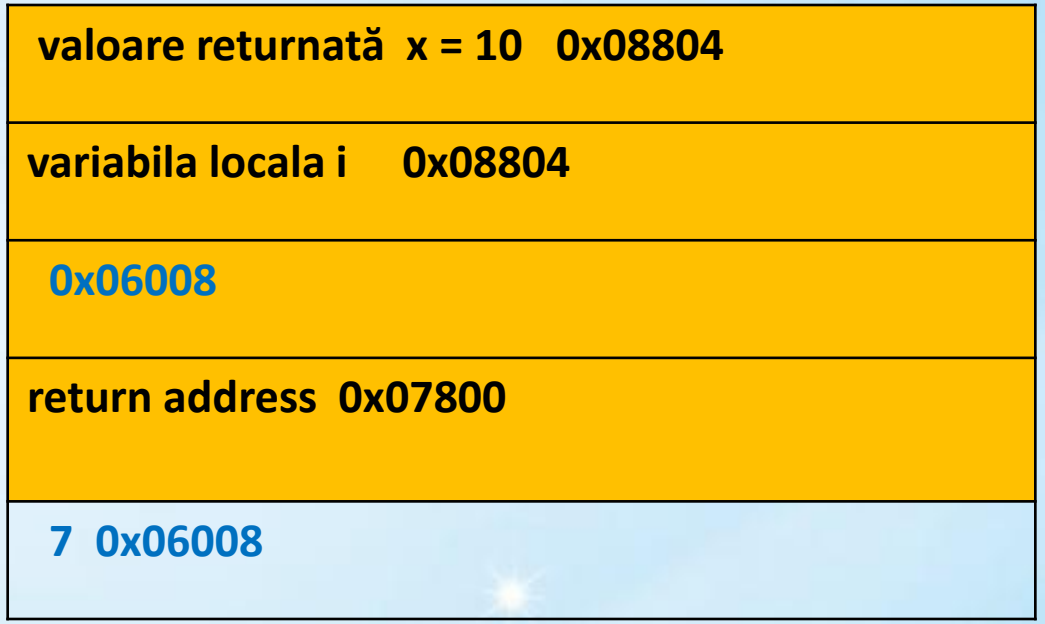
Transmiterea unui argument al unei funcții prin referință

```
int func(int &x)
{
    int i=3
    x = x+i;
    return x;
}

int main()
{
     int a = 7;
     cout<<func(a);
     cout<<a;

     return 0;
}
```

Call stack



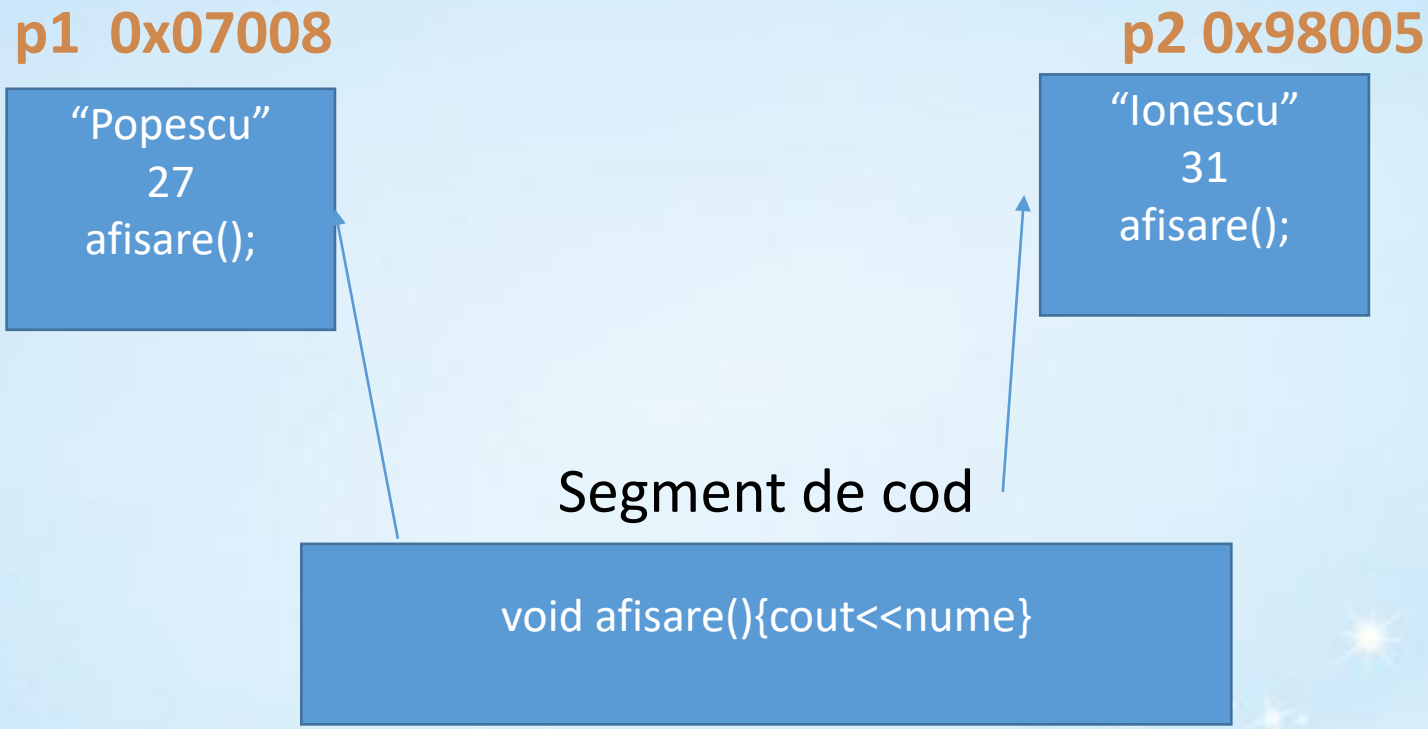
TRANSMITEREA UNUI OBIECT CA PARAMETRU AL UNEI FUNCȚII

1. **Transmiterea unui obiect prin valoare:** se construiește un obiect local prin constructorul de copiere care realizează o copie a obiectului pe stiva.

1. **Transmiterea prin referința (&) sau prin adresă (*):** nu se mai apelează constructorul de copiere!!!!

Date membre statice

- **Implicit datele membre sunt alocate pentru fiecare obiect al unei clase;**



Date membre statice

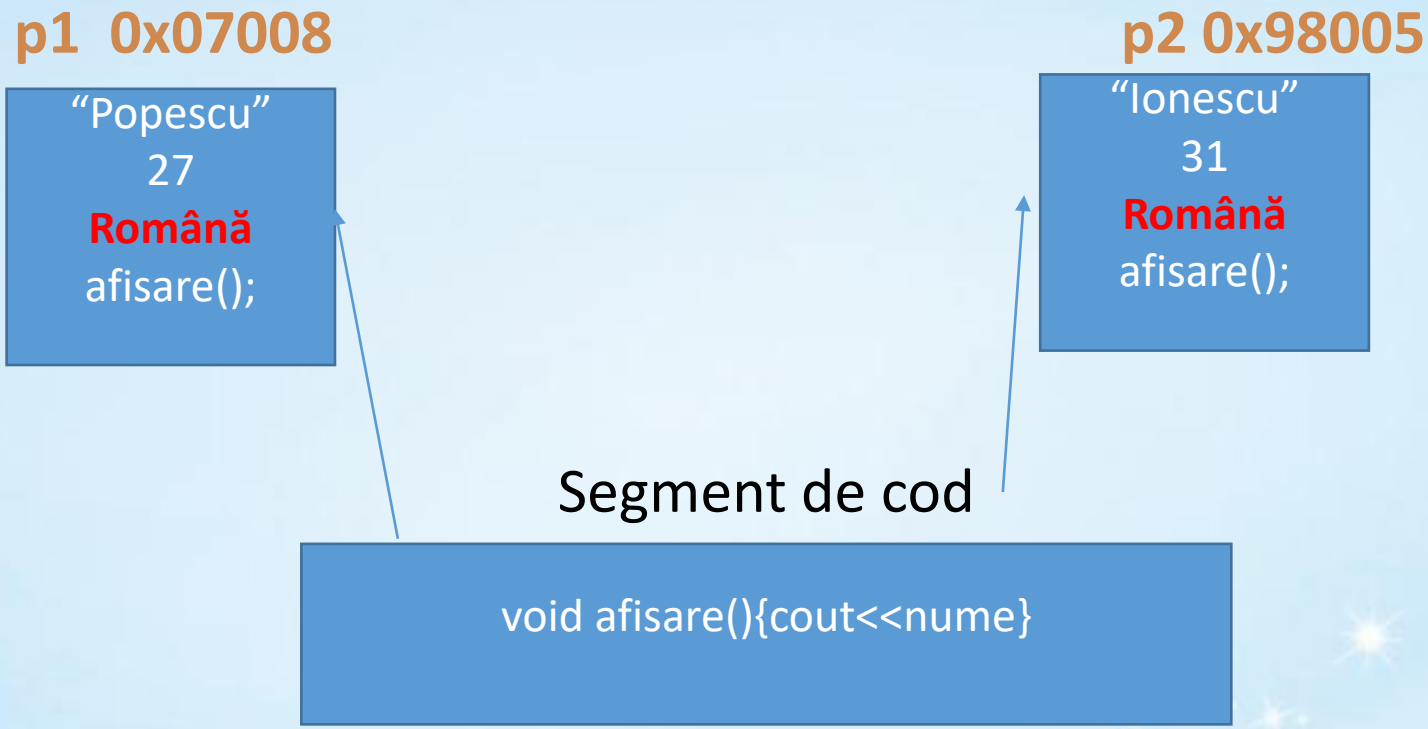
- Există situații în care anumite date membre au aceași valoare pentru toate obiectele unei clase

- Exemple
  - naționalitatea unei persoane
  - numărul de credite necesar unui student pentru a promova anul 1
  - numărul de obiecte cu o anumită proprietate

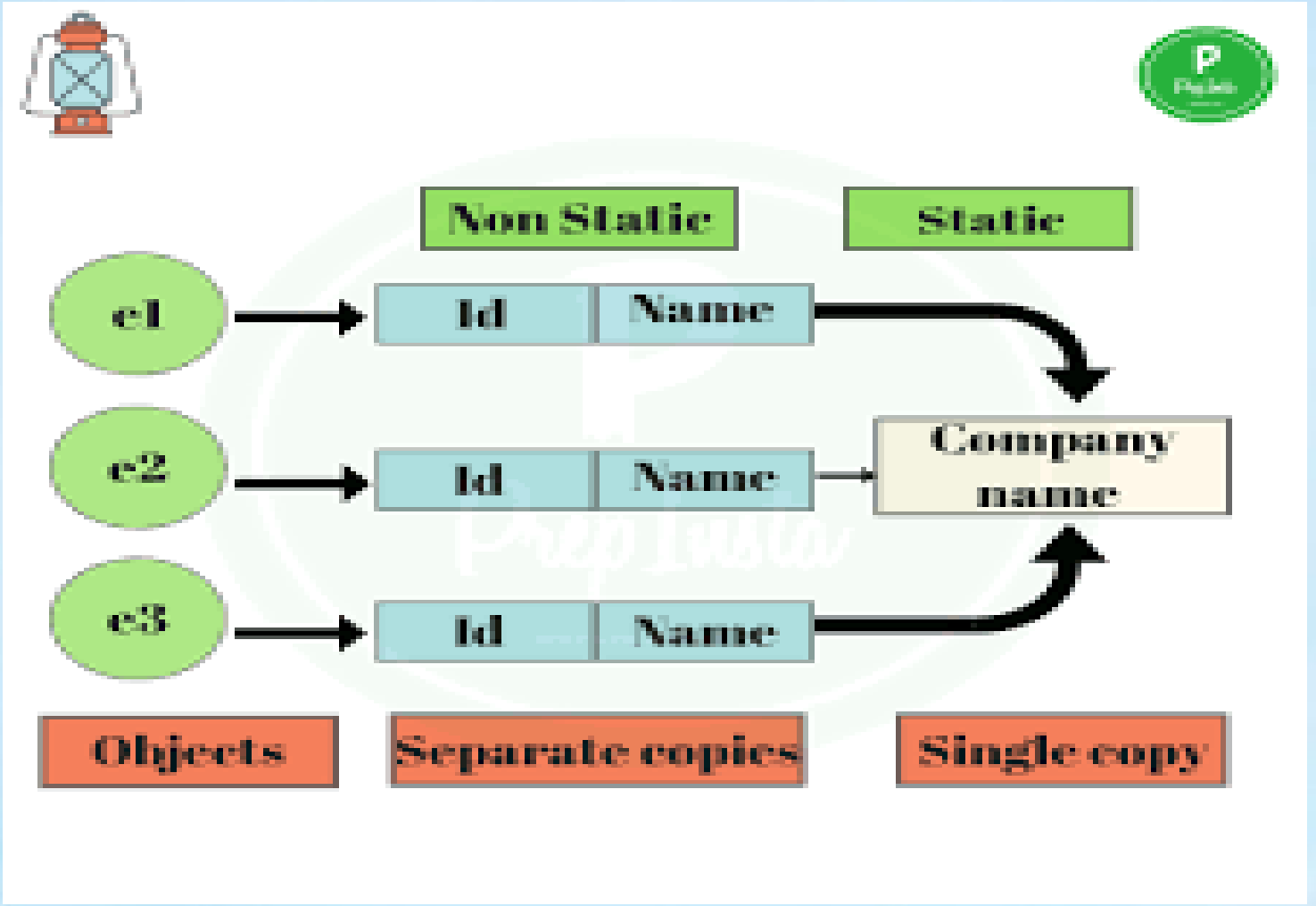
- Dacă aceste date sunt definite ca date membre ale unei clase, atunci se alocă spațiu de memorie pentru fiecare instanță, deși valoarea lor este aceeași!!!!

Date membre statice

- **Implicit datele membre sunt alocate pentru fiecare obiect al unei clase;**



Date membre statice



Date membre statice

- Datele **membre statice** sunt date membre ale unei clase ce sunt partajate de toate obiectele;

- O dată membră se alocă în memorie o singură dată, indiferent de numărul obiectelor

- Sintaxa declarării unei date membre statice:

```
class IdNumeClasa
{...
  static tip nume;
...
};
```

➢ *Exemplu de date membre statice:*

```
static float TVA;//TVA pentru produce
static int nr_studenti; // Nr total de studenți
```

➢La definirea unei clase, o dată membră statică nu este definită, ci doar declarată!!!!

➢Data membră statică nu este inițializată prin constructor!!!!

➢Alocarea memoriei pentru o dată membre statică (definirea) și inițializarea ei se realizează explicit în exteriorul clasei.

```
IdClasa::data_statica=valoare initiala;
```

Accesarea unei date membre statice

- Regurile de acces sunt aceelasi ca și regurile de acees la datele membre unui clase;

- Se apoate accesa membru static fie prin intermediul unui obiect, fie prin intermediul clasei:

nume_clasa::data_statica;

- Membrii statici există chiar dacă nu sunt încă obiecte create.

```
class Produs{
 char nume[100];
 float pret; //fara TVA
public:
static double PROCENT_TVA;
 Produs(char *nume, float pret)
{
 strcpy(this -> nume, nume);
 this -> pret = pret;}
 void afisare(){
  cout<<nume<<" "<<getPretCuTVA()<<endl;
}
float getPretCuTVA()
{
return pret*(1+PROCENT_TVA);
}};
float Produs::PROCENT_TVA = 0.24;
```

Declarare

Definire

```
int main(){
Produs p1("CPU", 100);
Produs p2("HDD", 200);
p1.afisare();
p2.afisare();

cout<<Produs::PROCENT_TVA;

}
```

CPU 124
HDD 248

Accesare prin numele clasei

Metode membre statice

- Sunt asociate unui clase, ci nu unui obiect.

- O referință la o funcție membră statică nu necesită un obiect;

```
nume clasa::functie statica( );
```

- Funcția membră statică care accesează date membre statice nu are argumente;

- Dacă funcția membră statică accesează date membre nestatice, atunci primește ca parametru o referinta la obiectul respectiv, **deoarece nu primește pointerul this**.

```
static functie(obiect &);
```