Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Facultatea de Automatică şi Calculatoare

PROGRAMARE II

Curs 9

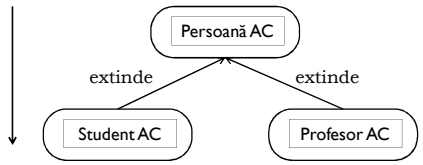
Moştenire (ordinea de apelare a constructorilor şi a destructorilor). Moştenirea multiplă.

Ordinea de apelare a constr. / destr.

- Ordinea de apelare a constructorilor pentru un obiect dintr-o clasă derivată:
  - Constructorul clasei de bază
  - Constructorul clasei derivate
- Ordinea de apelare a destructorilor pentru un obiect dintr-o clasă derivată:
  - Destructorul clasei derivate
  - Destructorul clasei de bază

Ordinea de apelare a constructorilor

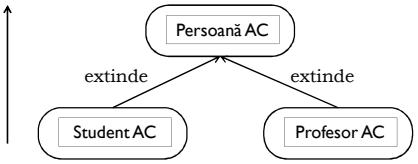
- Pentru un obiect StudentAC
- Ordinea de apelare: de la clasa de bază către clasa derivată



- Constructorul clasei Persoană AC
- Constructorul clasei Student AC

Ordinea de apelare a destructorilor

- Pentru un obiect StudentAC
- Ordinea de apelare: de la clasa derivată către clasa de bază (în ordinea inversă a constructorilor)



- Destructorul clasei Student AC
- Destructorul clasei Persoană AC

cod curs 9.pdf

Funcția main

```
int main ()
{
  PersoanaAC
p1("1234567890123","Ana","Iasi");
  p1.afisareProfil();

  StudentAC s2;
  s2.afisareProfil();

  StudentAC s1("1234567890122",
"Ion","Vaslui",2,10);
  s1.schimbareAdresa("Bucuresti");
  s1.inscriereAnStudiu(3);
  s1.afisareProfil();

  //constructor copiere
  cout << "Copiere " <<endl;

  StudentAC s3(s1);
  s3.afisareProfil();

  return 0;
}
```

```
constr. cu arg. PersoanaAC
Nume: Ana CNP: 1234567890123
Adresa: Iasi
constr. fara arg. PersoanaAC
constr. fara arg. StudentAC
Nume:  CNP: 0000000000000
Adresa:
An studiu: 0 Nota P2: 0
constr. cu arg. PersoanaAC
constr. cu arg. StudentAC
Nume: Ion CNP: 1234567890122
Adresa: Bucuresti
An studiu: 3 Nota P2: 10
Copiere
Nume: Ion CNP: 1234567890122
Adresa: Bucuresti
An studiu: 3 Nota P2: 10
destructor StudentAC
destructor PersoanaAC
destructor StudentAC
destructor PersoanaAC
destructor StudentAC
destructor PersoanaAC
destructor PersoanaAC
```

Exemplu:

- Fără funcția afișare student
- Debug s2
- Constructor de copiere s3 – exemplu
  - Fără constructor de copiere
  - Numai cu constructor de copiere student

Constructorii apelați implicit

- Pentru un obiect al unei clase derivate, constructorul apelat implicit pentru clasa de baza este constructorul fără listă de argumente (dacă există constructori declarați în clasă)
- Constructorul clasei derivate apelat implicit este constructorul fără listă de argumente (dacă există constructori declarați în clasă)
- Prin apeluri de forma

```
StudentAC::StudentAC(…) :
      PersoanaAC(cnp, nume, adresa), ...
```

se poate controla ce constructor din clasa de bază va fi folosit (se forțează compilatorul pentru a apela un anumit constructor)

Moştenirea "în lanț" (în cascadă)

```
class A
{
public:
    A()
    { cout << "A" << endl; }
};

class B: public A
{
public:
    B()
    { cout << "B" << endl; }
};

class C: public B
{
public:
    C()
       { cout << "C" << endl; }
};
```

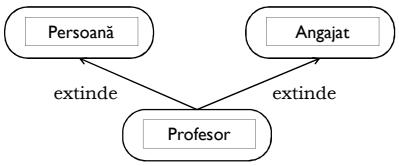
```
int main()
{
     C objC;
}
```

La rulare, se va afișa:

A
B
C

Moştenirea multiplă

- Permite unei clase derivate de a moşteni mai mult de un singur părinte
- Se consideră exemplul clasei Profesor care moşteneşte atât o clasă Persoană cât şi o clasă Angajat



Clasa Persoană

```
class Persoana
{
private:
    string m_sNume;
    int m_iVarsta;

public:
    Persoana(string nume, int varsta)
        : m_sNume (nume), m_iVarsta(varsta)
    {}

    int getVarsta()
       {
              return m_iVarsta;
       }
};
```

Clasa Angajat

```
class Angajat
{
private:
    double m_dSalariu;

public:
    Angajat(double salariu)
        : m_dSalariu(salariu)
    {}

    double getSalariu()
       {
              return m_dSalariu;
       }
};
```

Clasa Profesor

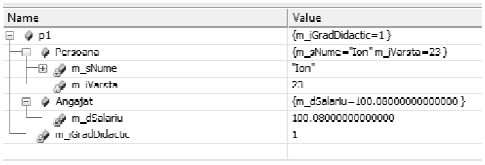
```
class Profesor: public Persoana, public Angajat
{
private:
     int m_iGradDidactic;

public:
    Profesor(char * nume, int varsta, double salariu, int
gradDidactic)
        : Persoana(nume, varsta),
          Angajat(salariu),
          m_iGradDidactic(gradDidactic)
    {
    }
};
```

Funcţia main

```
int main()
{
       Profesor p1("Ion",23,100.08,1);

       return 0;
}
```



Moştenire multiplă - probleme

- Cand o clasă moşteneşte 2 clase care au o funcţie membră comună
- Cand moştenirea este în "formă de diamant"

Exercitiu

- Cum trebuie modificate clasele daca se schimbă constructorul clasei Profesor în modul următor:

```
public:
    Profesor(string nume, int varsta, double salariu, int
gradDidactic)
        : Persoana(nume, varsta),
          m_iGradDidactic(gradDidactic)
    {
        m_dSalariu = salariu;
    }
```

```
error C2512: 'Angajat' : no appropriate default
constructor available

error C2248: 'Angajat::m_dSalariu' : cannot access private
member declared in class 'Angajat'
```

- De adaugat în clasă:

```
public:
      Angajat(){};

protected:
    double m_dSalariu;
```

Problema moştenire diamant

- Exemplu VS