**Laborator POO**

**Clasa. Set. Get. Constructori. Destructor.**

class forma {

string nume;

int h;

// !!declara!! forma f;

public:

void add(string a, int b) // f.add(„patrat”,3 );

{

nume = a; h = b;

}

// get= imi da acces la una din variabile; cout<<f.nume; cout<<f.get\_nume();

string get\_nume() { return nume; } // cout << f.get\_nume();

// set = imi permite sa modific valorile; f.nume=”triunghi” ar da eroare; f.set\_nume(”triunghi”); e corect.

void set\_nume(string a) { nume = a; } // f.set\_nume(„triunghi”);

**Mostenirea claselor ( inca nu e aprofundat )**

**Ex:** class A{

**MASTER :**

class A {}

class B : public A

class C

int x ;

public : void citire () { cin>>x ;}

};

class B : public A {

public : void citire() { cin>> x; } 404

} ;

class C {

A ob ;

public :

void citire () { cin>> ob.x; } 404

};

**//Exemple de functii ;**

class forma {

string nume;

int h;

public:

void add(string, int);

string get\_nume();

int get\_h();

void set\_nume(string);

void set\_h(int);

**// constructori** (SUPRAINCARCAREA FUNCTIILOR)

/\*forma() { nume = "recunoscut"; h = 0; } // <--- Prima parte de teorie la exmane;

forma(string);\*/

forma(string, int); //<--- Isi aloca singuri memorie. Set-ul nu face asta

**// destructor** ~forma();

}o; // !!! IMPORTANT !!! Daca avem o variabila globala si il omitem ( in ordinea de apel ) la examen nu luam niciun punct din 0.5

void forma::add(string s, int a) { nume = s; h = a; }

string forma::get\_nume() { return nume; }

int forma::get\_h() { return h; }

void forma::set\_nume(string s) { nume = s; }

void forma::set\_h(int x) { h = x; }

**\*Declarari + Default-uri\***

forma::forma(string s) { nume = s; h = 0; }

forma::forma(string a = "", int b = 0) { nume = a; h = b; } \*/ daca in apelare <<forma x,y("triunghi",3)>> x ia valorile default daca nu are parametri , iar y ia valorile date prin parametri \*/

**// destructor**

forma::~forma() { cout << "D"; }

int main()

{

forma x, y("triunghi");

//zona de vizibilitate

for ( int i = 1; i<=2; i++)

{

forma M("", i), N("", i);

}

// end\_zona de vizibilitate

forma z("dreptunghi", 3);

/\*

ORDINEA DE APEL CONSTRUCTOR & DESTRUCTOR

(se comporta ca o stiva)

Cx Cy Cz Dz Dy Dx

+ 1 var.globala ?

Co Cx Cy Cz Dz Dy Dx Do

+ zona de vizibilitate

{forma M("",1),N("",1);

}

{forma M("",2),N("",2);

}

C0 Cx Cy Cm1 Cn1 Dn1 Dm1 Cm2 Cn2 Dn2 Dm2 Cz Dz Dy Dx Do\*/}

**PROBLEMA**

**// 2.Se citeste numele si media fiecarui student din cei n ai semigrupei 1342. Sa se afiseze studentii in ordine descrescatoare a mediei.**

/\*

Student------------------Semigrupa

nume + medie dimensiune cetor de studenti

\*/

class student {

string nume;

float medie;

public:

student(string a = "", float b = 0){nume = a; medie = b;}

void citire() { cin >> nume >> medie; }

void afisare() { cout << "nume=" << nume <<" medie= "<< medie << endl; }

/// friend class semigrupa;

float get\_medie() { return medie; }

};

class semigrupa {

int dim;

student \*v;

public:

semigrupa(int n=0) { dim = n; v = new student[dim]; }

~semigrupa() { delete[] v; }

void citire();

void afisare() { for (int i = 0; i <= dim; i++) { v[i].afisare(); } }

void sortare();

};

void semigrupa::citire() {

cin >> dim;

v = new student[dim]; // Alocam memorie pentru dimensiunea citita;

for (int i = 0; i < dim; i++) { v[i].citire(); }

}

void semigrupa::sortare() {

int i, j;

for (i = 0; i < dim; i++)

for (j = i + 1; j < dim; j++)

/\*if (v[i].medie < v[j].medie) //medie e privat ; mergem in classa student si bagam friend class semigrupa;

swap(v[i], v[j]);\*/

if (v[i].get\_medie() < v[j].get\_medie())

swap(v[i], v[j]);

}

int main()

{

semigrupa s1342; // are dimensiune default = 0; semigrupa s1342(20) ar fi avut 20;

s1342.citire();

s1342.afisare();

s1342.sortare();

s1342.afisare();

}