***TEST Nr. 25***

***pentru atestarea nr.1 la disciplina ”Programare orientată pe obiecte”***

1. Formulați definiția obiectului în POO. **(5 p.)**

***Obiectul*** în programare reprezintă modulul de program care îndeplineşte următoarele cerinţe principale:

•întruneşte date(caracterizează proprietăţile obiectului) şi operaţiileasupra lor (se mai numesc metodeşi caracterizează comportarea obiectului sau posibilităţile obiectului, adică ceea ce poate face obiectul);

•posedă proprietăţile de moştenire, încapsulare şi polimorfism.

2. Enumerați tipurile de acces la membrii clasei și explicați modul de funcționare a fiecăruia. **(10 p.)**

Declararea câmpurilor şi a funcţiilor membre se repartizează pe secţiuni în dependenţă de regimul dorit de accesare a lor. Acest regim este stabilit prin utilizarea a trei specificatori de acces: public, protected, private.

***Private (privat)*** – cel mai dur regim de accesare din cele trei. Câmpurile şi funcţiile membre private sunt accesibile numai din interiorul acestei clase, adică acces la ele au numai funcţiile membre ale sale, sau funcţiile de tip prieten (friend) al acestei clase.

***Protected(protejat)*** este puţin mai liber în comparaţie cu regimul private.El stabileşte că câmpurile şi funcţiile membre ale clasei definite vor fi accesibile şi din interiorul claselor derivate ale clasei definite.

***Public*** este cel mai liber din cei trei.El admite accesarea câmpurilor şi a funcţiilor membre ale acestei secţiuni din orice loc al programului unde va fi vizibil obiectul concret al clasei definite.

3. Formulați definiția constructorului, regulile de scriere a constructorilor, tipurile de constructori, forma generală de declarare a constructorului unei clase. **(10 p.)**

***Constructorul*** este funcţia membră specială a clasei, destinată pentru iniţializarea fiecărui obiect al acesteia. Iniţializarea se face imediat după crearea obiectului (imediat după declararea variabilei de tip clasă), adică după ce el a fost declarat şi lui i s-a alocat memorie.

Exemple de acţiuni puse în responsabilitatea ***constructorilor*** pot fi următoarele: • iniţializarea câmpurilor; • alocarea de memorie şi iniţializarea câmpurilor de tip pointer; • deschiderea unor fişiere;

***Constructorul*** unei clase are următoarele particularităţi:

• numele constructorului trebuie să fie identic cu numele clasei (obligatoriu);

• pentru o clasă se pot defini mai mulţi constructori, fiecare iniţializând obiectele în moduri diferite. Constructorii trebuie să se deosebească între ei prin numărul de argumente sau tipul argumentelor;

• constructorul nu returnează nici o valoare şi deci, spre deosebire de celelalte funcţii, îi lipseşte din antet sau prototip descrierea tipului valorii returnate;

• constructorul poate fi apelat explicit ca o funcţie membră obişnuită. Dacă nu se apelează nici un constructor la crearea unui obiect, sistemul singur va apela unul din ei, şi anume – constructorul implicit.

Există 3 tipuri de constructori:

1) constructorul implicit;

2) constructori cu parametri;

3) constructorul de copiere.

Forma generală:

class <Nume\_clasa>

{

. . .

Nume\_clasa(<lista\_parametri>) {//instructiuni}

. . .

}<lista de obiecte>;

4. Găsiţi erori în următorul fragment de program şi explicaţi cum ele pot fi corectate:

class Corp\_Solid

{

public:

//double ~Corp\_Solid(double,double,int);

private:

double volum;

double densitate;

int cod;};

//dacă nu creăm noi un constructor, clasa îl va rea singur, //de aia, căutăm greșele în destructor. Destructorul nu //returnează vreun tip de date și nu poate //primi //parametri.

//corect va fi :

~Corp\_Solid();

private:

double volum;

double densitate;

int cod;}; **(10 p.)**

5. Fie dat următorul fragment de program:

class Poligon\_Regulat

{

char Denumire[40];

int Nr\_laturi;

double Lungime\_latura;

public:

double GetLatura() {return Lungime\_latura;};

. . . . . . . . . . . . .

};

main()

{

Poligon\_Regulat i(”Patrat”);

Poligon\_Regulat j(”Hexagon”,6,12.5);

j.Afisare();

}

Adăugați în clasa dată descrierile necesare, pentru ca instrucțiunile din funcția **main()** să lucreze corect. **(15 p.)**

#include <iostream>

#include <memory>

#include <initializer\_list>

#include <iostream>

#include <list>

#include <iterator>

using namespace std;

class Poligon\_Regulat

{

char Denumire[40];

int Nr\_laturi;

double Lungime\_latura;

public:

Poligon\_Regulat(char Denumire)

{

cout << Denumire << endl;

};

Poligon\_Regulat(char Denumire, int nr\_laturi, double Latura)

{

Nr\_laturi = nr\_laturi;

Lungime\_latura = Latura;

cout

<< Latura << endl;

}

double GetLatura()

{

return Lungime\_latura;

};

void Afisare()

{

cout << Nr\_laturi << endl;

}

};

main()

{

Poligon\_Regulat i('Patrat');

Poligon\_Regulat j('Hexagon', 6, 12.5);

j.Afisare();

double GetLatura();

}

6. Să se definească clasa **„Marfă”** cu proprietăţile: denumirea mărfii, cantitatea, unitatea de măsură a mărfii (bucăți, kg etc.) şi prețul mărfii pentru o unitate de măsură.

Să se definească constructorul fără parametri.

Să se definească constructorii cu parametri:

a) cu un parametru,

b) cu trei parametri,

c) cu patru parametri.

Să se definească constructorul de copiere şi destructorul.

Să se definească funcţiile membre:

a) de afişare a informaţiei despre marfă la ecran,

b) de calculare a costului întregii cantități de marfă,

c) de comparare a mărfii cu o altă marfă, transmisă acestei metode ca parametru: -1 – este mai mică, 0 – sunt egale, +1 – este mai mare (în sensul valorii întregului lot de marfă).

Să se declare în funcţia **main()** două obiecte de tip marfă şi să se afişeze la ecran acea marfă care este mai valoroasă, apelând metoda **c)**. **(30 p.)**

#include<iostream>

#include<cstring>

using namespace std;

class Marfa

{

private:

char Denumire[20];

double Cantitate;

char Unitate\_masura[20];

double Pret;

public:

Marfa ();

Marfa (char\*);

Marfa (char\*, double, char\*);

Marfa (char \*, double, char\*, double);

Marfa (Marfa &);

~Marfa ();

void Afisare ();

void CalculareCantitate (double);

void ComparareMarfa (Marfa);

};

Marfa::Marfa()

{

strcpy(Denumire, "Necunoscut");

strcpy(Cantitate, "-");

strcpy(Unitate\_masura, "Necunoscut");

Pret=0;

}

Marfa::Marfa(char \*D)

{

strcpy (Denumire, D);

strcpy (Cantitate,"-");

strcpy (Unitate\_masura, "Necunoscut");

Pret=0;

}

Marfa::Marfa (char \*D,double C, char \*U )

{

strcpy (Denumire, D);

strcpy (Cantitate, C);

strcpy (Unitate\_masura, "U");

Pret=0;

}

Marfa::Marfa(char \*D, double C, char \*U, double \*P)

{

strcpy (Denumire, D);

strcpy (Cantitate, C);

strcpy (Unitate\_masura, "U");

Pret=P;

}

Marfa::Marfa (Marfa &x)

{

strcpy (Denumire, x.denumir);

strcpy (Cantitate, x.cantitate);

strcpy (Unitate\_masura, x.unitate);

Pret=x.pret;

}

Marfa::~Marfa()

{

cout <<"Obiectul "<<Denumire<<" a fost distrus"<<endl;

}

void Marfa::Afisare()

{

cout <<"Denumirea:"<<Denumire<<endl;

cout <<"Cantitatea:"<<Cantitate<<endl;

cout <<"Unitatea de masura:"<<Unitate\_masura<<endl;

}

void Marfa::CalculareCantitate (double x)

{

x=Pret\*Cantitate;

}

void Marfa::CopiePret(Marfa x)

{

Pret=x.Pret;

}

void Marfa::ComparareMarfa( m)

{

}

main()

{

Marfa a; ///constructor fara parametru

a.Afisare()

Marfa b(Pere,5,kg,8);

b.Afisare();

Marfa c(prune,10,kg);

c.Afisare();

Marfa d(Visine);

}

}

7. Declaraţi clasele din ierarhia de moştenire de mai jos. Fiecare clasă trebuie să aibă constructor.

Latură

Triunghi

Piramidă

triunghiulară

Prizmă

triunghiulară

**(20 p.)**

class latura {

int x;

public:

latura(int x;

~latura();

void afisare();

};

class triunghi

{

char \*tip;

int laturi;

public:

ttriunghi (char \*t, int \*l)

~triunghi();

void afisare(); };

**BAREM DE NOTARE**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nota** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Punctaj acumulat** | **E** | **0-10** | **11-20** | **21-29** | **30-45** | **46-60** | **61-75** | **76-85** | **86-95** | **96-100** |