Grile.poo

Programare Orientata Obiect (Academia de Studii Economice din Bucure□ti)

1. Cu ce secventa de operatori este echivalata urmatoarea instructiune: a1 = 10 += a2?
a. a1.operator=(operator+=(a2,10));
b. a1.operator=(operator+=10,a2))
c. a1.operator=(a2.operator+=(10));
d. operator=(a1,a2.operator+=(10));

2. O functie independenta declarata friend in domeniul public dintr-o clasa si care primeste ca parametru o referinta la un obiect al clasei respective are acces
a. la toti membrii, dar ii poate dar consulta, nu si modifica;
b. la membrii public si protected;
c. doar la membrii public;
d. la toti membrii
e. la membrii protected;

3. Ce se apeleaza in codul urmator?
Student s; Student stud=s;

a. constructor fara parametri + operator=
b. constructor fara parametri + constructor de copier
c. constructor fara parametri + constructor cu parametri
d. constructor cu parametric + constructor de copiere

4. Daca avem o clasa derivate si instantiem un obiect din aceasta atunci:
a. partea din obiect, ce este mostenita din clasa parinte trebuie sa fie create prima
b. partea din obiect, ce este mostenita din clasa parinte nu trebuie sa fie construita
c. niciun raspuns prezentat
d. partea din obiect ce este specifica clasei derivate nu trebuie sa fie construita
e. partea din obiect ce este specifica clasei derivate trebuie sa fie create prima

5. Care din urmatoarele nu se mostenesc?
a. supraincarcarea operatorului +
b. niciun raspuns prezentat
c. constructorii si destructorii
d. datele membre publice
e. functiile care returneaza void

6. Constructorul de copiere se apeleaza atunci cand:
a. se instantiaza un obiect cu valori predefinite
b. se instantiaza un obiect cu valori constante
c. se instantiaza un obiect nou pe baza unui obiect existent
d. se modifica un obiect existent pe baza unu alt obiect existent
e. se instantiaza un obiect cu valori date de catre utilizator

7. Clasele ce permit parametrizarea tipurilor de date asociate unor variabile membru sunt numite:
a. friend
b. derivate
c. template
d. complexe
e. virtuale

8. Care dintre urmatoarele afirmatii despre destructor este adevarata?
a. o clasa poate avea unul sau mai multi destructori
b. poate avea, daca este cazul, unul sau mai multi parametric
c. se apeleaza intotdeauna implicit
d. se apeleaza explicit (prin delete) la pointeri la obiecte

9. Stream-urile standard sunt:
a. cerr
b. cout
c. toare variantele
d. cin

10. O metoda dintr-o clasa derivata care are acelasi nume cu o alta din clasa de baza:
a. niciun raspuns prezentat
b. va suprascrie metoda din clasa de baza
c. va fi executata imediat doar dup ace metoda din clasa de baza isi va termina executia
d. va genera un mesaj de eroare
e. va fi suprascrisa de catre metoda din clasa de baza

11. Care este diferenta intre rolul operatorului = si cel al constructorului de copiere?
a. constructorul de copiere creeaza un nou obiect, operatorul = lucreaza cu doua obiecte existente
b. operatorul = creeaza un nou obiecr cu aceleasi valori, constructorul de copiere copiaza doua obiecte
c. operatorul = dezaloca mai intai spatial de memorie afferent numelui
d. nu exista nici o diferenta, ambele functii seteaza un obiect cu date din alt obiect

12. In ce consta problema mostenirii in romb (a diamantului)?
a. derivarea unei clase din mai multe clase care au o baza comuna
b. derivarea unei clase din doua alte clase
c. derivarea unei clase din cel putin trei clase
d. niciuna din variantele prezentate
e. utilizarea de clase abstracte pentru a evita mostenirile multiple
f. utilizarea de interfete pentru a evita mostenirile multiple

13. Manipulatorii sunt:
a. alte functii special utilizate pentru formarea sirurilor de baiti
b. functii ce contin obligatoriu un constructor fara parametrii
c. atribute declarate in zona public

d. siruri de caractere

14. Fie declaratia:

Class c1 { /* … */ };
Class c2 : public c1 { /* … */ };

Clasa c2 fata de c1 este:
a. friend;
b. virtuala;
c. de baza;
d. declarata eronat, in loc de semnul : trebuia pus operatorul de rezolutie ::
e. derivata;

15. Fie o clasa D care mosteneste clasa B, ambele clase avand cate un destructor. Sa se precizeze, in cazul dezalocarii unui obiect de tipul D, care destructor se va executa primul?
a. depinde de instructiunile existente in destructori
b. nu exista un criteriu precis, acest lucru facandu-se aleator
c. al clasei care are definiti mai multi constructori
d. al clasei D, pentru zona specifica
e. al clasei B, pentru zona mostenita

16. Un instrument performat prin care se realizeaza polimorfismul il constituie:
a. functiile virtuale
b. constructorii
c. functiile inline
d. functiile friend
e. destructorii

17. Posibilitatea definirii unui obiect ca fiind o extensie a altuia:
a. este data de polimorfism
b. este asigurata prin mostenire
c. este data de existent claselor virtuale
d. este nepermisa
e. este data de încapsulare

18. Ce reprezinta this in interiorul constructorului unei clase C++?
a. pointer ce gestioneaza adresa obiectului construit
b. variabila optionala prin care se pot indica ce variabile sunt attribute si care nu
c. pointer ce gestioneaza adresa obiectului distrus
d. valoarea obiectului care apeleaza metoda constructor
e. nu se poate folosi this in constructor

19. O functie declarata friend in clasa de baza:
a. are acces pe zonele public si protected ale clasei derivate
b. ramane friend si are acces pe toata clasa derivata
c. nu are acces pe zona private mostenita in clasa derivata
d. daca derivarea este public, functia are acces pe toata clasa derivata
e. ramane friend in clasa derivata, pentru partea mostenita din baza

20. Care din urmatoarele afirmatii este adevarata privind supraincarcarea operatorilor?
a. nu se supraincarca operatorii . si ziseof()
b. cardinalitatea operatorilor se poate modifica in urma implementarii
c. asociativitatea operatorilor se modifica in functie de implementarea aleasa
d. precedent si directia de evaluare se poate modifica in urma supraincarcarii operatorilor

21. Daca o clasa derivata foloseste specificatorul public pentru mostenire, atunci:
a. membrii private din clasa de baza devin protected in clasa derivata
b. membrii publici din clasa de baza sunt inaccesibili in clasa derivata
c. membrii protejati din clasa de baza devin publici in clasa derivata
d. membrii publici din clasa de baza raman publici in clasa derivata
e. niciun raspuns prezentat

22. Polimorfismul se poate realiza prin
1- incapsulare
2- functii virtuale
3- supraincarcarea functiilor
4- derivare

Variantele corecte sunt:
a. 1+2+3+4
b. 2+3+4
c. 2+3
d. 1+2+3
e. 3

23. Un operator C++ obisnuit care se comporta intr-un mod special pentru un tip de data definit de utilizator se numeste?
a. supraspecializat
b. incapsulat
c. niciun raspuns prezentat
d. clasificat
e. supraincarcat

24. Se considera secventa de cod din imaginea alaturata. In programul principal main() se declara un obiect de tipul Magazin.
Care din urmatoarele instructiuni este incorecta?

```
class Magazin{
private:
    int nr_preturi;
    float* preturi;
public:
    .....
        float operator[](int pozitie)
    {
            if (pozitie >= 0 && pozitie < nr_preturi
                return preturi[pozitie];
            else
                throw "Pozitia este incorecta";
        }
    .....
}
```

a. float b = m[0];

b. m[0] = 3.4

c. int a = m[0] ???

d. cout << m[0] ???

25. Fiind data secventa urmatoare, indicati raspunsul corect:

```
#include <iostream>
using namespace std;

class C1 { public:     int x;       void f() { } };

class C2 { public:     int x;       virtual void f() { } };

void main()
{
    C1 o1; C2 o2;   cout << sizeof(o1) << "  " << sizeof(o2);
}
```

a. sizeof(o1) < sizeof(o2)

● b. sizeof(o1) > sizeof(o2)

c. nu se poate declara obiecte de tip c1 si c2, deoarece f() nu are corp executabil

d. sizeof(o1) = sizeof(o2)

e. nu se poate declara obiecte de tip C2, deoarece f() este virtuala pura

26. Ce afiseaza la consola programul urmator?

```
#include "iostream"
using namespace std;
class Muncitor {
public:
    int cod;
    int varsta;
    Muncitor(int C, int V) {
        cod = C;
        varsta = V;
    }
    Muncitor(const Muncitor& m) {
        cod = m.cod;
        varsta = m.varsta;
    }
    void Afiseaza() {
        cout << endl << "cod " << this->cod << " - " << this->varsta << " ani";
    }
};
void main()
{
    Muncitor *pm1 = new Muncitor(1, 23);
    Muncitor *pm2 = pm1;
    pm2->cod = 55;
    pm1->Afiseaza();
    pm2->Afiseaza();
}
```

a. Cod 55-23 ani. si Cod 55-23 ani

b. Programul este gresit deoarece nu este definit operatorul = pentru clasa Mucitor

c. Cod 1-23 ani. si Cod 55-23 ani

d. Cod 1-23 ani. si Cod 1-23 ani

27. Secventa urmatoare va afisa?

```
#include <iostream>
using namespace std;
class B
{
public:
    int x;
    B(int v = 0) :x(v) {}
    virtual int f() { return x; }
};

class D : public B
{
public:
    D(int v) : B(v) {}
    int f() { return 2 * x; }
};

B& select(B& b) { return b; }

void main()
{
    B b, *pb;
    D d(5);
    b = d;      cout << b.f() << " ";
    pb = &d;    cout << pb->f() << " ";
    cout << select(d).f();
}
```

a. 0 5 5

b. 5 5 5

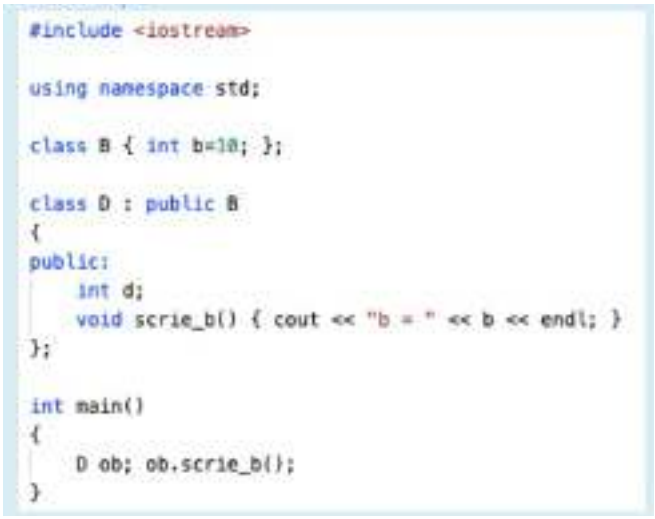
c. 5 10 10
d. 10 10 10
e. 5 10 5

28. Care din urmatoarele afirmatii este corecta pentru definitia clasei din imaginea alaturata?

```
#include "iostream"
using namespace std;
class Automobil
{
private:
    char denumire[20];
    static int nrAutomobile;
public:
    int capacitate;
    const int serie;
    Automobil() :serie(++nrAutomobile)
    {
        strcpy(denumire, "Nimic");
        capacitate = 1400;
    }
};
int Automobil::nrAutomobile = 0;
```

a. apar erori de compilare deoarece este gresit modul in care se initializeaza valoare campului nrAutomobile
b. apar erori de compilare deoarece este gresit modul in care se modifica valoarea campului nrAutomobile
c. apar erori de compilare deoarece este gresit modul in care se modifica valoare campului serie
d. apar erori de compilare deoarece este gresit modul in care se modifica valoare campului denumire
e. instructiunile sunt corecte

29. Fie declaratia:



```
#include <iostream>

using namespace std;

class B { int b=10; };

class D : public B
{
public:
    int d;
    void scrie_b() { cout << "b = " << b << endl; }
};

int main()
{
    D ob; ob.scrie_b();
}
```

a. prin derivare public, accesul la membrii mostenirii devine public
b. programul afiseaza valoarea lui b, deoarece b este implicit public
c. functia de acces are doar acces read-only asupra unui membru privat

d. functia scrie_b() nu are drept de acces asupra unui membru privat
e. programul afiseaza valoarea lui b, deoarece derivarea s-a facut public

30. Ce varianta de raspuns despre codul de mai jos este adevarata?

```
#include <iostream>
using namespace std;
class C
{
public:
    int x = 0;
    C(int v = 0) : x(v) {}
    C(const C& c) { cout << "\nConstructor de copiere"; }
    C& operator= (const C& c)
    {
        cout << "\nOperator de atribuire"; return *this;
    }
};

C f(C c) { return c; }

void main()
{
    C c1, c2 = c1, c3;
    c3 = f(c1); cin.get();
}
```

a. operatorul = si constructorul de copiere se apeleaza de cate doua ori fiecare
b. operator = se apeleaza o data, constructorul de copiere de doua ori
c. operator = se apeleaza o fata, constructorul de copiere de trei ori
d. operator = se apeleaza de doua ori, constructorul de copiere de trei ori
e. operator = se apeleaza o data, constructorul de copiere de patru ori

31. Fie programul:

```
class c {
int a;
public :
c();
c(const c&);
void operator =(c&);
} ;
void main() {
c a;
// instructiuni
c b=a;
// instructiuni
};
```

Linia c b=a determina:

a. o eroare, deoarece nu se permite combinarea atribuirii cu o declarative
b. executia metodei prin care se supraincarca operator=
c. apelul constuctorului implicit
d. apelul constructorului de copiere
e. executia atat a constructorului de copiere, cat si a metodei operator =

32. Ce va afisa urmatorul cod C++ in functia main?

```
int v[]{ 1, 5, 10, 20 };
int* pointer = v;
pointer++;
cout << *pointer;
```

a. Adresa unde este salvat vectorul v
b. O adresa oarecare de memorie
c. 1
d. 5
e. eroare de compilare
f. eroare de executie

33. Programul de mai jos:

```
#include <iostream>
using namespace std;

class C
{
public: int x;
      C(int  i = 0) : x(i) { }
};

void main()
{
    C a, b, v[] = { C(1), C(2), a };
    a.x = 3;  cout << v[2].x;
}
```

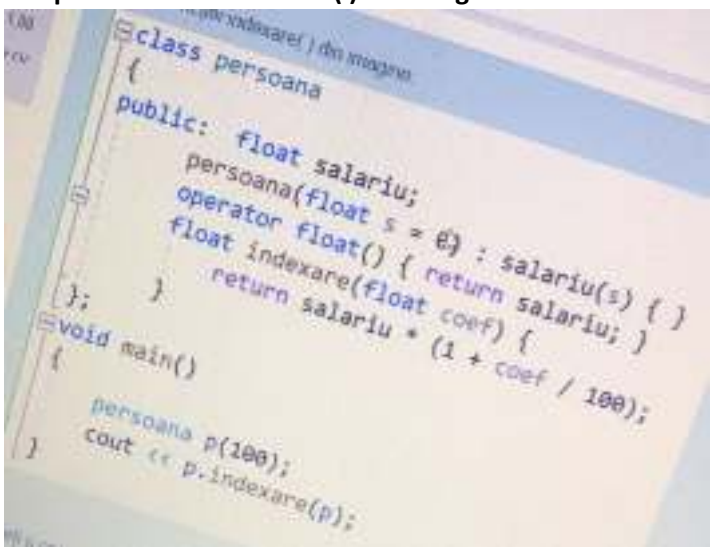
a. afiseaza 3
b. initializeaza incorect un vector de obiecte
c. afiseaza 2
d. afiseaza 0
e. afiseaza 1

34. Clasa de mai jos are membrii:

```
class c
{ float a ; void afis_a() ; };
```

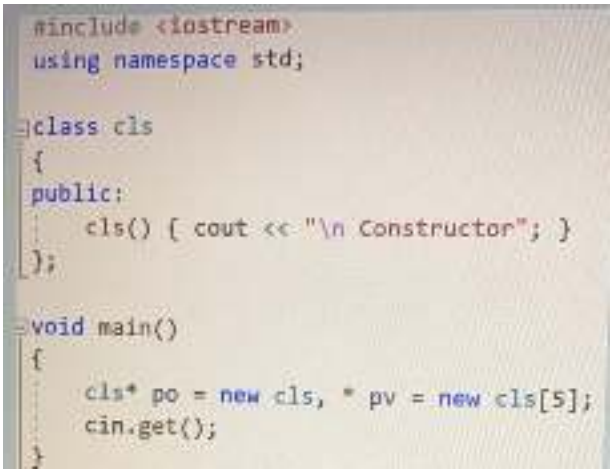
a. date private si metode publice
b. publici
c. protected
d. descrisi eronat, deoarece nu declara tipul de acces
e. privati

35.Apelul functiei indexare ( ) din imagine:



a. genereaza eroare, neexistand o supraincarcare ce primeste obiect persoana in intrare
b. se asimileaza cu indexare (int)
c. pentru adaptare la prototip apeleaza cast-ul definit de programator
d. genereaza eroare, prin folosirea recursive a obiectului p
e. se asimileaza cu indexare (void)

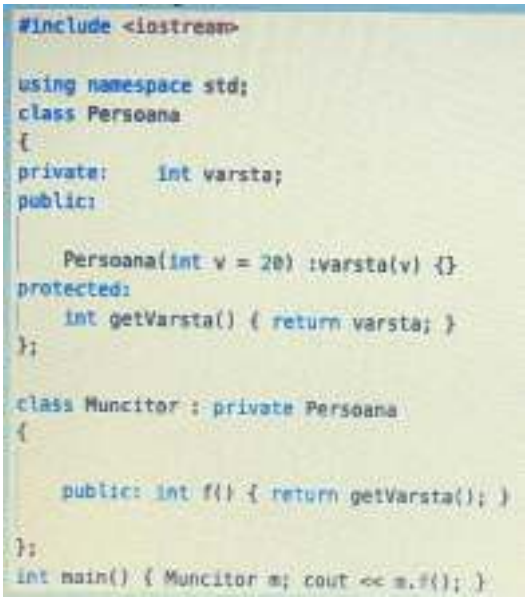
36. In programul:



Constructorul:

a. nu se apeleaza niciodata, programul lucrand cu pointeri, nu cu obiecte
b. se apeleaza de cinci ori
c. se apeleaza de 6 ori
d. se apeleaza o data, vectorul de obiecte fiind alocat de constructorul pus implicit de compilator
e. nu se apeleaza niciodata, alocarea facandu-se cu operatorul new

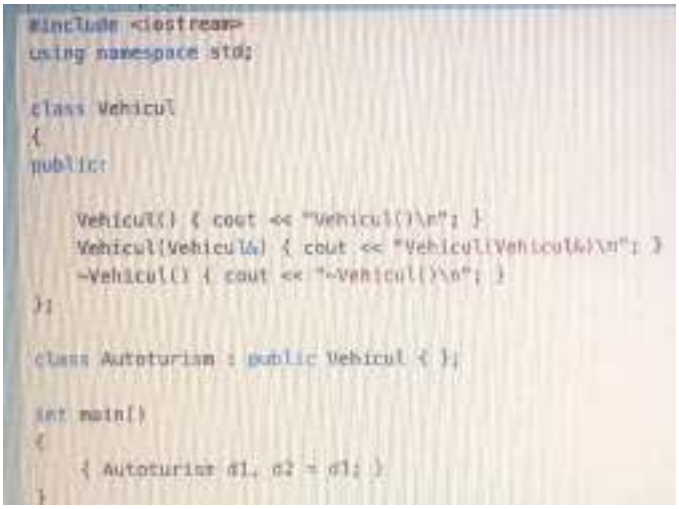
37. Se considera programul



```
#include <iostream>

using namespace std;
class Persoana
{
private:    int varsta;
public:

    Persoana(int v = 20) :varsta(v) {}
protected:
    int getVarsta() { return varsta; }
};

class Muncitor : private Persoana
{

    public: int f() { return getVarsta(); }

};
int main() { Muncitor m; cout << m.f(); }
```

Functia f() din clasa Muncitor are acces pe o zona private din clasa de baza?
a. da, printr-o metoda public sau protected mostenita public, private sau protected
b. da, doar daca getVarsta () ar fi mostenuta protected
● c. da, daor daca getVarsta () ar fi mostenita public
d. nu, deoarece getVarsta () prin derivare private devine private
e. nu, deoarece varsta este private si este mostenita private

38. Se considera programul



```
#include <iostream>
using namespace std;

class Vehicul
{
public:

    Vehicul() { cout << "Vehicul()\n"; }
    Vehicul(Vehicul&) { cout << "Vehicul(Vehicul&)\n"; }
    ~Vehicul() { cout << "~Vehicul()\n"; }
};

class Autoturism : public Vehicul { };

int main()
{
    { Autoturism a1, a2 = a1; }
}
```

Alegeti afirmatiile corecte:

(1) Constructorul clasei derivate, pus implicit de compilator, apeleaza constructorul clasei de baza
(2) Constructorul de copiere al clasei derivate, pus implicit de compilator, apeleaza constructorul de copiere al clasei de baza
(3) Destructorul clasei derivate, pus implicit de compilator, apeleaza destructorul clasei de baza

a. 1+2+3
b. niciuna din afirmatii nu se aplica in acest caz
c. 1+3
d. 1
e. 1+2

39. Exista atat forma prefixata, cat si forma postfixata pentru operatorii:
a. ++ si –
b. << si >>
c. == si !=
d. new si delete

40. Pentru a expune un membru al unei clase programului principal, in care din urmatoarele sectiuni ale unei clase trebuie declarant membrul?
a. user
b. common
c. public
d. exposed
e. unrestricted

41. Programul principal poate accesa membrii private ai unei clase?
a. doar prin intermediul altor membrii publici ai clasei
b. in mod direct
c. doar prin intermediul altor membrii private ai clasei
d. niciun raspuns prezentat

42. Obiectul este pentru clasa precum:
a. factura este pentru chitanta
b. parintele este pentru copil
c. biblioteca este pentru carte
d. presedintele este pentru Lincoin
e. Platon este pentru filosofi

43. Operatorii de comparatie sunt operatori:
a. unari
b. binari
c. temari
d. prefixati
e. postfixati

44. Un constructor implicit este:

a. niciun raspuns prezentat

b. un constructor cu valori prestabilite pentru toate argumentele

c. ambele variante prezentate

d. un constructor care nu are parametric

45. Ce reprezinta conceptul de memory leak in C++?

a. alocarea unui spatiu in memoria HEAP care sa nu mai fie referit de nici un pointer

b. initializarea incorecta a unui pointer

c. definirea gresita a constructorului de copiere

d. initializarea gresita a unui obiect

e. acest concept nu exista in c++

f. dezalocarea unui spatiu de memorie in HEAP

46. Care din variantele de mai jos afiseaza corect numele unei persoane?

```
#include <iostream>
using namespace std;

class Pers
{
    double salariu;
public:
    char nume[20];
    Pers(const char n[] = "Anonymous", double s = 0) :salariu(s)
    {
        strcpy_s(nume, n);
    }
    Pers* gA()
    {
        return this;
    }
};

void main()
{
    Pers p1 = Pers("Daniel", 5000), p2("John", 3500);
    cout << endl << (&p1)->nume;          // varianta 1
    cout << endl << &p1->nume;            // varianta 2
    cout << endl << p2.gA()->nume << "\n\n";     // varianta 3
    cout << endl << p2.this->nume << "\n\n";     // varianta 4
    system("pause");
}
```

a. 1+3

b. 1+2+3+4

c. 2+3

d. 1+2+3

e. 1+2

47. Indicati enuntul gresit despre functiile virtuale pure in C++

a. functii ce pot fi supradefinite in clasele derivate

b. clasa ce le contine este abstracta

c. functii care sunt definite fara a avea implementare (corp)
d. functii ce trebuie sa fie supraincarcate in clasa in care sunt definite
e. functii ce pot fi definite intr-o clasa

48. O metoda dintr-o clasa derivata care are acelasi nume cu o alta din clasa de baza
a. niciun raspuns prezentat
b. va fi executata imediat doar dupa ce metoda din clasa de baza isi va termina executia
c. ba genera un mesaj de eronare
d. va suprascrie metoda din clasa de baza
e. va fi suprascrisa de catre metoda din clasa de baza

49. In ipoteza ca exista definite clasele de baza B1, B2, B3 si B4, declaratia este:

```
#include <iostream>

using namespace std;

class B1 {};

class B2 {};

class B3 {};

class B4 {};

class  D : public B1, private B2, protected B3, B4 { public: int m = 0; };

int main()
{
    D d; cout << d.m;
}
```

este

Alegeți o opțiune:

- a. incorectă, pentru că la B4 nu se specifică tipul derivării;
- b. perfect validă;
- c. incompletă, deoarece clasa D nu are constructori;
- d. incompletă, deoarece clasa D nu are metode specifice;
- e. incorectă, neexistând derivare de tip protected

a. incorecta, pentru ca la B4 nu se specifica tipul derivarii
b. perfect valida
c. incompleta, deoarece clasa D nu are constructori
d. Incompleta, deoarece clasa D nu are metoda specifice
e. incorecta, neexistand derivare de tip protected

50.

```
class cls
{
    public: void* operator new(size_t dim)
    {
        cls* po = new cls[dim];
        cout << "\n Aloca obiect";
        return po;
    }
};

void main()
{
    cls* po = new cls;
    cls* pv = new cls[5];
}
```

a. o data, deoarece pentru vectori se foloseste varianta initiala a lui new
b. de zero ori, caci operatorul new nu se supraincarca
c. de doua ori, cate o data pentru fiecare pointer
d. de sase ori, cate o data pentru fiecare obiect alocat ????
e. de zero iru, caci varianta supraincarcata nu se apeleaza niciodata pentru pointeri

51. Programul din imagine afiseaza:

```
class Vector
{
    int* pe;
    int nr_c;
public:
    operator int() {
        return nr_c;
    }
    Vector(int);
};

Vector::Vector(int n)
{
    pe = new int[n];
    nr_c = n;
    while (--n) {
        pe[n] = n;
    }
}
void f(int i) {
    cout << i << endl;
}
void main()
{
    Vector x(10);
    cout << (int)x;
}
```

a. numerele de la 0 la 10
b. numerele de la 1 la 10
c. 10
d. numerele de la 0 la 9
e. 9

52. Ce afiseaza programul din imagine?

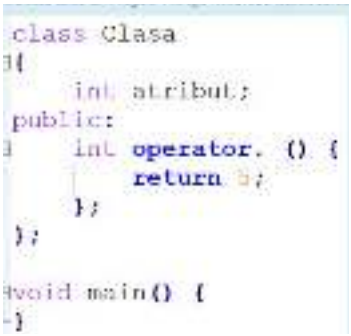
a. 200
b. 800 ???
c. eroare de compilare deoarece nu exista niciun constructor cu doi parametri in clasa Taxa
d. 1000

53. Care dintre variantele de mai jos este adevarata?

```
using namespace std;
class cls
{
public: static int x;
};

int cls::x = 0;
void main() { cls::x = 1; cls ob; ob.x = 1; }
```

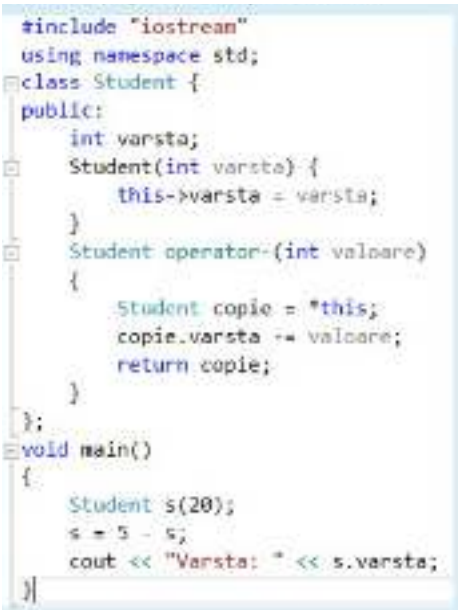
a. variabila x poate fi folosita inainte de a defini un obiect din clasa
b. variabila statica x este dublu definita, in clasa si in exterior
c. exista doua variabile x, una interna si alta externa clasei
d. nu pot exista clase fara membri nestatici
e. variabila x nu poate fi calificata pornind de la un obiect al clasei

54. Fiind data clasa din imagine. Functia realizeaza supraincarcarea operatorului " . " ?



a. nu, deoarece nu exista o variabila de tip struct in obiect
b. nu, deoarece operatorul " . " nu poate fi supraincarcat
c. nu, deoarece operatorul " . " se supraincarca numai printr-o functie friend
d. da, deoarece returneaza un int si exista o data de tip int in cadrul obiectului
e. da, respecta regulile de supraincarcare a operatorilor

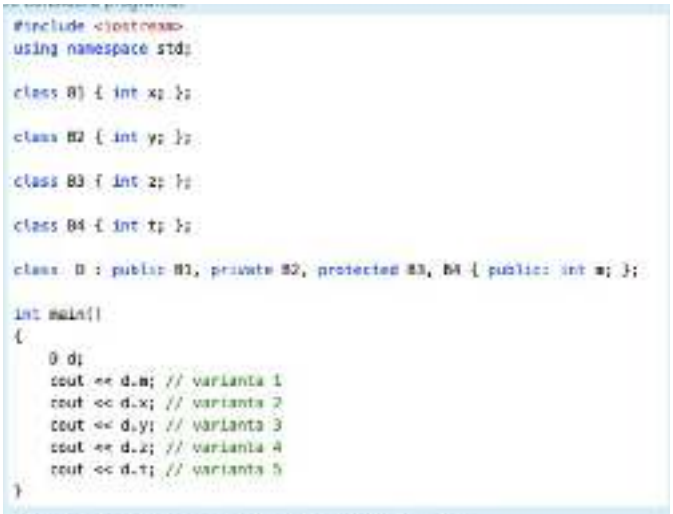
55. Ce afiseaza programul din imaginea alaturata?



```
#include "iostream"
using namespace std;
class Student {
public:
    int varsta;
    Student(int varsta) {
        this->varsta = varsta;
    }
    Student operator-(int valoare)
    {
        Student copie = *this;
        copie.varsta -= valoare;
        return copie;
    }
};
void main()
{
    Student s(20);
    s = 5 - 5;
    cout << "Varsta: " << s.varsta;
}
```

a. eroare de compilare pentru ca operatorul – nu este recunoscut
b. eroare de compilare pentru ca nu a fost supraincarcat operatorul = in clasa Student
c. varsta: -15
d. varsta: 15

56. Se considera programul



```
#include <iostream>
using namespace std;

class B1 { int x; };

class B2 { int y; };

class B3 { int z; };

class B4 { int t; };

class D : public B1, private B2, protected B3, B4 { public: int m; };

int main()
{
    D d;
    cout << d.m; // varianta 1
    cout << d.x; // varianta 2
    cout << d.y; // varianta 3
    cout << d.z; // varianta 4
    cout << d.t; // varianta 5
}
```

Variantele care permit accesul la variabile, pentru afisare sunt:
a. 1+2

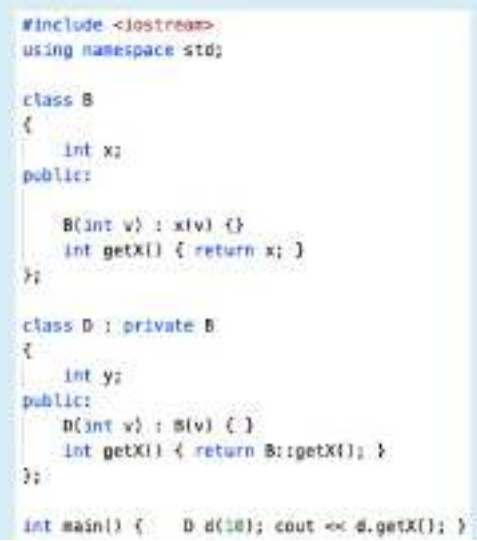
b. 1+2+5
c.1
d. 1+2+4
e. 1+2+4+%

57. Ce va afisa urmatorul program C++?

```
#include <iostream>

using namespace std;

class Base
{
public:
    int function(){ return 42; }
};

class Derived : public Base
{
public:
    int function() { return 24; }
};

int main()
{
    Base* pb = new Derived();
    cout << pb->function();
    delete pb;
}
```

a. 24
b. 0
c. 42
d. adresa unei functii
e. eroare de compilare
f. eroare de executie

58. Se considera urmatorul program:

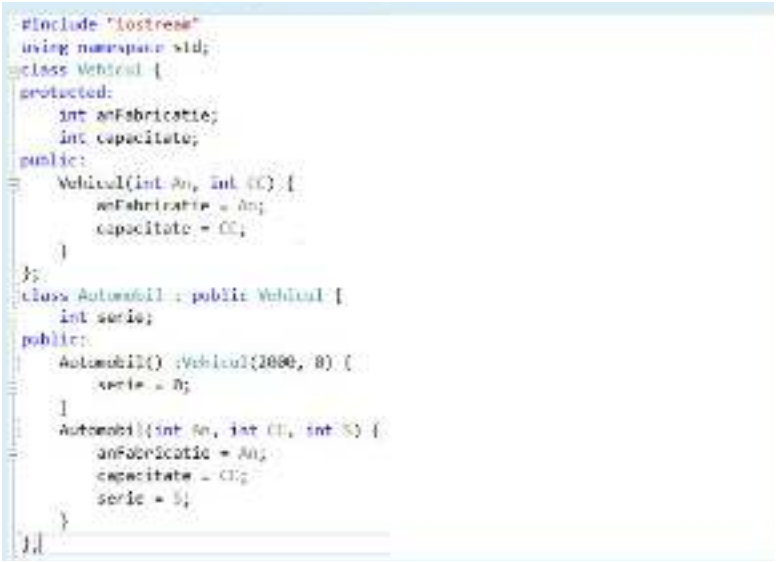


```
#include <iostream>
using namespace std;

class B
{
    int x;
public:

    B(int v) : x(v) {}
    int getX() { return x; }
};

class D : private B
{
    int y;
public:
    D(int v) : B(v) { }
    int getX() { return B::getX(); }
};

int main() {    D d(10); cout << d.getX(); }
```

Care din urmatoarele afirmatii este adevarata?

a. programul afiseaza 10
b. variabila x nu este accesibila deoarece se mosteneste private in D
c. transferul de sarcini intre constructori nu este permis, B() devine private
d. apelul B::getX() din clasa D nu este accesibil deoarece derivarea lui D este private
e. constructorul D() este apelat eronat

59. Care este cauza erorilor de compilare generate de secventa urmatoare de cod?



a. constructorul din clasa de baza este apelat gresit prin : Vehicul (2000, 0):
b. derivarea este definit gresita
c. clasa Automobil acceseaza campuri mostenite din Vehicul care sunt protejate la acces
d. constructorul cu parametri din Automobil incearca sa apeleze constructorul implicit din baza

60. Ce este gresit la urmatoarea secventa?

```
class Bloc {
private:
    const int id;
    int nr_etaje;
public:
    Bloc() {
        this->id = 2;
        this->nr_etaje = 4;
    }
}
```

a. constructorul creaza doar blocuri cu 4 etaje si cu id-ul 2
b. clasa are doar doua atribute
c. atributul nr_etaje nu este declarat constant asemanator cu atributul id
d. atributul constant este initializat in interiorul constructorului

Seminarul 1 POO

Minoiu Maria-Magdalena -- grupa 1055

1.Ce se va afisa pe ecran?

```
#include<iostream>
using namespace std;
void main()
{
    int var = 7;
    int* b = &var;
    cout << var << *b ;
}
```

a) Programul nu va rula;
b) Valoarea variabilei “var” urmata de adresa sa;
c) Adresa variabilei “var” urmata de valoarea sa;
d) De doua ori valoarea variabilei “var”;

2.Ce se va afisa pe ecran?

```
#include<iostream>
using namespace std;
void main()
{
    int var = 7;
    int* b = &var;
    cout << var << b ;
}
```

a) De doua ori valoarea variabilei “var”;
b) De doua ori adresa variabilei “var”;
c) Valoarea variabilei “var” urmata de adresa sa;
d) Adresa variabilei “var” urmata de valoarea sa;

3.Ce se va afisa pe ecran?

```
#include<iostream>
using namespace std;
void functie(int* p)
{
    *p = 4;
}
void main()
{
    int var = 7;
    cout << var;
    functie(&var);
    cout << " Dupa apel: " << var ;
}
```

a) 7 Dupa apel: 4
b) 4 Dupa apel: 7
c) 4 Dupa apel: 4

d) 7 Dupa apel: 7

4.Ce se va afisa e ecran?

```
#include<iostream>
using namespace std;
void functie(int p)
{
    p = 4;
}
void main()
{
    int var = 7;
    cout << var;
    functie(var);
    cout << " Dupa apel: " << var ;
}
```

a) 7 Dupa apel: 4
b) 4 Dupa apel: 7
c) 4 Dupa apel: 4
d) 7 Dupa apel: 7

5.Care varianta este corecta daca se doreste copierea sirului ION in prof.nume?

a) strcpy(“ION”,prof.nume);
b) strcpy(‘ION’,prof.nume);
c) strcpy(prof.nume,”ION”);
d) strcpy(prof.nume,’ION’);

6.Care este varianta corecta daca se doreste sa se afle lungimea sirului ION?

a) strlen(“ION”);
b) strlen(“ION”)+1;
c) strlen(‘ION’);
d) strlen(‘ION’)+1;

7.In cazul alocarii memoriei, care este operatorul echivalent cu malloc in C++?

a) delete;
b) free;
c) new;
d) add;

8.Cati octeti ocupa un pointer?

a) 2 octeti;
b) 4 octeti;
c) 8 octeti;

d) 16 octeti;

9.De ce s-au folosit parantezele drepte in urmatoarea linie de cod?

```
prof.nume = new char[strlen("Ion") + 1];
```

a) Deoarece operatorul new impune acest lucru;
b) Deoarece functia strlen impune acest lucru;
c) Deoarece este vorba de un sir de caractere;
d) Deoarece s-au folosit deja parantezele rotunde;

10.Alegeti varianta corecta de afisare care lipseste din functia *afisare* a programului urmator:

```
#include<iostream>
using namespace std;
struct profesor
{
    char* nume;
    int varsta;
    float salariu;

};
void afisare(profesor prof)
{
    ...
}
void main()
{
    profesor prof;
    prof.nume = new char[strlen("Ion") + 1];
    strcpy(prof.nume, "Ion");
    prof.varsta = 45;
    prof.salariu = 1200;
    afisare(prof);
}
```

a) cout << "Profesorul " << *prof.nume << " are varsta de " << prof.varsta << " ani si salariul(lei) " << prof.salariu ;
b) cout << "Profesorul " << prof.nume << " are varsta de " << prof.varsta << " ani si salariul(lei) " << prof.salariu ;
c) cout << "Profesorul " << nume << " are varsta de " << varsta << " ani si salariul(lei) " << salariu ;
d) cout << "Profesorul " <<*nume << " are varsta de " << varsta << " ani si salariul(lei) " << salariu ;

11.Ce functie are operatorul * in structura:

```
char* nume;
```

a) Extragere adresa;
b) Referire membru structura;
c) Dereferentiere;
d) Definire variabila pointer;

12. Ce functie are operatorul * in structura:

```
*p = 4;
```

a) Extragere adresa;
b) Referire membru structura;
c) Dereferentiere;
d) Definire variabila pointer;

13.Ce functie are operatorul *&* in structura:

```
b = &var;
```

a) Extragere adresa;
b) Referire membru structura;
c) Dereferentiere;
d) Definire variabila pointer;

14.Prin ce este caracterizata o variabila de tip POINTER?

a) Lungime si tip;
b) Lungime si nume;
c) Nume si tip;
d) Nume si adresa;

Raspunsuri:

1 - d

2 - c

3 - a

4 - d

5 - c

6 - a

7 - c

8 - b

9 - c

10 - b

11 - d

12 - c

13 - a

14 - c

GRILE POO – SEMINAR 02

Minoiu Maria-Magdalena

1.Care este rolul structurii *#include<string>***?**

a)Faciliteaza citirea sau scierea fluxului standard de intrare/iesire;

b)Este functia principala a programului;

c)Faciliteaza folosirea anumitor programe din bibliotecile limbajelor pe baza numelor acestora;

d)Este biblioteca necesara lucrului cu stringuri;

2.Ce se intampla cand se executa structura :

```
string sir = "Seminar 02";
sir += "-pointeri";
```

*a)Se adauga sirul “-pointeri” la sirul “Seminar 02”, se produce concatenare;

b)Se copiaza sirul “-pointeri” peste sirul “Seminar 02”;

c)Nu se intampla nimic pentru ca operatia “+=” nu este valabila pentru string;

d)Se adauga adresa sirului “-pointeri” la sirul “Seminar 02”;

3.Care dintre urmatoarele variante este echivalenta cu structura `sir += "-pointeri";` **?**

a) sir-append["-pointeri"];

b) sir.apend("-pointeri");

c) sir.append["-pointeri"];

d) sir.append("-pointeri");

4.Ce afiseaza urmatoarea structura:

```
string sir = "Seminar 02";
sir.append("-pointeri");
string sir2=sir.substr(sir.find('-') + 1, 8);
cout << sir2 << endl;
```

a) sirul “-pointer”;

b) sirul “-pointeri”;

c) sirul “pointeri”;

d) sirul “pointer”;

5.Ce returneaza urmatoarea structura:

```
string sir = "Seminar 02";
sir.append("-pointeri");
string sir2=sir.substr(sir.find('-') , 8);
cout << sir2 << endl;
```

a)Returneaza sirul “-pointer”;

b)Returneaza sirul “-pointeri”;

c)Returneaza sirul “pointeri”;

d)Returneaza sirul “pointer”;

6.Ce returneaza urmatoarea structura:

```
string sir = "Seminar 02";
sir.append("-pointeri");
string sir2=sir.substr(sir.find('-')+1, 7);
cout << sir2 << endl;
```

a)Returneaza sirul “-pointer”;

b)Returneaza sirul “-pointeri”;

c)Returneaza sirul “pointeri”;

d)Returneaza sirul “pointer”;

7. De ce nu este indicata alocarea statica in cazul vectorilor?

a)Pentru ca se foloseste prea multa memorie;

b)Pentru ca este prea mult de scris;

c)Pentru ca nu este recunoscuta de compilator;

d)Pentru ca lucrul cu vectorii implica alocarea dinamica;

8.Ce se va afisa dupa executia structurii:

```
int v[100];
cout << v << endl;
```

a)Primul element din vectorul v;

b)Adresa de unde incepe vectorul v;

c)Elementele vectorului v;

d)Eroare de executie;

9. Ce se va afisa dupa executia structurii:

```
int v[];
cout << v << endl;
```

a)Primul element din vectorul v;

b)Adresa de unde incepe vectorul v;

c)Elementele vectorului v;

d)Codul furnizeaza eroare;

10.Ce rol are penultima linie de cod din structura urmatoare:

```
int *vector;
int n = 5;
vector = new int[n];
delete []vector;
```

a)Aloca spatiu vectorului;

b)Initializeaza al cincilea element din vector;

c)Converteste al cincilea element din vector la int;

d)Initializeaza toate elementele vectorului cu valoarea 5;

11. Ce rol are ultima linie de cod din structura urmatoare:

```
int *vector;
int n = 5;
vector = new int[n];
delete []vector;
```

a)Sterge toate elementele vectorului;

b)Sterge primul element din vector;

c)Elibereaza memoria alocata pentru vector;

d)Elibereaza memoria alocata primului element al vectorului;

12. La ce este necesara directiva *#include<stdlib>* **?**

a)Faciliteaza citirea sau scierea fluxului standard de intrare/iesire;

b)Este functia principala a programului;

c)Faciliteaza folosirea anumitor programe din bibliotecile limbajelor pe baza numelor acestora;

d)Este biblioteca necesara lucrului cu functii utilizate pentru conversia valorilor numerice in sir;

13. Ce face functia *atoi* **?**

a)converteste un sir catre tipul int;

b)converteste un sir catre tipul float;

c)converteste o valoare de tip int catre un sir;

d)converteste o valoare de tip float catre un sir;

14. Ce face functia atof **?**

a)converteste un sir catre tipul int;

b)converteste un sir catre tipul float;

c)converteste o valoare de tip int catre un sir;

d)converteste o valoare de tip float catre un sir;

15.Ce face functia itoa **?**

a)converteste un sir catre tipul int;

b)converteste un sir catre tipul float;

c)converteste o valoare de tip int catre un sir de caractere;

d)converteste o valoare de tip float catre un sir de caractere;

16.Ce se va afisa pe ecran?

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    char *sir = "12345.67";
    n = atoi(sir);
    cout << "string = " << sir << endl << "float = " << n;
}
```

a)string = 12345.67
float = 12345.67

b) string = 12345.67
float = 12345

c) string = 12345
float = 12345

d) string = 12345.67
float = 12345.00

Rspunsuri:

1 - d
2 - a
3 - d
4 - c
5 - a
6 - d
7 - a
8 - b
9 - d
10 - a
11 - c
12 - d
13 - a
14 - b
15 - c
16 - b

GRILE POO – SEMINAR 3

Minoiu Maria-Magdalena

1.Ce este un CONSTRUCTOR in limbajul C++?
a) este o functie membra speciala a unei clase ce se apeleaza in mod automat pentru distrugerea unui obiect;
b)este o functie membra speciala a unei clase ce se apeleaza pentru crearea unui obiect;
c)este functia principala a programului, functia unde se construiesc obiectele;
d)este o metoda prin care se contruieste o biblioteca;

2.Ce este un DESTRUCTOR in limbajul C++?
a) este o functie membra speciala a unei clase ce se apeleaza in mod automat pentru distrugerea unui obiect;
b)este o functie membra speciala a unei clase ce se apeleaza pentru crearea unui obiect;
c)este functia principala a programului, functia unde se construiesc obiectele;
d)este o metoda prin care se distruge o biblioteca;

3.Care dintre urmatoarele este o declarare corecta a unui atribut static?
a) int (static constanta);
b) int constanta static;
c) static int constanta;
d) static(int constanta);

4.Unde este corect sa initializezi un atribut static?
a)In clasa din care face parte;
b)In functia principala a programului;
c)In afara clasei(global);
d)In antetul constructorului clasei din care face parte;

5.Unde este corect sa initializezi un atribut de tip const?
a)In clasa din care face parte;
b)In functia principala a programului;
c)In afara clasei(global);
d)In antetul constructorului clasei din care face parte;

6.Ce se afiseaza pe ecran?

```
7   class Laptop {
8   public:
9       string Marca;
10      static double tva;
11      static int nrExemplare;
12
13      Laptop(){
14          Marca = "Acer";
15          nrExemplare++;
16      }
17  };
18  double Laptop::tva = 22;
19  int Laptop::nrExemplare = 44;
20  void main()
21  {
22      Laptop L;
23      L.Marca = "Asus";
24      cout << L.Marca << endl;
25      cout << Laptop::tva << endl;
26      cout << Laptop::nrExemplare << endl;
27  }
```

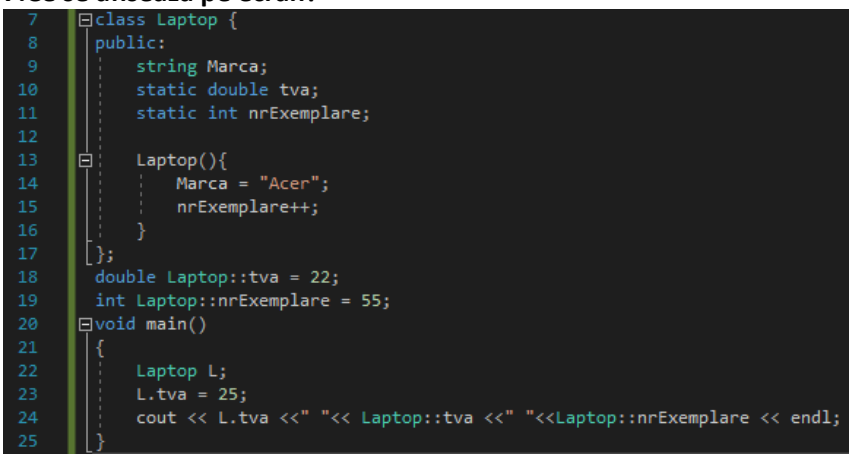
a) Acer
   22
   44
b) Asus
   22

45

c) Acer
22
45

d) Asus
22
44

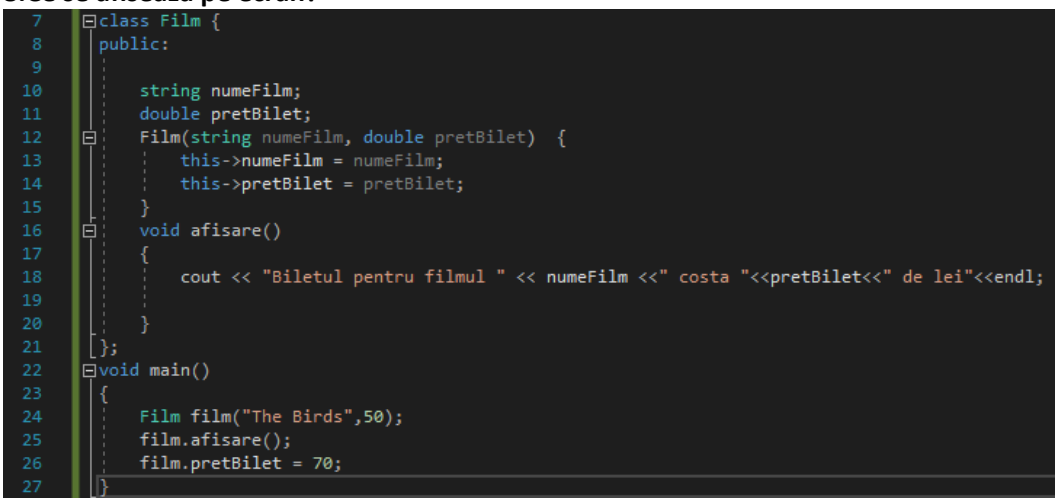
7.Ce se afiseaza pe ecran?



```
class Laptop {
public:
    string Marca;
    static double tva;
    static int nrExemplare;

    Laptop(){
        Marca = "Acer";
        nrExemplare++;
    }
};
double Laptop::tva = 22;
int Laptop::nrExemplare = 55;
void main()
{
    Laptop L;
    L.tva = 25;
    cout << L.tva <<" "<< Laptop::tva <<" "<<Laptop::nrExemplare << endl;
}
```

a)25 25 56
b)25 22 56
c)25 22 55
d)22 22 56

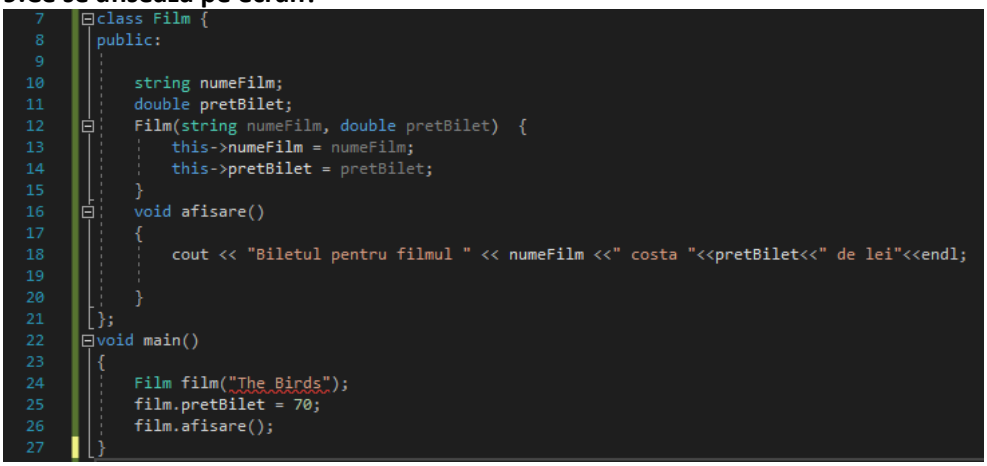
8.Ce se afiseaza pe ecran?



```
class Film {
public:

    string numeFilm;
    double pretBilet;
    Film(string numeFilm, double pretBilet)  {
        this->numeFilm = numeFilm;
        this->pretBilet = pretBilet;
    }
    void afisare()
    {
        cout << "Biletul pentru filmul " << numeFilm <<" costa "<<pretBilet<<" de lei"<<endl;

    }
};
void main()
{
    Film film("The Birds",50);
    film.afisare();
    film.pretBilet = 70;
}
```

a)Biletul pentru filmul The Birds costa 50 de lei;
b)Biletul pentru filmul The Birds costa 70 de lei;
c)Eroare de compilare;
d)Eroare de executie;

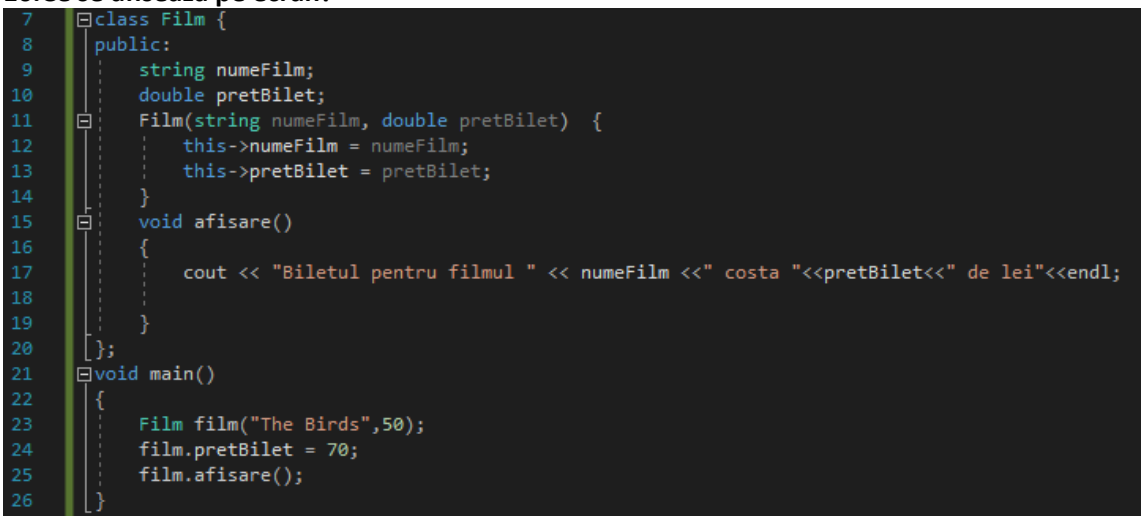
9.Ce se afiseaza pe ecran?



```
class Film {
public:

    string numeFilm;
    double pretBilet;
    Film(string numeFilm, double pretBilet)  {
        this->numeFilm = numeFilm;
        this->pretBilet = pretBilet;
    }
    void afisare()
    {
        cout << "Biletul pentru filmul " << numeFilm <<" costa "<<pretBilet<<" de lei"<<endl;

    }
};
void main()
{
    Film film("The Birds");
    film.pretBilet = 70;
    film.afisare();
}
```

a)Biletul pentru filmul The Birds costa 50 de lei;
b)Biletul pentru filmul The Birds costa 70 de lei;
c)Eroare de compilare;
d)Eroare de executie;

10.Ce se afiseaza pe ecran?



```
class Film {
public:
    string numeFilm;
    double pretBilet;
    Film(string numeFilm, double pretBilet)  {
        this->numeFilm = numeFilm;
        this->pretBilet = pretBilet;
    }
    void afisare()
    {
        cout << "Biletul pentru filmul " << numeFilm <<" costa "<<pretBilet<<" de lei"<<endl;

    }
};
void main()
{
    Film film("The Birds",50);
    film.pretBilet = 70;
    film.afisare();
}
```

a)Biletul pentru filmul The Birds costa 50 de lei;
b)Biletul pentru filmul The Birds costa 70 de lei;
c)Eroare de compilare;
d)Eroare de executie;

<u>Raspunsuri</u>

1.b)
2.a)
3.c)
4.c)
5.d)
6.b)
7.a)
8.a)
9.c)
10.b)

Grile 5 POO – Minoiu Maria-Magdalena

1.Ce tip returnat are, de obicei, o metoda de tip *set*?
a)tipul datei modificate;
b)int-returneaza 1 daca s-a facut modificarea, 0 altfel;
c)bool-returneaza true daca s-a facut modificarea, false altfel;
d)void-nu returneaza nimic, este doar o modificare;

2.Ce tip returnat are de obicei o metoda de tip *get*?
a)tipul datei accesate;
b)int-returneaza 1 daca accesarea a fost posibila, 0 altfel;
c)bool-returneaza true daca accesarea a fost posibila, false altfel;
d)void-nu returneaza nimic,este doar o accesare;

3.Ce se afiseaza pe ecran?

```
class Magazin {
    string denumire="SAMSUNG";
    int nrMagazine;
    string* adresaMagazine;
public:
    Magazin(){
        denumire = "Apple";
        nrMagazine = 2;
        adresaMagazine = new string[nrMagazine];
        adresaMagazine[0] = "Mall AFI";
        adresaMagazine[1] = "MegaMall";
    }

    void setDenumire(string denumireNoua) {
        if (denumireNoua.length() > 5) {
            denumire = denumireNoua;
        }
    }
    string getDenumire() {
        return denumire;
    }

};
void main()
{
    Magazin m;
    cout << m.getDenumire() << endl;
}
```

a)SAMSUNG;
b)Apple;
c)Eroare de executie;
d)Eroare de compilare;

4.Ce se afiseaza pe ecran?

```
class Magazin {
    string denumire="SAMSUNG";
    int nrMagazine;
    string* adresaMagazine;
public:
    Magazin(){
        denumire = "Apple";
        nrMagazine = 2;
        adresaMagazine = new string[nrMagazine];
        adresaMagazine[0] = "Mall AFI";
        adresaMagazine[1] = "MegaMall";
    }
    void setDenumire(string denumireNoua) {
        if (denumireNoua.length() > 5) {
            denumire = denumireNoua;
        }
    }
    string getDenumire() {
        return denumire;
    }
};
void main()
{
    Magazin m;
    cout << m.getDenumire() << endl;
    m.setDenumire("Nokia");
    cout << m.getDenumire() << endl;
}
```

a)Apple
Nokia
b)SAMSUNG
Apple
c)Apple
Apple
d)SAMSUNG
Nokia

5.Ce se afiseaza pe ecran?

```
class Magazin {
    string denumire="SAMSUNG";
    int nrMagazine;
    string* adresaMagazine;
public:
    Magazin(){
        denumire = "Apple";
        nrMagazine = 2;
        adresaMagazine = new string[nrMagazine];
        adresaMagazine[0] = "Mall AFI";
        adresaMagazine[1] = "MegaMall";
    }
    void setDenumire(string denumireNoua) {
        if (denumireNoua.length() > 5) {
            denumire = denumireNoua;
        }
    }
    string getDenumire() {
        return denumire;
    }
};
void main()
{
    Magazin m;
    cout << m.getDenumire() << endl;
    m.setDenumire("SAMSUNG");
    cout << m.getDenumire() << endl;
}
```

a)SAMSUNG
SAMSUNG
b)SAMSUNG
Apple
c)Apple
Apple
d)Apple
SAMSUNG

6.Ce tip returnat are un constructor in C++?
a)Tipul void pentru ca doar aloca spatiu;
b)Tipul bool pentru ca lucreaza numai cu atribute logice;
c)Nu are un tip returnat;
d)Tipul obiectului pe care il construieste;

7.Ce tip returnat are un destructor in C++?
a)Nu are un tip returnat;
b)Tipul void pentru ca doar elibereaza spatiu;
c)Tipul bool pentru ca lucreaza numai cu atribute logice;
d)Tipul obiectului pe care il distruge;

Raspunsuri

1.d)
2.a)
3.b)
4.c)
5.d)
6.c)
7.a)

Grile 6-7 POO

Minoiu Maria-Magdalena

1.Care este rolul unui constructor de copiere in limbajul C++?

a) Copiaza toate functiile publice dintr-o clasa deja existenta in una nou creata;
b) Copiaza toate functiile private dintr-o clasa deja existenta in una nou creata;
c) Initializeaza obiecte noi pe baza unor obiecte deja existente;
d) Initializeaza obiecte deja existente pe baza altor obiecte noi;

2.Cand se apeleaza constructorul de copiere?

a) Atunci cand se initializeaza la declarare un obiect nou pe baza unuia deja existent;
b) Atunci cand se doreste copierea unui obiect in alt obiect, ambele deja existente;
c) Atunci cand se afiseaza un obiect;
d) Atunci cand se aduna doua obiecte deja create;

3.Cand se apeleaza operatorul de atribuire(=)?

a) Atunci cand se initializeaza la declarare un obiect nou pe baza unuia deja existent;
b) Atunci cand se doreste copierea unui obiect in alt obiect, ambele deja existente;
c) Atunci cand se afiseaza un obiect;
d) Atunci cand se aduna doua obiecte deja create;

4.Considerand A si B doua obiecte din clasa CosCumparaturi. Ce se apeleaza la rularea urmatoarelor linii de cod?

```
15  void main()
16  {
17      CosCumparaturi A;
18      CosCumparaturi B = A;
19  }
```

a) Operatorul de atribuire;
b) Constructorul de copiere;
c) Operatorul de indexare;
d) Destructorul clasei CosCumparaturi;

5.Considerand A si B doua obiecte din clasa CosCumparaturi. Ce se apeleaza la rularea urmatoarelor linii de cod?

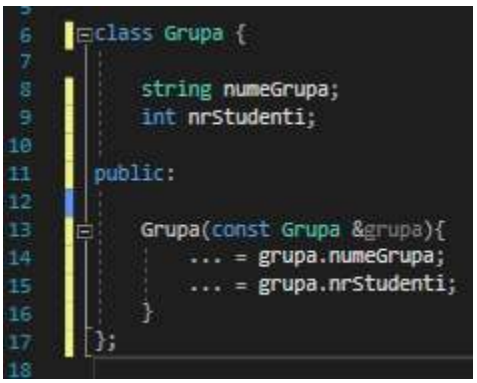
```
15  void main()
16  {
17      CosCumparaturi A;
18      CosCumparaturi B;
19      B = A;
20  }
```

a) Operatorul de atribuire;
b) Constructorul de copiere;
c) Operatorul de indexare;
d) Destructorul clasei CosCumparaturi;

6.Care varianta contine numai operatori aritmetici unari?

a) ++, --, !
b) ++, --, *
c) ++, --, /
d) ++, --, =

7.Ce trebuie scris in corpul urmatorului constructor de copiere in locurile libere pentru crearea unui nou obiect?



```
class Grupa {

    string numeGrupa;
    int nrStudenti;

public:

    Grupa(const Grupa &grupa){
        ... = grupa.numeGrupa;
        ... = grupa.nrStudenti;
    }
};
```

a) Grupa si grupa;
b) grupa si grupa;
c) this->grupa si g.grupa;
d) this->grupa si this->grupa;

8.Cum se evita un memory leak in cazul supraincarcarii operatorilor?

a) Prin eliberarea memoriei obiectului destinatie;
b) Prin alocarea in plus a unei zone de memorie pentru obiectul destinatie;
c) Prin eliberarea memoriei obiectului sursa;
d) Prin alocarea in plus a unei zone de memorie pentru obiectul sursa;

9.Fie prototipul Clasa operator=(const Clasa &sursa); **al operatorului de atribuire. De ce parametrul sursa este transmis prin referinta?**

a) Pentru ca se doreste evitarea constructorului de copiere;
b) Pentru ca se doreste evitarea recursive a operatorului de asignare;
c) Pentru ca este pointer;
d) Pentru ca se afla in this;

RASPUNSURI

1.c)

2.a)

3.b)

4.b)

5.a)

6.a)

7.d)

8.a)

9.a)

GRILE POO-SEMINAR 3

Minoiu Maria-Magdalena

1.Ce este o clasa?

a)Este o colectie de elemente de date(numite atribute) si de operatii;
b)O biblioteca specifica lucrului cu string-uri;
c)O biblioteca specifica lucrului cu obiecte;
d)O incapsulare a unei stari si a unui comportament;

2. Ce este un obiect?

a)Este o colectie de elemente de date(numite atribute) si de operatii;
b)O biblioteca specifica lucrului cu stringuri;
c)O biblioteca specifica lucrului cu obiecte;
d)O incapsulare a unei stari si a unui comportament;

3. Ce este un atribut?

a)Este o colectie de elemente de date(numite atribute) si de operatii;
b)O biblioteca specifica lucrului cu obiecte;
c)O informatie care califica obiectul caruia ii apartine;
d)O incapsulare a unei stari si a unui comportament;

4. Care este diferenta dintre o functie si o metoda?

a)Functia este declarata in clasa, metoda este declarata global;
b)Functia este declarata global, metoda este declarata in clasa;
c)Functia nu se declara pentru ca se afla implicit in program, metoda este declarata global;
d)Functia este declarata global, metoda nu se declara pentru ca se afla implicit in program;

5. De cine pot fi folosite datele si functiile membre dintr-o clasa daca dreptul de acces este *private*?

a)De catre functiile care apartin clasei si de functiile membre ale claselor derivate din clasa respectiva;
b)De catre toate functiile din program;
c)Doar de catre functiile care apartin clasei respective;
d)Nu pot fi folosite de nicio functie;

6. De cine pot fi folosite datele si functiile membre dintr-o clasa daca dreptul de acces este *protected*?

a)De catre functiile care apartin clasei si de functiile membre ale claselor derivate din clasa respectiva;
b)De catre toate functiile din program;
c)Doar de catre functiile care apartin clasei respective;
d)Nu pot fi folosite de nicio functie;

7.De cine pot fi folosite datele si functiile membre dintr-o clasa daca dreptul de acces este *public*?

a)De catre functiile care apartin clasei si de functiile membre ale claselor derivate din clasa respectiva;

b)De catre toate functiile din program;

c)Doar de catre functiile care apartin clasei respective;

d)Nu pot fi folosite de nicio functie;

8.De cine pot fi folosite datele si functiile membre dintr-o clasa daca nu se specifica dreptul de acces (in C++)?

a)De catre functiile care apartin clasei si de functiile membre ale claselor derivate din clasa respectiva;

b)De catre toate functiile din program;

c)Doar de catre functiile care apartin clasei respective;

d)Nu pot fi folosite de nicio functie;

9.Care este dreptul de acces pentru atributele din clasa Film implementata in C++?

```
 8  class Film
 9  {
10      string nume;
11      int anul_lansarii;
12      float nota_imdb;
13
14  }
```

a)public;

b)private;

c)protected;

d)secret;

10.Care atribute din clasa Tara sunt publice?

```
 8  class Tara
 9  {
10      public:
11      string nume;
12      private:
13      int nr_locuitori;
14      float PIB;
15
16  }
```

a)nume, PIB;

b)nume;

c)toate;

d)niciunul;

11.Care este declararea corecta a unui obiect din clasa Examen?

```
#include<iostream>
#include<string>
using namespace std;

class Examen{
public:
    string materie;
    int credite;
    float nota;
}
void main(){
 ...
}
```

a)Examen Examen;

b)Examen POO;

c)class Examen;

d)class Examen POO;

12.Care dintre urmatoarele variante de initializare pentru atributele obiectului *interregio* **este corecta?**

```
class Tren{
public:
    string destinatie;
    float pret_bilet;
};
int main(){
    Tren interregio;
    Tren * ptren;
    ptren = new Tren();

    ...
}
```

a)(*ptren).destinatie=”Constanta”; ptren.pret_bilet=70;

b))(*ptren).destinatie=”Constanta”; ptren->pret_bilet=70;

c) ptren.destinatie=”Constanta”; ptren.pret_bilet=70;

d) *ptren->destinatie=”Constanta”; *ptren->pret_bilet=70;

Raspunsuri:

1 - a

2 - d

3 - c

4 - b

5 - c

6 - a

7 - b

8 - c

9 - b

10 - b

11 - b

12 - b

1. Cat ocupa o variabila de tip int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

4 bytes

2.Cat ocupa o variabila de tip short int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

2 bytes

3.Cat ocupa o variabila de tip long int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

4 bytes

4.Cat ocupa o variabila de tip long long int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

4 bytes

5.Cat ocupa o variabila de tip bool (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

1 byte

6.Cat ocupa o variabila de tip float (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

4 bytes

7.Cat ocupa o variabila de tip double (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

8 bytes

8.Cat ocupa o variabila de tip long double (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

8 bytes

9.Cat ocupa o variabila de tip char (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

1 byte

10.Cat ocupa o variabila de tip pointer la int - int *(se ia un considerare un compilator din VS2015 pe 32 de biti)?

4 bytes

11.Cat ocupa o variabila de tip pointer la char- char *(se ia un considerare un compilator din VS2015 pe 32 de biti) ?

4 bytes

12.Ce reprezinta cuvantul cheie this in C++ ?

Este un pointer primit ca primul parametru de toate metodele non-statica din clasa si care contine adresa obiectului care a apelat metoda

13.Ce reprezinta this in constructor ?

Este un pointer ce contine adresa obiectului ce urmeaza sa fie construit

14.Ce reprezinta this in destructor ?

Este un pointer ce contine adresa obiectului ce urmeaza sa fie distrus

15.Ce reprezinta this intr-o functie membra ?

Este un pointer ce contine adresa obiectului care a apelat metoda.

16.Ce reprezinta this intr-o functie statica ?

O functie statica nu este asociata cu niciun obiect, deci nu primeste pointerul this.

17.Ce reprezinta this in constructorul de copiere ?

Este un pointer catre obiectul ce urmeaza sa fie creat.

18.Ce reprezinta this in cadrul supraincarcarii operatorului = ?

Este un pointer catre obiectul ce urmeaza sa fie modificat.

19.Ce este o functie membra ?

Este o functie al carei definitii sau prototip se afla in definitia unei clase ca orice alta variabila si opereaza pe orice obiect/instanta

a acelei clase si are acces la toti membrii clasei.

20.Ce este o functie statica ?

Este o functie membra, independenta de obiectele clasei.O functie statica poate fi apelata chiar daca nu exista nicio instanta a clasei.Ele sunt apelate

folosind numele clasei si operatorul de rezolutie.

21.Ce este un atribut static ?

Un atribut static este un atribut ce nu face parte din obiect, in schimb el este impartit de toate obiectele clasei.Nu poate fi initializat in interiorul unei clase

22.Ce este un atribut constant ?

Un atribut constant este un atribut ce nu poate fi modificat in timpul executiei o data ce a fost initializat.

23.Cand se poate initializa un atribut constant ?

Un atribut constant se poate initializa doar la crearea obietului, in lista de initializare a constructorului.

24.Ce reprezinta supraincarcarea ?

Supraincarcarea reprezinta abilitatea de a avea mai multe functii cu acelasi nume dar cu o lista de parametrii diferiti.

25.Care este utilitatea supraincarcarii ?

Putem avea mai multe functii cu acelasi nume, care fac acelasi lucru sau lucruri diferite, dar pentru tipuri de date diferite si chiar si tipuri de date create de noi.

26.Ce reprezinta supradefinirea ?

Supradefinirea este abilitatea de a avea 2 functii, una in clasa de baza si una in clasa derivata, cu acelasi nume si aceeasi lista de parametrii dar care sunt

diferite prin implementare.

27.Care este utilitatea supradefinirii ?

Ne permite sa avem 2 functii cu acelasi nume si aceeasi parametrii, una in clasa de baza si una in clasa derivata, care sunt diferite prin implementare.

28.Ce este o functie virtuala ?

O functie virtuala este o functie membra ce astepti sa fie redefinita intr-o clasa derivata.Cand te referi la un obiect al clasei derivate printr-un

pointer la clasa de baza apelezi varianta din clasa derivata a functiei ce a fost declarata virtuala in baza.

29.Prin ce se implementeaza conceptul de Polimorfism in C++ ?

Polimorfismul in C++ se implementeaza prin functii virtuale, supradefinire,supraincarcarea operatorilor si upcasting.

30.Care sunt modificatorii de acces si ce vizibilitate ofera datelor/functiilor membre in cazul derivarii claselor?

In C++ exsta 3 modificatori de acces:public, private si protected.

Atributele si metodele declarate publice pot fii accesate din orice parte a programului.

Atributele declarate protected pot fii accesate doar din interiorul clasei si din clasele derivate.

Atributele declarate protected pot fii accesate doar in interiorul clasei.

31.In ce context este util modificatorul de acces private ?

Modificatorul private este util la incapsularea datelor.Acesta nu permite accesul altor clase, nici macar a celor derivate, la atributele clasei.

32.Care sunt tipurile de constructori in cadrul unei clase si ce rol are fiecare?

Exista 3 tipuri de constructor:default,cu parametrii si de copiere.

Constructorul default construieste un obiect initializand atributele cu valori constante, ce nu sunt primite ca parametrii.

Constructorul cu parameterii construiecte un obiect cu parametrii primiti de la apelator.

Constructorul de copiere creaza copii a unor obiecte care sunt transmise sau returnate prin valoare sau creaza un obiect nou, initializand atributele cu valorile altui obiect.

33.Care este diferenta intre rolul operatorului = si cel al constructorului de copiere?

Operatorul = se apeleaza pentru 2 obiecte existente deja.

Constructorul de copiere creaza obiectul si atribuie atributelor valorile altui obiect deja existent.

34.Cand este apelat constructorul de copiere ?

Constructorul de copiere se apeleaza atunci cand transmitem sau returnam obiecte prin valoare sau cand vrem sa construim un obiect nou care sa fie o copie a unui alt obiect deja existent.

35.Cand este apelat operatorul = ?

Operatorul = este apelat atunci cand vrem sa atribuim unui obiect deja existent valorile unui obiect deja existent.

36.Ce este un memory leak ?

Un memory leak se produce atunci cand nu dezalocam spatiu pentru un obiect de care nu mai avem nevoie.

37.Ce este un dangling pointer?

Este un pointer ce indica spre o zona de memorie ce a fost eliberata deja.

38.Care este rolul destructorului ?

Rolul desctructorului este de a dezaloca memoria alocata la crearea sau in timpul folosirii obiectului.

39.Cand se apeleaza destructorul ?

Destructorul se apeleaza la sfarsitul blocului de cod in care a fost declarat obiectului.

40.Ce este memoria HEAP?

Heap este o portiunea de memorie unde putem aloca si dezaloca spatiu in timpul executiei.

41.Cum se aloca spatiu de memorie in HEAP?

Spatiu in Heap se aloca folosing opertorul new.

42.Cum se elibereaza memoria in HEAP ?

Memoria alocata in Heap se elibereaza folosind operatorul delete.

43.Cum se genereaza un memory leak ?

Un memory leak se genereaza atunci cand distrugem un obiect si nu eliberam memoria din heap alocata pentru el.

44.Care este rolul functiilor accesor in cadrul clasei?

Functiile accesor au rolul de a oferi o interfata prin care putem controla accesul la membrii privati ai clasei.

45.Ce rol au functiile friend in cadrul claselor si care sunt caracteristicile acestora ?

Functiile friend ofera acces la membrii privati ai clasei unor functii sau clase ce nu se afla in clasa.

Functiile friend nu primesc pointerul this.

46.Ce reprezinta conceptul de incapsulare ?

Este conceptul prin care restrictionam accesul direct la membrii unei clase.

47.Care este diferenta dintre supradefinire si supraincarcare?

Supradefinirea necesita existenta unei ierarhii de clase si cele 2 functii trebuie sa se afle in clase diferite.

Supraincarcarea poate fi folosita pentru 2 functii din aceeasi clasa si chiar si in afara claselor intre orice 2 functii.

48.Cum se poate afla dimensiunea unui vector de char?

Folosind functia strlen()

49.Care e vizibilitatea implicita in cadrul unei clase?

Private

50.Enumerati trei functionalitati ale operatorului * in C++.

Declararea unui pointer.

Extragerea valorii de la o adresa.

Inmultire.

51.De ce se folosesc in general functiile de acces in detrimentul vizibilitatii publice?

Pentru a nu oferi acces direct la membrii functiei si pentru a controla modul in care acestea sunt modificate.

52.De ce tip (clasa) este obiectul cin?

istream

53.De ce tip (clasa) este obiectul cout?

ostream

54.Ce reprezinta prescurtarea STL?

Standard Template Library

55.Dati exemplu de un container STL.

list, set, map

56.Dati exemplu de un algoritm STL.

swap(interschimba valorile pentru 2 obiecte),for_each, copy

57.Prin ce tipuri de metode se poate supraincarca un operator ?

Un operator se poate supraincarca printr-o functie membra sau o functie in afara clasei

58.Prin ce tipuri de metode se poate supraincarcare operatorul + pentru (obiect + int) ?

Operatoul + pentru (obiect + int) se poate supraincarca printr-o functie membra sau o functie in afara clasei.

59.Prin ce tipuri de metode se poate supraincarcare operatorul + pentru (int + obiect) ?

Operatorul + pentru (int + obiect) se poate supraincarca DOAR printr-o functie in afara clasei.

60.Cand trebuie utilizat "friend" la supraincarcarea operatorilor ?

Atunci cand avem atribute private si nu avem functii de acces.

61.Ce realizeaza constructorul de copiere ?

O copie a unui obiect.

62.Cum se apeleaza destructorul ?

Destructorul se apeleaza in ordinea inversa construirii obiectelor.

63.Care este diferenta dintre o variabila si un atribut ?

Un atribut este un camp al unei clase.

64.In ce context este util modificatorul de acces protected ?

Protected este folosit pentru a oferi acces direct unei clase derivate la membri clasei de baza, pastrand atributele clasei de baza privvate pentru restul programului.

65.Care este diferenta dintre specificatorii class si struct ?

Class are ca speficaficator default de acces private iar struct are public

66.Ce reprezinta o clasa ?

Este o structura de date definita de utilizator care contine date si functii iar accesul la aceste date se face prin specificatori de acces.

67.Ce reprezinta un obiect ?

Un obiect este o instanta a unei clase si contine toate campurile din clasa din care a fost instantiat.

68.Ce reprezinta o instanta a unei clase ?

Un obiect cu campurile din clasa respectiva

69.Ce este un constructor ?

O functie ce creaza un obiect

70.Ce reprezinta termenul de up-casting ?

Folosirea unui pointer de tipul clasei de baza pentru a gestiona obiecte din clasa derivata

71.Ce reprezinta termenul de down-casting ?

Folosirea unui pointer de tipul clasei derivate pentru a gestiona obiecte din clasa de baza

72.Ce este un framework de clase ?

O ierarhie de clase

73.Pot fi definite metode constante ?

NU

74.Care este ordinea de apel a constructorilor in cadrul ierarhiilor de clase ?

De la constructorul din baza catre cel din clasa derivata

75.Care este ordinea de apel a destructorilor in cadrul ierarhiilor de clase ?

De la cel din clasa derivata catre cel din clasa de baza.

76.Ce rol au functiile virtuale in cadrul ierarhiilor de clase ?

Anunta ca ele vor fi supradefinite in clasele derivate.

77.Ce este o functie virtuala pura ?

Este o functie fara corp ce urmeaza a fi supradefinita intr-o clasa derivata

78.Ce este o clasa abstracta ?

Ce contine minim o functie virtuala pura.

79.Ce restrictii impune o clasa abstracta ?

O clasa abstracta nu poate fi instantiata.(Nu se pot crea obiecte de acel tip)

80.Cum se realizeaza mostenirea multipla in C++ ?

Mentionand clasele din care se va deriva in ordinea in care se vrea sa se apeleze constructorii.

81.Ce sunt functiile inline ?

Functii ale caror apel vor fi inlocuite de codul lor la compilare

82.Ce reprezinta conceptul de is a ? Exemplificati

Derivare Ex:Autovehicul is a Masina

83.Ce reprezinta conceptul de has a ? Exemplificati

Includere Ex:Spital has a Medic

84.Ce este mostenirea virtuala?

De chestia asta nu a zis nimic la curs si nici nu am gasit mare lucru pe net si nu am inteles ce este

85.In ce moment are loc efectiv supradefinirea?

In timpul executiei

86.In ce moment are loc efectiv supraincarcarea?

La compilare

87.Care e vizibilitatea implicita in cadrul unei structuri?

Public

88.Cum se poate modela o relatie de tip 1:M intre doua clase?

Habar nu am ce vrea sa zica aici

89.Cum se poate face conversia baza-derivat?

prin dynamic_cast<>()

90.Cum rezolvam problema mostenirii in romb (deadly diamond of death)?

Evitam mostenirea multipla

91.In ce ordine se apeleaza constructorii in cazul unei derivari multiple?

In ordinea in care s-a facut derivarea

92.Cum se poate rezolva problema nefunctionarii unei clase sablon pe un anumit tip de data?

Din nou nu inteleg ce vrea sa zica prin intrebarea asta

93.Cum poate deveni un iterator invalid?

Daca adaugam sau stergem elemente din container (nu sunt 100% sigur)

94.Cum se defineste un pointer prin care se poate gestiona orice tip de zona de memorie ?

char*

95.Cum se poate accesa zona privata a unui obiect in C++ ?

Printr-o functie accesor sau printr-un pointer la char

96.Ce reprezinta termenul de late-binding ?

Alegerea functiei ce urmeaza sa fie executata la executie

97.Ce reprezinta termenul de early-binding ?

Alegerea functiei ce urmeaza sa fie executa la compilare.

98.Care este utilitatea functiilor virtuale ?

Putem crea vectori cu obiecte de tip diferit,daca ele fac parte din aceeasi ierarhie, si prin intermediul functiilor virtuale putem decide

ce functie apelam pentru fiecare obiect din vector.

99.La ce pot fi folosite clasele abstracte ?

La derivare

100.Poate fi folosit un pointer la o clasa abstracta pentru a gestiona obiecte de tipul claselor derivate ?

Da

101.Care este utilitatea atributelor constante ?

Putem defini atribute ce nu vor fi modificate dupa initializare

102.Cum se defineste o metoda care sa nu primeasca pointerul this ?

Statica

1. Cat ocupa o variabila de tip int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

4 bytes

2.Cat ocupa o variabila de tip short int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

2 bytes

3.Cat ocupa o variabila de tip long int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

4 bytes

4.Cat ocupa o variabila de tip long long int (se ia un considerare un compilator din VS2015 pe 32/64 de biti) ?

8 bytes

5.Cat ocupa o variabila de tip bool (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

1 byte

6.Cat ocupa o variabila de tip float (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

4 bytes

7.Cat ocupa o variabila de tip double (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

8 bytes

8.Cat ocupa o variabila de tip long double (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

8 bytes

9.Cat ocupa o variabila de tip char (se ia un considerare un compilator din VS2015 pe 32 de biti) ?

1 byte

10.Cat ocupa o variabila de tip pointer la int - int *(se ia un considerare un compilator din VS2015 pe 32 de biti)?

4 bytes

11.Cat ocupa o variabila de tip pointer la char- char *(se ia un considerare un compilator din VS2015 pe 32 de biti) ?

4 bytes

12.Ce reprezinta cuvantul cheie this in C++ ?

Este un pointer primit ca primul parametru de toate metodele non-statica din clasa si care contine adresa obiectului care a apelat metoda

13.Ce reprezinta this in constructor ?

Este un pointer ce contine adresa obiectului ce urmeaza sa fie construit

14.Ce reprezinta this in destructor ?

Este un pointer ce contine adresa obiectului ce urmeaza sa fie distrus

15.Ce reprezinta this intr-o functie membra ?

Este un pointer ce contine adresa obiectului care a apelat metoda.

16.Ce reprezinta this intr-o functie statica ?

O functie statica nu este asociata cu niciun obiect, deci nu primeste pointerul this.

17.Ce reprezinta this in constructorul de copiere ?

Este un pointer catre obiectul ce urmeaza sa fie creat.

18.Ce reprezinta this in cadrul supraincarcarii operatorului = ?

Este un pointer catre obiectul ce urmeaza sa fie modificat.

19.Ce este o functie membra ?

Este o functie al carei definitii sau prototip se afla in definitia unei clase ca orice alta variabila si opereaza pe orice obiect/instanta

a acelei clase si are acces la toti membrii clasei.

20.Ce este o functie statica ?

Este o functie membra, independenta de obiectele clasei.O functie statica poate fi apelata chiar daca nu exista nicio instanta a clasei.Ele sunt apelate

folosind numele clasei si operatorul de rezolutie.

21.Ce este un atribut static ?

Un atribut static este un atribut ce nu face parte din obiect, in schimb el este impartit de toate obiectele clasei.Nu poate fi initializat in interiorul unei clase

22.Ce este un atribut constant ?

Un atribut constant este un atribut ce nu poate fi modificat in timpul executiei o data ce a fost initializat.

23.Cand se poate initializa un atribut constant ?

Un atribut constant se poate initializa doar la crearea obietului, in lista de initializare a constructorului.

24.Ce reprezinta supraincarcarea ?

Supraincarcarea reprezinta abilitatea de a avea mai multe functii cu acelasi nume dar cu o lista de parametrii diferiti.

25.Care este utilitatea supraincarcarii ?

Putem avea mai multe functii cu acelasi nume, care fac acelasi lucru sau lucruri diferite, dar pentru tipuri de date diferite si chiar si tipuri de date create de noi.

26.Ce reprezinta supradefinirea ?

Supradefinirea este abilitatea de a avea 2 functii, una in clasa de baza si una in clasa derivata, cu acelasi nume si aceeasi lista de parametrii dar care sunt

diferite prin implementare.

27.Care este utilitatea supradefinirii ?

Ne permite sa avem 2 functii cu acelasi nume si aceeasi parametrii, una in clasa de baza si una in clasa derivata, care sunt diferite prin implementare.

28.Ce este o functie virtuala ?

O functie virtuala este o functie membra ce astepti sa fie redefinita intr-o clasa derivata.Cand te referi la un obiect al clasei derivate printr-un

pointer la clasa de baza apelezi varianta din clasa derivata a functiei ce a fost declarata virtuala in baza.

29.Prin ce se implementeaza conceptul de Polimorfism in C++ ?

Polimorfismul in C++ se implementeaza prin functii virtuale, supradefinire,supraincarcarea operatorilor si upcasting.

30.Care sunt modificatorii de acces si ce vizibilitate ofera datelor/functiilor membre in cazul derivarii claselor?

In C++ exsta 3 modificatori de acces:public, private si protected.

Atributele si metodele declarate publice pot fii accesate din orice parte a programului.

Atributele declarate protected pot fii accesate doar din interiorul clasei si din clasele derivate.

Atributele declarate protected pot fii accesate doar in interiorul clasei.

31.In ce context este util modificatorul de acces private ?

Modificatorul private este util la incapsularea datelor.Acesta nu permite accesul altor clase, nici macar a celor derivate, la atributele clasei.

32.Care sunt tipurile de constructori in cadrul unei clase si ce rol are fiecare?

Exista 3 tipuri de constructor:default,cu parametrii si de copiere.

Constructorul default construieste un obiect initializand atributele cu valori constante, ce nu sunt primite ca parametrii.

Constructorul cu parameterii construiecte un obiect cu parametrii primiti de la apelator.

Constructorul de copiere creaza copii a unor obiecte care sunt transmise sau returnate prin valoare sau creaza un obiect nou, initializand atributele cu valorile altui obiect.

33.Care este diferenta intre rolul operatorului = si cel al constructorului de copiere?

Operatorul = se apeleaza pentru 2 obiecte existente deja.

Constructorul de copiere creaza obiectul si atribuie atributelor valorile altui obiect deja existent.

34.Cand este apelat constructorul de copiere ?

Constructorul de copiere se apeleaza atunci cand transmitem sau returnam obiecte prin valoare sau cand vrem sa construim un obiect nou care sa fie o copie a unui alt obiect deja existent.

35.Cand este apelat operatorul = ?

Operatorul = este apelat atunci cand vrem sa atribuim unui obiect deja existent valorile unui obiect deja existent.

36.Ce este un memory leak ?

Un memory leak se produce atunci cand nu dezalocam spatiu pentru un obiect de care nu mai avem nevoie.

37.Ce este un dangling pointer?

Este un pointer ce indica spre o zona de memorie ce a fost eliberata deja.

38.Care este rolul destructorului ?

Rolul desctructorului este de a dezaloca memoria alocata la crearea sau in timpul folosirii obiectului.

39.Cand se apeleaza destructorul ?

Destructorul se apeleaza la sfarsitul blocului de cod in care a fost declarat obiectului.

40.Ce este memoria HEAP?

Heap este o portiunea de memorie unde putem aloca si dezaloca spatiu in timpul executiei.

41.Cum se aloca spatiu de memorie in HEAP?

Spatiu in Heap se aloca folosing opertorul new.

42.Cum se elibereaza memoria in HEAP ?

Memoria alocata in Heap se elibereaza folosind operatorul delete.

43.Cum se genereaza un memory leak ?

Un memory leak se genereaza atunci cand distrugem un obiect si nu eliberam memoria din heap alocata pentru el.

44.Care este rolul functiilor accesor in cadrul clasei?

Functiile accesor au rolul de a oferi o interfata prin care putem controla accesul la membrii privati ai clasei.

45.Ce rol au functiile friend in cadrul claselor si care sunt caracteristicile acestora ?

Functiile friend ofera acces la membrii privati ai clasei unor functii sau clase ce nu se afla in clasa.

Functiile friend nu primesc pointerul this.

46.Ce reprezinta conceptul de incapsulare ?

Este conceptul prin care restrictionam accesul direct la membrii unei clase.

47.Care este diferenta dintre supradefinire si supraincarcare?

Supradefinirea necesita existenta unei ierarhii de clase si cele 2 functii trebuie sa se afle in clase diferite.

Supraincarcarea poate fi folosita pentru 2 functii din aceeasi clasa si chiar si in afara claselor intre orice 2 functii.

48.Cum se poate afla dimensiunea unui vector de char?

Folosind functia strlen()

49.Care e vizibilitatea implicita in cadrul unei clase?

Private

50.Enumerati trei functionalitati ale operatorului * in C++.

Declararea unui pointer.

Extragerea valorii de la o adresa.

Inmultire.

51.De ce se folosesc in general functiile de acces in detrimentul vizibilitatii publice?

Pentru a nu oferi acces direct la membrii functiei si pentru a controla modul in care acestea sunt modificate.

52.De ce tip (clasa) este obiectul cin?

istream

53.De ce tip (clasa) este obiectul cout?

ostream

54.Ce reprezinta prescurtarea STL?

Standard Template Library

55.Dati exemplu de un container STL.

list, set, map

56.Dati exemplu de un algoritm STL.

swap(interschimba valorile pentru 2 obiecte),for_each, copy

57.Prin ce tipuri de metode se poate supraincarca un operator ?

Un operator se poate supraincarca printr-o functie membra sau o functie in afara clasei

58.Prin ce tipuri de metode se poate supraincarcare operatorul + pentru (obiect + int) ?

Operatoul + pentru (obiect + int) se poate supraincarca printr-o functie membra sau o functie in afara clasei.

59.Prin ce tipuri de metode se poate supraincarcare operatorul + pentru (int + obiect) ?

Operatorul + pentru (int + obiect) se poate supraincarca DOAR printr-o functie in afara clasei.

60.Cand trebuie utilizat "friend" la supraincarcarea operatorilor ?

Atunci cand avem atribute private si nu avem functii de acces.

61.Ce realizeaza constructorul de copiere ?

O copie a unui obiect.

62.Cum se apeleaza destructorul ?

Destructorul se apeleaza in ordinea inversa construirii obiectelor.

63.Care este diferenta dintre o variabila si un atribut ?

Un atribut este un camp al unei clase.

64.In ce context este util modificatorul de acces protected ?

Protected este folosit pentru a oferi acces direct unei clase derivate la membri clasei de baza, pastrand atributele clasei de baza privvate pentru restul programului.

65.Care este diferenta dintre specificatorii class si struct ?

Class are ca speficaficator default de acces private iar struct are public

66.Ce reprezinta o clasa ?

Este o structura de date definita de utilizator care contine date si functii iar accesul la aceste date se face prin specificatori de acces.

67.Ce reprezinta un obiect ?

Un obiect este o instanta a unei clase si contine toate campurile din clasa din care a fost instantiat.

68.Ce reprezinta o instanta a unei clase ?

Un obiect cu campurile din clasa respectiva

69.Ce este un constructor ?

O functie ce creaza un obiect

70.Ce reprezinta termenul de up-casting ?

Folosirea unui pointer de tipul clasei de baza pentru a gestiona obiecte din clasa derivata

71.Ce reprezinta termenul de down-casting ?

Folosirea unui pointer de tipul clasei derivate pentru a gestiona obiecte din clasa de baza

72.Ce este un framework de clase ?

O ierarhie de clase

73.Pot fi definite metode constante ?

NU

74.Care este ordinea de apel a constructorilor in cadrul ierarhiilor de clase ?

De la constructorul din baza catre cel din clasa derivata

75.Care este ordinea de apel a destructorilor in cadrul ierarhiilor de clase ?

De la cel din clasa derivata catre cel din clasa de baza.

76.Ce rol au functiile virtuale in cadrul ierarhiilor de clase ?

Anunta ca ele vor fi supradefinite in clasele derivate.

77.Ce este o functie virtuala pura ?

Este o functie fara corp ce urmeaza a fi supradefinita intr-o clasa derivata

78.Ce este o clasa abstracta ?

Ce contine minim o functie virtuala pura.

79.Ce restrictii impune o clasa abstracta ?

O clasa abstracta nu poate fi instantiata.(Nu se pot crea obiecte de acel tip)

80.Cum se realizeaza mostenirea multipla in C++ ?

Mentionand clasele din care se va deriva in ordinea in care se vrea sa se apeleze constructorii.

81.Ce sunt functiile inline ?

Functii ale caror apel vor fi inlocuite de codul lor la compilare

82.Ce reprezinta conceptul de is a ? Exemplificati

Derivare Ex:Autovehicul is a Masina

83.Ce reprezinta conceptul de has a ? Exemplificati

Includere Ex:Spital has a Medic

84.Ce este mostenirea virtuala?

De chestia asta nu a zis nimic la curs si nici nu am gasit mare lucru pe net si nu am inteles ce este

85.In ce moment are loc efectiv supradefinirea?

In timpul executiei

86.In ce moment are loc efectiv supraincarcarea?

La compilare

87.Care e vizibilitatea implicita in cadrul unei structuri?

Public

88.Cum se poate modela o relatie de tip 1:M intre doua clase?

Habar nu am ce vrea sa zica aici

89.Cum se poate face conversia baza-derivat?

prin dynamic_cast<>()

90.Cum rezolvam problema mostenirii in romb (deadly diamond of death)?

Evitam mostenirea multipla

91.In ce ordine se apeleaza constructorii in cazul unei derivari multiple?

In ordinea in care s-a facut derivarea

92.Cum se poate rezolva problema nefunctionarii unei clase sablon pe un anumit tip de data?

Din nou nu inteleg ce vrea sa zica prin intrebarea asta

93.Cum poate deveni un iterator invalid?

Daca adaugam sau stergem elemente din container (nu sunt 100% sigur)

94.Cum se defineste un pointer prin care se poate gestiona orice tip de zona de memorie ?

char*

95.Cum se poate accesa zona privata a unui obiect in C++ ?

Printr-o functie accesor sau printr-un pointer la char

96.Ce reprezinta termenul de late-binding ?

Alegerea functiei ce urmeaza sa fie executata la executie

97.Ce reprezinta termenul de early-binding ?

Alegerea functiei ce urmeaza sa fie executa la compilare.

98.Care este utilitatea functiilor virtuale ?

Putem crea vectori cu obiecte de tip diferit,daca ele fac parte din aceeasi ierarhie, si prin intermediul functiilor virtuale putem decide

ce functie apelam pentru fiecare obiect din vector.

99.La ce pot fi folosite clasele abstracte ?

La derivare

100.Poate fi folosit un pointer la o clasa abstracta pentru a gestiona obiecte de tipul claselor derivate ?

Da

101.Care este utilitatea atributelor constante ?

Putem defini atribute ce nu vor fi modificate dupa initializare

102.Cum se defineste o metoda care sa nu primeasca pointerul this ?

Statica

TEST C++
NUME:
GRUPA:

1. **Fie dat urmatorul program:**

```
#include <iostream.h>
using namespace std;
class c1    { public: int a; } ;
class c2 : private c1
{
public:
        c1::a;
        int b;
        void scrie_a(){ cout << "a = " << a << endl;}
} ;
void main()
{       c2 ob; ob.scrie_a (); }
```

Selectati afirmatia corecta:

a. functia de acces nu are drept de acces la membrul *a* deoarece derivarea s-a realizat private;
b. programul afiseaza valoarea lui *a*;
c. functia are doar acces *read-only* asupra unui membru dintr-o clasa derivata privat;
d. derivarea privata este incorect realizata;
e. prin derivare privata, accesul la membrii mosteniti ramane privat.

2. **Se dă clasa :**

```
class c  {
        double a, b ;
public :
        friend c operator + (c &, double ) ;
        friend c operator + (double, c & ) ;
};
```

In declaratia de mai sus apare:

a. supraincarcare prin functii independente;
b. supraincarcare prin functii membre in clasa c;
c. supraincarcare de constructori de clasa c;
d. supraincarcare de constructori de copiere ai clasei c;
e. declararea unor clase *friend*.

3. **Se dă structura :**

```
#include <iostream.h>
using namespace std;
struct persoana
{
        char nume[50];
        struct copil { char prenume[10]; int virsta;  } c[3];
} p1,*pp;

void main()
{
        pp=&p1; p1.c[0].virsta=5;
        cout << pp->c->virsta;                          // 1
        cout << pp->c[0].virsta;                // 2
        cout << (pp->c+1)->virsta;                      // 3
        cout << ((*pp).c+1)->virsta;                    // 4
        cout << (*pp).c[0].virsta;              // 5
        cout << (*pp).c->virsta;                // 6
}
```

Care din expresiile:

pp->c->virsta; // 1

pp->c[0].virsta; // 2
(pp->c+1)->virsta; // 3
((*pp).c+1)->virsta; // 4
(*pp).c[0].virsta; // 5
(*pp).c->virsta; // 6

afiseaza valoarea 5?

a. toate
b. 2+3+5
c. 2+3+5+6
d. 1+2+5+6
e. 1+2+4+5+6

4. Fie declaratia :

```
class c1   { /* ... */ } ;
class c2 : public c1    {/*... */} ;
```

Atunci clasa c2 fata de c1 este:

a) derivata; b) de baza; c) friend; d) virtuala; e) derivata friend

5. Fiind data clasa:

```
#include <conio.h>
#include <iostream>
using namespace std;
class persoana {

public:
          float salariu;
          persoana(float s=0): salariu(s){ }
          operator float( ) { return salariu; }
          float indexare(float coef)
                              { return salariu *(1+coef/100); }
};

void main( )
{   persoana p(100); cout << p.indexare(p); getch(); }
```

Apelul functiei indexare( ):

a. foloseste cast-ul definit de programator;
b. genereaza eroare, prin folosirea recursiva a obiectului p;
c. genereaza eroare, nexistînd o supraîncarcare ce primeste obiect persoana;
d. se traduce prin indexare(int);
e. se traduce prin indexare(void);

6. Având declaratia :

```
class persoana {
          int virsta ;
public :
          persoana () ;
          int spune_virsta() { return virsta ; }
};
```

Functia int spune_virsta( ) :

a. asigura acces read only la un membru privat al clasei;
b. este o functie friend;
c. este o metoda obisnuita;
d. asigura acces read - write la un membru privat al clasei;
e. este eronata, deoarece variabila virsta este privata.

7. Având o clasa definita astfel:

```
class ex {
int a ;
public :
        friend ostream& operator << (ostream& , ex  ) ;
};
```

Functia friend ostream& operator << (ostream& , ex) supraîncarca operatorul << ?

a. nu, neavînd suficienti parametri.
b. da, în scopul citirii datelor obiectului;
c. da, pentru implementarea operatiei de deplasare la stinga, pe obiecte;
d. nu, operatorul << neputindu-se supraincarca;
e. da, în scopul afisarii datelor obiectului;

8. Fie clasa :

```
class c {
        int a, b ;
public :
        c (int , int ) ;
        int det_a ( ) {return a ;}
        ~c () ;
};
```

Semnul ~ are rolul :

a. de a nega pe biti rezultatul returnat de metoda c( );
b. de a preciza existenta destructorului;
c. de a nega logic rezultatul returnat de metoda c( );
d. de a supraincarca constructorul clasei;
e. de a supraincarca operatorul ~

9. Se dau urmatoarele clase:

```
#include <iostream>
using namespace std;

class B1 { int x; };
class B2 { int y; };
class B3 { int z; };
class B4 { int t; };
class  D: public B1, private B2, protected B3, B4  { public: int m; };

void main()
{
        D d;
        cout << d.m; // varianta 1
        cout << d.x; // varianta 2
        cout << d.y; // varianta 3
        cout << d.z; // varianta 4
        cout << d.t; // varianta 5
}
```

Variantele care au acces la variabile pentru afisare sunt:

a) 1 + 2 + 4 + 5 b) 1 + 2 c) 1 + 2 + 4 d) 1 e) 1 + 2 + 5

10. Se dă clasa :

```
#include <iostream>
using namespace std;
#include <string.h>

class person
{
        double wage;
```

```
      public:
         char name[20];
         person(char n[]=" anonymous",double w=0):wage(w)
         {                  strcpy(name, n); }
         person * gA()
         {                  return this;       }
    };

void main()
{
         person p1=person("Daniel",5000), p2("John",3500);
         cout << endl  << (&p1)->name;                        // varianta  1
         cout << endl  << &p1->name;                          // varianta  2
         cout << endl << p2.gA()->name <<"\n\n";              // varianta  3
         cout << endl << p2.this->name <<"\n\n";    // varianta  4
}
```

Care din variantele de mai jos afiseaza corect numele unei persoane ?

a) 2+3 b) 1+2 c) 1+3 d) 1+2+3 e) 1+2+3+4

11. Fiind data urmatoarea secventa de cod:

```
#include <iostream>
using namespace std;

class persoana{
private:
         float salariu;
public:
         char nume[20];
         virtual float calc_sal( )  {return 0.;}          //  Salariu persoanei
};
class inginer : public persoana{
public:
         float calc_sal( )    {return 1.;}                // Salariu în regie
};
class muncitor : public persoana{
public:
         float calc_sal( )   {  return 2.;}               // Salariu în acord
};

void main( )
{
         persoana p, *pp; inginer i,*pi; muncitor m, *pm;
    pp = &p; pi=&i; pm=&m;

    pp=pi; cout << endl <<pp->calc_sal( );
    pp=pm; cout << endl <<pp->calc_sal( );

    p=i;  cout << endl << p.calc_sal( );
    p=m; cout << endl << p.calc_sal( );
}
```

programul afiseaza in ordine, valorile:

a) 0 0 1 2 b) 1 2 1 2 c) 0 1 0 2 d) 0 0 0 0 e) 1 2 0 0

12. Se da programul:

```
#include <iostream>
using namespace std;
class c{
         int a;
```

```
public :
        c() {}
        c(const c&);
        c& operator =(c&);
};

c& c::operator=(c &c){ cout << endl  << "copiere cu egal"; return c;}
c::c(const c &c) { cout << endl << "Constructor de copiere"; }
void main()
{
        c x,y=x;
        c b=x;  x=y;
};
```

programul:

a. apeleaza de doua ori operator=(), o data constructorul de copiere si o data constructorul implicit;
b. apeleaza de trei ori constructorul de copiere, o data constructorul implicit;
c. apeleaza de trei ori supraincarcarea operatorului =;
d. apeleaza de doua ori constructorul de copiere si de trei ori operator=();
e. apeleaza de doua ori constructorul de copiere, o data operator=() si o data constructorul implicit;

13. Pentru urmatorul program:

```
#include <iostream>
using namespace std;
class persoana{
public:
        int virsta;
        persoana(int v=30) : virsta(v){}
};
class profesor{
public:
        int virsta;
        profesor(int v=20) : virsta(v){}
    operator persoana(){ persoana p; p.virsta = virsta; return p; }
};

persoana f(persoana p)  { p.virsta++; return p; }

void main()
{
   persoana p; p=f(p);   cout << endl << p.virsta;
   profesor prof; p=f(prof);          cout << endl << p.virsta;
}
```

varstele afisate la rularea programului de mai sus sunt:

a. 30 20, ambele obiecte fiind temporare, la transmiterea prin valoare;
b. 31 21, datorita incrementarilor din functie;
c. 31 20, profesor fiind temporar, datorita conversiei implicite prin cast;
d. 30 21, persoana fiind temporar, datorita conversiei implicite prin cast;
e. 0 0, deoarece pentru obiecte temporare s-au apelat constructori de copiere.

14. Programul următor:

```
#include <iostream>
using namespace std;
class Clasa1{
     bool value;
public:
     virtual void f(){cout<<"Clasa1"<<endl;}
```

```
};
class Clasa2{
	int caracter;
	int articol;
public:
	void f(){cout<<"Clasa2"<<endl;}
};
void egale(){cout<<"sizeof(obC1) = sizeof(obC2)"<<endl;}
void obC1obC2(){cout<<"sizeof(obC1) < sizeof(obC2)"<<endl;}
void obC2obC1(){cout<<"sizeof(obC2) < sizeof(obC1)"<<endl;}
void main()
{
	Clasa1 obC1;
	Clasa2 obC2;
	void(*f)();
	sizeof(obC1) == sizeof(obC2)? f = egale : sizeof(obC1) < sizeof(obC2) ? f = obC1obC2 : f =
obC2obC1;
	(*f)();
}
```

tipãreste:

a. sizeof(obC2) < sizeof(obC1);
b. sizeof(obC1) < sizeof(obC2);
c. sizeof(obC1) = sizeof(obC2);
d. generează eroare la compilare datoritã unei indirectãri incorecte;
e. generează eroare la compilare deoarece operatorul conditional ?: nu permite efectuarea unor atribuiri.

15. In programul următor:

```
#include <iostream>
using namespace std;
class Persoana{
	int varsta;
	char* nume;
public:
	Persoana(int v=0, char* n="Oarecare"):varsta(v){
		this->nume = new char[strlen(n)+1];
		strcpy(this->nume,n);
		cout<<"Constructor"<<endl;}
	Persoana(Persoana& p){
		this->varsta = p.varsta;
		this->nume = new char[strlen(p.nume)+1];
		strcpy(this->nume, p.nume);
		cout<<"Constructor de copiere"<<endl;}
	void operator=(Persoana& p){
		this->varsta = p.varsta;
		delete[] this->nume;
		this->nume = new char[strlen(p.nume)+1];
		strcpy(this->nume, p.nume);
		cout<<"Operator="<<endl;}
	~Persoana(){ cout<<"Destructor"<<endl;}
};

void main()
{
	Persoana p1, p2(20, "Gigel");
	Persoana p3 = p1;
	p3 = p2;
	Persoana p4 = p1;
}
```

sunt apelate următoarele:

a. constructor – de patru ori, constructor de copiere – o datã, destructor – de patru ori;
b. constructor – de trei ori, constructor de copiere - de douã ori, destructor de cinci ori;
c. constructor – de douã ori, constructor de copiere – de douã ori, operator= - o datã, destructor – de patru ori;
d. constructor – de douã ori, constructor de copiere – o datã, operator= - de douã ori, destructor – de douã ori;
e. constructor – de douã ori, constructor de copiere – o datã, operator= - de douã ori, destructor – de patru ori.

16. De câte ori este apelat destructorul clasei Persoana în programul urmãtor?

```
#include <iostream>
using namespace std;
class Persoana{
public:
    Persoana() {cout<<"Constructor"<<endl;}
    ~Persoana() {cout<<"Destructor"<<endl;}
};

void main(){
        Persoana** ppp;
        ppp = new Persoana*[5];
        for(int i=0; i<5; i++)
        ppp[i] = new Persoana();
        //prelucrari
        for(int i=0; i<5; i++)
        delete ppp[i];
}
```

Raspuns:
a. 10;
b. 6;
c. 7;
d. 5;
e. niciunul din rãspunsurile anterioare.

17. Sã se precizeze ce afiseazã programul urmãtor:

```
#include <iostream>
using namespace std;
class Masina{
        int anFabr;
        char* culoare;
public:
        Masina(int an = 0, char* cul = ""){
                this->anFabr = an;
                this->culoare = new char[strlen(cul)+1];
                strcpy(this->culoare, cul);
        }
Masina& operator=(Masina& m){
        this->anFabr = m.anFabr;
        delete[] this->culoare;
        this->culoare = new char[strlen(m.culoare)+1];
        strcpy(this->culoare, m.culoare);
        return (*this);
        }
int getAnFabr() {return this->anFabr;}
void setAnFabr(int anFabricatie) {this->anFabr = anFabricatie;}
char* getCuloare() {return this->culoare;}
void setCuloare(char* c){
        delete[] this->culoare;
        this->culoare = new char[strlen(c)+1];
        strcpy(this->culoare, c);
        }
};
```

```
void main() {
        Masina m1(2000,"Alb");
        Masina m2(2001,"Negru");
        Masina m3 = m2;
        Masina m4(2003, "Rosu");
        m3.setCuloare("Verde");
        m4 = m1;
        m4.setCuloare("Albastru");
        cout<<m1.getAnFabr()<<" "<<m1.getCuloare()<<" ; ";
        cout<<m2.getAnFabr()<<" "<<m2.getCuloare()<<" ; ";
        cout<<m3.getAnFabr()<<" "<<m3.getCuloare()<<" ; ";
        cout<<m4.getAnFabr()<<" "<<m4.getCuloare()<<" ; ";
}
```

afiseazã:

a. 2000 Albastru ; 2001 Negru ; 2001 Verde ; 2000 Albastru ;
b. 2000 Alb ; 2001 Negru ; 2001 Verde ; 2000 Albastru ;
c. 2000 Alb ; 2001 Negru ; 2001 Negru ; 2000 Albastru ;
d. 2000 Alb ; 2001 Verde ; 2001 Verde ; 2000 Albastru ;
e. Niciun rãspuns corect

18. Se consideră clasa

```
class Student{
public:
        char * nume;
        int note[10];
int nrnote;
...
Student(int *v, int dim,char* num){...}
Student operator=(Student s){
nume = new char[strlen(s.nume)+1];
strcpy(nume,s.nume);
for(int i=0;i<s.nrnote;i++) note[i] = s.note[i];
nrnote = s.nrnote;
return *this;
}
~Student(){
        if(nume) delete[]nume;}
};
```

Execuţia programului următor:

```
void main()
{
int vector[] = {1,2,3};
        Student s1(vector,3,”Popescu”);;
        Student s2(vector,3,”Gigel”);
        s1 = s2;
}
```

are ca efect:

a. Copierea valorilor din s2 în s1 cu generare de memory leak datorită câmpului *nume*.
b. Copierea valorilor din s2 în s1 fară generare de memory leak
c. Copierea valorilor din s2 în s1 cu generare de memory leak datorită câmpului *note* care nu este dezalocat în destructor.
d. Eroare la execuţia operatorului = deoarece nu este alocat spaţiu pentru câmpul *note*.
e. Eroare la apelul constructorului cu parametrii nefiind respectat tipul parametrilor de intrare.

19. Pentru a defini corect (fără a genera în aplicaţii viitoare memory leaks, pointeri cu valoarea 0xcccccc sau iniţializări de pointeri diferiţi cu aceeaşi adresă) clasa Student ce are atributele

```
char nume[30];
int *note;
int nrNote
```

este nevoie obligatoriu de:

a. constructor cu parametrii/fără parametrii, constructor copiere, operator =;
b. constructor cu parametrii, constructor de copiere, destructor
c. constructor fără şi cu parametrii, constructor copiere, operator =, destructor
d. constructor cu parametrii, constructor implicit, operator = , destructor
e. constructor copiere, operator =, destructor

20. Dacă se consideră clasa

```
class Test{
private:
        int valoare;
        Test(int vb){valoare = vb;}
public:
int GetValoare() { return valoare;}
}
```

si forma supraîncărcată a operatorului +

```
int operator +(int vb, Test t){
return vb+t.GetValoare();
}
```

analizaţi instrucţiunile:

```
Test t(5);
int vb = 10 + t
```

a. Sunt corecte si vb ia valoare 15
b. Sunt corecte si vb ia valoarea 10;
c. Nu sunt corecte deoarece operatorul + nu este anuntat ca fiind *friend* in clasa
d. Nu sunt corecte deoarece operatorul trebuie supraîncărcat numai prin funcţie membră
e. Nu sunt corecte deoarece atributul *valoare* este privat si nu este accesibil din afara clasei

Grila	Raspuns
1	B
2	A
3	D
4	A
5	A
6	A
7	E
8	B
9	D
10	C
11	E
12	E
13	B
14	B,C
15	C
16	D
17	D
18	A
19	C
20	A

//

cod

foloseste cast u definit de programator

//

int v[]{1,5,10,20}

int* pointer=v

pointer++

cout<<*pointer;

afiseaza 5

//

operator unar: operator++

//

Clasele absracte: poate fi folosit ca reference-type(tip pointer)

//

la cod cu automobil: D. instructiuni corecte

//

a1=10+=a2;

A. a1.operator=(operator+=(10,a2));

//

cod persoana int spune_varsta

functie inline

//

cod base derived

eroare de compilare

//

extragerea atributelor relevante ale unui obiect: abstractizare

//

in ce context este util modificatorul de acces "protected"?

in definirea unor ierarhii de clase

//lista de initializatori a unui constructor produce efecte similare cu:

atribuirea

//

cod magazin

m[0]=3.4;

//

Constructorul de copiere:

se instaniaza un obiect pe baza unui obiect exstent

//

class vector

doar variantele 1 si 2

//

class Animal

Neidentificat.Neidentificat.

//

blocurile catch:

sa apara imediat dupa blocul try

//

class Vehicul

constructorul din clasa de baza este apelat gresit prin Vehicul(2000,0)

//

care din urmatoarele nu se mostenesc?

supraincarcarea operatorului+

//

clasa abstracta: cu cel putin o functie virtuala pura (nu poate instantia obiecte)

//functiile accesor

sa ofere acces in citire si scriee la atributele private ale clasei

39 grile 2019

12 grile 2017

GRILE 2017

//

Ce inseamna STL?

STL=Standard Template Library

//

downcasting:

o conversie a obiectelor din clasa de baza in clase derivate

//

STRUCT CAR

eroare de executie

//

```
void main()
{
	int x = 2;
	int* px = &x;
	--px;
	px += 2;
	px -= 1;
	cout << *px;
}
```

afiseaza 2

//

```
void main()
```

```
{
	const char* string = "Something";
	int val = sizeof(string);
	cout << val;
}
```

afiseaza 4

```
//
void main()
{
	int x = 2;
	int*px = NULL;
	cout << px;
}
```

afiseaza 000000

```
//
class Car
{
public:  Car()
{
	cout << "constructor" << endl;
}
		~Car()
		{
			cout << "destructor" << endl;
		}
};
```

```
void main()
{
	Car c;
}
```

afiseaza constructor si destructor

```
//
class Car
{
public:
	char producer[30];
	int year;
	float power = 90.5;

	int getPower()
	{
		return power;
	}

};

void main()
{
	Car c;
	cout << c.getPower();
}
```

afiseaza 90

```
//
car cu 2 comstr si un destr
cinstructor, constructor, destructor, destructor


//
void main()
{
	int n=0;
	cout << "n= ";
	cin >> n;
	int vector[n];
	cout << vector[0];
}
```
nu compileaza

```
//
long square(int &x)
{
	x = x * x;
	return x;
}


void main()
{
	int x = 5;
```

```
	cout << x << endl;
	square(x);
	cout << x << endl;
}
```

afiseaza 5 25

```
//
class Car
{
public:
	char producer[30];
	int year;
	float power = 90.5;

};

void main()
{
	Car c;
	Car c2 = c;
	cout << c2.year;
}
```

afiseaza adresa lui year

//

Daca o clasa derivata foloseste specificatorul public pt mostenire:

membri protejat din clasa de baza devin publici in clasa derivata

//

C.Cum se initializeaza un atribut constant?

in antetul constructorilor, in listav de initializare

//

C.constructorul de copiere o data si cel de clasa niciodata

//

Care e problema mostenirii in romb?

C.derivarea unei clase din mai multe clase care au o baza comuna

//

B.care e diferenta dintre rorlul operatorului= si cel al constructorului de copiere?

operatorul=creeaza un obiect nou cu aceleasi valori, constructorul de copiere lucreaza cu 2 obiecte deja existente

//

D. destrucorul de apeleaza de 3 ori

//

Cand un argument este transferat prin referinta, atunci:

B. functia nu poate accesa in mod direct valoarea argumentului

//

Cand compilatorul nu poate face diferenta dintre 2 constructori supraincarcati, cum se numeste starea in care acestia regasesc din perspectiva compilatorului?

D. ambigua

//

Un operator c++ obisnuit care se comporta intr-un mod special pentru un tip de data definit de utilizator se numeste?

D.supraincarcat

//Daca avem o clasa derivata si instantiem un obiect din aceasta:

C. obiectul din clasa parinte nu trebuie sa fie construit

//

O functie independenta declarata friend in domeniul publicdintr-o clasa si care primeste ca parametru o referinta la un obiect al clasei respective are acces:

B. la toti membrii

//

Care sunt modificatorii de acces?

B. private, protected, public

//

Care din urmatoarele afirmatii este adevarata privind supraincarcarea operatorilor?

B. asociativitatea operatorilor se modifica in functie de implementarea aleasa

//

Programul principal poate accesa membri privati ai unei clase?

C. doar prin intermediul ltor membri publici ai clasei

//

In c++ conceptul de polimorfism este implementat prin:

D. prin supraincarcare de operatori/metode si prin supradefinire metode

//

cod 1-23 ani, cod-55-23 ani

//

Semnul ~:

C. de a defini destructorul

//

istream operator>>

C. nimic, erori de compilare

//

un program in c++ contine urmatorul antet de functie int function(double d, char c). Care din urmatoarele functii mai pot fi utilizate in acelasi program?

B. int function(int d, char c)

//

cod

foloseste cast u definit de programator

//

int v[]{1,5,10,20}

int* pointer=v

pointer++

cout<<*pointer;

afiseaza 5

//

operator unar: operator++

//

Clasele absracte: poate fi folosit ca reference-type(tip pointer)

//

la cod cu automobil: D. instructiuni corecte

//

a1=10+=a2;

A. a1.operator=(operator+=(10,a2));

//

cod persoana int spune_varsta

functie inline

//

cod base derived

eroare de compilare

//

extragerea atributelor relevante ale unui obiect: abstractizare

//

in ce context este util modificatorul de acces "protected"?

in definirea unor ierarhii de clase

//lista de initializatori a unui constructor produce efecte similare cu:

atribuirea

//

cod magazin

m[0]=3.4;

//

Constructorul de copiere:

se instaniaza un obiect pe baza unui obiect exstent

//

class vector

doar variantele 1 si 2

//

class Animal

Neidentificat.Neidentificat.

//

blocurile catch:

sa apara imediat dupa blocul try

//

class Vehicul

constructorul din clasa de baza este apelat gresit prin Vehicul(2000,0)

//

care din urmatoarele nu se mostenesc?

supraincarcarea operatorului+

//

clasa abstracta: cu cel putin o functie virtuala pura (nu poate instantia obiecte)

//functiile accesor

sa ofere acces in citire si scriee la atributele private ale clasei

39 grile 2019

12 grile 2017

GRILE 2017

//

Ce inseamna STL?

STL=Standard Template Library

//

downcasting:

o conversie a obiectelor din clasa de baza in clase derivate

//

STRUCT CAR

eroare de executie

```
//
void main()
{
	int x = 2;
	int* px = &x;
	--px;
	px += 2;
	px -= 1;
	cout << *px;
}
```
afiseaza 2

```
//
void main()
{
	const char* string = "Something";
	int val = sizeof(string);
	cout << val;
}
```
afiseaza 4

```
//
void main()
{
	int x = 2;
	int*px = NULL;
	cout << px;
```

```
}
```

afiseaza 000000

```
//
class Car
{
public:  Car()
{
        cout << "constructor" << endl;
}
                ~Car()
                {
                        cout << "destructor" << endl;
                }
};

void main()
{
        Car c;
}
```

afiseaza constructor si destructor

```
//
class Car
{
public:
        char producer[30];
```

```
	int year;
	float power = 90.5;

	int getPower()
	{
		return power;
	}

};

void main()
{
	Car c;
	cout << c.getPower();
}
```

afiseaza 90

//

car cu 2 comstr si un destr

cinstructor, constructor, destructor, destructor

//

```
void main()
{
	int n=0;
	cout << "n= ";
	cin >> n;
	int vector[n];
	cout << vector[0];
```

```
}
```

nu compileaza

```
//
long square(int &x)
{
	x = x * x;
	return x;
}


void main()
{
	int x = 5;
	cout << x << endl;
	square(x);
	cout << x << endl;
}
```

afiseaza 5 25

```
//
class Car
{
public:
	char producer[30];
	int year;
	float power = 90.5;
```

```
};


void main()
{
	Car c;
	Car c2 = c;
	cout << c2.year;
}
```

afiseaza adresa lui year