**Test: Facilitati C++**

* **În C++ realizarea operațiilor de I/O are loc prin intermediul "fluxurilor de intrare/ieşire ", care sunt obiecte care conţine datele şi metodele necesare operaţiilor cu acel flux. Pentru realizarea operației de citire a fluxului de date se va utiliza:**

The correct answer is: cin

* **Manipulatorii sunt funcţii speciale, care pot fi folosite împreună cu operatorii de inserţie într-un flux de ieşire sau de extracţie dintr-un flux de intrare, în scopul modificării caracteristicilor formatului informaţiilor de intrare/ieşire. Manipulatorul care defineşte caracterul de umplere este**

The correct answer is: setfill ( )

* **Fie dată următoarea secvență de program. Scrieți mesajul care va fi afișat urmare a executării secvenței**

double c=123.1234567;

cout<<”c=”<<setw(7)<<setprecision(2)<<c;

The correct answer is: c= 123.12

* **Selectați instrucțiunea care pentru un pointer de tip int cu numele a alocă un spațiu de memorie și îl inițializează cu valoarea 5.**

The correct answer is: a=new int (5);

* **Deschiderea fişierului se efectuează prin intermediul metodei open:**

nume\_file.open(adresa\_file, mod\_de\_deschidere);

Precizați care va fi valoarea argumentului **mod\_de\_deschidere** pentru ca fișierul să fie deschis pentru scriere/adăugare:

The correct answers are: ios::out, ios::app, ios::ate

* **Din opțiunile de mai jos selectați funcția  prin intermediul căreia poate fi citit doar un caracter dintr-un fișier în limbajul C++**

The correct answer is: get ( )

* **Un pointer este o variabilă care conţine adresa unui Obiect (altă variabilă sau funcţie)**

The correct answer is 'True'.

* **Fie dată următoarea secvență de program. Scrieți mesajul care va fi afișat urmare a executării secvenței**

int \*c, b=17;

c=&b; (\*c)+=3;

cout<<”b=”<<b;

The correct answer is: b=20

* **Fie dată următoarea secvență de program.**

struct student {

char nume[20], prenume[20]; int an\_studiu;

}\*p ;

student st={"Anatol","Manole",2}; p=&st;

**Scrieți instrucțiunea prin care utilizând pointerului p se va modifica anul de studiu al studentului st din 2 în 3.**

The correct answer is: p->an\_studiu=3;

**Test: Constructori și clase**

**Selectați proprietățile caracteristice unui constructor:**

* Constructorul este o funcție membră care se apelează automat cand într-un program se creeaza un obiect al unei clase
* Constructorul are același nume cu cel al clasei

**Un membru al clasei (dată sau o metodă) declarat la nivelul de protecție private:**

* nu poate fi accesat din afara clasei

**Selectați doar opțiunile pe care le considerati corecte:**

* O clasa poate avea doar un singur destructor
* Constructorii si destructorii sunt apelați automat

**Următoarea secvența de program:**

class Time {....  
public:   
 Time(int h=0, int m=0, int s=0){ hour = h; minute = m; secunde = s; }  
}

**defineste:**

* un constructor cu parametri ce iau valori implicite

**Selectați opțiunea care corespunde principiul incapsularii:**

* Combinarea datelor și metodelor într-o singura structură de date, definind totodată modul în care obiectul și restul programului pot referi datele din obiect

**Fie data următoarea secvență de program:**

text.c  
double sqrt(){...}  
void f(double x){...sqrt(x);...}  
int main(){  
 …  
 double a=sqrt(9);  
 f(sqrt(121.49));  
…}

**Aceasta este organizată avand la baza principiul:**

* programarii procedurale

**Se considera urmatoarea secventă de program:**

class carte{  
 char titlu[40], autor[25];  
public:  
 int an\_aparitie;  
 float pret;  
private:  
 void citire();  
};  
carte c;

**Selectati doar instructiunile pentru care datele sau metodele obiectului care sunt accesibile.**

* cin >>c.pret;

**Selectati proprietatile caracteristice unui destructor:**

* Un destructor nu returneaza nici o valoare
* Un destructor nu are nicioadata parametri
* Destructorul unei clase este apelat automat atunci cand un obiect al clasei este șters din memorie

**Selectati optiunea care corespunde principiului abstractizarii:**

* Procesul de grupare a datelor si metodelor de prelucrare specifice rezolvarii unei probleme

**Fie data urmatoarea secventa de program:**

text.c

//declaratii date

int main(){  
 //declaratii date locale  
 //instructiuni  
}

**Aceasta este organizată avant la baza principiul:**

* Nestructurata

**Test: Particularități: clase și obiecte. Funcții și clase friend**

**Se considera următoarea secvența de program:**

class Cls1{  
 static int s;  
public:  
 S() {v = 0;}  
 int gets() {return s;}  
 int getv() {return v;}  
 void incs() {s++;}  
 void incv() {v++;}  
};

int Cls1::s;

int main(){  
 Cls1 x,y;  
 x.incs(); x.incs();  
 cout <<”y.s: “<<y.gets()<<endl;  
}

**Dupa executarea secventei de program, la consola se va afisa valoarea:**

* 2

**Fie data clasa A1, dupa cum urmeaza:**

class A1{  
 int a;  
 void f\_a(int a) {this->a=a;}  
 A1& g\_a(){a++; return \*this;}  
 friend A1 prieten (A1 &, int);

};  
A1 prieten (A1 t, int a);  
 this->a=a+t1.a;  
 return \*this;  
}

**Selectati doar optiunile pentru care utilizarea pointerului this este corecta:**

* A1& g\_a{ a++;return \*this;}
* void f\_a(int a){this ->a=a;}

**Se consideră următoarea secvență de program:**

class cls{  
 public:  
 static int s;  
};

int cls : : s=0;  
int main(){  
 int i=7;  
 cls : : s=i;  
 cout<<cls : : s;  
}

**Utilizarea lui s este:**

* corectă, deoarece membrii statici exista înainte de a se crea obiecte din clasă

**O funcție friend diferă de o metoda obișnuită a unei clase prin faptul ca:**

* nu primește pointerul implicit la obiect this

**Fie data următoarea secvență de program:**

class Point{  
 public:  
 Point(int,int);  
 int X( );  
 int Y( );  
 Point & incr\_punct( );  
   
 private:  
 int x , y;  
};

Point& Point : : incr\_punct( ){  
 ++x;  
 ++y;  
 ...  
}

**Cu care din următoarele instrucțiuni necesită a fi substituite … astfel incat metoda incr\_punct sa returneze obiectul curent:**

* return \*this;

**Selectați doar opțiunile pe care le considerati corecte:**

* Pointerul **this** este folosit implicit pentru a referi datele membre și funcții membre ale unui obiect
* **This** reprezinta un pointer a carui valoare este întotdeauna adresa obiectului

**Se considera următoarea secvența de program:**

class ex1;  
class ex2{  
 public:  
 friend class e1;  
};

class ex1{

//..

};

**Relatia dintre clase este:**

* clasa ex1 este friend al clasei ex2

**O functie friend, declarata in cadrul unei clase in domeniul public care primește ca parametru o referinta la un obiect al clasei respective are acces:**

* la toți membrii clasei

**Fie date clasele A1 și B1 după cum urmează:**

class A1{  
 int a;  
 void f\_a();  
 friend void prieten(A1 &, B1 &);  
};

class B1{  
 public:  
 int b;  
 void f\_b();  
 friend void prieten(A1 &, B1 &);  
};

**Selectați datele și metodele din clasele A1 și B1 care vor putea fi accesate din cadrul funcției prieten:**

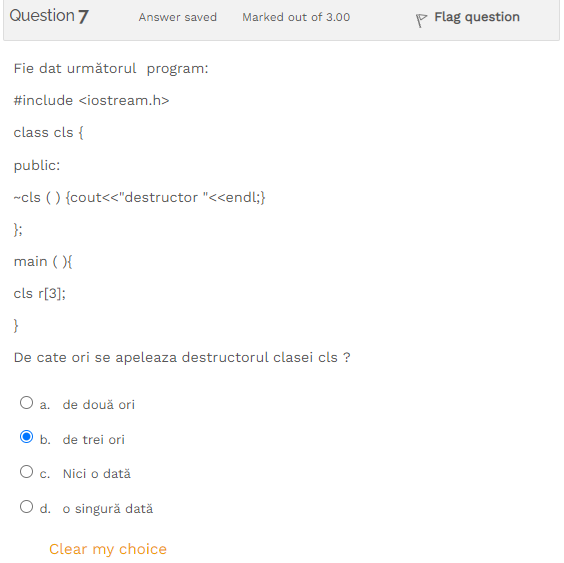
* f\_a()
* f\_b()
* a
* b

**Selectați doar opțiunile pe care le considerati corecte:**

* Functii prietene (friend) sunt funcții ne-membre ale unei clase, care au acces la datele membre private ale unei clase
* Funcții prietene ale unei clase trebuie precizate în definiția clasei

**Atestarea 1**

**Fie dat următorul program:**



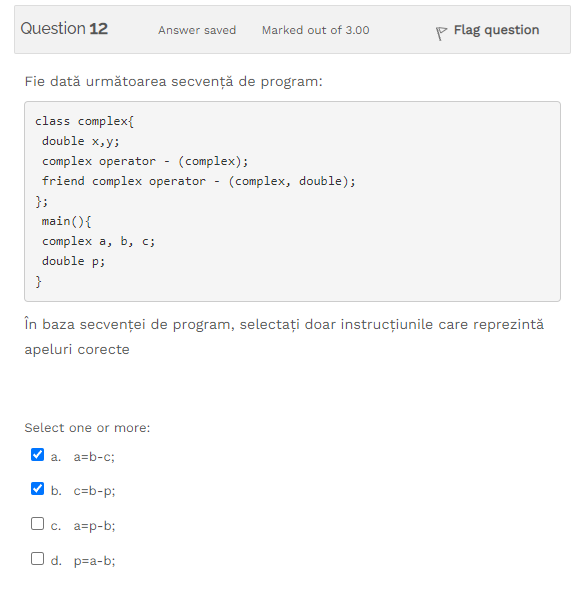
**De cate ori se apeleaza destructorul clasei cls?**

* De trei ori

**Care este denumirea corectă a cursului, în cadrul caruia suneți evaluat**

* Programarea orientată pe obiecte

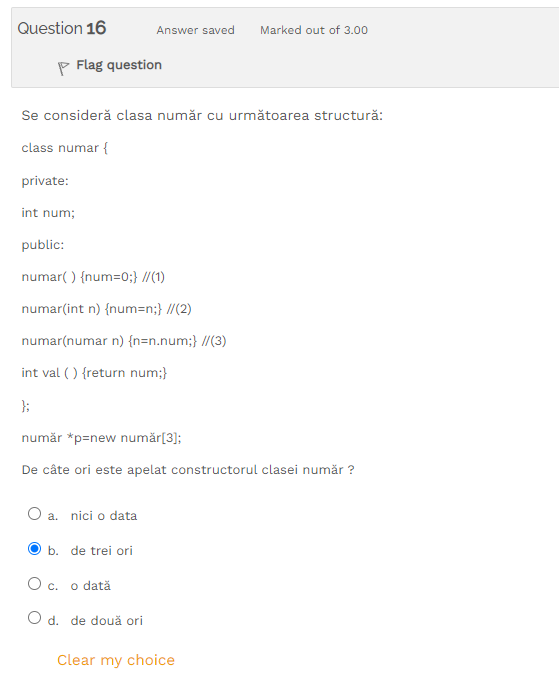
**Fie dată următoarea secvență de program:**



**În baza secvenței de program, selectați doar instrucțiunile care reprezintă apeluri corecte:**

* a = b – c;
* c = b – p;

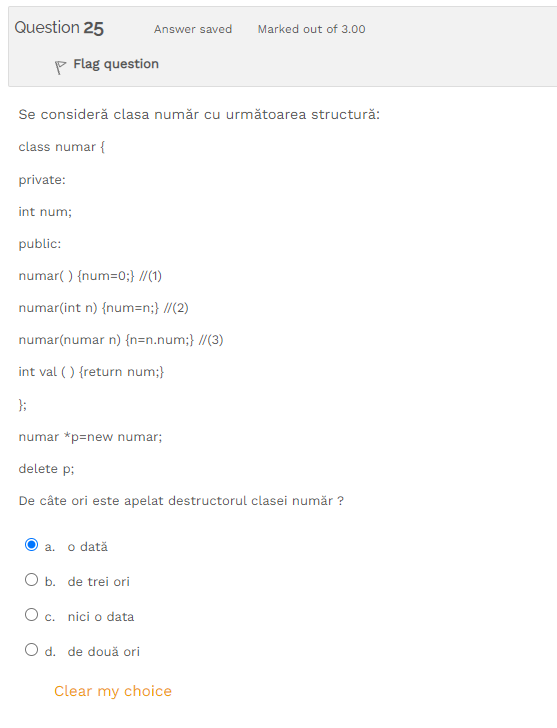
**Se consideră clasa număr cu următoarea structură:**



**De câte ori este apelat constructorul clasei număr ?**

* de trei ori

**Se consideră clasa număr cu următoarea structură:**



**De câte ori este apelat desctructorul clasei număr ?**

* o dată

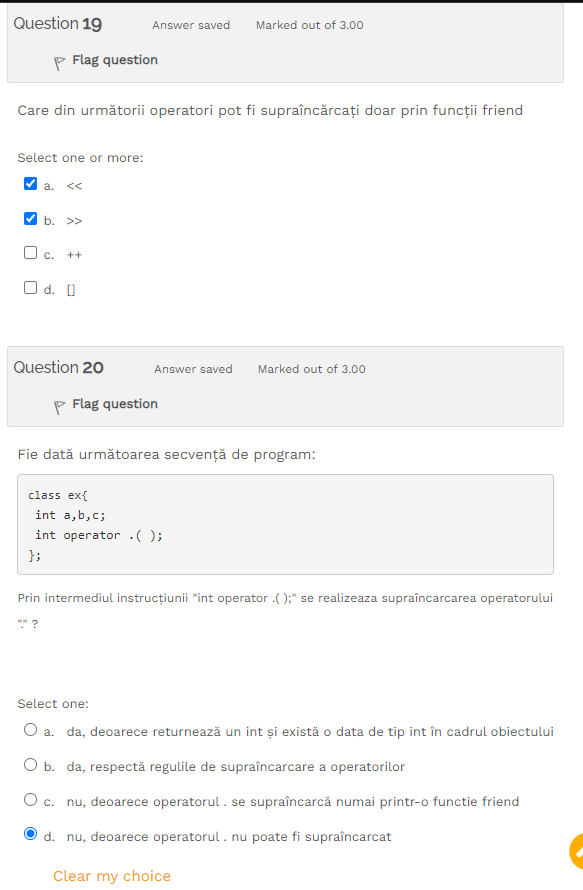
**Care dintre următorii operatori nu pot fi supraîncărcați**

* sizeof
* ::

**Care dintre următorii operatori pot fi supraîncărcați doar prin funcții friend**

* <<
* >>

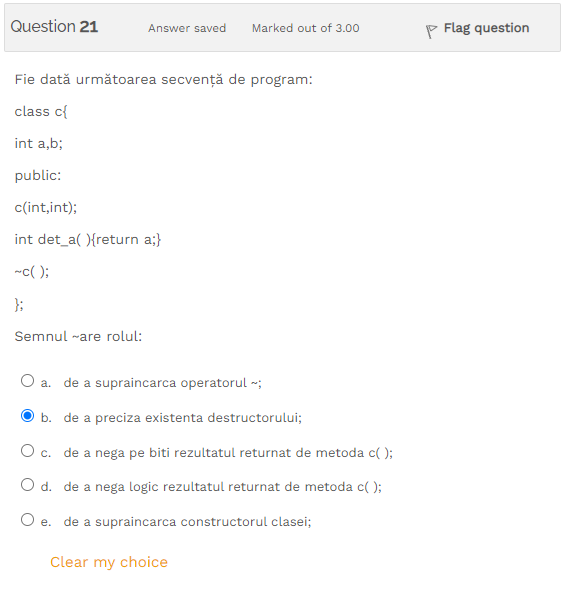
**Fie dată următoarea secvență de program:**



**Prin intermediul instrucțiunii “int operator .();” se realizeaza supraîncărcarea operatorului “.” ?**

* Nu, deoarece operatorul . nu poate fi supraîncărcat

**Fie dată următoarea secvență de program:**

****

**Semnul ~ are rolul:**

* De a preciza existenta destructorului

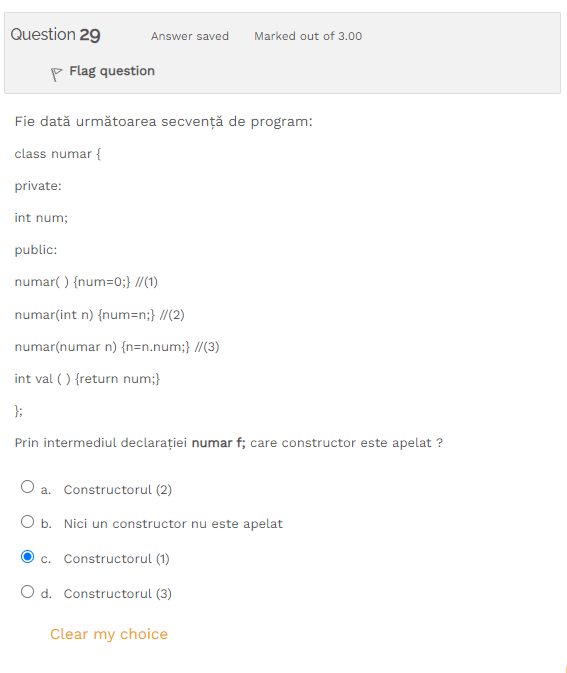
**Selectați regulile care necesită a fi respectate la supraîncărcarea operatorilor**

* Unui operator i se poate atribui orice semnificație, însă se recomandă ca aceasta să fie cât mai apropiată de semnificația naturală
* Nu se poate modifica precedența și asociativitatea operatorilor

**Selectați doar arfimațiile pe care le considerați a fi corecte**

* În cazul supraîncărcării operatorilor prin intermediul funcțiilor membru numărul de parametri este cu 1 mai mic decât paritatea operatorului
* Dacă o clasă nu are supraîncărcat operatorul egal atunci compilatorul va genera automat o supraîncărcare standard care va realiza copierea bit cu bit a datelor membru

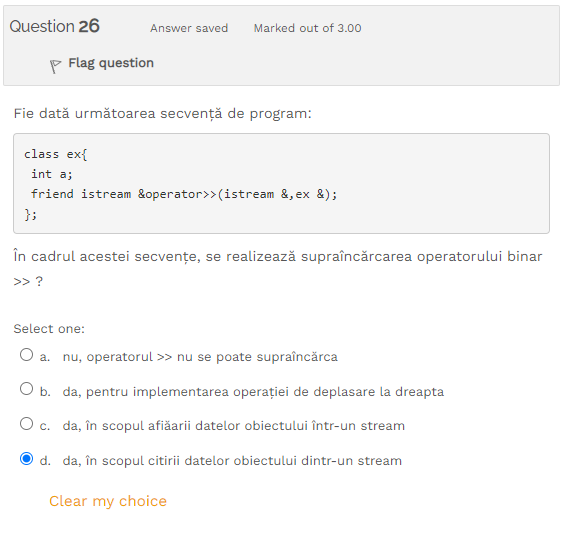
**Fie dată următoarea secvență de program:**



**Prin intermediul declarației numar f; care constructor este apelat?**

* Constructorul (1)

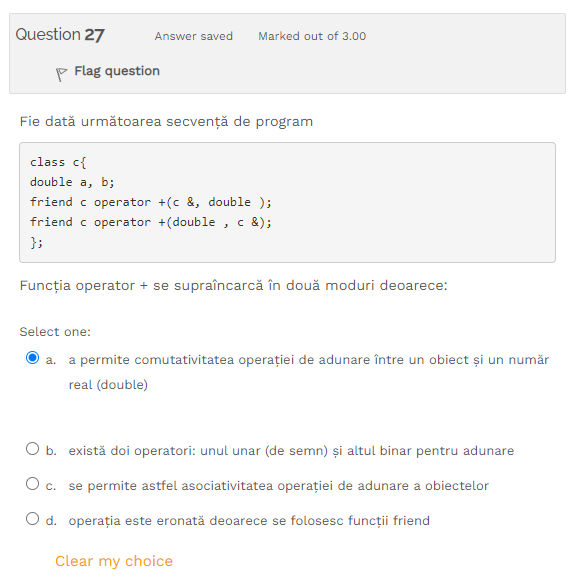
**Fie dată următoarea secvență de program:**



**În cadrul acestei secvențe, se realizează supraîncărcarea operatorului binar >> ?**

* Da, în scopul citirii datelor obiectului dintr-un stream

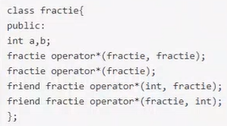
**Fie dată următoarea secvență de program:**



**Funcția operator + se supraîncarcă în două moduri deoarece:**

* A permite comutivitatea operației de adunare între un obiect și un număr real (double)

**Fie dată următoarea secvență de program:**

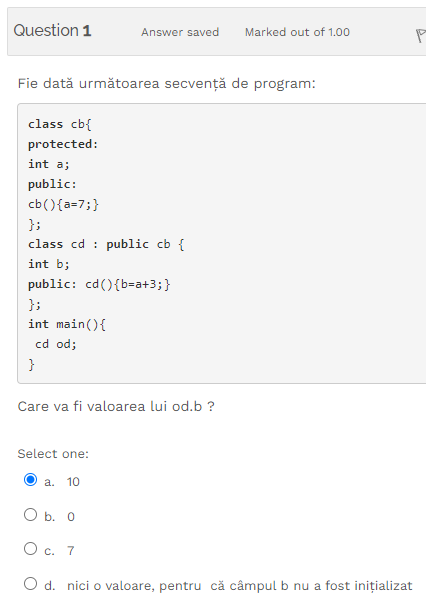
****

**Selectați instrucțiunea, prototipul căreia corespunde operației de inmulțire dintre două fracții:**

* fractie operator \*(fractie);

**Test: Moștenire. Derivarea simplă a claselor**

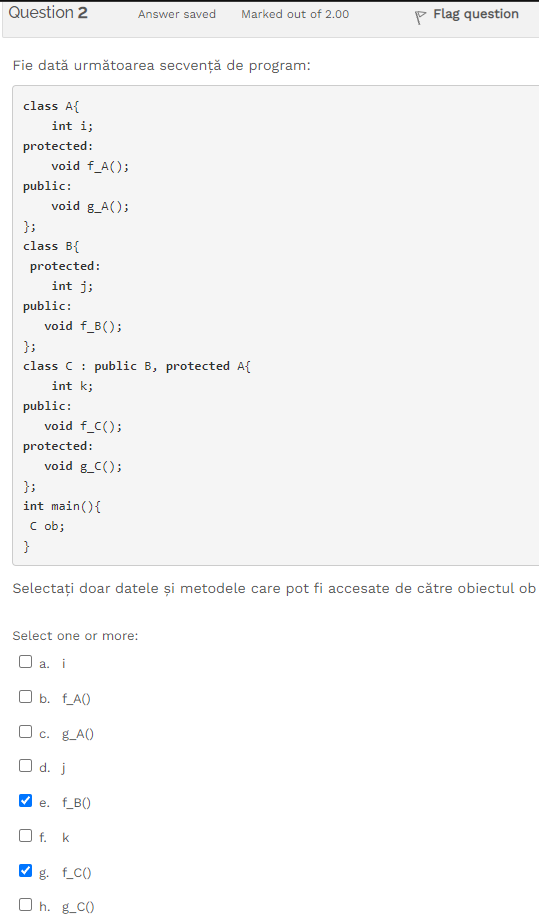
**Fie dată următoarea secvență de program:**

****

**Care va fi valoarea lui od.b ?**

* 10

**Fie dată următoarea secvență de program:**



**Selectați doar datele și metodele care pot fi accesate de către obiectul ob**

* f\_B()
* f\_C()

**Într-o relație de moștenire dintre două clase nu se moștenesc:**

* constructorii
* supraîncarcarea operatorului de atribuire =
* destructorul

**Fie dată o clasă CD care moștenește clasa CB, ambele clase având cate un destructor,  în cazul definirii unui obiect de tipul CD care destructor se va executa primul ?**

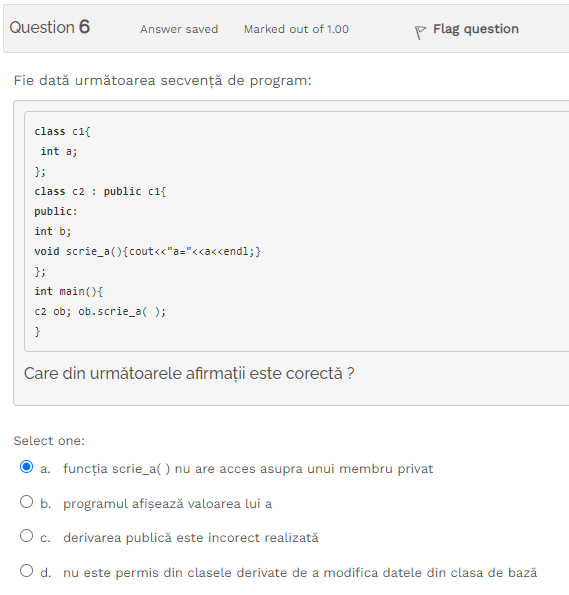
* al clasei CD

**Realizați corespondența dintre definiție și termenul utilizat**

Într-o relație de moștenire clasa de la care se preia structura şi comportamentul se numește ***clasa de bază***

Într-o relație de moștenire clasa care preia structura şi comportamentul se numește ***clasa derivată***

**Fie dată următoarea secvență de program:**



**Care din următoarele afirmații este corectă**

* funcția scrie\_a() nu are acces asupra unui membru privat

**Moştenirea este o formă de reutilizare a codului în care noile clase sunt create din clase existente prin:**

* toate afirmațiile enumerate

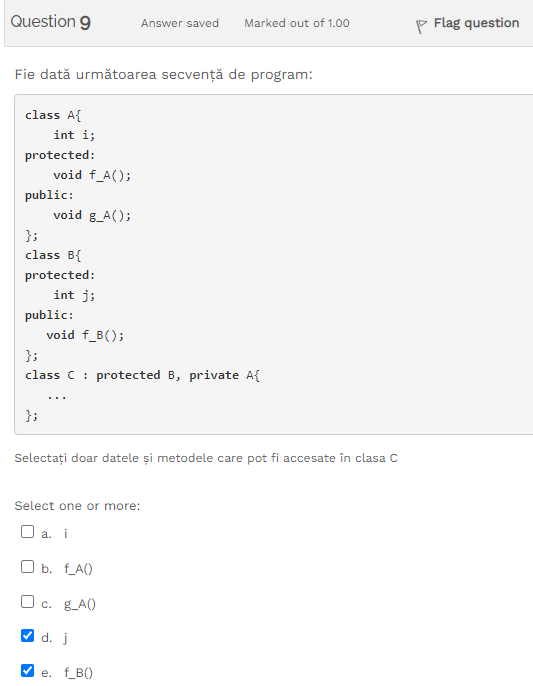
**Fie dată următoarea secvență de program:**



**Scrieți care va fi rezultatul execuției secvenței de program**

* 235

**Fie dată următoarea secvență de program:**



**Selectați doar datele și metodele care pot fi accesate în clasa C**

* j
* f\_B()

**Fie clasa D derivată public din clasa B. In acest caz o metodă publică din clasa D poate accesa o dată din secțiunea privată a clasei B ?**

* nu

**Test:** [**Funcții virtuale și polimorfism**](https://else.fcim.utm.md/mod/resource/view.php?id=15280)

**Într-o relație de moștenire, precizați care dintre următoarele funcții speciale pot fi declarate virtuale**

* destructorul

**O funcţie membră specială invocată printr-o referinţă sau un pointer la o anumită clasă de bază, de regulă publică, care este definită şi în clasele derivate se numește**

* funcție virtuală

**Pentru care din următoarele situații este util polimorfismul și funcțiile virtuale**

* Când într-o fază intermediară a proiectării şi implementării unei aplicaţii nu sunt cunoscute toate clasele care vor fi folosite
* Noile clase care sunt adăugate sunt integrate prin legarea dinamică

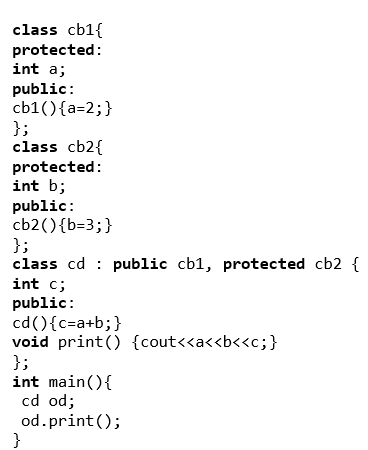
**Fie dată următoarea secvență de program:**

**class** AAA{  
 **protected**:  
 **virtual void** f();  
};  
**class** BBB: **public** AAA{  
 **protected**:  
 **void** f();  
};  
**class** CCC : **public** BBB{  
 **protected**:  
 **void** f();  
};

**Funcția f() definită în cele trei clase este**

* virtuală pentru toate cele trei clase

**Fie dată următoarea secvență de program**



**Scrieti care va fi rezultatul executiei secventei de program**

* 235

**Fie dată următoarea secvență de program:**

**class** A{ //....  
};  
**class** B : **public** A{  
// ...  
}

**Care din următoarele declarații sunt corecte**

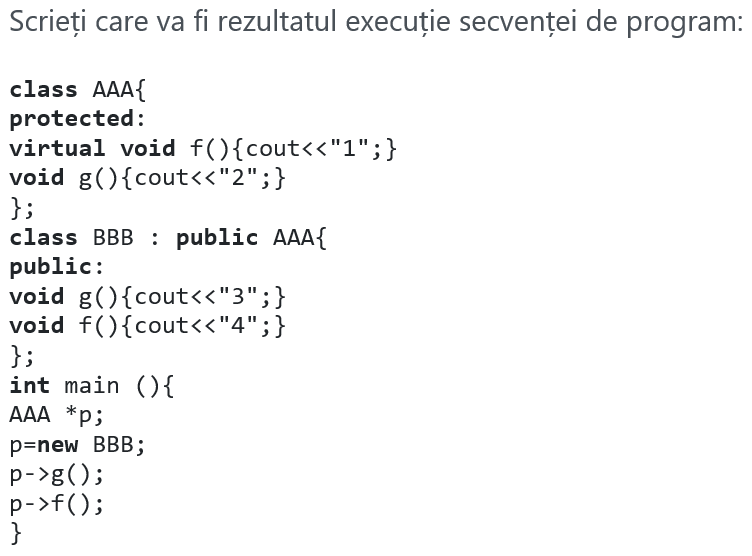
I.  A \*p; p=**new** B;  
II. B \*q; q=**new** A;

* Doar I

**Proprietatea unor entități de a se comporta diferit în funcție de tipul lor și a reacționa diferit la același mesaj se numește**

* Polimorfism

**Scrieti care va fi rezultatul executiei secventei de program**



* 24

**Polimorfismul se implementează prin intermediul:**

* funcțiilor virtuale

**Fie dată următoarea secvență de program:**

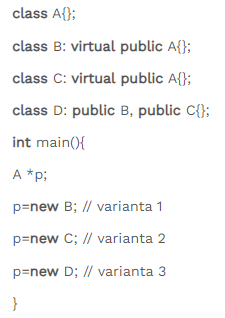
**class** AAA{  
**protected**:  
**int** d;  
**public**:  
**virtual int** f(){**return** 0;}  
**virtual void** g(){};  
};

**Din opțiunile enumerate specificați tipul clasei AAA**

* clasă abstractă

**Test: Moștenire. Derivarea multiplă a claselor**

**Fie dată următoarea secvență de program:**



**Care din următoarele afirmații este corectă**

* 1 + 2 + 3

**Care din următoarele relații este specifică moștenirii multiple**

* imprimantă, scaner, printer multifuncțional
* flux de date, flux de intrare, flux de ieșire, flux de intrare / ieșire

**Principiile moștenirii simple sunt valabile şi în cazul moștenirii multiple ?**

* True

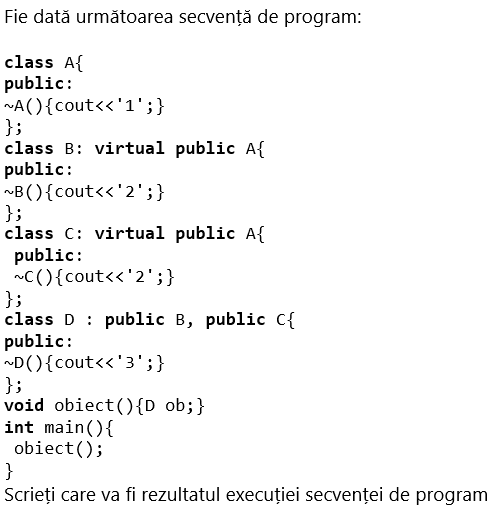
**Fie dată următoarea declarație:**

**class** A{};  
**class** C : **public** A{};  
**class** B : **public** A{};  
**class** E : **public** B{};  
**class** D : **public** B{};  
**class** H : **public** D{};  
**class** F : **public** E, **public** D{};  
**class** G : **public** F, **public** H{};

**Precizați care din clasele enumerate mai sus necesită a fi moștenite prin intermediul cuvântului virtual ?**

* B, D

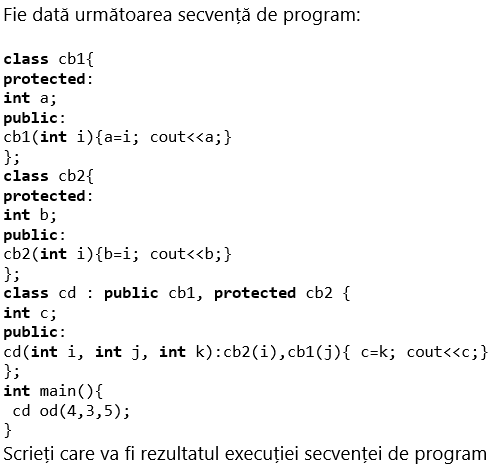
**Fie dată următoarea secvență de program:**



**Scrieti care va fi rezultatul executiei secventei de program**

* 3221

**Fie dată următoarea secvență de program:**



**Scrieti care va fi rezultatul executiei secventei de program**

* 345

**Fie dată următoarea secvență:**

**class** B1 { **int** x; };  
**class** B2 { **int** y; };  
**class** B3 { **int** z; };  
**class** B4 { **int** t; };  
**class** D: **public** B1, **private** B2, **protected** B3, **protected** 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 au acces la variabile pentru afișare sunt:**

* 1

**Test: Ierarhii de clase**

**Se consideră următoare secvență de program:**

**void** ... :: ... (){

TStudent \*q;  
 q=**new** TStudent;   
 cin>>q;  
 spate->next=q;   
 spate=q;  
}

**Aceasta reprezintă o secvență prin care se realizează operația de:**

* inserare a unui student în cadrul unei structuri de tip coadă

**O relație dintre clasele: grupă de elevi și șef de grupă se atribuie relației de:**

* Asociere

**Realizați corespondența dintre definiție și termenul utilizat**

* **Asociere**: O echipă are un manager
* **Agregare**: Grădina conține Flori
* **Compoziție**: Floarea este compusă din Petale

**Agregarea este folosită atunci când**

* Se dorește utilizarea trăsăturilor unei clase în interiorul altei clase

**Realizați corespondența dintre definiție și termenul utilizat**

* relaţie în care obiectele unei clase comunică/cunosc cu/despre de obiectele altei clase: **Asociere**
* relaţie în care obiectele unei clase  se regăsesc în cadrul altor obiecte: **Agregare**

**Prin compoziție avem în vedere o relație în care:**

* Subobiectele agregate aparțin exclusiv agregatului din care fac parte, iar durata de viață coincide cu cea a agregatului

**Care din următoarele exemple se atribuie relației de compoziție**

* Carte, pagini
* Automobil, motor

**Realizați corespondența dintre definiție și termenul utilizat**

* Obiectele ştiu unul despre altul astfel încât pot lucra împreună: **Asociere**
* Fiecare parte poate fi membru al unui singur obiect agregat: **Compoziție**
* Protejează integritatea obiectului, funcționează ca un singur tot întreg, controlul  se realizează printr-un obiect: **Agregare**

**O relație dintre clasele: automobil și volan se atribuie relației de:**

* Compoziție

**Conceptul de agregare se manifestă prin:**

* faptul că o clasă are o clasă conținută în ea

**Test: Programarea generică**

**Construirea unei funcții șablon este recomandat atunci când ...**

* sunt necesare mai multe funcții, care au parametri de tipuri diferite, dar care realizează aceleași prelucrări

**Presupunând că am definit funcția f ca fiind parametrizată,  care returnează un număr întreg, cu doi parametri diferiți  și considerând că avem intenția de a apela funcția f, câte funcții va crea compilatorul urmare a execuției următoarelor instrucțiuni ?**

**int** main(){  
 **int** a=f<**int, double**>(2,3.4);  
 **int** b=f<**int, int**>(2,3);  
 **int** c=f<**double, double**>(2.4,3.4);  
 **int** d=f<**int, int**>(2,2);  
 **int** e=f<**int, double**>(2.4,3.2);  
 **int** x=f<**double, int**>(2,3.4);  
}

* 4

**Presupunând că am definit funcția m ca fiind parametrizată,  care returnează un număr întreg, cu doi parametri diferiți  și considerând că avem intenția de a apela funcția m, care din următoarele declarații sunt corecte ?**

* int p=m<int, double>(2,3.4);
* cout<<m<int, char>(2,'a');

**Clasele care permit parametrizarea tipurilor de date asociate unei variabile membru sunt numite:**

* template

**Care din următoarele opțiuni precedă definiția unei clase șablon ?**

* template <class T>

**Funcțiile template nu pot fi supraîncărcate**

* False

**Selectați doar afirmațiile pe care le considerați a fi corecte**

* [Programarea generică](https://else.fcim.utm.md/mod/resource/view.php?id=16888) reprezintă o metodă de programare în care funcţiile şi clasele au parametri formali cu tip nedefinit
* Template-ul implementează conceptul de tip parametrizat
* clasă template specifică modul în care pot fi construite clase individuale, diferite prin tipul sau tipurile de date asupra cărora se operează

**Presupunând că am definit clasa șablon X cu un singur tip ca parametru și considerând că avem intenția de a declara un obiect ob, pe care să-l inițializăm la creare cu valoarea 7, care din următoarele declarații este corectă ?**

* X<int> ob(7);

**Pentru care din următoarele situații are sens de a implementa o funcție template**

* realizarea operației de sortare a elementelor: unui tablou de obiecte, unui tablou de numere întregi și a unui tablou de cuvinte
* realizarea operației de căutare a unei valori în cadul unei liste, stive, cozi

**Funcțiile care pot fi apelate pentru mai multe tipuri de date primite ca parametri se numesc**

* funcții template

**Test: Containeri și iteratori**

**Fie dat containerul a de tip număr întreg care conține elementele: 2 3 5. Ce va conține containerul a după următorul apel ?**

a.**insert**(a.**begin**(), 4);

* 4 2 3 5

**Selectați din opțiunile propuse doar cele pe care le considerați a fi corecte**

* Containerul deque se comportă atât ca o stivă, cât și ca o coadă
* Containerul vector permite inserarea / ștergerea elementelor doar la sfârșit

**Prin care din Algoritmii enumerați pot fi excluse elemente din cadrul unui container**

* remove
* unique

**Având în vedere funcționalitățile iteratorilor, care dintre următoarele opțiuni reprezintă o ierarhizare corectă a acestora**

* iteratori de intrare/ieșire; iteratori de înaintare; iteratori bidirecționali; iteratori cu acces aleatoriu

**Care funcție din cele enumerate exclude un anumit element dintr-un container de tip listă ?**

* remove()

**Care din următoarele containere nu acceptă iteratori**

* stack
* queue

**Selectați afirmația care definește un iterator**

* generalizare a pointerilor, fiind obiecte care indică alte obiecte

**Care din algoritmii enumerați nu modifică ordinea elementelor într-un container ?**

* fiind
* count

**Care din algoritmii enumerați verifică dacă elementele din două domenii sunt egale**

* equal

**Alegerea unui anumit tip de container depinde de:**

* funcţionalităţile oferite de container
* eficienţa (complexitatea) funcționalităților containerului

**Algoritmii au ca parametri iteratori și/sau funcții de tip obiect**

* True

**Precizați rezultatul execuției următoarei secvențe de program:**

**bool** ff(int i) { **return** (i<0);}  
...  
 **vector<int>** mv;  
 mv.push\_back(-2);  mv.push\_back(-3);  
 mv.push\_back(40);  mv.push\_back(-2);  
 mv.push\_back(60);  mv.push\_back(-60);  
 int nrP=(**int**) count\_if(mv.begin(),mv.end(), ff);  
 cout<<nrP;

* 4

**Realizați corespondența dintre termen și definiție**

|  |  |
| --- | --- |
| * permit accesul la elementele unui container, independent de modul în care aceste elemente sunt stocate **iteratori** |  |
| * operează asupra elementelor unei secvenţe doar indirect, prin intermediul iteratorilor **algoritmi** |  |
| * un obiect care stochează o colecţie de alte obiecte (elementele sale) **container** |  |

**Test: Tratarea excepțiilor**

**Fie dată următoarea secvență de program:**

**class** Err{  
 **public**:  **const char**\* mesaj;  
 Err(**const char**\* m){mesaj=m;} };  
**void** f(**int** n){  
 **switch** (n){  
  **case** 1: **throw** 0.5;   
  **case** 2: **case** 3: **throw** Err("Eroare n este 2 sau 3");   
  **case** 4: **throw** n; }   
 cout<<"Terminare funtie f()"<<endl; }  
**int** main (){  
 **int** n; cout<<"n="; cin>>n;  
 **try**{   
  cout<<"Execut functia f("<<n<<")\n";  
  f(n);   
  cout<<"Executie cu succes"<<endl; }   
 **catch** (double n){ cerr<<"Tratare eroare de tip double: "<<n; }   
 **catch** (Err e){ cerr<<"Tratare eroare de tip Err: "<<e.mesaj;}   
 **catch** (...){ cerr<<"Tratare eroare necunoscuta"; }  
}

**Identificați datele de intrare corespunzătoare mesajului afișat**

* Tratare eroare de tip E: Eroare n este 2 sau 3 – date de intrare: **n = 2**
* Tratare eroare de tip double: 0.5 – date de intrare: **n = 1**
* Tratare eroare necunoscuta – date de intrare: **n = 4**

**Mecanismul de tratare a  unei eroi presupune: scrierea codului care ar urma să se execute, apoi scrierea codului ce tratează situațiile neprevăzute**

* True

**Erorile pot fi tratate în timpul rulării unui program. O metodă de tratare a erorilor ar fi  de a introduce în program secvenţe de cod adaptate prevenirii unor posibile situaţii nedorite. Selectați din opțiunea/opțiunile propuse care constituie dezavantaje ale acestei abordări**

* codul programului este oarecum „poluat” cu secvenţe de procesare a erorilor
* codul programului devine mult mai greu de citit de un programator

**Erorile pot fi tratate în timpul rulării unui program. O metodă de tratare a erorilor ar fi  de a introduce în program secvenţe de cod adaptate prevenirii unor posibile situaţii nedorite. Selectați din opțiunea/opțiunile propuse care constituie avantaje ale acestei abordări**

* persoana care citeşte codul poate vedea modalitatea de prelucrare a erorii în vecinătatea codului
* poate determina dacă metoda de tratare a excepţiei este implementată corect

**Ce este o excepție**

1. o eroare care poate să apară la rularea unui program
2. un eveniment care se produce în timpul execuţiei unui program şi care perturbă fluxul normal al instrucţiunilor acestuia
3. o eroare care poate să apară la rularea unui program, sau altfel spus un eveniment deosebit ce poate apare pe parcursul execuției unui program și care necesită o deviere de la cursul normal de execuției al programului
4. **Toate opțiunile enumerate desfinesc noțiunea de excepție**

**Realizați corespondența dintre termen și semnificația acestuia**

* catch - **bloc de instrucțiuni care se executa în cazul în care apare o excepție**
* try - **marchează blocul de instrucțiuni care poate arunca excepții**
* throw - **mecanism prin care putem arunca pentru a semnala codului client apariția unei probleme**

**Care din următoarele exemple pot fi considerate excepții**

* încercarea de deschidere a unui fișier ce nu există
* depășirea limitelor unui tablou
* împărțirea la 0

**Selectați doar opțiunea/opțiunile pe care le considerați a fi corecte**

* Clauza catch nu trebuie neapărat sa fie în același metodă unde se aruncă excepția
* Când se aruncă o excepție, se caută cel mai apropiat bloc de catch care poate trata excepția
* Dacă nu avem clauză catch în funcția în care a apărut excepția, se caută clauza catch în funcția care a apelat funcția

**Realizați corespondența dintre termen și definiție**

* raportează violarea unei precondiţii – domain\_error
* în afara domeniului – out\_of\_range
* un obiect cu o lungime mai mare decât valoarea maximă reprezentabilă - lenght\_error