TEORIE POO(toate variantele din 2004-prezent)

Subiect 2004:

1) Prin ce se caracterizeaza o variabila statica a unei clase:

Proprietetati:

- variabilele statice sunt precedate de cuvantul- cheie static.
- o singura copie din acea variabila va exista pentru toata clasa
- o variabila statica declarata in clasa nu este definita(nu are zona de memorie alocata). Pentru a putea fi definite, sunt redeclarate in exteriorul clasei respective, folosind operatorul de rezolutie.
- Dat fiind ca variabilele statice sunt instantiate indiferent daca sunt declarate sau nu obiecte, putem accesa variabilele statice fara a ne folosi de vreun obiect, prin intermediul operatorului de rezolutie.

Exemplu:

```
  Class A
{
   static int x;
}
int A::x=5;
```

Exemplu de utilizare:

pentru a retine numar de instante ale unui obiect.(si a limita instatierea mai multor obiecte ex Singletone)

2)Prin ce se caracterizeaza o metoda statica a unei clase?

Sintaxa:

```
class A{

static void numara();

}
```

Proprietati:

-functiile statice pot opera doar asupra variabilelor statice din clasa.

- in cadrul acestor functii nu exista pointerul *this , pentru ca aceste metode se apeleaza indiferent daca exista sa nu obiecte instantiate.

-functiile statice nu pot avea natura virtuala

- nu pot exista mai multe variante ale aceleasi functii, una statica si una nestatica.

Exemplu de utilizare:

metodele statice sunt utilizate pentru accesarea datelor statice, care au fost declarate ca fiind private(principiul incapsularii)

3) Descrieti pe scurt diferenta dintre transferul prin valoare si transferul prin referinta in cazul apelului unei functii.

TRANSFER PRIN VALOARE	TRANSFER PRIN REFERINTA
-copierea valorii transmise ca parametru actual in parametrul formal(care este creat pe stiva la lansarea executiei functiei), ca o variabile locala. -pot aparea expresii sau nume de variabile.	-se evita consumul de resurse necesar pentru creerea unei alta variabile. - variabila transmisa prin referinta constanta nu va fi modificata -**NU** pot aparea expresii, ci doar nume de variabile(avem eroare logica)
sintaxa transmitere prin varloare: *int suma(int a, int b){..}*	**sintaxa transmitere prin referinta:** *void suma(int &S ,int a,int b){..}*

OBS: Daca transmitem obiecte prin apel prin referinta, nu se mai creeaza noi obiecte temporare, ci se lucreaza direct pe obiect trimis ca referinta.(nu se mai apeleaza copy-constructorul si nici destructorul).

4) Spuneti care este diferenta dintre incluziune de clase si mostenire de clase si cand se foloseste fiecare metoda.

INCLUZIUNEA DE CLASE	MOSTENIREA DE CLASE
- in cadrul unei clase avem campuri de date de tipul alor clase. -cand vrem ca doar anumite parti ale unei clase sa fie incluse in declararea unei clase noi	-mecanismul prin care o clasa este creata prin preluarea tuturor elementelor unei alte clase si adaugarea unor elemente noi. -clasa de la care se pleaca se numeste clasa de baza, iar clasa la care se ajunge este clasa derivata. la derivare putem asocia clasei un **atribut de acces.(vezi tabel)**
sintaxa: *class A{...};* *class B* *{* *A o1;* *int x;* *}*	**sintaxa** *class A{...};* *class B: public A* *{...}*

Accesul asupra membrilor mosteniti:

When the component is declared as:	When the class is inherited as:	The resulting access inside the subclass is:
public	public	Public
protected		protected
private		none
public	protected	protected
protected		protected
private		none
public	private	private
protected		private
private		none

5) Spuneti care dintre urmatoarele reprezinta mecanisme prin care se obtine polimorfismul de functii:

functiile friend , functiile inline , constructorii, functiile virtuale, destructorii.

Raspuns: functii inline,constructori, functii virtuale.

Obs: nu poate exista decat un singur destructor pentru fiecare clasa.

Subiect 2005(Mate info)

6) Descrieţi pe scurt diferenţa dintre funcţiile care returnează valoare şi cele carereturnează referinţă.

INTOARCERE PRIN VALOARE	**INTOARCERE PRIN REFERINTA**
-copierea valorii furnizate de functie intr-o variabila temporara,de tipul funtiei.	- crearea unei referinte temporare de tipul functiei in functia apelanta , catre variabila intoarsa ca rezultat de functia apelata.

7) Descrieţi pe scurt cum se pot defini funcţii de conversie între tipuri (clase).

Conversiile de tip se pot realiza prin:

- supraincarcarea operatorului unar „cast”(se poate face conversia dintr-un tip clasa intr-un tip fundamental sau intr-un alt tip clasa)
  sintaxa:
  operator TipData();
- prin intermediul constructorilor (consta in definirea unui constructor care primeste ca parametru tipul la care se face conversia). Constructorul intoarce intotdeauna un rezultat de tipul clasei de care apartine, prin urmare prin intermediul cosntructorilor se poate face conversia doar de la un tip fundamental sau un tip clasa la un alt tip clasa.
  OBS: in cazul conversiei de la un tip clasa la alt tip clasa, constructorul trebuie sa aiba acces la campurile private ale tipului clasa in care de

doreste a se face conversia si prin urmare trebuie declarat ca functie friend.

Exemplu :

Fie clasa B (tip de date predefinit sau clasa) si se doreste a se face conversia la tipul A

1) supraincarcam operatorul de cast al tipului in clasa B.

```
class B
{
        public : operator A();
}
```

2) supraincarcare constructorului de copiere cu un singur parametru de tip B:

```
A(B);
```

OBS: Constructorii cu un parametru sunt tratati ca metode de conversie implicita, chiar daca nu pentru asta au fost creati. Solutie: se pune in fata constructorului cuvantul explicit.

8)Spuneti care este diferenta dintre o clasa generica(template) si o calsa abstracta si in ce situatie se foloseste fiecare dintre ele:

CLASA GENERICA(TEMPLATE)	**CLASA ABSTRACTA**
-o clasa templete este un sablon utilizat pentru generarea unor clase concrete, care difera prin tipul anumitor date membre -este o metoda de implementare a polimorfismului de compilare	-clasa abstracta este o clasa in care avem cel putin o metoda virtuala pura. -reprezinta un tip incomplet care este folosit ca fundament pentru calsa derivata. - este o metoda a polimorfismului de executie
Sintaxa: *template<class Tip>class nume-clasa { ....}*	**Sintaxa:** *class A{ virtual f()=0}*

Odata ce am definit o clasa template, putem crea un exemplar al acesteia folosind: nume-clasa<tip obiect>ob;	Clasele virtuale sunt folosite cand cand ceea ce dorim sa modelam are caracteristici generale.

SUBIECT 2007

8) Spuneţi ce reprezintă o funcţie virtuală şi în ce condiţii o funcţie virtuală defineşte o clasa abstractă.

O functie virtuala este o functie care este definita in clasa de baza folosind cuvantul-cheie „virtual" si redefinita in clasa derivata. Redefinirea functiei in clasa derivata are prioritate in fata definitiei din clasa de baza.

- functiile virtuale reprezinta o modalitate de implentare a polimorfismului de executie.
- O functie virtuala pura este o functie care nu are definitie in calsa de baza.

Sintaxa pentru functie virtuala pura:

virtual nume_functie()=0;

-functiile virtuale pure trebuie redefinite in clasa derivata.
- o clasa care contine cel putin o functie virtuala pura este o clasa abstracta(si nu poate fi instantiata)

9) Spuneţi ce reprezintă o funcţie prietenă (friend) a unei clase.

- o functie friend are acces la membrii private si protected ai unei clase, fara sa fie funtie membra a clasei respective.

- pentru a declara o functie friend includem prototipul ei in clasa respectiva, precedat de cuvnatul cheie „friend".

ex: supraincarcarea operatorului „>>" pentru citire, respectiv „<< „ pentru afisare.

```
friend ostream &operator<<(ostream&os, const clasa& p);
```

10) Spuneti daca o variabila constanta poate fi transmisa ca parametru al unei functii si daca da, in ce situatie. Justificati !

O variabila constanta poate fi transmisa ca parametru al unei functii doar cand tipul parametrului functiei :

- **NU** este de tip referinta neconstanta (avem eroare pentru ca incercam sa convertim o variabila constanta la o referinta neconstanta)
- **NU** este de tip neconstant(se apeleaza constructorul de copiere care este de tip A(A &a) , prin urmare avem aceeasi eroare).

11) Descrieti pe scurt ce reprezinta obiectul implicit al unei metode:

- obiectul implicit este obiectul care apeleaza metoda.
- in cadrul metodei nu mai trebuie specificat carui obiect apartin campurile, pentru ca se subintelege ca ele apartin cuvantului implicit.
- obiectul implicit al unei metode se poate apela si folosind pointerul *this.
- este transmit in metoda prin referinta(referinta constanta daca metoda este constanta)

Exemplu:
pentru variabile : this->a;
pentru metode : this->f();

12) Descrieţi pe scurt ce reguli verifică supraîncărcarea operatorilor.

- nu putem defini operatori noi.(doar supraincarca pe cei existenti)
- nu putem modifica numarul operanzilor unui operator existent si nici sintaxa lor.
- nu putem modifica prioritatea si nici asociativitatea operatorilor existenti.

-exista operatori care **NU** se pot supraincarca:

exemplu : „ .” „ :: ” „ , ” „ ?: ”

-operatorii pot fi supraincarcati ca metode sau ca functii indepenente.

OBS: Nu toti operatorii care pot fi supraincarcati ca metode pot fi supraincarcati ca functii independente:

exemplu : „ =” „()” „[]” „->”

SUBIECT 2008-CTI

13) Descrieti trei metode de proiectare diferite prin care elementele unei unei clase se pot regasi in dublu exemplar, sub diferse forme, in definitia altei clase.

A. mostenire multipla, nevirtuala(fie B o clasa de baza si D1 si D2 clase derivate din B. Atunci cand D3 deriva din D1 si D2, clasa D3 are elemente in dublu exemplar din B)
B. prin compunere(poti sa ai doua obiecte din doua clase diferite, care mostenesc aceeasi clasa de baza)
C. mostenire si compunere (a aceleasi clase de baza)

14) Descrieti mostenirea virtuala si scopul in care aceasta este folosita :

Mostenirea virtuala este o metoda de implementare a polimorfismului de executie.

-mostenirea virtuala este necesara cand avem 2 clase derivate din clasa de baza si o a treia care mosteneste cele 2 clase(apare problema diamantului – anumite elemente sunt mostenite de doua ori, ceea ce conduce la ambiguitate).

15) Enumerati 3 metode de implementare a polimorfismului de compilare

-supraincarcarea functiilor/operatorilor

-template-uri (sabloane)

-parametrii cu valori implicite

16) Descrieti pe scurt comportamentul operatorului dynamic_cast<>

-se poate aplica pe pointeri sau referinta

-realizeaza convertirea unui obiect de un anumit tip la alt tip.

Sintaxa:

dynamic_cast<tipul la care vrem sa convertim>(ceea ce vrem sa convertim)

17) Descrieti diferenta dintre un pointer si o referinta

POINTER *	**REFERINTA &**
-pointeaza(arata) catre un obiect -obiectul spre care pointeaza poate fi schimbat -pointerul declarat de un anumit tip nu mai poate pointa catre un alt tip.	- este un pointer constant care se deferentiaza automat(un alt nume pentru un obiect) -odata atribuita unui obiect nu mai poate fi schimbata -prin modificare referintei se schimba si obiectul referit.

SUBIECT 2008-VARA

18) Descrieti mecanismul de tratare al exceptiilor

Motivatie: tratarea unitara a exceptiilor, automatizare a exceptiilor

-sunt folosite cuvintele cheie try, throw si catch.

Se implementeaza in felul urmator:

-semnalizarea aparitiei unei exceptii prin intermediul unei valori care semnifica eroarea respectiva

-receptionarea valorii prin intermediul careia se intransmite eroarea

-tratarea(rezolvarea) exceptiei semnalizate si receptionate.

sintax:

try

{ // instructiuni se executa pana la posibila aparitie a erorii

if(test) throw valoare

//instructiuni care se executa daca nu survine eroarea

catch(tip valoare)

{ // instructiuni pentru tratarea/rezolvarea exceptiei }

OBS: Daca aruncam o anumita valoare si nu exista un catch care sa prinda valoarea respectiva , atunci vom avea eroare de executie „Unhandled exception”. Aceasta problema se poate rezolva printr-un catch care prinde orice tip de valoare:

catch(...)(fara tip)

SUBIECTE 2009

19) Descrieti cum se comporta desconstructorii la mostenire:

- ordinea de executie a destructorilor este inversa ordinii de executie a constructorilor(adica de la clasa derivata la clasa de baza). In cazul mostenirii multiple, desconstructorii se apeleaza de la dreapta la stanga in lista de derivare.

20) Descrieti cum este implementat mecanismul de control al tipul in timpul executiei (RTTI)

RTTI= Runtime Type Information

RTTI cuprinde:

A) operatorul typeid : permite aflarea tipului obiectului la executie atunci cand ai doar un pointer sau o referinta catre acel tip.
exemplu:
int myint=50;
cout<<typeid(myint).name();

B) operatia dynamic_cast<>

-se poate aplica pe pointeri sau referinta

-realizeaza convertirea unui obiect de un anumit tip la alt tip.

Sintaxa:

dynamic_cast<tipul la care vrem sa convertim>(ceea ce vrem sa convertim)

21) Descrieti diferenta dintre o clasa si un obiect

CLASA	OBIECT
-tip abstract de date, care contine campuri de date(structuri de date) si metode. -nu se aloca memorie atunci cand este creata -este creata folosind cuvantul cheie class	-este o instanta a unei clase. -se aloca memorie atunci cand este creat.

22) Descrieti crearea dinamica de obiecte

-obiectele se pot crea dinamic folosind instructiunea **new** , care aloca memorie in timpul executie in zona de heap.

- constructorul initializeaza dona de memorie alocata.

-eliberarea memoriei se face cu ajutorul instructiunii delete.

- cand un obiect este creat dinamic, metodele si campurile sale se acceseaza cu „->” in loc de „ .”

23) Descrieti functiile sablon si dati exemplu de 3 situatii in care acestea NU genereaza o versiunea a functiei dintr-un sablon disponibil pentru functia respectiva.

o clasa templete este un sablon utilizat pentru generarea unor clase concrete, care difera prin tipul anumitor date membre

-este o metoda de implementare a polimorfismului de compilare

Sintaxa:

template<class Tip>class nume-clasa{ ....}

Odata ce am definit o clasa template, putem crea un exemplar al acesteia folosind:

nume-clasa<tip obiect>ob;

3 situatii in care un apel de functie nu genereaza o versiune a functiei dintr-un sablon:

-cand functia sablon are in lista de parametrii doi parametrii de tipul sablonului, iar noi apelam functia pentru 2 tipuri de date diferite.

-cand functia sablon are un singur parametru de tipul sablonului si mai exista o functie definita cu acelasi nume si cu tip specificat, atunci se va executa functia cu tipul deja specificat.

-cand functia sablon are in lista de parametrii 2 parametrii de tipul sablonului si noi apelam functia pentru un tip de date si un pointer la acel tip.

24) Descrieti diferenta dintre polimorfismul de compilare si cel de executie .

POLIMORFISMUL DE COMPILARE	**POLIMORFISMUL DE EXECUTIE**
-se decide la compilare **Curpinde:** -supraincarcare functiilor/operatorilor -template-uri(sabloane) parametrii cu valori implicite	-se decide la executie(in functie de anumite informatii disponible la momentul executie) **Cuprinde:** -instantierea dinamica -mostenirea -metodele virtuale

25) Descrieti in ce consta mecanismul de incapsulare

Mecanismul de incapsulare este procesul prin care se creeaza un nou tip de date(abstract) definind o clasa ca fiind formata din campuri de date(structuri de date) si metode.

Fiecare camp sau metoda are un atribut de acces:

-public : poate fi accesat de oriunde din program

-protected : poate fi accesat din clasa de baza si din clasele derivate

-private : poate fi accesat doar din clasă.

OBS: Daca nu este pus un atribut de acces, atunci implicit acesta este private:

Exemplu:

```
class A
{
private:
  int x, int y
public: A(){};
}
```

SUBIECTE 2010

26) Descrieti constructorul de copiere

-este un constructor cu un singur parametru de tipul clasei.

-forma implicita(data de compilator) face copierea bit cu bit a informatiei

Utilizare:

-pentru initializarea obiectelor din alte obiecte de acelasi tip

-la transmiterea parametrilor unei functii prin valoare

-la intoarecerea rezultatului unei functii prin valoarea.

27) Cum se pot supraincarca operatorii ca functii independente in C++

Supraincarcarea operatorilor se poate face ca functii membre sau ca functii independente, friend.(pentru a putea avea acces la campurile private ale clasei).

In cazul supraincarcarii ca functie independenta:

-nu avem pointerul *this.

-avem nevoie de toti operanzii ca parametru(pentru ca nu mai exista obiect implicit)

sintaxa:

tip retunrat nume clasa::operator #(lista argumente)

{ //supraincarcare operator }

Exemplu:

supraincarcarea operatorului „<<” pentru afisare sau „>>” pentru citire.

```
friend ostream &operator<<(ostream&os, const clasa& p);
```

OBS: Nu se pot supraincarca ca functie friend operatorii:

„ =” „ ()” „[]”

28) Descrieti in ce consta polimorfismul de executie folosind metode virtuale:

Mostenirea virtuala este o metoda de implementare a polimorfismului de executie.

-mostenirea virtuala este necesara cand avem 2 clase derivate din clasa de baza si o a treia care mosteneste cele 2 clase(apare problema diamantului – anumite elemente sunt mostenite de doua ori, ceea ce conduce la ambiguitate).

29) Descrieti cele doua feluri de folosire a cuvantului „virtual” la mostenire si in ce cazuri se folosesc:

A) Mostenirea virtuala se realizeaza folosind cuvantul cheie virtual.

Exexplu:

Class D: virtual public A

Utilitate:

pentru a rezolva ambiguitatea diamantului(doua clase deriva din clasa de baza si avem o a treia clasa care deriva din cele doua clase).

B) Cand definim functiile utilizand cuvantul cheie virtual, pentru a le putea redefini in clasa derivata

Exemplu:

```
class A
{
  int x;
public:
virtual int sum(){ //definitie functie }; // functie initiala
}
class B:public A
{
        int y;
 public:
 virtual int sum(){//redefinire functie };
}
```

30) Cum se poate supraincarca operatorul [] si care este utilizarea uzuala a acestuia.

Operatorul [], alaturi de operatorul () si operatorul „=” NU se pot supraincarca folosind functie friend. Acestia trebuie supraincarcati ca functii membre

-este considerat operator binar;

-operatorul [] poate fi folosit si la stanga unei atribuiri(obiectul intors este referinta)

Exemplu:

```
int v[100]

int & operator[](int i){return vector[i]}
```

31) Ce este lista de initializare a unui constructor si care este utilitatea ei.

-apare in implementarea constructorului, intre antentul si corpul acestuia.

- lista contien operatorul „ : ” urmat de numele fiecarui membru si valoarea acestuia, in ordinea in care memebrii apar in definita clasei

Utilizare:

-initializarea variabilelor statice

-initializarea referintelor

-initializarea campurilor pentru care nu exista un constructor implicit.

-initializare membrilor clasei de baza

-din motive de eficienta.

Exemplu:

```
 class A
{
int x, y;
public: A(int x, int y)::A(7,9){};
}
```

VARIANTE 2013

32) Sa se scrie diferenta dintre un pointer catre un obiect constat si un pointer constat catre un obiect:

POINTER CATRE OBIECT CONSTANT	POINTER CONSTANT CATRE OBIECT
- valoarea obiectului nu poate fi modificata, dar putem modifca obiectul catre care arata pointerul. SINTAXA: const tip de date * p	-pointerul nu poate fi modifica, dar valoarea catre care pointeaza da SINTAXA: tip de date const*p

IN 2014 SE REPETA INTREBARILE DE PANA ACUM

MULT SUCCES! 😊 😊 😊