Programare Orientata pe Obiecte
-teorie-

1. Descrieti pe scurt in ce consta mecanismul de incapsulare:

Mecanismul de incapsulare este procesul in care se creeaza un nou tip de date (abstract) definind o clasa ca fiind formata din campuri(structuri de date) si metode(functii/algoritmi). Fiecare camp sau metoda are un atribut de acces : **private** (poate fii accesat doar de metodele clasei) , protected (poate fi accesat si de metodele claselor derivate), public (poate fii accesat de oriunde din program). Daca nu este pus un atribut de acces atunci atributul de acces implicit va fi private.

Exemplu de definire a unei clase : class A { }; .

2. Spuneti ce este obiectul implicit al unei metode si descrieti pe scurt proprietatile pe care le cunoasteti despre acesta.

Obiectul implicit este obiectul care apeleaza metoda. In cadrul metodei campurile obiectului implicit nu vor mai avea specificat carui obiect apartin, subintelegandu-se ca ele apartin obiectului implicit (adica a celui pe care o apeleaza). Obiectul implicit al un metode se poate accesa si cu ajutorul pointerului *this in modul acesta : this->a; (pentru variabile) / this->f(); (pentru metode).

3. Descrieti pe scurt mostenirea virtuala si scopul in care este folosita.

Mostenirea virtuala este o metoda de implementare a polimorfismului de executie . Mostenirea virtuala este necesara atunci cand avem 2 clase derivate dintr-o clasa de baza iar o a patra clasă derivată din cele două. Se foloseşte derivarea virtuală pentru a evita o problemă de ambiguizare (problema diamantului).

4. Descrieti pe scurt proprietatile unui camp constant al unei clase.

Un camp constant al unei clase este un camp a carei valoare nu se poate modifica. El se poate initializa doar in lista de initializare a constructorului si nicaieri in alta parte. Pentru a implementa un camp constant se adauga inainte tipului de date cuvantul cheie const. De asemenea daca acest camp constant este un obiect , el nu poate apela.

5. Descrieti pe scurt mecanismul de tratare a exceptiilor.

Sunt folosite cuvintele cheie try, throw si catch. Se implementeaza in felul urmator :

* Semnalizarea aparitie un exceptii prin intermediul unei valori care reprezinta exceptia respectiva
* Receptionarea valorii prin care este semnalizata exceptia
* Tratarea (rezolvarea) exceptiei semnalizate si receptionate

In C++ se reprezinta astfel :

```
try {
 // instructiuni care se executa pana la posibila aparitie a exceptiei
if(test) throw valoare
// instructiuni care se executa daca nu survine exceptia}
catch (tip valoarea){
//instructiuni pentru tratarea / rezolvarea exceptiei }
```

Daca aruncam o valoare de anumit tip si nu exista un catch care sa prinda valori de acel timp , atunci vom avea eroare de executie ("Unhandled exception"). Exista deasemenea si posibilitatea de a pune catch(...) (fara tip) care prinde orice tip de valoare.

6. Descrieti pe scurt diferenta dintre un pointer si o referinta.

Referinta este practic un pointer constant care se dereferentiaza automat (un alias pentru un obiect care odata atribuita unui obiect nu se mai schimba). Prin modificarea referintei se modifica si obiectul in sine. Pointerul "pointeaza" (arata) catre un obiect , dar obiectul spre care pointeaza poate fii schimbat. In schimb, pointerul cand este declarat de un anumit tip nu mai poate pointa catre un tip diferit.

7. Descrieti pe scurt functiile virtuale si scopul in care sunt folosite.

O functie virtuala este o functie care este declarata ca fiind virtual in clasa de baza si redefinita de o clasa derivata. Pentru a declara o functie ca fiind virtuala , declararea sa este precedata de cuvantul cheie "virtual". Redefinirea functiei in clasa derivata modifica si are prioritate fata de definitia functiei din clasa de baza. Functiile virtuale reprezinta o metoda de implementare a polimorfismului de executie. O functie virtuala pura este o functie virtuala care nu are definitie in clasa de baza . Pentru a declara o astfel de functie se foloseste urmatoarea forma :

virtual tip nume_functie(//lista de parametri) = 0;

O clasa care contine cel putin o functie virtuala pura se numeste abstracta.

8. Descrieti pe scurt functiile statice si scopul in care sunt folosite.

Declararea unei functii statice este precedata de cuvantul cheie "static". O astfel de functie are urmatoarele restrictii :

*Ea poate sa aiba acces doar la alti membri (campuri de date/functii) de tip static ai clasei si bineinteles la functiile si datele globale.

*Ele nu pot avea un pointer de tip *this (deci nu au obiect implicit)

*Nu poate exista o versiune static si una non-static ale aceleasi functii.

Functiile statice pot fi folosite fara a mai declara un obiect, prin intermediul operatorului de rezolutie :: , ele fiind independente de obiect. Functiile statice si campurile de date statice au fost create pentru a se pastra principiul incapsularii. Scopul functiilor statice este acela ca pot "preinitializa" datele particulare de tip static, inainte de creearea efectiva a unui obiect.

9. Descrieti pe scurt diferenta dintre transferul parametrilor unei functii prin valoare si prin referinta constanta.

Transferul prin valoare reprezinta copierea valorii transmise ca parametru actual in parametru formal , care este creat pe stiva la lansarea in executie a functiei , ca o variabila locala (automatica), in timp ce transferul prin referinta constanta evita consumul suplimentare de memorie pentru crearea unei alte variabile si ne asigura in acelasi timp ca variabila transmisa ca referinta constanta nu va fi modificata .

10. Descrieti pe scurt pe scurt cum se comporta destructorii la mostenire.

Ordinea de executie a destructorilor este inversa ordinii de executie a constructorilor (adica de la clasele derivate spre clasa de baza). In cazul mostenirii multiple ordinea de executie a destructorilor e de la dreapta la stanga in lista de derivare.

11. Creare dinamica de obiecte.

Obiectele se pot crea dinamic folosind instructiunea "new" pentru alocarea memoriei in momentul executiei in heap. Constructorul initializeaza zona de memorie alocata. Eliberarea memoriei alocata astfel se poate face cu ajutorul instructiunii "delete". Cand un obiect este creat dinamic, metodele si campurile de date se acceseaza cu ajutorul operatorului "->" si nu cu ajutorul operatorului "." .

12. Proprietatile campurilor statice.

Proprietatile campurilor statice sunt urmatoarele :

* Campurile statice sunt comune tuturor obiectelor din clasa.
* Sunt independente de obiectele din clasa.
* Sunt create la lansare programului.
* Sunt declarate in interiorul clasei, dar trebuie declarate si global.

13. Proprietatile destructorului.

Proprietatile destructorilor sunt :

* Intr-o clasa exista un singur destructor.
* Exista un destructor implicit, pe care il putem suprascrie.
* In cazul mostenirii ordinea de executie a destructorilor este inversa ordinii de executie a constructorilor (adica de la clasele derivate spre clasa de baza).
* Se declara astfel : class A { ~A();};.
* Destructorul se apeleaza automat in momentul iesirii din blocul in care a fost declarat obiectul.

14. Descrieti diferenta dintre transferul parametrilor unei functii prin pointeri si prin referinta.

Transferul parametrilor prin pointeri reprezinta copierea adresei transmise ca parametru actual in parametrul formal, care este creat pe stiva la lansarea in executie a functiei ca variabila locala, modificarile asupra parametrilor fiind vizibile si in afara functiei, in timp ce transferul parametrilor prin referinta reprezinta crearea unei referinte locale catre variabila transmisa ca parametru actual la lansarea in executie a functiei.

15. Descrieti pe scurt cum este implementat mecanismul de control al tipului in timpul executiei (RTTI).

RTTI cuprinde operatorul typeid si operatia dynamic_cast<>. "typeid()" permite aflarea tipului obiectului (clasa din care face parte) la executie atunci cand ai doar un pointer sau o referinta catre acel tip. "dynamic_cast<>" se poate aplica pe pointeri sau referinta, realizeaza convertirea unui obiect de un anumit tip la alt tip si are urmatoarea implementare : dynamic_cast <tipul la care vrem sa convertim> (ceea ce vrem sa convertim).

16. Descrieti pe scurt diferenta dintre o clasa si un obiect.

Clasa reprezinta un tip abstract de date , format din campuri de date (structuri de date) si metode (functii/algoritmi), in timp ce, un obiect este o instanta a unei clase.

17. Descrieti pe scurt functiile sablon si dati exemplu de trei situatii in care un apel de functie nu genereaza o versiune a functiei dintr-un sablon disponibil pentru functia respectiva .

Functiile sablon sunt o metoda de implementare a polimorfismului de compilare. Cu un sablon este posibil sa cream functii generice si clase generice. O functie generica defineste un set general de operatii care vor fi aplicate unor tipuri de date variate. Unei astfel de functii tipul de date asupra caruia va opera ii este transmis ca parametru. O astfel de functie este creata cu ajutorul cuvantului cheie "template" iar forma generala a unei defenitii de functie de tip template este :

template <class Tip> tip_returnat nume_functie (lista parametri) {//corpul functiei}

Trei 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 doua tipuri de date diferite.

*Cand functia sablon are in lista de parametrii un parametru de tipul sablonului si cand mai avem o functie definita cu acelasi nume si un tip specificat , atunci se va executa functia cu tipul deja specificat.

*Cand functia sablon are in lista de parametrii doi parametrii de tipul sablonului, iar noi apelam functia pentru un tip si un pointer la acel tip.

18. Descrieti pe scurt diferenta dintre polimorfismul de compilare si cel de executie.

Polimorfismul este o caracteristica a programarii orientate pe obiecte care se refera la comportamentul metodelor (posibilitatea ca o functie sa se comporte diferit in contexte diferite). Exista doua tipuri de polimorfism : de compilare (care se decide la compilare) si de executie (care se decide la excutie in functie de anumite informatii disponibile la momentul executiei). Polimorfismul de compilare cuprinde : supraincarcarea functiilor/operatorilor , sabloane si parametrii cu valori implicite , in timp ce polimorfismul de executie cuprinde : insantiere dinamica, mostenire si metode virtuale.

19. Descrieti pe scurt diferenta dintre functiile care returneaza valoarea si cele care returneaza referinta

Intoarcerea rezultatului prin valoare reprezinta copierea valorii furnizate de functia apelata pentru o instructiune return intr-o variabila temporara, de tipul functiei , creata in functia apelanta, in timp ce intoarcerea rezultatului prin referinta reprezinta crearea unei referinte temporare de tipul functiei in functia apelanta , catre variabila intoarsa ca rezultat de functia apelata.

20. Descrieti pe scurt cum se pot defini functii de conversie intre tipuri (clase).

Avem situatia urmatoare : B - tip de date deja existent (clasa sau tip predefinit) si definim o clasa noua A .

*B->A : se defineste un constructor in clasa A cu un singur parametru de tip B : A(B).

*A->B : supraincarcam operatorul "cast" al tipului B in clasa A : (B)A. Sintaxa pentru acest caz este : class A{ public: operator B(); }

Observatie : 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.

21. Spuneti care este diferenta dintre clasa generica (template) si clasa abstracta si in ce situatii se foloseste fiecare dintre ele.

O clasa generica este o clasa care defineste toti algoritmii definiti de ea , dar tipul de date care este manevrat efectiv va fi specificat ca un parametru la crearea obiectului acelei clase. Forma generala a declararii unei clase generice este : template<class Tip>class nume-clasa{ ....} . Odata ce am construit o clasa generica putem crea un anumit exemplar al acesteia folosind forma generala : nume-clasa<tip>ob;. Pe de alta parte clasa abstracta este o clasa in care avem cel putin o metoda virtuala pura. O alta diferenta este ca clasa generica reprezinta o metoda a polimorfismului de compilare , in timp ce clasa abstracta reprezinta o metoda a polimorfismului de executie. O clasa abstracta constituie un tip incomplet care este folosit ca fundament pentru clasele derivate. Clasele generice sunt folositoare cand o clasa contine caracteristici generale.

22. Descrieti pe scurt diferenta dintre transferul prin valoare si transferul prin referinta al parametrilor in cazul apelului unei functii.

Transferul prin valoare reprezinta copierea valorii transmise ca parametru actual in parametru formal , care este creat pe stiva la lansarea in executie a functiei , ca o variabila locala (automatica), in timp ce transferul prin referinta reprezinta crearea unei referinte locale catre variabila transmisa ca parametru actual la lansarea in executie a functiei.

23. Spuneti care este diferenta dintre incluziunea de clase si mostenirea de clase si cand se foloseste fiecare metoda.

Mostenirea este mecanismul prin care o clasa noua este creata prin preluarea tuturor elementelor unei clase existente si adaugarea unor elemente noi specifice. Clasa de la care se pleaca se numeste clasa de baza, iar clasa la care se ajunge se numeste clasa derivata. La derivare putem asocia clasei de baza un atribut de acces : public (elementele din clasa de baza isi pastreaza atributele de acces in clasa derivata), private (elementele din clasa de baza devin private in clasa derivata) si protected (elementele publice din clasa de baza devin protected in clasa derivata). Incluziunea de clase se manifesta atunci cand intr-o clasa avem campuri de date de tipul altor clase. Mostenirea se foloseste cand vrem ca o clasa sa incorporeze in declararea sa alta clasa, astfel v-om construi o ierarhie de clase, in timp ce incluziunea este folosita cand vrem ca doar anumite parti a unei clase deja existente sa fie incluse in declararea unei clase noi. Observatie : Incluziunea -> verbul “a avea” ; Mostenire -> verbul “a fi”.

24. Descrieti pe scurt constructorul de copiere si situatiile in care acesta este folosit.

Constructorul de copiere este un constructor cu un singur parametru de tipul clasei. El are o forma implicita care copiaza octet cu octet . Utilizarea constructorului de copiere :

* Pentru initializarea obiectelor din alte obiecte de acelasi tip
* La transferul parametrilor unei functii prin valoare
* La intoarcerea rezultatului unei functii prin valoare

25. Spuneti daca o variabila constanta poate fi transmisa ca parametru al unei functii si daca da in ce situatii. Justificati.

O variabila constanta poate fi transmisa ca parametru al unei functii doar cand tipul parametrului functiei nu este de tip referinta neconstanta si cand nu este de tip neconstant , deoarece in momentul in care unul dintre parametrii functiei este de tipul mentionat atunci

am avea in primul caz : convertirea unei variabile constante la o referinta neconstanta (eroare) , iar in al doilea caz : se apeleaza constructorul de copiere care este de tipul A(A &a) , deci din nou va fi aceeasi eroare.

26. Spuneti ce reprezinta o functie prietena a unei clase.

Este posibil sa permitem unei functii care nu este membru sa aiba acces la membri paritculari ai unei clase folosint un friend (o functie prietena). O functie friend are acces la membrii private si protected ai clasei careia ii este prietena . Pentru a declara o functie friend includem prototipul ei in acea clasa precedat de cuvantul cheie friend .

27. Descrieti pe scurt ce reguli verifica supraincarcarea operatorilor.

Regulile care verifica supraincarcarea operatorilor sunt urmatoarele :

* Nu putem defini operatori noi. Doar supraincarcam operatori deja 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 independente. Nu toti operatorii care pot fi supraincarcati ca metode pot fi supraincarcati ca functii independente ! (exemplu: “= “, “( )”, “[ ] “,”->”)

28. Descrieti trei metode de proiectare diferite prin care elementele unei clase se pot regăsi în dublu exemplar, sub diverse forme, în definitia altei clase.

Cele trei metode de proiectare sunt urmatoarele:

* mostenire multipla, nevirtuala (cand mostenesti o clasa si inca una care e derivata din prima clasa mostenita)
* prin compunere (poti sa ai doua obiecte din doua clase diferite care mostenesc aceeasi clasa de baza
* mostenire si compunere (a aceleiasi clase de baza)

29. Enumerati trei metode de implementare a polimorfismului de compilare.

Polimorfismul de compilare se poate realiza prin :

* Supraincarcare functii si operatori
* Sabloane
* Parametrii cu valori implicite

30. Descrieti pe scurt comportamentul operatorului dynamic_cast.

Operatorul "dynamic_cast<>" se poate aplica pe pointeri sau referinta, realizeaza convertirea unui obiect de un anumit tip la alt tip si are urmatoarea implementare : dynamic_cast <tipul la care vrem sa convertim> (ceea ce vrem sa convertim).