Intrebari Teoretice Examen

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

INTREBARI TEORETICE EXAMEN

1. Cat ocupa o variabila de tip int?
   32 bits (4 bytes)

2. Cat ocupa o variabila de tip short int?
   16 bits (2 bytes)

3. Cat ocupa o variabila de tip long int?
   Cel putin 32 bits(4 bytes)

4. Cat ocupa o variabila de tip long long int?
   8 bytes

5. Cat ocupa o variabila de tip bool?
   1 byte

6. Cat ocupa o variabila de tip float?
   Float- reprezinta un nr real stocat in virgula mobila, in general IEEE 754 simpla precizie. Dimensiunea va fi de 4 octeti si numarul va avea cel putin 7 zecimale.
   4 bytes

7. Cat ocupa o variabila de tip double?
   Double- reprezinta un nr real stocat in virgula mobila, in general IEE 754 dubla precizie. Dimensiunea va fi de 8 octeti si numarul va avea cel putin 15 zecimale exacte.
   8 bytes

8. Cat ocupa o variabila de tip long double?
   Este >= decat double, care larandul lui este >= float.
   8 bytes

9. Cat ocupa o variabila de tip char?
   8 bits (1 byte)

10. Cat ocupa o variabila de tip pointer la int - int *?
    Pe 32 de bits, un pointer ocupa 4 bytes.

11. Cat ocupa o variabila de tip pointer la char- char *?
    4 bytes

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

Un pointer. Este utilizat in special in metode, pt a accesa obiectul curent, salvand adresa acelui obiect.

13. Ce reprezinta *this* in constructor ?

Adresa obiectului curent. (ce urmeaza sa fie construit)

14. Ce reprezinta *this* in destructor ?

Adresa obiectului curent. (ce urmeaza sa fie distrus)

15. Ce reprezinta *this* intr-o functie membra ?

Arata adresa obiectului pentru care este apelata functia membra.

16. Ce reprezinta *this* intr-o functie statica ?

Metodele statice nu primesc pointerul „this" si nu pot prelucra atributele non-statice ale clasei.

Pointerul „this" nu poate fi folosit pentru un obiect care apeleaza functii statice (deoarece nu acestea nu faac parte din clasa)

17. Ce reprezinta *this* in constructorul de copiere ?

Adresa obiectului in care se copiaza (obiectul prelucrat).

18. Ce reprezinta *this* in cadrul supraincarcarii operatorului = ?

Adresa obiectului in care se copiaza (obiectul prelucrat).

19. Ce este o functie membra ?

Numita si metoda, este o functie declarata sau definita in cadrul clasei care defineste comportamente pentru obicetele clasei; primeste parametrul „this".

Metodele accesibile din afara clasei formeaza interfata unui obiect.

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 ?

Se definesc folosind cuvantul cheie static. Sunt metode ce fac referire la clasa si nu la obiect (asemanatoare unor metode globale). Nu primesc pointerul „this" si nu pot prelucra atributele non-statice ale clasei.

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 ?

Se definesc folosind cuvântul cheie static și sunt atribute ale clasei, nu ale obiectului.

Valoarea atributului este comună pentru toate obiectele (odată schimbată valoarea pentru un obiect se schimbă pentru toate celelalte).

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 ?

Este un atribut al clasei care nu se mai poate modificat in timpul executiei odata ce a fost initializat.

23. Cand se poate initializa un atribut constant ?

La declararea atributului sau prin constructor la declararea obiectului (Un atribut constant se poate initializa doar la crearea obietului, in lista de initializare a constructorului.)

24. Ce reprezinta supraincarcarea ?

Supraincarcarea este prima forma de polimorfism din P.O.O., fiind considerata polimorfism slab sau „early binding”.

Presupune existența a cel puțin două funcții (din același context) care au același nume, dar diferă prin numărul sau tipul parametrilor (prin signatură).

25. Care este utilitatea supraincarcarii ?

Supraincarcarea functiilor presupune declararea si definirea unor functii cu acelasi nume, astfel incat, in functie de parametrii transmisi, la compilare, sa se poata decida care dintre functiile cu acelasi nume sa fie adresata.

26. Ce reprezinta supradefinirea ?

Supradefinirea functiilor presupune declararea si definirea unor functii cu exact aceeasi signatura (nu difera prin numarul sau tipul parametrilor), iar alegerea functiei potrivite (aferente apelului) se face la momentul executiei (“late binding”).

27. Care este utilitatea supradefinirii ?

Atunci cand se doreste crearea unei noi functionalitati, se deriveaza o clasa din cea existenta si se supradefinesc functiile, astfel ramanand doar obiectul a carui adresa este salvata ca pointer sa fie modificat.

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 funcție virtuală sau metodă virtuală este o funcție al cărei comportament este determinat de către definiția unei funcții cu aceeași semnătură cea mai îndepărtată pe linia succesorală a obiectului în care este apelată. Acest concept este o foarte importantă parte din porțiunea de polimorfism a paradigmei de programare pe obiecte (POO).

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

DEF polimorfism: Aceeasi entitate poate avea mai multe intelesuri, functionalitati in functie de context.

- **Pointerii implementeaza o forma incipienta de polimorfism (un pointer la o functie poate face diferite tipuri de prelucrari prin schimbarea adresei la care pointeaza).**

- **Supaincarcarea functiilor este prima forma de polimorfism din POO, ea implementand conceptul de polimorfism in C++.**

- **Supradefinirea este cea de a doua forma de polimorfism.**

- **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?

- **privat (private) - membrii (atributele și metodele) definiți în această zonă pot fi accesați doar de clasa în care sunt definiți - domeniu implicit**

- **protejat (protected) - membrii definiți în această zonă de vizibilitate pot fi accesați doar de clasa în care sunt definiți și de clasele derivate din ea (ce o moștenesc - mai multe detalii despre derivare în cursurile viitoare)**

- **public (public) - membrii definiti în această zonă de vizibilitate pot fi accesati de orice entitate (clasa respectivă, clasele derivate, alte clase, funcția main, etc.)**

Funcțiile de acces (getteri-citire si setteri-scriere) sunt funcții special definite pentru a asigura accesul controlat asupra atributelor private ale clasei.

Atributele private nu pot fi accesate de clasa derivata, insa acestea tot se mostenesc.

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

Se foloseste doar atunci cand vrem ca datele si functiile membre dintr-o clasa sa fie folosite doar de catre functiile care apartin clasei respective.

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?

Acestia sunt utilizati atunci cand sunt create obiecte.

- **constructor implicit (fără parametri)**
- **constructor(i) cu parametri**
- **constructor de copiere**

Constructorul implicit poate fi creat de utilizator si creat in lipsa existentei oricarui alt constructor.

Constructorul cu parametri are oricati parametri doreste utilizatorul si acestia au sau nu valori implicite.

Constructorul de copiere copiază informații dintr-un obiect existent într-un obiect nou.

Daca nu este creat explicit, atunci se creaza automat un constructor de copiere in clasă ce realizează copiere byte cu byte a obiectului curent. Acest lucru poate crea probleme atunci când clasa are membri pointeri.

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

Apelat automat atunci când se copiază informații dintr-un obiect existent intr-un alt obiect existent (spre deosebire de constructorul de copiere ce copiază informații dintr-un obiect existent într-un obiect nou)

34. Cand este apelat constructorul de copiere ?

Compilatorul apelează totuşi implicit constructorul de copiere în două situaţii:

- **transmiterea unui obiect ca parametru într-o funcție prin valoare**
- **returnarea unui obiect prin valoare dintr-o funcție**

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* ?

Memory leak apar în C ++ atunci când programatorii alocă memorie utilizând un cuvânt cheie nou şi uită să repartizeze memoria utilizând funcţia delete () sau operatorul delete []. Una dintre cele mai multe scurgeri de memorie apare în C ++ prin utilizarea unui operator de ştergere greşit.

37. Ce este un dangling pointer?

Un dangling pointer este un pointer care indică date nevalide sau date care nu mai sunt valabile.

38. Care este rolul destructorului ?

Este utilizat la distrugerea obiectelor (dezalocarea memoriei alocate in constructori, eliberarea resurselor, etc).

39. Cand se apeleaza destructorul ?

O funcţie de distrugere este apelată automat atunci când obiectul iese din scop: (1) funcţia se termină (2) programul se încheie (3) se termină un bloc care conţine variabile locale (4) se apelează un operator de ştergere

40. Ce este memoria HEAP?

Se numeşte heap, deoarece este o grămadă de spaţiu de memorie disponibil programatorilor pentru alocare şi dezalocare. Dacă un programator nu gestionează bine această memorie, se poate scurge memoria în program.

41. Cum se aloca spatiu de memorie in HEAP?

Memoria este alocată în timpul executării instrucţiunilor scrise de programatori, cu ajutorul comenzii new.

42. Cum se elibereaza memoria in HEAP ?

Memoria in heap este eliberata prin utilizarea destructorului.

43. Cum se genereaza un *memory leak* ?

Cand se uita a se sterge memeria alocata utilizand delete. 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?

Se pot utiliza şi metode de acces pentru a putea citi/afişa atribute private.

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

In limbajul C++ se poate acorda acces anumitor functii sau anumitor clase asupra tuturor membrilor clasei (fie ei privati sau protected).

Aceste funcţii sau clase se numesc prietene (friend) şi trebuie anunțate în cadrul clasei in care se permite accesul în următorul mod:

- **Pentru clase: friend class NumeClasa;**
- **Pentru functii: friend tip_returnat NumeFunctie(tipParam1, tipParam2, etc);**
- **Functiile friend nu primesc pointerul this.**

46. Ce reprezinta conceptul de incapsulare ?

Entitatile pot ascunde informatii sau comportamente de alte entitati.

47. Care este diferenta dintre supradefinire si supraincarcare?

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

Cu functia strlen().

49. Care e vizibilitatea implicita in cadrul unei clase?

Private.

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

Inmultire **Marcarea alocarii dinamice**

Declararea unui pointer **Extragerea valorii de la o adresa**

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?

Obiect de tip (clasa) istream (stream-ul standard de intrare).

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

Obiect de tip (clasa) ostream (stream-ul standard de iesire).

54. Ce reprezinta prescurtarea STL?

Standard Template Library (Biblioteca standard de sabloane)

55. Dati exemplu de un container STL.

a) **Secventiale (elementele sunt salvate consecutiv):**
   vector, list, deque
b) **Asociative (asociaza fiecarui element o relatie de ordine sau o alta valoare):**
   set, multiset, map, multimap
c) **Adaptive (adapteaza containerele secventiale a. i. sa aiba o anumita modalitate de functionare- de ex. FIFO, LIFO):**
   stack, queue, priority_queue

56. Dati exemplu de un algoritm STL.

Sortare, copiere, parcurgere, cautare, alte prelucrari.

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

Prin metoda (functie membra) si prin functie globala (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 globala (in afara clasei).

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

DOAR prin functie globala, care contine un parametru de tip int si un parametru de tipul clasei; astfel se conserva comutativitatea operatorului +.

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

Cand supraincarcarea se face global si vrem sa ii acordam acces functiei globale asupra tuturor membrilor clasei (fie ei privati sau protected).

Atunci cand avem atribute private ce trebuiesc folosite 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 ?

Se foloseste doar atunci cand vrem ca datele si functiile membre dintr-o clasa sa fie folosite doar de catre functiile care apartin clasei si de functiile membre ale claselor derivate din clasa respectiva.

65. Care este diferenta dintre specificatorii class si struct ?

? în structuri vizibilitatea implicită este cea publică! + intre cele două tipuri de entități există o diferență la nivel de concept: structurile se utilizează pentru construcții simple, pe când clasele pentru cele cu o complexitate ridicată.

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

66. Ce reprezinta o clasa ?

Clasele sunt entitati asemanatoare structurilor ce incapsuleaza caracteristici(sub forma de atribute) si comportamente (sub forma de metode) prin abstractizarea entitatilor din lumea reala. Sunt baza POO si permit crearea de tipuri de date noi.

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

In C++ singura diferenta practica dintre clase si structuri este data de faptul ca in structuri, vizibilitatea implicita este cea publica. Cu toate acestea intre cele două tipuri de entități există o diferență la nivel de concept: structurile se utilizează pentru construcții simple, pe când clasele pentru cele cu o complexitate ridicată.

67. Ce reprezinta un obiect ?

Obiectele sunt instantele unei clase (variabile de tipul unei clase).

68. Ce reprezinta o instanta a unei clase ?

Un obiect/ o variabila de tipul unei clase.

69. Ce este un constructor ?

Constructorii sunt metode speciale ce au numele clasei si nu au tip returnat. Ei sunt utilizati la crearea obiectelor (initializarea atributelor).

70. Ce reprezinta termenul de up-casting ?

Sau conversia derivat-baza . Permite transformarea obiectelor de tip derivat in obiecte de tipul clasei de baza (toate telefoanele mobile pliabile sunt telefoane mobile pana la urma) . Functioneaza implicit daca derivarea este publica

71. Ce reprezinta termenul de down-casting ?

(**invers ^) Conversia baza-derivat.**

72. Ce este un framework de clase ?

O colectie de clase si functii globale sablon, predefinite, ce usureaza lucrul cu structuri de date, parcurgeri, algoritmi, s.a. (O ierahie de clase)

73. Pot fi definite metode constante ?

Nu, doar statice.

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

Ordinea de apel a constructorilor este baza-derivat (mai intai se apeleaza constructorul din clasa de baza si mai apoi cel din clasa derivata)

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

Ordinea de apel a destructorilor este derivat-baza (mai intai se apeleaza destructorul din clasa derivata si mai apoi 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.

Se asigura apelul functiei corecte (in cazul in care sunt functii cu acelasi nume, nr. de parametri in clasa de baza si clasa derivata) pentru un obiect, indiferent de tipul de referinta folosit pentru apel.

77. Ce este o functie virtuala pura ?

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

Pot exista situatii cand o functie nu are neaparat sens pentru o clasa de baza, ci doar pentru clasele derivate . Functiile virtuale ce nu au corp si doar obliga clasele derivate sa le supradefinieasca.

78. Ce este o clasa abstracta ?

O clasa ce contine cel putin o metoda virtuala pura poarta denumirea de clasa abstracta

79. Ce restrictii impune o clasa abstracta ?

Clasele abstracte nu se pot instantia (nu se pot crea obiecte de tipul clasei abstracte), ele au rolul doar de baza pentru derivari

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.

Derivarea din mai multe clase simultan, Utilizarea acesteia modeleaza faptul ca tipul derivat este, in acelasi timp, si oricare dintre tipurile de baza .Se pot deriva oricat de multe clase, iar tipul de derivare se decide pentru fiecare clasa ce este derivata in parte .Ordinea de apel a constructorilor/destructorilor in acest caz este data de ordinea derivarii (nu de cea definita explicit in cazul unor constructori) . Upcasting-ul se poate face catre oricare dintre clasele de baza

81. Ce sunt functiile inline ?

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

Deobicei (depinde de compilator) metodele scrise în interiorul clasei sunt considerate metode inline e Metodele scrise în afara clasei pot fi transformate în metode inline prin folosirea cuvântului cheie inline (acest lucru va duce la mutarea lor automată în clasă în faza de preprocesare, asemănător cu ce se întâmplă cu define)

82. Ce reprezinta conceptul de *is a* ? Exemplificati

Derivarea. Ex: Masina de pompieri este o masina.

83. Ce reprezinta conceptul de *has a* ? Exemplificati

Compunerea. Ex: Spitalul are un medic.

84. Ce este mostenirea virtuala?

Membrii comuni sunt mosteniti o singura data.

85. In ce moment are loc efectiv supradefinirea?

Are loc la momentul executiei.

86. In ce moment are loc efectiv supraincarcarea?

Are loc la momentul compilarii.

87. Care e vizibilitatea implicita in cadrul unei structuri?

Publica.

88. Cum se poate modela o relatie de tip 1:M intre doua clase?
89. Cum se poate face conversia baza-derivat?
90. Cum rezolvam problema mostenirii in romb (deadly diamond of death)?
91. In ce ordine se apeleaza constructorii in cazul unei derivari multiple?
92. Cum se poate rezolva problema nefunctionarii unei clase sablon pe un anumit tip de data?
93. Cum poate deveni un iterator invalid?
94. Cum se defineste un pointer prin care se poate gestiona orice tip de zona de memorie ?
95. Cum se poate accesa zona privata a unui obiect in C++ ?
96. Ce reprezinta termenul de late-binding ?
97. Ce reprezinta termenul de early-binding ?
98. Care este utilitatea functiilor virtuale ?
99. La ce pot fi folosite clasele abstracte ?

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

101. Care este utilitatea atributelor constante ?
102. Cum se defineste o metoda care sa nu primeasca pointerul *this* ?