



## Examen poo tott

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



Scan to open on Studocu

Examen 23 iunie 2021, ora 7:30

online.ase.ro/mod/quiz/attempt.php?attempt=635659

Aplicații Gmail YouTube Maps SeriTimp – Google... Proiect Serii de tim... Programe româneș... SAS OnDemand for... Proiect echipa 7 - SIC

Lista de lectură

BL@ASE

ROMÂNĂ (RO)

ROMEOCONSTANTIN MORUZ

1 Întrebare

Nu a primit răspuns încă

Marcat din 1,00

Întrebare cu flag

(3p) Se consideră o aplicație pentru gestionarea activității unui **broker de asigurări**. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări atribute precum: numele/denumirea asiguratului, suma asigurată, tipul asigurării, durata asigurării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcție ()** care calculează valoarea asigurării într-o valută pentru care cursul de schimb este transmis ca parametru.

(1p) Se va defini **operatorul de conversie la double** a unei asigurări.

(2p) **Implementați metodele** necesare pentru scriere/citirea de obiecte în/din **fișiere binare**.

(2p) Exemplificați conceptul "**has-a**" folosind clasa definită și o nouă clasă. Clasa include o metodă care calculează valoarea medie a unei asigurări.


(1p) Calculați valoarea totală a asigurărilor definite în funcția main() cu ajutorul unui **container STL**.

1

Încheiați încercarea ...

Timp rămas 0:12:35

This document is available on



Tastați aici pentru a căuta

Downloaded by Oranceanu Marius (oranceanu@gmail.com)

23°C Sunny 09:13 23.06.2021

(3p) Se consideră o aplicație pentru gestionarea activității unei companii care pune la dispoziție servicii de streaming video (filme, seriale). Se vor urmări atribute specifice, precum: preț abonament, durată, colecție de filme, colecție de seriale, statistici vizualizări (minute) etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

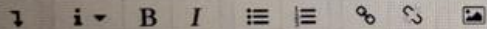
(1,5p) Se vor defini **operatorii += și -=** pentru adăugarea/eliminarea unei producții din colecție.

(1p) Definiți **operatorul de conversie la double**, care va returna cea mai vizionată producție.

(2p) **Specializați o clasă** existentă pentru a gestiona diferite tipuri de producții de streaming (filme, seriale, documentare etc.).

(1,5p) Implementați **o metodă** care determină primele cinci filme și primele cinci seriale, cele mai vizionate.

(1p) Exemplificați conceptul de **metodă/clasă template** în C++.



(3p) Se consideră o aplicație pentru gestionarea activității unui **broker de asigurări**. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări attribute precum: numele/denumirea asiguratului, suma asigurată, tipul asigurării, durata asigurării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcție ()** care calculează valoarea asigurării într-o valută pentru care cursul de schimb este transmis ca parametru.

(1p) Se va defini **operatorul de conversie la double** a unei asigurări.

(2p) Implementați metodele necesare pentru scriere/citirea de obiecte în/din **fișiere binare**.

(2p) Exemplificați conceptul "**has-a**" folosind clasa **denumita** și o nouă clasă. Clasa include metode care calculează valoarea medie a unei asigurări.

(1p) Calculați valoarea totală a asigurărilor definite în funcția **main()** cu ajutorul unui **container STL**.

**Puncte** 0,00/1,00  
**Notează** 0,00 din maxim 10,00 (0%) posibil

trebare  
primit  
uns  
cat din 1,00  
întrebare cu

**(3p)** Se consideră o aplicație pentru gestionarea activității unei firme care oferă servicii de securitate și supraveghere video. Definiți o clasă care modelează un sistem de supraveghere. Se vor urmări atribute specifice, precum: numărul de camere de supraveghere, evenimentele generate de fiecare cameră (număr și tip), datele specifice unei camere, durata de timp scursă de la ultima pornire, starea camerei etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

**(1p)** Se va defini **operatorul []**, astfel încât să se poată accesa o anumită cameră de supraveghere după un identificator.

**(1p)** Oferiți posibilitatea ca prin **operator+=** să fie adăugat o nouă cameră de supraveghere.

**(2p)** Exemplificați conceptul de relație de tip „is a” prin specializarea clasei/uneia dintre clasele definite.

**(2p)** Implementați **două metode**: o metodă care identifică și returnează camerele care îndeplinesc o anumită condiție, iar a doua metodă care calculează amplitudinea pe baza numărului de evenimente detectate de fiecare cameră.

**(1p)** Propuneți parametrizarea unui tip din cadrul clasei definite, astfel încât aceasta să poată fi utilizată și cu alte tipuri de date (**clasă template**).



# Programare Orientată Obiect, Tip-C, Sem-1, Zi (2020-2021)

Pagina principală / Cursurile mele / Programare-C,Sem1(2634fy) / General / Examen 26 ianuarie 2020

1 întrebare

Nu a primit  
răspuns încă

Marcat din 1,00

🚩 Întrebare cu  
flag

(3p) Se consideră o aplicație pentru gestionarea activității unei magazin care vinde **baterii externe de tipul powerBank**. Se vor urmări atribute specifice, precum: capacitate, greutate, numar intrari/iesiri, culoare etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic**, **constructori**, **metodele specifice claselor cu membri alocați dinamic** și **operatorul de afișare**. Folosiți un membru **static** sau **const**.

(1p) Se vor defini **operator+=** și **operator-=** care permit încărcarea powerbank-ului respectiv descărcarea acestuia.

(1p) Definiți **operatorul ==** care compară două obiecte de tip PowerBank și returnează true dacă toate valorile atributelor sunt egale între ele.

(2p) Exemplificați conceptul de relație de tip „is a” prin specializarea clasei PowerBank. Testați soluția prin instanțierea noli clase.

(2p) Explicați conceptele de **early binding** și **late binding**.

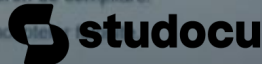
(1p) Exemplificați conceptul de funcție template în C++.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare.

Implementarea soluției trebuie să fie corectă și să nu conțină erori de compilare.

This document is available on



Downloaded by Oranceanu Marius (oranceanu@gmail.com)

NAVIGARE ÎN TEST



Înceiați încercarea ...

Timp rămas 0:09:06

Activate Windows

Go to Settings to activate Windows

Micro-cantitativă - S9 Assignment-1042.pdf

## Status răspuns

Grup

Status răspuns

Grup implicit

Draft (netrimis)

Utilizatorii care trebuie să trimită un răspuns: ANGHEL GEORGE, BAJENARU ANDREEA LOREDANA, CARAGEA CATALIN MIHAI, CRISTINA LAVINIA ELENA, DAVID MIHAEL MARIA, DRAGOMIR GABRIEL, DUMITRESCU RAZVAN, GRIGOROI A GEORGIANA, MIHAI CARMEN ALEXANDRA, OBOGEANU THEODORA ELENA, PADURARU TEODORA, PANDURU OANACOSMINA, PAUN CARLA ELENA, PETRE ANAMARIA, PETRE CHRISTIANA ALEXANDRA, PETRE TEODORA ROBERTA, PETRI A LIVIA MARIA, PICU VICTOR STEFAN, PIRVU ADINAGABRIELA, PIRVU RARES, POENARU MARINA, POPA DIANA, POPOVICI ALEXANDRU, POTAMIANOS CRINAGABRIELA, PRAJESCU TABITA ANA, PREDETEANU ALINA MIHAI, PRIAN MALINAGABRIELA, RADA ANDRADALORENA, RADU CATALINAGABRIELA, RADU RALUCACRISTINA, RADULESCU CEZAR, RADUT ELENA ADRIANA, ROBU ILINCA, ROTARU ANDREEA, SACUIU ADINA ELENA, SARU ANDREEA RUXANDRA, SAVIN TEODORA, SCUTELNICU VLAD, SIMINICIANU ANDREEA, SPASENIE VIORICA, STAN GABRIEL VLAD, TOPANA FLORINA PAULINA, VILCU ELENABIANCA

Fără notă

marți, 15 decembrie 2020, 09:00

Sarcina a fost depășită de: 41 zile 16 ore

marți, 15 decembrie 2020, 08:51

 PetreChristiana\_TemaSeminar.pdf Comentarii (0)

DELL

(3p) Se consideră o aplicație pentru gestionarea activității unei firme care oferă **servicii de imprimare 3D**, folosind diferite materiale (lemn, plastic, silicon etc.), pe baza modelelor primite de la clienți. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări atribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

(1,5p) Se va defini **operatorul <** care permite compararea a două obiecte și va fi utilizat în cadrul unei funcții de sortare.

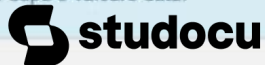
(1p) Prin intermediul **operatorului +=** se va combina obiectul curent cu un obiect primit ca parametru.

(2p) Exemplificați conceptul de relație de tip „**is a**” prin specializarea clasei/unela dintre clasele definite.

(1,5p) Implementați câte o **metodă** pentru **scrierea/citirea** unui obiect într-un **fișier binar**.

(1p) Propuneți un **container STL** ce permite regăsirea cu ușurință a unor obiecte după o valoare dată.

This document is available on



Downloaded by Oranceanu Marius (oranceanu@gmail.com)



**(3p)** Se consideră o aplicație pentru gestionarea petițiilor primite în cadrul unei primării. Definiți o clasă care modelează o astfel de solicitare. Se vor urmări attribute specifice, precum: data depunerii, numele și prenumele solicitantului, categoria în care se încadrează petiția, descrierea solicitării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

**(1p)** Se va defini **operatorul !** pentru indica dacă o petiție nu are răspuns și a depășit termenul legal (30 de zile).

**(1p)** Se va defini **operatorul ~** pentru schimba starea unei petiții (deschisă -> rezolvată).

**(2p)** Definiți **operatorii << și >>** pentru scrierea/citirea în/din **fișiere text**.

**(2p)** **Specializați** clasa definită și exemplificați **conceptul de virtualizare**.

**(1p)** Propuneți un **container STL** care permite gruparea petițiilor după categorii și regăsirea cu ușurință a acestora.

(3p) Se consideră o aplicație pentru gestionarea petițiilor primite în cadrul unei primării. Definiți o clasă care modelează o astfel de solicitare. Se vor urmări atribute specifice, precum: data depunerii, numele și prenumele solicitantului, categoria în care se încadrează petiția, descrierea solicitării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul !** pentru indica dacă o petiție nu are răspuns și a depășit termenul legal (30 de zile).

(1p) Se va defini **operatorul ~** pentru schimba starea unei petiții (deschisă -> rezolvată).

(2p) Definiți **operatorii << și >>** pentru scrierea/citirea în/din **fișiere text**.

(2p) **Specializați** clasa definită și exemplificați **conceptul de virtualizare**.

(1p) Propuneți un **container STL** care permite gruparea petițiilor după categorii și regăsirea cu ușurință a acestora.

This document is available on



Downloaded by Oranceanu Marius (oranceanu@gmail.com)

1 întrebare

Nu a primit  
răspuns încă

Marcat din 1,00

Întrebare cu  
flag

(3p) Se consideră o aplicație pentru gestionarea activității unui **broker de asigurări**. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări atribute precum: numele/denumirea asiguratului, suma asigurată, tipul asigurării, durata asigurării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcție ()** care calculează valoarea asigurării într-o valută pentru care cursul de schimb este transmis ca parametru.

(1p) Se va defini **operatorul de conversie la double** a unei asigurări.

(2p) **Implementați metodele** necesare pentru scriere/citirea de obiecte în/din **fișiere binare**.

(2p) Exemplificați conceptul "**has-a**" folosind clasa definită și o nouă clasă. Clasa include o metodă care calculează valoarea medie a unei asigurări.

(1p) Calculați valoarea totală a asigurărilor definite în funcția main() cu ajutorul unui **container STL**.

Rich text editor toolbar with icons for bold, italic, list, link, unlink, and image.

### NAVIGARE ÎN TEST

1

[Înceiați încercarea ...](#)

Timp rămas **0:12:35**

(3p) Se consideră o aplicație pentru gestionarea activității unui magazin care vinde și închiriază jocuri video. Se vor defini atribute specifice, precum: titlu joc, consola compatibilă, vârsta minimă recomandată, tipul tranzacției (cumpărare/închiriere), starea jocului (închiriat/disponibil) prețuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocati dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

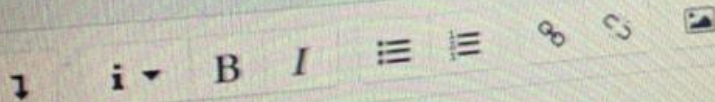
(1.5p) Se va defini **operatorul funcție ()** pentru căutarea jocurilor care îndeplinesc un anumit criteriu.

(1.5p) Supraincărcați **operatorii += și -=** pentru închirierea/returnarea de jocuri, respectiv **operatorul []** pentru referirea unui joc pe baza poziției.

(1p) Scrieți o **metodă** care determină valoarea totală a încasărilor din închirieri într-o anumită perioadă.

(2p) **Specializați o clasă** definită și exemplificați **conceptul de virtualizare**.

(1p) Propuneți un **container STL** pentru identificarea rapidă a jocurilor după vârsta minimă recomandată.





(3p) Se consideră o aplicație pentru gestionarea activității unei firme de catering. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări atribute precum: numele/denumirea clientului, data și durata evenimentului, produse și cantități comandate, prețuri asociate etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

(1p) Se va defini operatorul `[]` pentru a accesa un produs după poziția dată.

(1p) Definiți operatorul `+=` pentru adăugarea unui produs la o comandă. Dacă produsul există deja în comandă, se va incrementa doar cantitatea.

(2p) Scrieți într-un fișier text comenzile care îndeplinesc un criteriu dat, primit ca parametru.

(2p) Exemplificați conceptul de virtualizare prin utilizarea unei clase abstracte.

(1p) Propuneți un container STL carea permite efectuarea rapidă a operațiilor de inserare/ștergere de produse într-o/dintr-o comandă.





(3p) Se consideră o aplicație pentru gestionarea activității unui magazin care vinde și închiriază **jocuri video**. Se vor urmări atribute specifice, precum: titlu joc, consola compatibilă, vârsta minimă recomandată, tipul tranzacției (vânzare/cumpărare), starea jocului (închiriat/disponibil) prețuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

(1.5p) Se va defini **operatorul funcție ()** pentru căutarea jocurilor care îndeplinesc un anumit criteriu.

(1.5p) Supraîncărcați **operatorii += și -=** pentru închirierea/returnarea de jocuri, respectiv **operatorul []**, pentru referirea unui joc pe baza poziției.

(1p) Scrieți o **metodă** care determină valoarea totală a încasărilor din închirieri într-o anumită perioadă.

(2p) **Specializați o clasă** definită și exemplificați **conceptul de virtualizare**.

(1p) Propuneți un **container STL** pentru identificarea rapidă a jocurilor după vârsta minimă recomandată.



**(3p)** Se consideră o aplicație pentru gestionarea activității unui **furnizor de energie electrică**. Se vor urmări atribute specifice, precum: nume/denumire client, consum lunar efectiv, consum lunar estimat, număr contract, durată contract, preț kWh etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

**(1p)** Se va defini **operatorul funcție ()** pentru a modifica valoarea consumului lunar efectiv dintr-o anumită lună din contract.

**(1p)** Definiți **operatorul de conversie la double**, care va returna valoarea totală a diferențelor lunare de consum.

**(2p)** Scrieți **două metode**: o metodă pentru determinarea lunii cu cea mai mică diferență de consum și o metodă care calculează valoarea totală a consumului efectiv.

**(2p)** Exemplificați **conceptul de virtualizare** prin utilizarea unei clase abstracte.

**(1p)** Propuneți un **container STL** carea permite efectuarea rapidă a operațiilor de regăsire după numărul contractului.

**Puncte** 0,00/1,00

**Notează** 0,00 din maxim 10,00 (0%) posibil

Finalizare

1 întrebare

Nu a primit  
răspuns

Marcat din 1,00

🚩 Întrebare cu  
flag

(3p) Se consideră o aplicație pentru gestionarea activității unui **furnizor de energie electrică**. Se vor urmări atribute specifice, precum: nume/denumire client, consum lunar efectiv, consum lunar estimat, număr contract, durată contract, preț kWh etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcției ()** pentru a modifica valoarea consumului lunar efectiv dintr-o anumită lună din contract.

(1p) Definiți **operatorul de conversie la double**, care va returna valoarea totală a diferențelor lunare de consum.

(2p) Scrieți **două metode**: o metodă pentru determinarea lunii cu cea mai mică diferență de consum și o metodă care calculează valoarea totală a consumului efectiv.

(2p) Exemplificați **conceptul de virtualizare** prin utilizarea unei clase abstracte.

(1p) Propuneți un **container STL** carea permite efectuarea rapidă a operațiilor de regăsire după numărul contractului.

This document is available on



Downloaded by Oranceanu Marius (oranceanu@gmail.com)



# Programare Orientată Obiect, Tip-C, Sem-1, Zi (2020-2021)

[Pagina principală](#) / [Cursurile mele](#) / [Programare-C,Sem1\(2634fy\)](#) / [General](#) / [Examen 26 ianuarie 2021](#)

## 1 Întrebare

Nu a primit  
răspuns încă

Marcat din 1,00

🚩 Întrebare cu  
flag

(3p) Se consideră o aplicație pentru gestionarea activității unui **broker de asigurări**. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări atribute precum: numele/denumirea asiguratului, suma asigurată, tipul asigurării, durata asigurării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocati dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcție ()** care calculează valoarea asigurării într-o valută pentru care cursul de schimb este transmis ca parametru.

(1p) Se va defini **operatorul de conversie la double** a unei asigurări.

(2p) **Implementați metodele** necesare pentru scriere/citirea de obiecte în/din **fișiere binare**.

(2p) Exemplificați conceptul "**has-a**" folosind clasa definită și o nouă clasă. Clasa include o metodă care calculează valoarea medie a unei asigurări.

(1p) Calculați valoarea totală a asigurărilor definite în funcția main() cu ajutorul unui **container STL**.

## NAVIGARE ÎN TEST



[Încheiați încercarea ...](#)

⏱ Timp rămas 0:04:47



Examen 26 ianuarie 2021

online.ase.ro/mod/quiz/attempt.php?attempt=369717

HL@ASE

IRINAANDREEA PIRLITU

1 întrebare  
Nu a primit răspuns încă  
Marcat din 1,00  
Întrebare cu flag

(2p) Se consideră aplicația pentru **gestiunea documentelor** dintr-un dosar de angajare, folosind aspecte comune precum CV-ul, copie CI, copie diplomă, etc. Definiți o clasă care modelează un aspect propriu acestei activități. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic**. Folosiți un membru static sau const.

(1p) Supraincărcați **operator<<** pentru afișarea informațiilor dintr-un dosar de angajare.

(2p) Să se genereze un raport într-un fișier text cu informațiile dintr-un dosar de angajare.

(2p) Să se exemplifice utilizarea mecanismul try-catch pentru a gestiona modificarea valorii unui atribut pentru care a fost implementată metoda setter. Metoda setter trebuie să genereze o excepție specifică aplicației (definită de programator) pentru date invalide. Utilizatorul este avertizat dacă introduce date invalide de la consolă având posibilitatea să le reintroducă.

(1p) Definiți și testați o metodă statică pentru afișarea numărului total de dosare create la un moment dat. Metoda va contoriza dosarele create chiar dacă nu au fost depuse.

(2p) Să se exemplifice conceptul de relație de "has a" între clase prin crearea unei clase pentru reprezentarea unei persoane care depune un dosar de angajare.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare. Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.

This document is available on

File Edit View Project Build Debug Test Analyze Tools

Extensions Window Help

Debug x86

Live Share

examen.cpp

(Global Scope)

```
47 }
48 void setCNP(string cnp)
49 {
50     CNP = cnp;
51 }
52 string getCNP()
53 {
54     return CNP;
55 }
56 void afisare()
57 {
58     cout << CNP << " " << serieCI;
59 }
60 };
61
62 class Dosar
63 {
64 private:
65     Document* docs;
66
67 };
68 int main()
69 {
70
71
72     return 0;
73 }
74
```

96 % No issues found

Err 74 Chk 1 SPC CRLF

Error List

Entire Solution 0 Errors 0 Warnings 0 Messages Build + IntelliSense

Search Error List

Code	Description	Project	File	Line
------	-------------	---------	------	------







nu a primit raspuns inca

Marcat din 1,00

🚩 Întrebare cu flag

**(3p)** Se consideră o aplicație pentru gestionarea petițiilor primite în cadrul unei primării. Definiți o clasă care modelează o astfel de solicitare. Se vor urmări attribute specifice, precum: data depunerii, numele și prenumele solicitantului, categoria în care se încadrează petiția, descrierea solicitării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

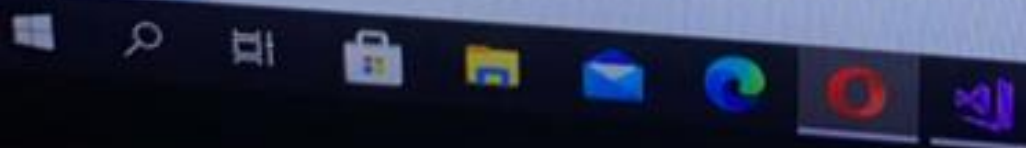
**(1p)** Se va defini **operatorul !** pentru indica dacă o petiție nu are răspuns și a depășit termenul legal (30 de zile).

**(1p)** Se va defini **operatorul ~** pentru schimba starea unei petiții (deschisă → rezolvată).

**(2p)** Definiți **operatorii << și >>** pentru scrierea/citirea în/din fișiere text.

**(2p)** Specializați clasa definită și exemplificați **conceptul de virtualizare**.

**(1p)** Propuneți un **container STL** care permite gruparea petițiilor după categorii și regăsirea cu ușurință a acestora.





Derivate Partur Omic din clasa Asociatie, deoarece cotizatiile sunt aceleasi pentru toti membrii.

(1p) Să se implementeze o modalitate de afişare a obiectelor de bază şi derivate.

(2p) Pentru clasa ce modelează asociaţia de persoane, să se supraîncarce:

- operator **+=** pentru primirea unui nou membru;

- operator **-=** pentru excluderea unui membru din asociaţie.

(2p) Precizaţi şi testaţi conversiile implicite care sunt acceptate între diferitele obiecte, între **pointeri** şi între **referinţe** de obiecte. Converteţi prin **operator cast** un simpatizant în membru plin.

(1p) Declaraţi în clasa de bază metoda virtuală **getCotizaţie()**, care în clasa de bază returnează 0, iar în clasele derivate returnează valori după cum urmează:

- pentru un membru plin, 100 lei,

- pentru un membru simpatizant 50 lei.

(1p) Explicaţi conceptul de *late binding* şi apelând implementările metodei **getCotizaţie()** şi calculaţi valoarea totală adunată la nivel de partid, din cotizaţii.

(1p) Folosiţi o clasa STL pentru a da o altă implementare a asociaţiei şi parcurgând structura aleasă, verificaţi dacă funcţionează şi în acest caz conceptul de *late binding*.



### Biblioteca împrumuturi

Să se scrie o aplicație orientată obiect pentru gestionarea împrumuturilor de cărți dintr-o bibliotecă. Se va avea în vedere că o carte poate exista în mai multe exemplare, fiecare exemplar fiind identificat printr-un cod unic. Un cititor se identifică prin CNP și nume și poate împrumuta câte un exemplar dintr-un număr de cărți diferite.

(2p) Pentru domeniul dat, să se definească clase cu membri de tip *public*, *private*, *protected*, *const*, *static*.

(1p) Elaborați constructor cu parametri, default constructor, constructor de copiere, destructor și operatorii pentru așezare.

(2p) Supraincarcați în clasa *Cititor* operatorii *+* și *-* pentru a împrumuta, respectiv returna un exemplar dintr-o carte.

(1p) Implementați operatorii *++* pentru a ține dacă două exemplare se referă la aceeași carte. Folosiți operatorul de comparație la căutarea unei cărți pentru a o returna.

(1p) Implementați două funcții de tip accesori (*get* și *set*) și două metode private clasei (0,25 puncte/funcție/operator).

(2p) Furnizați funcții sau operatori pentru salvarea și restaurarea cărților în și din fișiere binare, permanente.

(1p) Transformați una din clase într-o clasă template sau inițializați o clasă template STL, pentru domeniul dat.

**(3p)** Se consideră o aplicație care permite gestiunea unei scări de bloc. Se vor urmări aspectele comune privind denumire, adresa, apartamente, locatari etc.. Definiți o clasă care modelează un aspect propriu acestei activități. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic, constructori, metodele specifice** claselor cu **membri alocați dinamic și operatorul de afișare**. Folosiți un membru **static** sau **const**.

**(1p)** supraîncărcați **operator[]** pentru accesarea elementelor din vectorul alocat dinamic al clasei. Operatorul poate fi folosit și pentru citire și pentru modificare

**(1p)** Creați o nouă clasă Bloc, care să conțină mai multe obiecte de tipul Scară. Elementele pot fi gestionate și prin pointeri. Colecția de elemente de tip Scară este gestionată printr-un vector alocat dinamic

**(2p)** Oferiți posibilitatea ca prin **operator+=** să fie adăugată o nouă scară în cadrul unui obiect de tipul Bloc.

**(2p)** În cadrul clasei Bloc implementați două metode: una returnează scara cea mai productivă, iar cea de a doua metodă returnează scara cea mai puțin productivă din cadrul blocului.

**(1p)** Clasa Bloc trebuie să conțină o metodă care să ofere posibilitatea de calculare a unei sume pentru toate obiectele gestionate.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare.

Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.

(3p) Se consideră o aplicație pentru gestionarea activității unei firme care oferă **servicii de imprimare 3D**, folosind diferite materiale (lemn, plastic, silicon etc.), pe baza modelelor primite de la clienți. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări atribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

(1,5p) Se va defini **operatorul <** care permite compararea a două obiecte și va fi utilizat în cadrul unei funcții de sortare.

(1p) Prin intermediul **operatorului +=** se va combina obiectul curent cu un obiect primit ca parametru.

(2p) Exemplificați conceptul de relație de tip „**is a**” prin specializarea clasei/unela dintre clasele definite.

(1,5p) Implementați câte o **metodă** pentru **scrierea/citirea** unui obiect într-un **fișier binar**.

(1p) Propuneți un **container STL** ce permite regăsirea cu ușurință a unor obiecte după o valoare dată.





(3p) Se consideră o aplicație pentru gestionarea activității unei companii care pune la dispoziție servicii de streaming video (filme, seriale). Se vor urmări atribute specifice, precum: preț abonament, durată, colecție de filme, colecție de seriale, statistici vizualizări (minute) etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

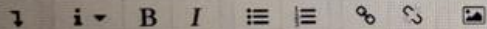
(1,5p) Se vor defini **operatorii += și -=** pentru adăugarea/eliminarea unei producții din colecție.

(1p) Definiți **operatorul de conversie la double**, care va returna cea mai vizionată producție.

(2p) **Specializați o clasă** existentă pentru a gestiona diferite tipuri de producții de streaming (filme, seriale, documentare etc.).

(1,5p) Implementați **o metodă** care determină primele cinci filme și primele cinci seriale, cele mai vizionate.

(1p) Exemplificați conceptul de **metodă/clasă template** în C++.



Examen POO 23 Iunie 2021 - Res: +

online.ase.ro/mod/quiz/attempt.php?attempt=636209

BL@ASE

ROMÂNĂ (RO)

ELENABIANCA CODREANU

1 Întrebare

Nu a primit răspuns încă

Marcat din 1,00

Întrebare cu flag

(3p) Se consideră o aplicație care permite gestiunea preparatelor culinare dintr-un restaurant. Se vor urmări aspectele comune privind denumire, culoare, numar calorii, etc.. Definiți o clasă care modelează un aspect propriu acestei activități. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) supraîncărcați operator++ pentru incrementarea unui atribut al obiectelor. Clasa permite atât utilizarea de preincrementare cât și postincrementare.

(1p) Creați o enumerare prin care să se specifice dacă felul de mâncare este vegan, vegetarian, raw-vegan, etc. Modificați clasa anterioară, astfel încât să conțină un atribut de tipul enumerării create.

(2p) Să se exemplifice conceptul de relație "has a" între clase pentru a gestiona elemente de tipul clasei definită anterior (pot fi folosiți și pointeri). Noua clasă poate fi un meniu care să conțină mai multe feluri de mâncare sau o masă. Această clasă nouă este un container care va gestiona colecția de obiecte printr-un vector alocat dinamic.

(2p) Oferiți posibilitatea ca prin operator+= să fie adăugat un nou obiect în colecția de obiecte gestionate de clasa (container) adăugată.

(1p) Clasa container trebuie să conțină o metodă care să ofere posibilitatea de calculare a unei sume pentru toate obiectele gestionate.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare.

Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.

1

i B I

NAVIGARE ÎN TEST

1

[Încheiați încercarea ...](#)

Temp rămas 0:12:28

Type here to search

Downloaded by Oana Marius Ioranceanu@gmail.com

85°F Mostly sunny

ENG US

12:46 PM 6/23/2021

**1 Întrebare**  
Nu a primit răspuns încă.  
Marcat din 1,00  
Întrebare cu flag

(3p) Se consideră o aplicație pentru gestiunea contactelor de pe un telefon. Se vor urmări aspectele comune privind numele, numărul de telefon, apeluri, mesaje, etc. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări atribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Toate contactele pot fi comparate cu operator<, astfel, un contact se consideră mai mic decât alt contact în funcție de numele acestora.

(2p) Specializați clasa care descrie pentru un contact și informații cu privire la locul în care acesta lucrează.

(1p) Oferiți posibilitatea ca prin operator+= să fie adăugat un nou apel sau un nou mesaj în relația cu un contact anume.

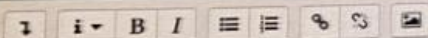
(2p) Exemplificați conceptul de virtualizare prin oferirea unei funcționalități diferite pentru Contact și ContactAngajat.

(1p) Propuneți o metodă pentru a ține informațiile despre contactele dintr-un telefon, astfel încât înregistrările să se facă în mod unic – să nu existe două contacte cu același nume.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare.

Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.





(3p) Se consideră o aplicație pentru gestiunea contactelor de pe un telefon. Se vor urmări aspectele comune privind numele, numărul de telefon, apeluri, mesaje, etc. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări atribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare**. Folosiți un membru static sau const.

(1p) Toate contactele pot fi comparate cu **operator<**; astfel, un contact se consideră mai mic decât alt contact în funcție de numele acestora.

(2p) **Specializați clasa** care descrie pentru un contact și informații cu privire la locul în care acesta lucrează.

(1p) Oferiți posibilitatea ca prin **operator+=** să fie adăugat un nou apel sau un nou mesaj în relația cu un contact anume.

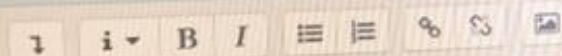
(2p) Exemplificați conceptul de virtualizare prin oferirea unei funcționalități diferite pentru **Contact** și **ContactAngajat**.

(1p) Propuneți o metodă pentru a ține informațiile despre contactele dintr-un telefon, astfel încât înregistrările să se facă în mod unic – să nu existe două contacte cu același nume.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acestora va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare.

Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.



### PartidDerivare

(2p) Să se realizeze o aplicație orientată obiect pentru modelarea activităților unei **asociații de persoane**, ca vector dinamic de persoane (CNP, nume).

Definiți clasa **PartidPolitic** în care persoanele au în plus cotizație.

Se va folosi derivarea *private*, în condițiile în care clasele dispun și de membri în domeniul de accesibilitate *private*.

Studiați implicațiile pe care le au două variante de modelare:

- derivare Membru din clasa Persoana, partidul fiind asociație de membri;
- derivare **PartidPolitic** din clasa Asociație, deoarece cotizațiile sunt aceleași pentru toti membrii.

(1p) Să se implementeze o modalitate de afișare a obiectelor de bază și derivate.

(2p) Pentru clasa ce modelează asociația de persoane, să se supraîncarce:

**operator +=** pentru primirea unui nou membru;

**operator -=** pentru excluderea unui membru din asociație.

(2p) Precizați și testați conversiile implicite care sunt acceptate între diferitele **obiecte**, între **pointeri** și între **referințe** de obiecte. Converteți prin **operator cast** un simpatizant în membru plin.

(1p) Declarați în clasa de bază metoda virtuală **getCotizație()**, care în clasa de bază returnează 0, iar în clasele derivate returnează valori după cum urmează:

pentru un membru plin, 100 lei.



(3p) Se consideră o aplicație pentru gestionarea activității unei firme care oferă **servicii de imprimare 3D**, folosind diferite materiale (lemn, plastic, silicon etc.), pe baza modelelor primite de la clienți. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări atribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

(1,5p) Se va defini **operatorul <** care permite compararea a două obiecte și va fi utilizat în cadrul unei funcții de sortare.

(1p) Prin intermediul **operatorului +=** se va combina obiectul curent cu un obiect primit ca parametru.

(2p) Exemplificați conceptul de relație de tip „**is a**” prin specializarea clasei/unela dintre clasele definite.

(1,5p) Implementați câte o **metodă** pentru **scrierea/citirea** unui obiect într-un **fișier binar**.

(1p) Propuneți un **container STL** ce permite regăsirea cu ușurință a unor obiecte după o valoare dată.





### CosCumparaturi

Să se realizeze o aplicație orientată obiect pentru a gestiona **coșul de cumparaturi** al unui client.

(3p) Se vor defini două clase:

- una elementară, care conține datele aferente unui item (**produs**) și implementează metode specifice;
- o clasă pentru a modela **mulțimea de produse** din coșul de cumpărături.

Clasa de bază va conține membri de tip **public**, **privat**, **protected**, **const**, **static** și operator de afișare. Cel puțin un membru **private** are asociate funcții de acces **get-set**.

Clasele dispun de constructor cu parametri, default constructor, constructor de copiere, destructor și **operator<<** pentru afișare.

(2p) **operator[]**, pentru localizarea unui produs din coș, un operator pentru compararea între obiecte și încă doi operatori adecvați domeniului (aritmetici sau cast sau indexare sau de alt tip) (0.5 puncte/funcție/operator);

(1p) funcții sau operatori pentru salvarea și restaurarea obiectelor în/din fișiere **binare**, permanente.

(2p) **operator +=** și **operator -=** vor asigura adăugarea / scăderea unui obiect din coș, folosind pentru identificare **operator[]**.

(2p) Transformați una din clase într-o clasă template sau instanțați o clasă template STL, pentru clasa ce ține mulțimea produselor. Alegeți o modalitate de parcurgere a structurii template aleasă și calculați valoarea totală a coșului de cumpărături.

1 i B I ≡ ≡ ∞ ∞

I

## Bibliotecalmpurmuturi

Să se scrie o aplicație orientată obiect pentru gestiunea **împrumuturilor de cărți dintr-o bibliotecă**. Se va avea în vedere că o carte poate exista în mai multe exemplare, fiecare exemplar fiind identificat printr-un cod unic. Un **cititor** se identifică prin CNP și nume și poate împrumuta câte un exemplar din mai multe cărți diferite.

(2p) Pentru domeniul dat, să se definească clase cu membri de tip **public, private, protected, const, static**

(1p) Elaborați constructor cu parametri, default constructor, constructor de copiere, destructor și **operator<<** pentru afișare.

(2p) Supralăncărați în clasa **Cititor** **operator+=** și **operator-=** pentru a împrumuta, respectiv returna un exemplar dintr-o carte.

(1p) Implementați **operator==** pentru a testa dacă **două exemplare** se referă la aceeași carte; folosiți operatorul de comparare la căutarea unei cărți pentru a o returna.

(1p) Implementați două funcții de tip accesori (get și set) și două metode proprii clasei (0.25 puncte/funcție/operator).

(2p) Furnizați funcții sau operatori pentru salvarea și restaurarea cărților în/din fișiere **binare**, permanente.

(1p) Transformați una din clase într-o clasă template sau instanțiați o clasă template STL, pentru domeniul dat.

NAVIGARE ÎN



Închideți Încercați



## TransportPublic

(2p) Să se realizeze o aplicație orientată obiect pentru modelarea activităților unui parc de mijloace de transport. Pentru gestiunea unitară se va folosi o clasă generală **Vehicul** ( id, tip, număr locuri, valoarea de inventar) din care se derivează clasele:

- **Autobuz** care are în plus capacitate cilindrică, în centimetri cubi;
- **Tramvai** care are în plus număr de vagoane;
- **Troleibuz** care are în plus puterea, exprimată în kilowați.

Se vor folosi toate cele trei tipuri de derivare: *public*, *private*, *protected*, iar clasele dispun de membri în toate domeniile de accesibilitate.

Precizați și testați conversiile implicite care sunt acceptate între diferitele obiecte, între pointeri și între referințe de obiecte.

(1p) Să se implementeze o modalitate de afișare a obiectelor de bază și derivate .

(2p) Considerând parcul de mijloace de transport ca un **vector de pointeri** la clasa de bază, să se supraîncarce:

- **operator +=** pentru a adăuga un nou mijloc de transport;
- **operator -=** pentru a surprinde scoaterea din folosință a unui mijloc de transport, identificat prin id .

(2p) Declarați în clasa de bază metoda virtual pură **getAmortizare()**, care în clasele derivate returnează valori după cum urmează:

- pentru un autobuz, 10% din valoarea de inventar,
- pentru un tramvai, 12% din valoarea de inventar,
- pentru un troleibuz, 8% din valoarea de inventar .

(1p) Explicați conceptul de *late binding* și apelând implementările metodei **getAmortizare()** calculați valoarea totală a amortizării anuale la nivelul întregului parc de mijloace de transport.

(2p) Folosiți clasa **STL list** pentru a da o altă implementare a parcului de mijloace de transport, folosind **STL** și în acest caz conceptul de *late binding*.





## Programare Orientata Obiect, Tip-C, Sem1, 21 (2020-2021)

Pagina principală / Cursuri / 2020 - 2021 / Licență / Programare-C,Sem1(341et) / General / Examen 25 ian 2021

## 1 Întrebare

Nu a primit  
răspuns încă

Marcat din 1,00

🚩 Întrebare cu  
flag

(2p) Se considera o aplicație pentru gestiunea unei colecții de markere folosite pentru scrierea pe o tablă dintr-o sală. Se vor urmări aspectele comune privind culoare, dimensiune, producător, etc. Pentru datele membre private sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic**, **constructori**, **metodele specifice** claselor cu **membri alocați dinamic**. Folosiți un membru **static** sau **const**.

(1p) Supraincărcați **operator<<** pentru afișarea unui marker.

(2p) **Specializați** clasa care descrie un **marker electronic** având noi câmpuri precum baterie, rază acțiune, etc

(1p) Oferiți posibilitatea de comparare a două markere prin **operator==**, compararea realizându-se pentru minim două atribute.

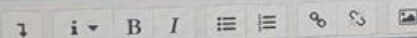
(2p) Exemplificați conceptul de virtualizare prin oferirea unei funcționalități diferite pentru **Marker** și **MarkerElectronic**.

(2p) Propuneți o metodă pentru a ține informațiile despre marker-ele dintr-o sală. Dorindu-se ca pentru fiecare marker să fie reținut și proprietarul acestuia. Un marker poate să aibă un singur proprietar, însă un proprietar poate să aibă mai multe markere. Propunerea realizată trebuie să permită identificarea proprietarului foarte ușor după marker.

## NAVIGARE ÎN TEST



Încheiați încercarea ...



Tastați aici pentru a căuta

This document is available on



studocu

Downloaded by Oranceanu Marius (oranceanu@gmail.com)

11:25  
25.01.2021

## SerieStatistica

- (2p) Definiți clase care modelează lucrul cu o **serie statistică** sub forma unui vector dinamic de perechi **valoare-frecvență**, **sortat** după valoare.
- (2p) Furnizați metode pentru calculul unor indicatori statistici (medie, dispersie, coeficient de corelație).
- (2p) Supraîncărcați **operator+=** pentru a obține o serie agregată a două serii statistice.
- (1p) Supraîncărcați **operator+=** pentru a adăuga o nouă pereche (**valoare-frecvență**) la o serie statistică existentă, menținând caracterul ei sortat.
- (1p) Supraîncărcați **operator-=** pentru a elimina o pereche (**valoare-frecvență**) identificată prin valoare, dintr-o serie statistică existentă.
- (2p) Transformați una din clase într-o clasă template sau instanțiați o clasă template **STL** care să faciliteze lucru cu serii statistice de forma unui vector de perechi **valoare-frecvență**. Indicați cum operează metodele elaborate mai sus în contextul clasei template.

(3p) Se consideră o aplicație pentru gestionarea activității unui **broker de asigurări**. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări attribute precum: numele/denumirea asiguratului, suma asigurată, tipul asigurării, durata asigurării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcție ()** care calculează valoarea asigurării într-o valută pentru care cursul de schimb este transmis ca parametru.

(1p) Se va defini **operatorul de conversie la double** a unei asigurări.

(2p) Implementați metodele necesare pentru scriere/citirea de obiecte în/din **fișiere binare**.

(2p) Exemplificați conceptul "**has-a**" folosind clasa **denumita** și o nouă clasă. Clasa include metode care calculează valoarea medie a unei asigurări.

(1p) Calculați valoarea totală a asigurărilor definite în funcția **main()** cu ajutorul unui **container STL**.

**Puncte** 0,00/1,00  
**Notează** 0,00 din maxim 10,00 (0%) posibil

trebare  
primit  
uns  
cat din 1,00  
întrebare cu

**(3p)** Se consideră o aplicație pentru gestionarea activității unei firme care oferă servicii de securitate și supraveghere video. Definiți o clasă care modelează un sistem de supraveghere. Se vor urmări atribute specifice, precum: numărul de camere de supraveghere, evenimentele generate de fiecare cameră (număr și tip), datele specifice unei camere, durata de timp scursă de la ultima pornire, starea camerei etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

**(1p)** Se va defini **operatorul []**, astfel încât să se poată accesa o anumită cameră de supraveghere după un identificator.

**(1p)** Oferiți posibilitatea ca prin **operator+=** să fie adăugat o nouă cameră de supraveghere.

**(2p)** Exemplificați conceptul de relație de tip „is a” prin specializarea clasei/uneia dintre clasele definite.

**(2p)** Implementați **două metode**: o metodă care identifică și returnează camerele care îndeplinesc o anumită condiție, iar a doua metodă care calculează amplitudinea pe baza numărului de evenimente detectate de fiecare cameră.

**(1p)** Propuneți parametrizarea unui tip din cadrul clasei definite, astfel încât aceasta să poată fi utilizată și cu alte tipuri de date (**clasă template**).



# Programare Orientată Obiect, Tip-C, Sem-1, Zi (2020-2021)

Pagina principală / Cursurile mele / Programare-C,Sem1(2634fy) / General / Examen 26 ianuarie 2020

1 întrebare

Nu a primit  
răspuns încă

Marcat din 1,00

🚩 Întrebare cu  
flag

(3p) Se consideră o aplicație pentru gestionarea activității unei magazin care vinde **baterii externe de tipul powerBank**. Se vor urmări atribute specifice, precum: capacitate, greutate, numar intrari/iesiri, culoare etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic**, **constructori**, **metodele specifice claselor cu membri alocați dinamic** și **operatorul de afișare**. Folosiți un membru **static** sau **const**.

(1p) Se vor defini **operator+=** și **operator-=** care permit încărcarea powerbank-ului respectiv descărcarea acestuia.

(1p) Definiți **operatorul ==** care compară două obiecte de tip PowerBank și returnează true dacă toate valorile atributelor sunt egale între ele.

(2p) Exemplificați conceptul de relație de tip „is a” prin specializarea clasei PowerBank. Testați soluția prin instanțierea noli clase.

(2p) Explicați conceptele de **early binding** și **late binding**.

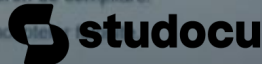
(1p) Exemplificați conceptul de funcție template în C++.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare.

Implementarea soluției trebuie să fie corectă și să respecte cerințele.

This document is available on



Downloaded by Oranceanu Marius (oranceanu@gmail.com)

NAVIGARE ÎN TEST



Înceiați încercarea ...

Timp rămas 0:09:06

Activate Windows

Go to Settings to activate Windows



(3p) Se consideră o aplicație pentru gestionarea activității unei firme care oferă **servicii de imprimare 3D**, folosind diferite materiale (lemn, plastic, silicon etc.), pe baza modelelor primite de la clienți. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări atribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

(1,5p) Se va defini **operatorul <** care permite compararea a două obiecte și va fi utilizat în cadrul unei funcții de sortare.

(1p) Prin intermediul **operatorului +=** se va combina obiectul curent cu un obiect primit ca parametru.

(2p) Exemplificați conceptul de relație de tip „**is a**” prin specializarea clasei/unela dintre clasele definite.

(1,5p) Implementați câte o **metodă** pentru **scrierea/citirea** unui obiect într-un **fișier binar**.

(1p) Propuneți un **container STL** ce permite regăsirea cu ușurință a unor obiecte după o valoare dată.



### PartidDerivare

(2p) Să se realizeze o aplicație orientată obiect pentru modelarea activităților unei **asociații de persoane**, ca vector dinamic de persoane (CNP, nume).

Definiți clasa **PartidPolitic** în care persoanele au în plus cotizație.

Se va folosi derivarea *private*, în condițiile în care clasele dispun și de membri în domeniul de accesibilitate *private*.

Studiați implicațiile pe care le au două variante de modelare:

- derivare Membru din clasa Persoana, partidul fiind asociație de membri;
- derivare **PartidPolitic** din clasa Asociație, deoarece cotizațiile sunt aceleași pentru toți membrii.

(1p) Să se implementeze o modalitate de afișare a obiectelor de bază și derivate.

(2p) Pentru clasa ce modelează asociația de persoane, să se supraîncarce:

**operator +=** pentru primirea unui nou membru;

**operator -=** pentru excluderea unui membru din asociație.

(2p) Precizați și testați conversiile implicite care sunt acceptate între diferitele **obiecte**, între **pointeri** și între **referințe** de obiecte. Converteți prin **operator cast** un simpatizant în membru plin.

(1p) Declarați în clasa de bază metoda virtuală **getCotizație()**, care în clasa de bază returnează 0, iar în clasele derivate returnează valori după cum urmează:

pentru un membru plin, 100 lei.



pentru toti membrii.

(1p) Să se implementeze o modalitate de afişare a obiectelor de bază şi derivate.

(2p) Pentru clasa ce modelează asociaţia de persoane, să se supraîncarce:

- operator **+=** pentru primirea unui nou membru;

- operator **-=** pentru excluderea unui membru din asociaţie.

(2p) Precizaţi şi testaţi conversiile implicite care sunt acceptate între diferitele obiecte, între **pointeri** şi între **referinţe** de obiecte. Converteţi prin **operator cast** un simpatizant în membru plin.

(1p) Declaraţi în clasa de bază metoda virtuală **getCotizaţie()**, care în clasa de bază returnează 0, iar în clasele derivate returnează valori după cum urmează:

- pentru un membru plin, 100 lei,

- pentru un membru simpatizant 50 lei.

(1p) Explicaţi conceptul de *late binding* şi apelând implementările metodei **getCotizaţie()** şi calculaţi valoarea totală adunată la nivel de partid, din cotizaţii.

(1p) Folosiţi o clasa STL pentru a da o altă implementare a asociaţiei şi parcurgând structura aleasă, verificaţi dacă funcţionează şi în acest caz conceptul de *late binding*.



# Programare Orientată Obiect, Tip-C, Sem-1, Zi (2020-2021)

[Pagina principală](#) / [Cursurile mele](#) / [Programare-C,Sem1\(2634fy\)](#) / [General](#) / [Examen 26 ianuarie 2021](#)

1 Întrebare

Nu a primit răspuns încă

Marcat din 1,00

Întrebare cu flag

(3p) Se consideră o aplicație pentru gestionarea activității unui **broker de asigurări**. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări atribute precum: numele/denumirea asiguratului, suma asigurată, tipul asigurării, durata asigurării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocati dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcție ()** care calculează valoarea asigurării într-o valută pentru care cursul de schimb este transmis ca parametru.

(1p) Se va defini **operatorul de conversie la double** a unei asigurări.

(2p) **Implementați metodele** necesare pentru scriere/citirea de obiecte în/din **fișiere binare**.

(2p) Exemplificați conceptul **"has-a"** folosind clasa definită și o nouă clasă. Clasa include o metodă care calculează valoarea medie a unei asigurări.

(1p) Calculați valoarea totală a asigurărilor definite în funcția main() cu ajutorul unui **container STL**.

↶ i B I

NAVIGARE ÎN TEST

1

[Încheiați încercarea ...](#)

Timp rămas 0:04:47



Examen 26 ianuarie 2021

online ase.ro/mod/quiz/attempt.php?attempt=369717

HL@ASE

IRINAANDREEA PIRLITU

1 Întrebare  
Nu a primit răspuns încă  
Marcat din 1,00  
Întrebare cu flag

(2p) Se consideră aplicația pentru **gestiunea documentelor** dintr-un dosar de angajare, folosind aspecte comune precum CV-ul, copie CI, copie diplomă, etc. Definiți o clasă care modelează un aspect propriu acestei activități. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic, constructorii, metodele specifice claselor cu membri alocați dinamic**. Folosiți un membru static sau const.

(1p) Supraîncărcați **operator<<** pentru afișarea informațiilor dintr-un dosar de angajare.

(2p) Să se genereze un raport într-un fișier text cu informațiile dintr-un dosar de angajare.

(2p) Să se exemplifice utilizarea mecanismul try-catch pentru a gestiona modificarea valorii unui atribut pentru care a fost implementată metoda setter. Metoda setter trebuie să genereze o excepție specifică aplicației (definită de programator) pentru date invalide. Utilizatorul este avertizat dacă introduce date invalide de la consolă având posibilitatea să le reintroducă.

(1p) Definiți și testați o metodă statică pentru afișarea numărului total de dosare create la un moment dat. Metoda va contoriza dosarele create chiar dacă nu au fost depuse.

(2p) Să se exemplifice conceptul de relație de "has a" între clase prin crearea unei clase pentru reprezentarea unei persoane care depune un dosar de angajare.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare. Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.

File Edit View Project Build Debug Test Analyze Tools

Extensions Window Help

Debug x86

Live Share

examen.cpp

```
28 };
29 class CI : public Document
30 {
31 private:
32     string CNP;
33     string serieCI;
34
35 public:
36     CI():Document()
37     {
38         CNP = "nedefinit";
39         serieCI = "ned";
40         //nume = "ned";
41     }
42     CI(string cnp, string serie, string num):Document(num)
43     {
44         CNP = cnp;
45         serieCI = serie;
46         //nume = num;
47     }
48     void setCNP(string cnp)
49     {
50         CNP = cnp;
51     }
52     string getCNP()
53     {
54         return CNP;
55     }
56 }
```

96 % No issues found Lrc 26 Cht 22 SPC CRLF

Error List

Entire Solution 0 Errors 0 Warnings 0 Messages Build + IntelliSense

Search Error List

Code	Description	Project	File	Line
------	-------------	---------	------	------

**(3p)** Se consideră o aplicație pentru gestionarea activității unui **furnizor de energie electrică**. Se vor urmări atribute specifice, precum: nume/denumire client, consum lunar efectiv, consum lunar estimat, număr contract, durată contract, preț kWh etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

**(1p)** Se va defini **operatorul funcție ()** pentru a modifica valoarea consumului lunar efectiv dintr-o anumită lună din contract.

**(1p)** Definiți **operatorul de conversie la double**, care va returna valoarea totală a diferențelor lunare de consum.

**(2p)** Scrieți **două metode**: o metodă pentru determinarea lunii cu cea mai mică diferență de consum și o metodă care calculează valoarea totală a consumului efectiv.

**(2p)** Exemplificați **conceptul de virtualizare** prin utilizarea unei clase abstracte.

**(1p)** Propuneți un **container STL** carea permite efectuarea rapidă de operații de **push** și **pop** după un **număr dat**.

Puncte 0,00/1,00

Notează 0,00 din maxim 10,00 (0%) posibil

1 întrebare

Nu a primit  
răspuns

Marcat din 1,00

🚩 Întrebare cu  
flag

(3p) Se consideră o aplicație pentru gestionarea activității unui **furnizor de energie electrică**. Se vor urmări atribute specifice, precum: nume/denumire client, consum lunar efectiv, consum lunar estimat, număr contract, durată contract, preț kWh etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul funcție ()** pentru a modifica valoarea consumului lunar efectiv dintr-o anumită lună din contract.

(1p) Definiți **operatorul de conversie la double**, care va returna valoarea totală a diferențelor lunare de consum.

(2p) Scrieți **două metode**: o metodă pentru determinarea lunii cu cea mai mică diferență de consum și o metodă care calculează valoarea totală a consumului efectiv.

(2p) Exemplificați **conceptul de virtualizare** prin utilizarea unei clase abstracte.

(1p) Propuneți un **container STL** carea permite efectuarea rapidă a operațiilor de regăsire după numărul contractului.

**(3p)** Se consideră o aplicație pentru gestiunea contactelor de pe un telefon. Se vor urmări aspectele comune privind numele, numărul de telefon, apeluri, mesaje, etc. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări atribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic, constructori, metodele specifice** claselor cu **membri alocați dinamic** și **operatorul de afișare**. Folosiți un membru **static** sau **const**.

**(1p)** Toate contactele pot fi comparate cu **operator<**; astfel, un contact se consideră mai mic decât alt contact în funcție de numele acestora.

**(2p) Specializați clasa** care descrie pentru un contact și informații cu privire la locul în care acesta lucrează.

**(1p)** Oferiți posibilitatea ca prin **operator+=** să fie adăugat un nou apel sau un nou mesaj în relația cu un contact anume.

**(2p)** Exemplificați conceptul de virtualizare prin oferirea unei funcționalități diferite pentru **Contact** și **ContactAngajat**.

**(1p)** Propuneți o metodă pentru a ține informațiile despre contactele dintr-un telefon, astfel încât înregistrările să se facă în mod unic – să nu existe două contacte cu același nume.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțin erori de compilare.

Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.





Examen 26 ianuarie 2021

online.ase.ro/mod/quiz/attempt.php?attempt=369717

HL@ASE

IRINAANDREEA PIRLITU

1 întrebare  
Nu a primit răspuns încă  
Marcat din 1,00  
Întrebare cu flag

(2p) Se consideră aplicația pentru **gestiunea documentelor** dintr-un dosar de angajare, folosind aspecte comune precum CV-ul, copie CI, copie diplomă, etc. Definiți o clasă care modelează un aspect propriu acestei activități. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic**. Folosiți un membru static sau const.

(1p) Supraincărcați **operator<<** pentru afișarea informațiilor dintr-un dosar de angajare.

(2p) Să se genereze un raport într-un fișier text cu informațiile dintr-un dosar de angajare.

(2p) Să se exemplifice utilizarea mecanismul try-catch pentru a gestiona modificarea valorii unui atribut pentru care a fost implementată metoda setter. Metoda setter trebuie să genereze o excepție specifică aplicației (definită de programator) pentru date invalide. Utilizatorul este avertizat dacă introduce date invalide de la consolă având posibilitatea să le reintroducă.

(1p) Definiți și testați o metodă statică pentru afișarea numărului total de dosare create la un moment dat. Metoda va contoriza dosarele create chiar dacă nu au fost depuse.

(2p) Să se exemplifice conceptul de relație de "has a" între clase prin crearea unei clase pentru reprezentarea unei persoane care depune un dosar de angajare.

Punctul din oficiu este inclus în prima cerință. Neimplementarea acesteia va conduce la notarea examenului cu 1.

Pentru a fi luate în considerare, soluțiile trebuie să nu conțină erori de compilare. Implementarea soluției trebuie să fie însoțită de descrierea conceptelor folosite.

File Edit View Project Build Debug Test Analyze Tools

Extensions Window Help

Debug x86

Live Share

examen.cpp

(Global Scope)

```
47 }
48 void setCNP(string cnp)
49 {
50     CNP = cnp;
51 }
52 string getCNP()
53 {
54     return CNP;
55 }
56 void afisare()
57 {
58     cout << CNP << " " << serieCI;
59 }
60 };
61
62 class Dosar
63 {
64 private:
65     Document* docs;
66
67 };
68 int main()
69 {
70
71
72     return 0;
73 }
74
```

96 % No issues found

Lrc 74 Chc 1 SPC CRLF

Error List

Entire Solution 0 Errors 0 Warnings 0 Messages Build + IntelliSense

Search Error List

Code	Description	Project	File	Line
------	-------------	---------	------	------

**(3p)** Se consideră o aplicație pentru gestionarea petițiilor primite în cadrul unei primării. Definiți o clasă care modelează o astfel de solicitare. Se vor urmări attribute specifice, precum: data depunerii, numele și prenumele solicitantului, categoria în care se încadrează petiția, descrierea solicitării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

**(1p)** Se va defini **operatorul !** pentru indica dacă o petiție nu are răspuns și a depășit termenul legal (30 de zile).

**(1p)** Se va defini **operatorul ~** pentru schimba starea unei petiții (deschisă -> rezolvată).

**(2p)** Definiți **operatorii << și >>** pentru scrierea/citirea în/din **fișiere text**.

**(2p)** Specializați clasa definită și exemplificați **conceptul de virtualizare**.

**(1p)** Propuneți un **container STL** care permite gruparea petițiilor după categorii și regăsirea cu ușurință a acestora.

This document is available on



Downloaded by Oranceanu Marius (oranceanu@gmail.com)

**(3p)** Se consideră o aplicație pentru gestionarea petițiilor primite în cadrul unei primării. Definiți o clasă care modelează o astfel de solicitare. Se vor urmări atribute specifice, precum: data depunerii, numele și prenumele solicitantului, categoria în care se încadrează petiția, descrierea solicitării etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

**(1p)** Se va defini **operatorul !** pentru indica dacă o petiție nu are răspuns și a depășit termenul legal (30 de zile).

**(1p)** Se va defini **operatorul ~** pentru schimba starea unei petiții (deschisă -> rezolvată).

**(2p)** Definiți **operatorii << și >>** pentru scrierea/citirea în/din **fișiere text**.

**(2p)** Specializați clasa definită și exemplificați **conceptul de virtualizare**.

**(1p)** Propuneți un **container STL** care permite gruparea petițiilor după categorii și regăsirea cu ușurință a acestora.

(3p) Se consideră o aplicație pentru gestionarea activității unei firme care oferă **servicii de imprimare 3D**, folosind diferite materiale (lemn, plastic, silicon etc.), pe baza modelelor primite de la clienți. Definiți o clasă care modelează un aspect propriu acestei activități. Se vor urmări attribute specifice, precum: tipul materialului, numărul de exemplare, dimensiunile, categorii, costuri etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru **static** sau **const**.

(1,5p) Se va defini **operatorul <** care permite compararea a două obiecte și va fi utilizat în cadrul unei funcții de sortare.

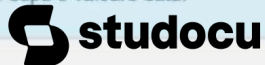
(1p) Prin intermediul **operatorului +=** se va combina obiectul curent cu un obiect primit ca parametru.

(2p) Exemplificați conceptul de relație de tip „**is a**” prin specializarea clasei/unela dintre clasele definite.

(1,5p) Implementați câte o **metodă** pentru **scrierea/citirea** unui obiect într-un **fișier binar**.

(1p) Propuneți un **container STL** ce permite regăsirea cu ușurință a unor obiecte după o valoare dată.

This document is available on



Downloaded by Oranceanu Marius (oranceanu@gmail.com)



(3p) Se consideră o aplicație pentru gestionarea activității unei companii care pune la dispoziție servicii de streaming video (filme, seriale). Se vor urmări atribute specifice, precum: preț abonament, durată, colecție de filme, colecție de seriale, statistici vizualizări (minute) etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1,5p) Se vor defini **operatorii += și -=** pentru adăugarea/eliminarea unei producții din colecție.

(1p) Definiți **operatorul de conversie la double**, care va returna cea mai vizionată producție.

(2p) **Specializați o clasă** existentă pentru a gestiona diferite tipuri de producții de streaming (filme, seriale, documentare etc.).

(1,5p) Implementați **o metodă** care determină primele cinci filme și primele cinci seriale, cele mai vizionate.

(1p) Exemplificați conceptul de **metodă/clasă template** în C++.

