



Subiecte Examen POO

Programare Orientata Obiect (Academia de Studii Economice din București)



Scan to open on Studocu

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) Supraincărcați operator`+ =` pentru a obține o serie agregată a două serii statisticice.
- (1p) Supraincărcați operator`+ =` pentru a adăuga o nouă pereche (**valoare-frecvență**) la o serie statistică existentă, menținând caracterul ei sortat.
- (1p) Supraincărcați operator`- =` pentru a elimina o pereche (**valoare-frecvență**) identificată prin valoare, dintr-o serie statistică existentă.
- (2p) Transformați una din clase **Intr-o clasă template sau Instanțați o clasă template STL** care să faciliteze lucru cu serii statisticice 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 petițiilor primite în cadrul unei localități. 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 dispozitie 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 !** 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 gestiunea unei colecții de **markere electronice** folosite pentru scrierea pe o tablă dintr-o sală. Se vor urmări aspectele comune privind culoare, dimensiune, producator, nivel curent acumulator 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) Supraîncărcați operatorii << și >> pentru afișarea, respectiv citirea unui marker electronic.

(2p) Specializați clasa care descrie un **marker intelligent** având noi câmpuri precum: grosime, tip linie, nivel presiune 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 aiba mai multe markere. Propunerea realizată trebuie să permită identificarea proprietarului foarte ușor după marker.

(3p) Se consideră o aplicație pentru gestionarea petițiilor primite în cadrul unei localități. 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 dispozitie 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 !** 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 magazin care vinde carne și produse din carne. Se vor urmări atribute specifice, precum: tipuri produse (carne proaspătă, congelată, produse din carne), tipuri de carne, data expirării, preț 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 *=** care permite acordarea unei reduceri produselor care expiră în ziua următoare.

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

(2p) Definiți o clasă pentru a gestiona produsele existente în stoc. **Operatorul +=** permite adăugarea unui produs nou (dacă există, se actualizează stocul), iar **operatorul funcție ()** va elimina toate produsele care nu sunt sănătoase în stoc.

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

(1p) Propuneți un **container STL** care permite regăsirea tuturor produselor pe baza unei specii de pește.

- (3p)** Se consideră o aplicație pentru gestionarea activității unei magazine care vinde carne și produse din carne. Se vor urmări atribută specifică, precum: tipuri produse (carne proaspătă, congelată, produse din carne), tipuri de carne, data expirării, preț 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, metode specifice claselor cu membri alocati dinamic și operatorul de afișare. Folosiți un membru static sau const.
- (1p)** Se va defini **operatorul *=** care permite acordarea unei reduceri produselor care expiră în ziua următoare.
- (1p)** Definiți **operatorul ==** care compară două produse și returnează true dacă toate valorile atributelor sunt egale între ele.
- (2p)** Definiți o clasă pentru a gestiona produsele existente în stoc. **Operatorul +=** permite adăugarea unui produs nou (dacă există, se actualizează stocul), iar **operatorul funcție ()** va elimina toate produsele care nu sunt sănătoase în stoc.
- (2p)** Explicați conceptele de **early binding** și **late binding**.
- (1p)** Propuneți un **container STL** care permite regăsirea tuturor produselor pe baza unei specii de pește.

- (3p)** Se consideră o aplicație pentru gestionarea activității unei magazine care vinde carne și produse din carne. Se vor urmări atribută specifică, precum: tipuri produse (carne proaspătă, congelată, produse din carne), tipuri de carne, data expirării, preț 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, metode specifice claselor cu membri alocati dinamic și operatorul de afișare. Folosiți un membru static sau const.
- (1p)** Se va defini **operatorul *=** care permite acordarea unei reduceri produselor care expiră în ziua următoare.
- (1p)** Definiți **operatorul ==** care compară două produse și returnează true dacă toate valorile atributelor sunt egale între ele.
- (2p)** Definiți o clasă pentru a gestiona produsele existente în stoc. **Operatorul +=** permite adăugarea unui produs nou (dacă există, se actualizează stocul), iar **operatorul funcție ()** va elimina toate produsele care nu sunt în stoc.
- (2p)** Explicați conceptele de **early binding** și **late binding**.
- (1p)** Propuneți un **container STL** care permite regăsirea tuturor produselor pe baza unei specii de pește.

Examen POO (Serie A) - 25 ianuarie 2022

online.ase.ro/mod/quiz/attempt.php?attempt=151929&cmi...

Apps profil student BLENDED LEARNING Other bookmarks Reading list

HL@ASE Română (ro)

GAVAN Andreea-Elena

Programare Orientată Obiect, Tip-C, Sem-1, Zi (2021-2022)

Acasă / Cursuri / 2021-2022 / Licenta / Programare-CSem1(2634gn) / General / Examen POO (Serie A) - 25 ianuarie 2022

Timp rămas 0:02:29

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

(3p) Se consideră o aplicație pentru gestionarea activității unei **firme distribuție**. Definiți o clasă care modelează un aspect specific acestei activități. Se vor urmări atribute precum: numele/denumirea clientului, 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 alocati 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 comanda, se va incrementa doar cantitatea.

(2p) Scrieți **într-un fișier text** comenziile care indeplinesc un criteriu dat, primit ca parametru.

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

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

Type here to search

(3p) Se consideră o aplicație pentru gestiunea cărților de vizită primite de către o persoană. Se vor urmări aspectele comune privind numele, numărul de telefon, adresă de email, nume, companie, 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) Toate cărțile de vizită pot fi comparate cu **operator>=**; comparația este realizată după un atribut la alegeră.

(2p) Specializați clasa care descrie pentru un o carte de vizită în format electronic.

(1p) Supraîncărcați operatorul **>>** care permite citirea informațiilor despre o carte de vizită de la tastatură .

(2p) Exemplificati conceptul de virtualizare prin oferirea unei funcționalități diferite pentru cele două clase.

(1p) Propuneți o metodă pentru a gestiona cărțile de vizită, astfel încât înregistrările să se facă în mod unic – să nu existe două cărți de vizită ale aceleiași persoane cu același nume.

(3p) Se consideră o aplicație pentru gestionarea locuințelor dintr-o localitate. Se vor urmări atribute specifice, precum: suprafață, număr camere, cu sau fără etaj, 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 definește **operator()(int)** (operator funcție) care primește numărul de locatari. Operatorul returnează true dacă fiecare persoană are posibilitatea să aibă propria camera în locuință sau false daca sunt mai puțin camera decât persoane.

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

(2p) Exemplificați conceptul de relație de tip „is a” prin specializarea clasei Locuintă. Testați soluția prin instantierea noii clase.

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

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

(3p) Se consideră o aplicație pentru gestiunea unei colecții de **markere electronice** folosite pentru scrierea pe o tablă dintr-o sală. Se vor urmări aspectele comune privind culoare, dimensiune, producător, nivel curent, acumulator 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 alocăți dinamic**. Folosiți un membru **static** sau **const**.

(1p) Supraîncărcați operatorii << și >> pentru afișarea, respectiv citirea unui marker electronic.

(2p) Specializați clasa care descrie un **marker intelligent** având noi câmpuri precum: grosime, tip linie, nivel presiune 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 aiba mai multe markere. Propunerea realizată trebuie să permită identificarea proprietarului foarte ușor după marker.

(3p) Se consideră o aplicație pentru gestiunea cărților de vizită primite de către o persoană. Se vor urmări aspectele comune privind numele, numărul de telefon, adresă de email, nume, companie, 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) Toate cărțile de vizită pot fi comparate cu **operator>=**; comparația este realizată după un atribut la alegeră.

(2p) Specializați clasa care descrie pentru un o carte de vizită în format electronic.

(1p) Supraîncărcați operatorul **>>** care permite citirea informațiilor despre o carte de vizită de la tastatură .

(2p) Exemplificati conceptul de virtualizare prin oferirea unei funcționalități diferite pentru cele două clase.

(1p) Propuneți o metodă pentru a gestiona cărțile de vizită, astfel încât înregistrările să se facă în mod unic – să nu existe două cărți de vizită ale aceleiași persoane cu același nume.

(3p) Se consideră o aplicație pentru gestionarea activității unei magazine care vinde carne și produse din carne. Se vor urmări atribută specifică, precum: tipuri de produse (carne proaspătă, congelată, produse din carne), tipuri de carne, data expirării, preț 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, metode specifice claselor cu membri alocati dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) Se va defini **operatorul *=** care permite acordarea unei reduceri produselor care expiră în ziua următoare.

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

(2p) Definiți o clasă pentru a gestiona produsele existente în stoc. **Operatorul +=** permite adăugarea unui produs nou (dacă există, se actualizează stocul), iar **operatorul funcție ()** va elimina toate produsele care nu sunt în stoc.

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

(1p) Propuneți un **container STL** care permite regăsirea tuturor produselor pe baza unei specii de pește.

(3p) Definiți clase care să permită abstractizarea conceptului de frizerie (adresa, număr frizeri, denumire, etc), folosind membri de tip public, private, protected, const, static. Clasa conține cel puțin un câmp alocat dinamic, constructori și 2 metode accesor (set va valida valoarea primită) pentru un atribut la alegere.

(1p) Supraîncărcați operatorii de citire și afișare la consolă. Operatorul de citire permite utilizarea de șiruri de caractere cu spații.

(2p) Să se exemplifice conceptul de clasă abstractă (Serviciu) și derivare (Tuns, Frezat, Bărbierit). Clasa abstractă impune o metodă de calcul a prețului.

(2p) Exemplificați conceptul de virtualizare în clasa frizerie prin utilizarea unui vector de pointeri la servicii. Se va adăuga o metodă ce calculează prețul total al serviciilor solicitate.

(1p) Supraîncărcați operatorii += și -= pentru a adăuga și șterge servicii.

(1p) Să se utilizeze un container STL adekvat ce ajută la găsirea rapidă a prețului unui serviciu pe baza denumirii acestuia.

online.ase.ro/mod/quiz/attempt.php?attempt=361131

sisc ESEU SPORT

HL@ASE RÖMÂNĂ (RO) NICOLEANAMARIA ALE

BibliotecaImprumuturi

Să se scrie o aplicație orientată obiect pentru gestiunea **Imprumuturilor 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) Elaborează constructor cu parametri, default constructor, constructor de copiere, destrutor și **operator<<** pentru afișare.

(2p) Supraincarcați 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, folosind 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 intr-o clasă template sau instantiați o clasă template **STL**, pentru domeniul dat.

(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**.

(3p) Se consideră o aplicație pentru gestiunea unei colecții de **markere electronice** folosite pentru scrierea pe o tablă dintr-o sală. Se vor urmări aspectele comune privind culoare, dimensiune, producător, nivel curent, acumulator 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 alocăți dinamic**. Folosiți un membru **static** sau **const**.

(1p) Supraîncărcați operatorii << **și** >> pentru afișarea, respectiv citirea unui marker electronic.

(2p) Specializați clasa care descrie un **marker intelligent** având noi câmpuri precum: grosime, tip linie, nivel presiune 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 aiba mai multe markere. Propunerea realizată trebuie să permită identificarea proprietarului foarte ușor după marker.

- (3p)** Se consideră o aplicație pentru gestionarea activității unui furnizor de energie electrică alternativă. Se vor urmări atribute specifice, precum: nume/denumire client, sursă energie, 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 alocăți dinamic și operatorul de afișare. Pentru sursele de energie utilizați constante enumerative (enum). Folosiți un membru static sau const.
- (1p)** Se va defini **operatorul funcție 0** 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** care permite efectuarea rapidă a operațiilor de regăsire după numărul contractului.

(3p) Se consideră o aplicație pentru gestiunea cărților de vizită primite de către o persoană. Se vor urmări aspectele comune privind numele, numărul de telefon, adresă de email, nume, companie, 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) Toate cărțile de vizită pot fi comparate cu **operator>=**; comparația este realizată după un atribut la alegeră.

(2p) Specializați clasa care descrie pentru un o carte de vizită în format electronic.

(1p) Supraîncărcați operatorul **>>** care permite citirea informațiilor despre o carte de vizită de la tastatură .

(2p) Exemplificati conceptul de virtualizare prin oferirea unei funcționalități diferite pentru cele două clase.

(1p) Propuneți o metodă pentru a gestiona cărțile de vizită, astfel încât înregistrările să se facă în mod unic – să nu existe două cărți de vizită ale aceleiași persoane cu același nume.

Examen P.O.O. 01.02.2022 Launch Meeting - Zoom

online.ase.ro/mod/quiz/attempt.php?attempt=180941&cmid=104017

HL@ASE Română (ro) ANTAL-VAIDA Raluca

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

(3p) Se consideră o aplicație folosită pentru a gestiona personajele dintr-un joc pe calculator/mobil. Se vor urmări atribute specifice, precum: nume, numar lovitură, puncte per lovitura. Datele membre sunt private sau protected. Clasa va permite următoarele apeluri în programul principal:

```
PersonajJoc p1;
PersonajJoc p2("Batman", 3, new int[] { 1, 2, 3 });
PersonajJoc p3(p2);
p2.setLovituri(2, new int[] { 5, 5 });
```

(1p) Să se suprînscrace operatorul += în forma obiect+=valoare pentru a modifica valoarea unui atribut.

(1p) Definiți operatorul >= care compară două obiecte și returnează true în funcție de valorile unui atribut (indicat de comisie).

(2p) Exemplificati conceptul de relație de tip „is a” prin specializarea clasei anterioare. Clasa nouă adaugă un atribut nou și constructorul cu parametri apelează explicit constructorul clasei de bază. Testați soluția prin instantierea noii clase.

(2p) Explicați și exemplificati conceptul de clasa abstractă. Derivați una dintre clasele existente din clasa abstractă și testați în main().

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

Text editor toolbar: A, B, I, etc.

System tray: -9°C Partly su..., ENG, 10:08

(Dp) Se consideră o aplicație pentru gestionarea activității unei firme care oferă servicii de securitate și supraveghere video. Definiți o clasă care modelază un sistem de supraveghere. Se vor urmări attribute 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 pompă, 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, constructor, metodele specifice claselor cu membri alocati dinamic și operatorului de afișare. Folosiți un membru static sau const.

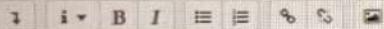
(1p) Se va defini operatorul [], astfel încât să se poată accesa o anumită cameră de supraveghere după un identificator.

(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/unei din clasele definite.

(2p) Implementați două metode: o metodă care identifică și returnează camerele care îndeplinește anumite condiții, 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 (clase template).



(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 alocati dinamic și operatorul de afișare. Folosiți un membru static sau const.

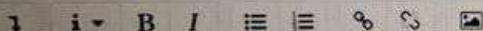
(1,5p) Se vor defini operatorii `+=` și `=` pentru adăugarea/eliminarea unei producții din colecție.

(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/clasa template în C++.



(3p) Definiți o clasă care să permită gestionarea datelor aferente temei pentru acasă (cerințe, termen limită, tip fișier de încărcat, puncte, etc.), folosind membri de tip public, private, protected, const, static. Clasa conține cel puțin un câmp alocat dinamic, constructor și două metode accesor (set va valida valoarea primită pentru un atribut la alegere.

(1p) Supraîncărcați operator+= valoare care să permită creșterea punctajului.

(1p) Exemplificați în main() mecanismul de tratare a exceptiilor try-catch prin modificarea metodei set implementată anterior și prin definirea unei exceptii proprii.

(2p) Definiți o metodă care să permită scrierea/serializarea subiectului într-un fișier de tip binar. Metoda primește numele fișierului ca parametru.

(2p) Să se exemplifice utilizarea unei relații de tip „has a” de tip 1:M (unu la mai mult) prin definirea unei clase suplimentare care să gestioneze mapa de teme ale unui seminar. Se pot folosi colectii STL.

(1p) Supraîncărcați un operator la alegere care să permită adăugarea de teme în mapa definită anterior.



Status	Terminat
pletat la	luni, 5 iulie 2021, 09:05
necesar	55 min 39 secs
Puncte	0,00/1,00
Notează	0,00 din maxim 10,00 (0%) posibil

InchirieriCamioane

Proiectați și dezvoltați o aplicație orientată obiect pentru o firmă ce deține 20 de camioane pentru închiriere către clienți .

(3p) Definiți clase și afișați obiecte de tip **camion** și de tip **firmă**, necesare aplicației; se vor folosi constructori cu parametri cu valori implicate .

Considerând că închirierea se face la nivel de zi întreagă, aplicația trebuie să permită:

(2p) Înregistrarea unei închirieri a unui camion pentru o zi dintr-un an, către un client persoană fizică, identificat prin CNP, nume și telefon

(1p) vizualizarea situației (stării) unui camion într-o zi (liber sau închiriat)

(2p) Calculați încasările din închirieri la nivelul întregului parc de camioane, considerând chiria de 100 lei / camion /zi.

(1p) Indicați modificările ce trebuie operate pentru ca aplicația să permită închirierea de camioane și către alte firme, nu doar către persoane fizice.

(1p) Transformați una din clase într-o clasă template pentru a permite închiriere către ambele tipuri de clienți.

Finalizare ve

Examen POO 25 ianuarie 2021 X Launch Meeting - Zoom X Slack | general | Curs POO 2020 X +

https://online.ase.ro/mod/quiz/attempt.php?attempt=362057

HL@ASE ROMÂNĂ (RO) IULIAALEXANDRA CAPLEA

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

Pagina principală / Cursurile mele / Programare-C,Sem1(8422cy) / Examen 25 ianuarie 2021 / Examen POO 25 ianuarie 2021 - Seria A

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

(3p) Se consideră o aplicație care permite gestiunea unei scară de bloc. Se vor urmări aspectele comune privind denumire, adresa, apartamente, locatari etc.. Definiți o clasă care modelază 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 alocati dinamic și operatorul de afișare. Folosiți un membru static sau const.

(1p) supraîncărcați operatorul pentru accesarea elementelor din vectorul alocat dinamic al clasei. Operatorul poate fi folosit și pentru ștergere și pentru modificare

(1p) Creați o nouă clasă Bloc, care să conțină mai multe obiecte de tip 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 operatorul să fie adăugată o nouă scară în cadrul unui obiect de tip Bloc.

(2p) În cadrul clasei Bloc implementați două metode: una returneză scară cea mai productivă, iar cea de a doua metodă returneză scară 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 concepției folosite.

NAVIGARE ÎN TEST

1 Închelați Încercarea ...
Timp rămas 0:13:37

BibliotecaImprumuturi

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, destrutor și **operator<<** pentru afișare.

(2p) Supraîncărcați 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 accesor (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.



InchirieriCamioane

Proiectați și dezvoltați o aplicație orientată obiect pentru o firmă ce deține 20 de camioane pentru închiriere către clienți.

(3p) Definiți clase și afișați obiecte de tip **camion** și de tip **firmă**, necesare aplicației; se vor folosi constructori cu parametri cu valori implicate.

Considerând că închirierea se face la nivel de zi întreagă, aplicația trebuie să permită:

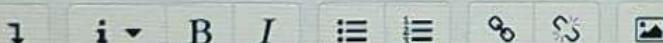
(2p) Înregistrarea unei închirieri a unui camion pentru o zi dintr-un an, către un client persoană fizică, identificat prin CNP, nume și telefon

(1p) vizualizarea situației (stării) unui camion într-o zi (liber sau închiriat)

(2p) Calculați încasările din închirieri la nivelul întregului parc de camioane, considerând chiria de 100 lei / camion /zi.

(1p) Indicați modificările ce trebuie operate pentru ca aplicația să permită închirierea de camioane și către alte firme, nu doar către persoane fizice.

(1p) Transformați una din clase într-o clasă template pentru a permite închiriere către ambele tipuri de clienți.



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

Pagina principală / Cursuri / 2020 - 2021 / Licenta / Programare-C.Sem1(2634fy) / General
/ Examen 6 iulie 2021, ora 7:30

1 întrebare

Nu a primit
răspuns încă

Marcat din 1,00

▼ Întrebare
cu flag

(3p) Se consideră o aplicație pentru gestionarea activității unei benzinării. Se vor urmări atribute specifice, precum: tip combustibil, preț/litr, pompe, capacitate rezervare stație etc. Datele menite sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin patru câmpuri, din care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocati dinamice, operatorul de afișare. Folosiți un membru static sau const.

(2p) Se va defini operatorul += pentru alimentarea unui rezervor din stație asociat unui anumit tip de combustibil cu o valoare dată. Verificați să nu fie depășită capacitatea rezervorului. Operatorul -= va utiliza atunci când este alimentat un autovehicul, modificând corespunzătorul combustibilului disponibil. Operatorul != se utilizează pentru a verifica dacă un rezervor este gol.

(1,5p) Implementați o metodă pentru generarea unui raport (fișier text) cu vânzările pe fiecare tip de combustibil.

(1p) Scrieți o metodă pentru determinarea tipului de combustibilului cel mai vândut.

(1,5p) Exemplificați în funcția main() mecanismul de tratare a exceptiilor try-catch prin modificați unui operator implementat anterior, folosind o excepție proprie.

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

Mute

Stop Video

Participants
2

New Share

You are screen sharing

||

Pause Share

Annotate

Remove Control

More

ENG

06-07-21

Polinom

(2p) Să se elaboreze clase care să faciliteze lucru cu polinoame, un polinom fiind stocat în memorie ca un **vector dinamic de termeni** de forma unor perechi (coeficient-grad), sortată **descrescător** după grad.

(1p) Se va scrie un constructor de clasă pentru crearea unui obiect dintr-un număr variabil de perechi (coeficient, grad).

(1p) Supraîncărcări ale **operator<<** conlucrăză pentru afișarea unui **termen**, respectiv a întregului **polinomul**.

(0.5p) Destructorul de clasă eliberează memoria dinamică ocupată de obiect.

(0.5p) Supraîncărcăți **operator cast** la **double** astfel încât să returnează valoarea polinomului într-un punct dat.

(1p) Supraîncărcăți **operator--** pentru **derivarea formală** a polinomului (pentru fiecare termen coeficientul se înmulțește cu gradul, iar gradul scade cu 1; termenul liber adică de grad zero, dacă există va dispărea prin derivare).

(2p) Supraîncărcăți, la alegere, **două** dintre operatorii menționați mai jos:

- ++** pentru **integrare formală**
- **+, -, *, / și %** pentru adunare, scădere, înmulțire, împărțire și restul împărțirii a două polinoame
- **+** pentru **adăugarea unui nou termen** și **-** pentru **eliminarea unui termen**
- **<< și >>** pentru operații de **intrare - ieșire** la nivel de obiect
- **operator!=** pentru test de polinom nul.

(2p) Supraîncărcăți la alegere **două** dintre **funcțiile** de mai jos:

- **integ(double,double)** - integrare numerică, pe un interval folosind operatorii **++** și **()**
- **deriv(double)** - evaluarea derivatei într-un punct, folosind operatorii **-** și **()**
- **cmmMc()** pentru cel mai mare divizor comun a două polinoame și **cmmc()** pentru cel mai mic multiplu comun a două polinoame, folosind operatorii **/ și ***
- **rad(double,double)** - determinarea unei rădăcini dintr-un interval dat

(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 clienti. Definiți o clasă care modeleză 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 dispozitie 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 afisare. 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) Exemplificăți conceptul de relație de tip „*is a*” prin specializarea claselor/dintre clasele definite.

(1,5p) Implementați căte 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 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 alocate dinamic și operatorul de afișare. Folosiți un membru static sau const.

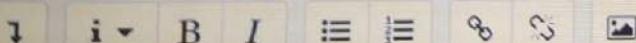
(1p) Se va defini operatorul `[]` pentru a accesa un produs după poziția dată.

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

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

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

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



(3p) Se consideră o aplicație folosită pentru a gestiona traseul dintre puncte turistice. Se vor urmări atribute specifice, precum: distanță, puncte turistice, durată, puncte de interes, hartă etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa conține cel puțin trei câmpuri, dintre care unul este alocat dinamic, un constructor cu parametri, destructor, metode accesori (get și set) pentru unul dintre atribute. Metoda set() validează datele de intrare. Folosiți un membru static sau const.

(1p) Să se suprîncarce operatorul - în forma obiect - valoare pentru a modifica valoarea unui atribut.

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

(2p) Exemplificati conceptul de relație de tip „is a” prin specializarea clasei anterioare. Clasa nouă adaugă un atribut nou și constructorul cu parametri apelează explicit constructorul clasei de bază. Testați soluția prin instanțierea noii clase.

(2p) Explicați și exemplificati conceptul de late-binding prin definirea unei metode virtuale în clasa de bază.

(1p) Exemplificati 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 însorită de descrierea conceptelor folosite.