



Variante POO

Programare Orientata Obiect (Academia de Studii Economice din București)



Scan to open on Studocu

1.SUBIECT

(3p) Se consideră o aplicație pentru gestionarea activității unei magazine care vinde baterii externe de tipul powerBank. Se vor urmări atribută specifică precum: capacitate greutate număr intrări/ieșiri, 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 alocati dinamic și operatorul de afișare. Folosiți un membru static sau const

(1p) Se vor defini operator+= și operator-= care permit incarcarea powerbank-ului respectiv descarcarea acestuia

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

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

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

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

2.SUBIECT (2p) Se consideră 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 alocati dinamic. Folosiți un membru static sau const.

(1p) Supraîncărcați 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 attribute.

(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.

SUBIECT

(3p) Se considera o aplicatie pentru gestiunea cartilor de vizita primite de catre o persoana. Se vor urmări aspectele comune privind numele numanul de telefon, adresa de email, nume. companie, etc.

Definiti o clasa care modeleaza un aspect propriu acestei activitati. Se vor urmani atribute specifice, precum tipul materialului, numarul de exemplare, dimensiunile categorii, costuri etc.

Datele membre sunt private si sunt puse la dispozitie metode de acces

Clasa contine cel putin patru campuri dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocati dinamic si operatorul de afisare

Folositi un membru static.

(1p) Toate cartile de vizita pot fi comparate cu operator \geq comparatia este realizata dupa un atribut la alegere

(2p) Specializați clasa care descrie pentru un a carte de vizita in format electronic

(1p) Supraincarcati operator care permite citirea informatiilor despre o carte de vizita de la tastatura

(2p) Exemplificati conceptul de virtualizare si late binding pun oferinea unei functionalitati diferite pentru cele doua clase

(1p) Propuneti o metoda pentru a gestiona cartile de vizita, astfel incat

inregistranle sa se faca in mod unic sa nu exista doua carti de vizita ale aceleasi persoane cu acelasi nume. Punctul din oficiu este inclus in prima cerinta. Neimplementarea acesteia va conduce la notarea examenului 1.

Pentru a luate in considerare solutile trebuie sa nu contina erori de compilare. Implementarea solutiei trebuie sa te insotita de descrierea conceptelor folosite.

SUBIECT

(3p) Se considera o aplicatie pentru gestiunea cartilor de vizita primite de catre o persoana. Se vor urmări aspectele comune privind numele numanul de telefon, adresa de email, nume. companie, etc.

Definiti o clasa care modeleaza un aspect propriu acestei activitati. Se vor urmani atribute specifice, precum tipul materialului, numarul de exemplare, dimensiunile

cateorii, costuri etc.
Datele membre sunt
private si
sunt puse la dispozitie
metode de acces
Clasa contine cel putin
patru campuri dintre care
unul este alocat dinamic,
constructori, metodele
specifice claselor cu
membri alocati dinamic si
operatorul de afisare
Folositi un membru static.
(1p) Toate cartile de vizita
pot fi comparate cu
 $\operatorname{operator} \geq$ comparatia
este realizata dupa un
atribut la

alegere

(2p) Specializați clasa care descrie pentru un a carte de vizita in format electronic

(1p) Supraincarcati operator care permite citirea informatiilor despre o carte de vizita de la tastatura

(2p) Exemplificati conceptul de virtualizare si late binding pun oferinea unei functionalitati diferite pentru cele doua clase

(1p) Propuneti o metoda pentru a gestiona cartile de vizita, astfel incat

inregistranle sa se faca in mod unic sa nu exista doua carti de vizita ale aceleasi persoane cu acelasi nume. Punctul din oficiu este inclus in prima cerinta. Neimplementarea acesteia va conduce la notarea examenului 1.

Pentru a luate in considerare solutile trebuie sa nu contina erori de compilare. Implementarea solutiei trebuie sa te insotita de descrierea conceptelor folosite.

3.SUBIECT (3p) Se considera o aplicatie pentru gestiunea cartilor de vizita primite de catre o persoana. Se vor urmări aspectele comune privind numele numanul de telefon, adresa de email, nume, companie, etc. Definiti o clasa care modeleaza un aspect propriu acestei activitati. Se vor urmari atribute specifice, precum tipul materialului, numarul de exemplare, dimensiunile categorii, costuri etc. Datele membre sunt private si sunt puse la dispozitie metode de acces. Clasa contine cel putin patru campuri dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocati dinamic si operatorul de afisare. Folositi un membru static.

(1p) Toate cartile de vizita pot fi comparate cu operator>= comparatia este realizata dupa un atribut la alegere

(2p) Specializați clasa care descrie pentru un a carte de vizita in format electronic

(1p) Supraîncarcați operator care permite citirea informatiilor despre o carte de vizita de la tastatura

(2p) Exemplificati conceptul de virtualizare si late binding pun oferinea unei functionalitati diferite pentru cele doua clase

(1p) Propuneti o metoda pentru a gestiona cartile de vizita, astfel incat inregistrante sa se faca in mod unic sa nu exista doua carti de vizita ale acelasi persoane cu acelasi nume

4.SUBIECT ProdusDerivat (2p) Definiți o clasă abstractă Produs care conține două metode:

• calcul valoare prin produsul dintre cantitate și pret, • respectiv calcul TVA pe bază de procent din valoare. Clasa abstractă declară și datele necesare implementării acestor metode.

(1p) Derivați din ea o clasă care să modeleze comerțul electronic cu produse de încărcătură și instanțați obiecte.

(1p) Supraîncărcați operator<< pentru afișarea informațiilor moștenite și specifice.

(2p) Definiti metode sau supraîncărcați operatori pentru creșterea / scăderea stocului prin aprovizionare / vânzare de produse.

(1p) Testați comportamentul virtual al metodei de calcul TVA pentru două categorii diferite de produse și servicii.

(1p) Comparați calculul TVA folosind funcții virtuale cu varianta în care fiecare produs are un membru ce conține procentul de TVA care i se aplică. Arătați avantajele și dezavantajele luând în considerare spațiul de memorie, volumul actualizărilor la schimbarea legislației, necesitatea recompilării claselor etc.

(2p) Adăugați în clasă două metode statice pentru salvarea, respectiv restaurarea, unei colecții de obiecte de tipul clasei (vector static) în/din fișiere binare

5.SUBIECT (3p) Se consideră o aplicatie 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, numar contract durată contract, pret kWh etc. Datele membre sunt private și sunt puse la dispoziție metode de acces. Clasa contine cel putin patru câmpuri, dintre care unul este alocat dinamic. constructori, metodele specifice claselor cu membri alocati dinamic și operatorul de afisare. Pentru sursele de energie utilizați constante enumerative (enum). Folositi 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) Definiti operatorul de conversie la double, care va returna valoarea totala a differentelor lunare de consum.

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

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

SUBIECT

(3p) Se consideră o aplicație pentru gestionarea activității unei magazin care vinde baterii externe. Se vor urmări atribute specifice, precum:

capacitate, nivel curent,
producător, greutate,
număr
intrări/iesiri, tipuri
intrări/ieșiri 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.
Folositi un membru
static sau const.

(1p) Se vor defini operator`+=` și operator`-=` care permit încărcarea bateriei externe, respectiv descărcarea acestora.

(1 p) Definiți operatorul `==` care compară două obiecte de tip baterie externă și returnează true dacă

toate valorile atributelor sunt egale între ele.

(2p) Exemplificați conceptul de relație de tip „is a” prin specializarea clasei de tip baterie externă. Testați

soluția prin instanțierea noii clase.

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

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

6.SUBIECT (3p) Se consideră o aplicație pentru gestionarea activității unei magazin care vinde baterii externe. Se vor urmări atribute specifice, precum: capacitate, nivel curent, producător, greutate, număr intrări/iesiri, tipuri intrări/iesiri 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. Folositi un membru static sau const.

(1p) Se vor defini operator+= și operator-= care permit încărcarea bateriei externe, respectiv descărcarea acesteia.

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

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

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

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

7.SUBIECT (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 alocați dinamic și operatorul de afișare. Folosiți un membru static sau const.

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

(1 p) 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 care permite efectuarea rapidă a operațiilor de inserare/stergere de produse într-o/dintr-o comandă.

8.SUBIECT (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 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 attribute.

(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.

9.SUBIECT (3p) Se consideră o aplicație folosită pentru a gestiona rezervarea unui sejur de vacanță. Se vor urmări attribute specifice, precum: destinația, data de început, data de final, numarul de persoane, tip mese, tip camera, însotitori etc. Datele membre sunt private sau protected. 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 attribute. Folosiți un membru constant. Metoda set() validează datele de intrare.

(1p) Sa se supraincarce operatorul + în forma valoare + obiect pentru a modifica valoarea unui atribut.

(1p) Să se supraincarce operatorul de indexare [] care permite accesul în mod citire și scriere la un element dintr-un atribut de tip vector.

(2p) Exemplificati conceptul de relatie de tip „has a” prin definirea unei clase noi care să gestioneze mai multe obiecte de tipul clasei anterioare. Implementați o metodă care să permită adaugarea unui obiect în colecție. Testați soluția în main().

(2p) Exemplificati conceptul de serializare și deserializare a unui obiect prin scrierea valorilor acestuia într-un fișier binar.

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

10.SUBIECT 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 adauga 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.

11.SUBIECT (3p) Definiți o clasă care să permită gestiunea datelor aferente unui subiect de examen (titlu, barem pentru fiecare cerință, data examenului, etc.), folosind membri de tip public, private, protected, const, static. Clasa conține un câmp alocat dinamic de perechi (cerință, punctaj), constructori și două metode accesore (set va valida valoarea primită) pentru un atribut la alegere.

(1p) Supraincărcați operator+ care să permită implementarea operației string + obiect pentru completare enunt.

(2p) Supraincărcați operator<< pentru a permite scrierea subiectului într-un fișier de tip text. Testați în main pe un fișier text creat local.

(2p) Să se exemplifice utilizarea unei relații de tip has a de tip 1:1 (unu la unu) prin definirea unei clase suplimentare care să gestioneze evaluarea unui student. Pentru o clasa se definește constructorul de copiere.

(1p) Exemplificați relația dintre constructorii celor două clase, aflate în relația de 1:1, în situația în care clasa inițială nu are constructor implicit.

(1p) Explicați și exemplificați utilitatea pointerilor prin intermediul unui pointer la obiectele uneia dintre clasele definite anterior.