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, dimensiuni, categorie, 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**.

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

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

c) Exemplificați conceptul de relație de tip "has-a" prin gestiunea unei colecții de servicii de imprimare.

d) Implementați câte o **metodă pentru scrierea/citirea** unui obiect într-un fișier binar.

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

Se consideră o aplicație pentru gestionarea 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**.

a) Supraincărcați operatorii **<< și >>** pentru afișarea, respectiv citirea unui marker electronic.

b) Specializați clasa care descrie un **marker inteligent** având noi câmpuri precum: grosime, tip linie, nivel presiune etc.

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

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

e) 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.

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 din mai multe cărți diferite.

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

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

c) Supraincărcați în clasa **Cititor** operatorii **+=** și **-=** pentru a împrumuta, respectiv returna un exemplar dintr-o carte.

d) 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.

e) Implementați două funcții de tip accesor (**get și set**) și două metode proprii clasei (0.25 puncte/functie/operator).

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

g) Transformați una din clase într-o **clasă template** sau instanțiați o clasă template STL, pentru domeniul dat.