Lista temelor, cu toate cerintele pentru **prima lucrare practica**.

Cerinte comune tuturor temelor:

- implementare in C++ folosind clase
- datele membre private
- constructori de initializare (cu si fara parametri), constructor de copiere, destructor, operator de atribuire
- get, set pentru toate datele membre
- **ATENTIE:** functiile pe care le-am numit mai jos *metode* (fie ca sunt supraincarcari de operatori, fie altfel de functii), **pot fi implementate ca functii prieten** in loc de metode ale claselor respective, daca se considera ca aceasta alegere este mai naturala;
- supraincarcarea operatorilor << si >> pentru iesiri si intrari de obiecte, dupa indicatiile de mai jos, astfel incat sa fie permise (si ilustrate in program);
- sa existe metode publice prin care se realizeaza citirea si afisarea informațiilor complete a n obiecte, memorarea și afisarea acestora.
- programul sa aiba un meniu interactiv

Necesar: programul sa nu contina erori de compilare si sa ruleze in CodeBlocks

Coding style este un plus

Cerinte specifice fiecarei teme:

Tema 0. Clasa ”Numar_Complex” //se va implementa in timpul orelor de laborator

- membrii privati pentru partea reala si partea imaginara (double);
- constructori pentru initializare si copiere;
- destructor (în cazul alocarii statice, se seteaza dimensiunea vectorului la zero, iar în cazul alocarii dinamice, se dezaloca zona de memorie utilizata);
- metode publice pentru setat/furnizat partea reala si partea imaginara;
- metoda publica de afisare (sub forma "a", "i*a", "-i*a", "a+i*b", "a-i*b");
- metoda publica pentru determinarea modulului unui numar complex;
- suma a doua numere complexe, implementata prin supraincarcarea op +;
- produsul a doua numere complexe, implementat prin supraincarcarea op *;
- împărțirea a doua numere complexe, implementata prin supraincarcarea op /.

Tema 1. Clasa ”Vector_Complex”

- clasa este prietena a clasei **Numar_Complex** definita în **Tema 0**
- membri privati pentru vectorul propriu zis si numarul de elemente;
- constructor pentru initializarea cu un numar dat pe toate componentele (primeste ca parametru numarul respectiv si numarul componentelor);
- constructori pentru initializare si copiere;
- destructor (în cazul alocarii statice, se seteaza dimensiunea vectorului la zero, iar în cazul alocarii dinamice, se dezaloca zona de memorie utilizata);
- supraincarcarea operatorilor << și >> sa utilizeze supraincarea acestora în cadrul clasei Numar_Complex;

- metoda publica pentru determinarea vectorului modulelor, folosind metoda de determinare a modulului din clasa numar complex;
- metoda publica pentru sortarea crescatoare dupa module a vectorului;
- metoda publica pentru calculul sumei tuturor elementelor vectorului, care sa utilizeze operatorul + din clasa de numere complexe;

Tema 2. Clasa ”Matrice_Complexa”

- clasa este prietena a clasei **Numar_Complex** definita în **Tema 0**;
- membri privati pentru matricea propriu zisa, nr linii și nr coloane;
- constructor pentru initializarea cu un numar dat pe toate componentele (primeste ca parametru numarul respectiv, numarul de linii și numărul de coloane);
- constructori pentru initializare si copiere;
- destructor (în cazul alocarii statice, se seteaza dimensiunile matricei la zero, iar în cazul alocarii dinamice, se dezaloca zona de memorie utilizata);
- supraincarcarea operatorilor << și >> sa utilizeze supraincarea acestora în cadrul clasei Numar_Complex;
- metoda publica pentru calculul sumei a doua matrici, implementata prin supraincarcarea operatorului + si pe baza functiei de supraincarcare a lui + din clasa numar complex;
- metoda publica pentru calculul produsului a doua matrici, implementat prin supraincarcarea operatorului * si pe baza functiilor de supraincarcare ale lui + si * din clasa numar complex.

Tema 3. Clasa ”Vector” (vector de numere întregi) // usor

- membri privati pentru vectorul propriuzis si numarul de elemente;
- constructor pentru initializarea cu un numar dat pe toate componentele (primeste ca parametru numarul respectiv si numarul componentelor);
- constructori pentru initializare si copiere;
- destructor (în cazul alocarii statice, se seteaza dimensiunea vectorului la zero, iar în cazul alocarii dinamice, se dezaloca zona de memorie utilizata);
- metoda-operator public de atribuire =;
- metoda publica pentru reactualizarea numarului de componente si initializarea componentelor cu un numar dat (primeste ca parametru numarul respectiv si numarul componentelor);
- metoda publica pentru calculul sumei tuturor elementelor vectorului;
- metoda publica pentru găsirea maximului și a pozitiei lui;
- metoda publica pentru sortarea crescătoare a vectorului;

Tema 4. Clasa ”Stiva_de_caractere” (implementata dinamic)

Se considera ***Class Nod{ char info; Nod* next;}***

- constructori de inițializare și parametrizati pentru clasa Nod;
- destructor pentru clasa Nod;

Clasa Stiva_de_caractere are:

- membru privat, un "Nod*" (varful stivei);
- un constructor care initializeaza varful stivei cu NULL;
- un destructor care dezaloca toate elementele stivei;

- metode publice de adaugare a unui element în stiva (***push***), de stergere a unui element (***pop***), de afisare a varfului stivei (***top***) și pentru a testa daca stiva e vida (***isempty***);
- metoda publica de fisarea stivei, concomitent cu golirea ei, realizata prin supraincarcarea operatorului <<;
- supraincarcarea operatorului >>, realizata prin push-uri succesive;
- metoda publica pentru inversarea unui sir de caractere folosind o stiva;
- metoda publica, realizata prin supraincarcarea operatorului -, care sa considere doua stive și sa elimine, concomitent, elementele din ambele stive adaugand caracterul ce are codul ASCII mai mare într-o noua stiva, ca în exemplul de mai jos:

Stiva_de_caractere S1,S2;
S1 = {E,X,A,M,E,N}; S2 = {P,O,O,L,A,B,O,R,A,T,O,R} S1 - S2 = {R,O,T,A,X,O}.

Tema 5. Clasa ”Coada_de_caractere” (implementata dinamic)
Se considera ***Class Nod{ char info; Nod* next;}***

- constructori de inițializare și parametrizati pentru clasa Nod;
- destructor pentru clasa Nod;

Clasa Coada_de_caractere are:

- membri privati, "Nod*, Nod*" (primul și ultimul element al cozii);
- un constructor care initializeaza coada cu NULL;
- un destructor care dezaloca toate elementele cozii;

- metode publice de adaugare a unui element în stiva (***push***), de stergere a unui element (***pop***) și pentru a testa daca e vida (***isempty***);
- metoda publica de fisarea a cozii, concomitent cu golirea ei, realizata prin supraincarcarea operatorului <<;
- supraincarcarea operatorului >>, realizata prin push-uri succesive;
- metoda publica pentru concatenarea a doua cozi de caractere, obtinand o alta coada de caractere, implementata prin supraincarcarea operatorului +;
- metoda publica, realizata prin supraincarcarea operatorului -, care sa considere doua cozi și sa elimine, concomitent, elementele din ambele cozi adaugand caracterul ce are codul ASCII mai mare într-o noua coada, ca în exemplul de mai jos:

Coada_de_caractere C1,C2;
C1 = {E,X,A,M,E,N}; C2 = {P,O,O,L,A,B,O,R,A,T,O,R} C1 - C2 = {P,X,O,M,E,N}.

Tema 6. Clasa ”Lista_circulara” (implementata dinamic)
Se considera ***Class Nod{ int info; Nod* next;}***

- constructori de inițializare și parametrizati pentru clasa Nod;
- destructor pentru clasa Nod;

Clasa Lista_circulara are:

- cel puțin un membru privat „Nod*” reprezentând nodul considerat prim;
- metoda publica de adaugare a unui element pe o poziție;
- supraincarcare a operatorului >>, realizata prin utilizarea succesiva a metodei decarata anterior;

- metoda publica de stergere a unui element de pe o poziție;
- metoda publica pentru inversarea legaturilor listei;
- metoda publica care primește parametrul întreg k și realizatea eliminarea elementelor listei circulare din k în k pana la golirea acesteia (elementele vor fi afisate în ordinea în care sunt eliminate);
- supraincarcarea operatorului +, care sa efectueze concatenarea a doua liste circulare, rezultand într-o noua lista circulara (ca în exemplul de mai jos).

Lista_circulara L1 = { 1 → 2 → 3 → 1} , L2 = {4 → 5 → 6 → 4}
L1 + L2 = { 1 → 2 → 3 → 4 → 5 → 6 → 1}

Tema 7. Clasa "Lista_dublu_inlantuita"
Se considera ***Class Nod{ int info; Nod* prev, next;}***
- constructori de inițializare și parametrizati pentru clasa Nod;
- destructor pentru clasa Nod;

Clasa Lista_dublu_inlantuita are:
- doi membri privati „Nod*" reprezentând primul și ultimul element;
- metoda publica de adaugare a unui element pe o poziție;
- supraincarcare a operatorului >>, realizata prin utilizarea succesiva a metodei decarata anterior;
- supraincarcare a operatorului << pentru afisarea listei in ambele sensuri, in aceeasi funcție;
- metoda publica de stergere a unui element de pe o poziție;
- supraincarcarea operatorului +, care sa efectueze concatenarea a doua liste dublu inlantuite, rezultand într-o noua lista dublu inlantuita.

Tema 8. Clasa "Multime" (multimi finite de numere intregi reprezentate ca tablouri unidimensionale)
- membri privati pentru vectorul propriuzis si numarul de elemente;
- constructori pentru initializare si copiere;
- destructor (în cazul alocarii statice, se seteaza dimensiunea vectorului la zero, iar în cazul alocarii dinamice, se dezaloca zona de memorie utilizata);
- metoda publica pentru transformare a unui vector in multime, prin eliminarea duplicatelor din respectivul vector;
- reuniune a doua multimi, implementata prin supraincarcarea operatorului +;
- intersectie a doua multimi, implementata prin supraincarcarea operatorului *;
- diferenta a doua multimi, implementata prin supraincarcarea operatorului -.

Tema 9. Clasa "Multime_dinamic" (multimi finite de numere intregi reprezentate ca liste inlantuite). Cerintele sunt aceleași ca la Tema 8, adaptate pentru liste alocate dinamic.

Tema 10. Clasa "Polinom" - reprezentat ca tablou unidimensional (prin gradul polinomului si vectorul coeficientilor (coeficientii vor apartine la monoame de grade consecutive), de dimensiune egala cu gradul plus 1, de la cel al termenului liber la cel de grad maxim.
- fiecare coeficient va fi de tip double;
- membri privati pentru vectorul propriuzis si numarul de elemente;
- constructori pentru initializare si copiere;

- destructor (în cazul alocarii statice, se seteaza dimensiunea vectorului la zero, iar în cazul alocarii dinamice, se dezaloca zona de memorie utilizata);
- metoda publica pentru calculul valorii unui polinom intr-un punct;
- suma a doua polinoame, implementata prin supraincarcarea operatorului +;
- diferenta a doua polinoame, implementata prin supraincarcarea operatorului -;
- produsul a doua polinoame, implementat prin supraincarcarea operatorului *;

Tema 11. Clasa "Polinom_dinamic" - reprezentat ca lista inlantuita (ca polinoame rare, prin lista perechilor (coeficient,exponent), ordonata crescator dupa exponent, si nu neaparat cu exponentii consecutivi.
Cerintele sunt aceleași ca la Tema 10, adaptate pentru liste alocate dinamic.

Tema 12. Clasa „student" avand // usor

- membri privati pentru nume(string), anul nasterii (intreg), numar_credite (intreg), medie_generala (double)
- constructor de initializare, destructor
- metode publice pentru setare/furnizare informatii pentru fiecare atribut
- supraincarcare pe operator >> si << pentru citirea si afisarea studentilor

Definiti clasa „grupa" avand

- ca membri privati un vector la „student" declarat dinamic, un intreg reprezentand numarul studentilor si un double reprezentand media generala a intregii grupe
- constructor de initializare, destructor care dezaloca toti studentii pointati in componentele vectorului
- eliminare student din grupa
- adaugare student in grupa
- verificare daca exista un student dupa nume

Cele doua clase de mai sus pot avea si alti membri/metode, dar se va respecta principiul incapsularii datelor (orice setare/furnizare a informatiilor continute in membrii privati se va face doar prin intermediul unor metode).

Tema 13.
D1. Definiti clasa "persoana", avand: // usor spre mediu

- membri privati pentru nume (string), anul nasterii (intreg), sex (caracter);
- sase metode publice pentru setarea/furnizarea informatiilor din cei trei membri privati.

Definiti clasa "baza_de_date" avand

- ca membri privati un vector la "persoana" declarat dinamic si un intreg reprezentand numarul persoanelor;
- ca metode publice un constructor care initializeaza vectorul cu NULL pe fiecare componenta, un destructor care dezaloca toate "persoanele" pointate de componentele vectorului, o metoda pentru adaugat o "persoana", trei metode pentru eliminat persoanele avand un anumit nume, respectiv an al nasterii, respectiv sex (cele trei metode vor avea acelasi nume, prin supraincarcare), doua metode pentru afisat persoanele continute in baza de date in ordinea alfabetica a numelor, respectiv crescatoare a varstelor.

Cele doua clase de mai sus pot avea si alti membri/metode, dar se va respecta principiul incapsularii datelor (orice setare/furnizare a informatiilor continute in membrii privati se va face doar prin intermediul unor metode).

Scrieti un program care sa gestioneze o baza de date folosind clasele de mai sus. El va afisa un meniu (in mod text), care va oferi functii de adaugat o persoana, eliminat persoanele dupa nume, varsta, sex, afisat
persoanele in ordine alfabetica sau dupa varsta.

Tema 14.
D5 Clasa "vector" (vector de double), avand: // usor

- membri privati pentru vectorul propriuzis si numarul de elemente;
- constructor pentru initializarea cu un numar dat pe toate componentele (primeste ca parametru numarul respectiv si numarul componentelor);
- constructori pentru initializare si copiere;
- metode publice pentru citire si afisare;
- metoda-operator public de atribuire =;
- metoda publica pentru reactualizarea numarului de componente si initializarea componentelor cu un numar dat (primeste ca parametru numarul respectiv si numarul componentelor);
- operatori friend: + (suma element cu element), - (diferenta element cu element), +=, -= (cu efectul cunoscut), == (testarea egalitatii), != (testarea neegalitatii),
- functie friend "length" care furnizeaza numarul de elemente.

Actiunile vor fi implementate astfel ca atunci cand nu este posibila operatia (de exemplu adunarea unor vectori de dimensiune diferita) sa se afiseze un mesaj de eroare.

Program care citeste mai multi vectori si calculeaza suma lor si maximul lor.

Tema 15.
D6 Clasa "matrice" (matrice de double), avand: // usor

- membri privati pentru matricea propriuzisa, numarul de linii si numarul de coloane;
- constructor pentru initializarea cu un numar dat pe toate componentele (primeste ca parametru numarul respectiv si numarul de linii si de coloane);
- constructori pentru initializare si copiere;
- metode publice pentru citire si afisare;
- metoda-operator public de atribuire =;
- metoda publica pentru reactualizarea numarului de linii si coloane si initializarea componentelor cu un numar dat (primeste ca parametru numarul respectiv, numarul liniilor si al coloanelor);
- operatori friend: + (suma), - (diferenta), * (produsul dintre doua matrici si dintre o matrice si un intreg), == (testarea egalitatii),
- functii friend "nrlinii", "nrcoloane", "nrelemente" care furnizeaza numarul liniilor, coloanelor, respectiv elementelor matricii;

Program pentru rezolvarea unei ecuatii matriciale de forma "A*X+B=0" (A,B,X,0 sunt matrici).

Tema 16.
D10 Clasa "individ", avand: // mediu spre greu

- membrii privati: i (intreg) = pozitia (dintr-un tablou unidimensional de 30 de elemente declarat static in main);
  tip (char) = „numele” speciei ('+' sau '0');

varsta (intreg) = varsta (de la 0 la o valoare maxima fixa aleasa de programator);
energie (double) = energia;
viu (unsigned char) = este 1 daca e viu si 0 daca e mort;

- constructor care primeste ca parametrii pozitia si tipul si initializeaza corespunzator membrii respectivi; varsta se initializeaza cu 0, energia cu o valoare fixa aleasa de programator, viu cu 1;

- metode private:
hraneste = creste energia proprie cu o valoare random intre 1 si 10;
inmulteste = adauga un fiu in stanga sau in dreapta individului daca pozitia este libera (fiul are acelasi tip, varsta este 0, iar energia este de doua ori mai mare fata de cea a parintelui)
ataca = pentru fiecare individ x invecinat si de alt tip, daca energia proprie este mai mare decat energia lui x, din energia proprie se scade energia lui x iar x este omorat (i se aplica metoda "***moare***");
imbatraneste = creste varsta cu 1; scade energie cu o valoare constanta aleasa de programator; daca a atins o varsta maxima aleasa de programator sau daca energia sa a devenit <=0, individul curent este omorat (i se aplica metoda "***moare***");
moare = viu devine 0;

- metode publice:
actualizare = aplica succesiv metodele: hraneste, inmulteste, ataca, imbatraneste;
esteviu = returneaza 1 daca viu==1 si 0 altfel;
gettip = returneaza tipul.

Se citesc maxim 30 de indivizi (supraincarcare pentru >> si <<).

Realizare meniu cu functia de: citire indivizi (se citeste si numarul de indivizi aici), actualizare, este viu, get tip.

Tema 17. Clasa „carte”

-membri privati: denumire carte, autor principal, al doilea autor, numar de pagini, pret, rating (goodreads).

-metode: constructori, destructor, supraincarcare pe >> si <<, constructor de copiere, supraincarcare pe =, supraincarcare pe operatorii specifici de comparare (compararea se face pe baza ratingului).

Sa se citeasca de la tastatura un vector declarat dinamic de carti, sa se poata compara doua carti, sa se poata modifica informatiile unei carti din meniu si sa se afiseze toate cartile.

Tema 18. Clasa „film”

La fel ca tema 17 doar ca avem membri privati: denumire film, gen, regizor, rating (imdb).

Tema 19. Clasa „teatru”

La fel ca tema 17 doar ca avem membri privati: denumire_piesa, numar actori, actori, rating.

Membrul „actori” este un vector dinamic de obiecte din clasa actor ce contine ca membri privati: nume actor (string), varsta(int).