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

Cerinte comune tuturor temelor:

- implementare in C++ folosind clase
- **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 o metoda publica prin care se realizeaza citirea informațiilor complete a n obiecte, memorarea și afisarea acestora.

Cerinte specifice fiecarei teme:

Tema 1. Clasa ”Numar_Complex”

- 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 /.
- metoda publica **sqrt** pentru furnizarea radicalului de ordinul 2 al unui complex.

Să se realizeze un program de rezolvare a ecuatiei de ordinul doi cu coeficienti complecsi.

Tema 2. Clasa ”Fractie”

- membrii privati pentru numarator si numitor (int);
- 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 numaratorul si numitorul;
- metoda publica pentru simplificare;
- metoda publica de afisare (sub forma "a", "a/b", "-a/b", dupa caz);
- metode publice pentru suma, diferența, produsul si împărțirea a doua numere raționale, implementate prin supraincarcarea operatorilor +, -, *, /
- metode publice pentru inmultirea unui numar rațional cu un numar întreg, realizata prin supraincarcarea operatorlui *.

Să se realizeze un program de rezolvare a unui sistem de doua ecuatii liniare cu doua necunoscute si coeficienti rationali.

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

- 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;
- produsul scalar a doi vectori de aceeasi lungime, implementat prin supraincarcarea operatorului *.

Tema 4. Clasa ”Vector_Complex”

- clasa este prietena a clasei **Numar_Complex** definita în **Tema 1**;
- 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);
- 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;
- metoda publica pentru a calcula produsul scalar a doi vectori de aceeasi lungime, care sa foloseasca suma si produsul a doi complecsi din clasa numar complex, si sa supraincarce operatorul *.

Tema 5. Clasa ”Matrice_Complexa”

- clasa este prietena a clasei **Numar_Complex** definita în **Tema 1**;
- membri privati pentru matricea propriuzisa, 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 6. Clasa ”Vectori_de_vectori”

Se considera ***Class Vectori_de_vectori { int dim; Vector *v;};***
- clasa este prietena a clasei **Vector** definita în **Tema 3**;
- 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 Vector;
- metoda publica prin care se creeaza o matrice de numere întregi cu nr_linii = dim și nr_coloane = maximul dimensiunilor vectorilor declarati, elementele matricei fiind cele declarate în vectori, completandu-se cu zerouri în cazul în care dimensiunea unui vector linie e mai mica decât nr_coloane.
- metoda publica prin care să se calculeze suma a doua obiecte de tip Vectori_de_vectori, realizata prin supraincarcarea operatorului + rezultatul fiind dat sub forma de matrice, ca în exemplul de mai jos:

Vector_de_vectori A,B;

```
A = { {1,2}              B = { {4}              A + B = 5  2  0  0
      {1,2,3} }                {4,5,6,2}                5  7  8  2
                               {3,8} }                  3  8  0  0.
```

Tema 7. 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 8. 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 9. 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 10. 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 11. 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 12. Clasa ”Multime_dinamic” (multimi finite de numere intregi reprezentate ca liste inlantuite). Cerintele sunt aceleași ca la Tema 11, adaptate pentru liste alocate dinamic.

Tema 13. 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 14. 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 13, adaptate pentru liste alocate dinamic.

Tema 15. Clasa ”ABC”- arbori binari de cautare, in reprezentare inlantuita.
Se considera ***Class Nod{ int info; Nod* st, dr;}***

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

Clasa ”ABC” are:
- membru privat de tip „Nod*” reprezentând rădăcina arborelui;
- metode publice pentru inserarea unui element, care sa supraincarce operatorul +, care va fi aplicat intre un numar reprezentand elementul nou introdus si un arbore;
- supraincarcarea operatorului >>, prin inserari succesive;

- supraincarcarea operatorului <<, care sa afiseze arborele în toate cele 3 metode (preordine, inordine, postordine;
- metoda publica pentru stergerea unui element;
- metoda publica pentru determinarea inaltimii arborelui;
- metoda publica pentru afisarea listei frunzelor.

Tema 16. Clasa ”Graf_neorientat”- reprezentate cum doreste programatorul.
- constructori de inițializare și de copiere;
- destructor;
- metoda publica pentru afisarea grafului, care sa supraincarce operatorul << și sa ilustreze toate modalitatile de reprezentare a grafului;
- parcurgerea in latime;
- parcurgerea in adancime;
- determinarea matricii (existentei) drumurilor;
- determinarea componentelor conexe nu ca grafuri, ci ca liste de noduri;
- o metoda care sa determine daca graful este conex, care poate folosi oricare dintre metodele anterioare;
- o metoda de supraincarcare a operatorului +, care sa determine, din doua grafuri neorientate avand aceeasi multime de noduri, graful cu aceeasi multime de noduri ca si acele doua grafuri, si cu multimea muchiilor egala cu, reuniunea multimilor muchiilor acelor doua grafuri.

Tema 17. Clasa ”Graf_ponderat”- a grafuri neorientate cu ponderi atasate muchiilor, in ce mod doreste programatorul.
- constructori de inițializare și de copiere;
- destructor;
- metoda publica pentru afisarea grafului, care sa supraincarce operatorul << și sa ilustreze toate modalitatile de reprezentare a grafului;
- parcurgerea in latime;
- parcurgerea in adancime;
- determinarea matricii ponderilor drumurilor cu ponderi minime;
- determinarea nodurilor intermediare de pe drumul de pondere minima intre doua noduri;
- o metoda care sa determine daca graful este conex, care poate folosi oricare dintre metodele anterioare;
- o metoda de supraincarcare a operatorului *, care sa determine, din doua grafuri ponderate avand aceeasi multime de noduri, graful ponderat cu aceeasi multime de noduri ca si acele doua grafuri, si cu multimea muchiilor egala cu intersectia multimilor muchiilor celor doua grafuri, cu fiecare muchie avand ca pondere minimul dintre ponderile muchiilor corespunzatoare din acele doua grafuri.

Tema 18. Clasa ”Numar_intreg_mare”- numere intregi mari (reprezentate ca indicator de semn si liste dinamice de cifre, incepand cu cifra unitatilor - vor fi afisate uzual, incepand cu cifra dominanta si incheind cu cifra unitatilor):
- clasa numar intreg mare sa contina metode pentru supraincarcarea operatorilor << si >> pentru iesiri si intrari, precum si pentru calculul: sumei a doua numere intregi mari, prin supraincarcarea operatorului +, diferentei dintre doua numere intregi mari, prin supraincarcarea operatorului -, produsului dintre doua numere intregi mari, prin supraincarcarea operatorului *, maximului dintre valorile absolute a doua numere intregi mari;
- sa se creeze o clasa vector de numere intregi mari, prietena a clasei numar intreg mare, care sa contina metode pentru: supraincarcarea operatorilor << si >>,

folosind metodele de supraincare pentru << si >> din clasa numar intreg mare; produs scalar dintre doi vectori de numere intregi mari cu acelasi numar de elemente, care sa supraincarce operatorul * si sa foloseasca produsul si suma din clasa numar intreg mare; calculul valorii absolute maxime dintr-un vector de numere intregi mari, folosind maximul dintre valorile absolute a doua numere intregi mari preluat din clasa numar intreg mare.

Tema 19. Clasa ”Numar_rational_mare”- numere rationale mari (reprezentate ca perechi de numere intregi mari, fiecare element al unei astfel de perechi fiind reprezentat ca la proiectul 18 de mai sus, elementele unei astfel de perechi reprezentand respectiv numaratorul si numitorul fractiei care defineste numarul rational mare):

- sa se creeze o clasa numar intreg mare, cu supraincarcari pentru operatorii: << si >> pentru iesiri si intrari; + pentru suma a doua numere intregi mari; * pentru produsul a doua numere intregi mari; - pentru diferenta a doua numere intregi mari; / si respectiv % pentru catul si restul impartirii intregi a doua numere intregi mari; sa mai contina si o metoda pentru calculul celui mai mare divizor comun al valorilor absolute a doua numere intregi mari (sugestie: sa se aplice aici algoritmul lui Euclid, utilizand metodele pentru determinarea catului si a restului impartirii intregi);

- clasa numar rational mare sa fie clasa prietena a clasei numar intreg mare, iar metodele ei sa apeleze metodele necesare din clasa numar intreg mare; clasa numar rational mare sa contina metode de supraincarcare pentru operatorii: << si >> pentru iesiri si intrari de obiecte; + pentru suma a doua numere rationale mari; * pentru produsul a doua numere rationale mari; sa contina si o metoda pentru scrierea unui numar rational mare ca fractie ireductibila, prin impartirea numaratorului si numitorului la cel mai mare divizor comun al valorilor lor absolute.

Tema 20. Clasa ”Numar_real_mare”- numere reale mari (reprezentate ca perechi de numere intregi mari, astfel incat valoarea unui numar real mare reprezentat astfel sa fie egala cu primul numar intreg mare din pereche inmultit cu 10 la puterea al doilea numar intreg mare din pereche; fiecare element al unei astfel de perechi sa fie reprezentat ca la proiectul 18 de mai sus):

- clasa numar intreg mare sa contina: supraincarcari pentru operatorii << si >> pentru iesiri si intrari de obiecte; supraincarcari pentru operatorii + si * pentru adunarea a doua numere intregi mari si respectiv produsul a doua numere intregi mari; o metoda pentru determinarea maximului dintre valorile absolute a doua numere intregi mari; o metoda care: aplicata lui 0 ca obiect al clasei numar intreg mare, sa intoarca 0 si sa nu modifice obiectul 0, iar, aplicata la un numar intreg mare nenul, sa elimine toate zerourile consecutive de la sfarsitul acelui numar intreg mare nenul, adica dintre cifrele cele mai putin semnificative, si sa intoarca numarul de zerouri pe care le-a eliminat (acel numar de zerouri reprezinta puterea naturala maxima a lui 10 care divide acel numar intreg mare, iar obiectul modificat astfel reprezinta catul impartirii obiectului initial la acea putere a lui 10);

- clasa numar real mare sa fie clasa prietena a clasei numar intreg mare si sa contina urmatoarele metode, care sa apeleze metodele necesare din clasa numar intreg mare: supraincarcari pentru operatorii << si >> pentru iesiri si intrari de obiecte; scrierea unui numar real mare nenul astfel incat primul numar din perechea care il reprezinta sa nu aiba zerouri consecutive la sfarsit, adica la cifrele cel mai putin semnificative; supraincarcari pentru operatorii + si * pentru adunarea a doua numere reale mari si respectiv produsul a doua numere reale mari; o metoda pentru determinarea maximului dintre valorile absolute a doua numere reale mari.