**Probleme POO**

1. De la tastatură se citeşte o matrice de valori reale. Să se afişeze pe ecran matricea, astfel încât fiecare element al matricei să fie scris aliniat la dreapta, pe opt caractere şi cu două zecimale exacte.

2.23 1.23 512.15

11.45 111.45 10.24

1. Să se aloce dinamic memorie pentru un vector de vectori de elemente de tip double cu următoarea proprietate: primul vector are un element, al doilea are două elemente, în general al k-lea vector are k elemente, k{1, 2, ..., n}. Să se citească de la tastatură n (numărul de vectori) precum şi elementele vectorilor. Să se construiască un nou vector format cu mediile aritmetice ale celor n vectori. În final să se elibereze toate zonele de memorie alocate dinamic.
2. Sa se scrie o functie care primeste ca parametri de intrare un vector de numere intregi si numarul de elemente ale acestui vector si returneaza, ca parametri de iesire doi vectori, alocati dinamic si numarul de elemente ale acestora, fiecare dintre ei continand numerele pare si, respectiv impare din vectorul de intrare. In programul principal se citeste numarul de elemente ale vectorului de intrare si elementele acestuia, se apeleaza functia mentionata mai sus si se afiseaza cei doi vectori de numere pare si, respectiv, impare.
3. Scrieti functii sablon pentru:

- determinarea minimului dintre trei valori numerice;

- alocarea dinamica a unui vector de valori;

- determinarea maximului dintr-un vector de valori numerice;

- sortarea unui sir prin metoda merge sort;

- produsul a doua matrici;

- ridicarea la putere a unei matrici.

1. Scrieţi funcţii pentru introducerea unui element într-o stivă de caractere, scoaterea unui element din stivă, afişarea conţinutului stivei şi eliberarea memoriei ocupate de stivă. Stiva se va memora dinamic folosind pointeri către tipul struct.
2. Scrieti un program care sa afiseze string-ul “POO” de N ori (N - constant) fara a folosi instructiuni repetitive, functii recursive sau goto.
3. Sa se implementeze o clasa cerc care are ca membri raza și centrul acestuia. Adaugați metode necesare clasei cerc: getter-i, setter-i, calcul perimetru, calcul arie, etc.
4. Sa se implementeze o clasa dreptunghi care are ca membri doua puncte care reprezinta coltul din dreapta sus a dreptunghiului si respectiv coltul din stanga jos. Adaugati metode necesare clasei dreptunghi: getter-i, setter-i, calcul perimetru, calcul arie, etc.
5. Scrieti o clasa Angajat cu membrii: ID, nume, salariu. Implementati metodele: getter-I, setter-i, crestere salariu, scadere salariu.
6. Implementati clasa rational. Supraincarcati operatorii +, -, ++(preincrementare), ++(postincrementare), \*, /, <<, >>, <, >, =, ==. Se citesc de la tastatura un sir de n numere rationale. Sa se scrie functii care determina numitorul comun numerelor rationale din sir, cel mai mic si el mai mare numar rational din sir.
7. Sa se implementeza clasa Multime. Clasa trebuie sa aiba ca membri un vector de n, elemente intregi, distincte, alocat dinamic si, n, numarul de elemente ale multimii. Pentru aceasta clasa supraincarcati operatorii += pentru a adauga un nou element in multime si pentru a reuni multimea curenta cu o alta multime, -= pentru a scoate un element din multime (daca acesta exista in multime) si pentru a lasa in multime doar elementele care nu apartin multimii cu care se face diferenta. Sa se implementeze, de asemenea operatorii + si - pentru reuniunea si respectiv diferenta dintre 2 multimi. Testati toate functiile implementate.
8. Implementati clasa Matrice care are ca membri dimensiunile si elementele matricii. Sa se supraincarce operatorii +, -, \* pentru adunarea, scaderea si respectiv inmultirea a doua matrici. De asemenea sa se implementeze operatorul ^ pentru ridicarea la putere a unei matrici patratice si operatorii -=, +=, \*=. in programul principal sa se verifice tori operatorii implementati.
9. Implementati clasa Coada cu liste inlantuite. Implementati toate metodele necesare in clasa coada (enQueue, deQueue, isEmpty, …). In programul principal testati functionalitatea clasei implementate.
10. Implementați clasa Angajat(nume (char\*), prenume(char\*), salariu(float)) în care numele si prenumele se vor aloca dinamic. Sa se supraincarce operatorul < pentru a compara doi angajati dupa nume, operatorul + pentru a aduna la salariul angajatului un procent la salariu, operatorul - pentru a scadea salariul angajatului cu o anumita valoare (astfel incat salariul sa-I ramana >= 0), operatorii << pentru afisarea datelor despre un angajat. In programul principal sa se citeasca de la tastatura un sir de persoane (fiecare cu nume, prenume si salariu). Sa se sorteze alfabetic persoanele. Cresteti salariul primei persoane din lista cu 20% si scadeti salariul ultimei persoane cu 10 lei;
11. Sa se defineasca o clasa Vector3D, care are ca membri componentele x , y si z. Sa se supraincarce operatorul ! pentru determinarea lungimii vectorului (!v = ), operatorul < pentru compararea a doi vectori dupa lungime, operatorul + pentru adunarea a doi vectori, operatorul \* pentru produsul dintre un vector si un scalar (c\*v =(c \* x, c \* y, c \* z), operatorul \* pentru produsul scalar dintre doi vectori (v1\*v2 = x1 \* x2 + y1 \* y2 + z1 \*z2), operatorii << si >> pentru citirea si afisarea vectorilor. Sa se citeasca de la tastatura un sir de n obiecte Vector3D, sa se afiseze produsul scalar dintre vectorii cu lungimea cea mai mare si, respectiv, cu lungimea cea mai mica si sa se afiseze lungimea vectorului suma: v = 1\*v1 + 2\*v2+ 3\*v3 + …..+ n\*vn.
12. Sa se implementeze o clasa Pixel, care are ca membri coordonatele varfului x , y si culoarea c - un numar intre 0 si 255. Sa se supraincarce operatorul + a unui pixel cu un intreg; d + pixel1 = pixel2 (pixel2.x = pixel1.x+d, pixel2.y = pixel1.y+d, pixel2.culoare = pixel1.culoare), operatorul ! pentru determinarea distantei pixelului pana la origine ( d = sqrt(pixel.x \*pixel.x + pixel.y \* pixel.y)), operatorul < pentru compararea culorii a doi pixeli, (pixel1 < pixel2 daca pixel1.culoare < pixel2.culoare, operatorii << si >> pentru citirea si afisarea pixelilor. Sa se citeasca de la tastatura un sir de n obiecte Pixel si sa se adune cu a (a este un numar citit de la tastatura) pixelii care au culoarea un numar impar cu - a pixelii care are culoarea un numar par. Sa se afiseze pixelul care se afla la distanta cea mai mica de origine.
13. Implementaţi clasa Persoana care contine numele, prenumele si varsta unei persoane. Sa se supraincarce operatorul < pentru compararea a doua persoane dupa nume si operatorii << si >> pentru afisarea si, respectiv, citirea unei persoane de la tastatura. Din clasa Persoana sa se derive clasele Angajat, care are campul salariu, si clasa Student, care contine campul medie. Sa se supraincarce operatorii << si >> pentru clasele angajat si student. Sa se citeasca de la tastatura un sir de Persoane (studenti si angajati si sa se sorteze in ordine alfabetica). Dupa sortare, sa se creasca salariul tuturor angajatilor cu 10% si sa se afiseze toti studentii cu prenumele x (x citit de la tastatura).
14. Să se implementeze o clasă șablon pentru lucrul cu matrice pătratică. Clasa va

avea două câmpuri:

* - ordin, de tip întreg;
* - elemente, de tip generic (se va aloca dinamic memoria pentru matrice).

Se vor supraîncărca diverși operatori:

* << pentru afișare, >> pentru citire;
* = pentru atribuire;
* == și ! = pentru comparare;
* + pentru sumă, − pentru diferență și ∗ pentru produs;
* +=, −= şi ∗=;
* [ ] pentru accesarea unei linii din matrice;
* ! pentru calculul determinantului;
* ^ pentru ridicarea la putere.
* ~ pentru transpusa matricii.

Să se scrie o functie principala care citeste de la tastatura o matrice patratica de numere reale si un numar natural ‘n’. Se cere sa se afiseze pe ecran matricea At + (At) 2 + … + (At) n.

1. Să se implementeze o clasă în C++ numită BigInt care poate gestiona numere întregi foarte mari (mai mari decât tipul de date standard int sau long long). Clasa trebuie să suporte operațiile de bază: adunare, scădere, înmulțire și împărțire, rest. Clasa contine un array de cifre, alocat dinamic si numarul total de cifre. In clasa trebuie sa se implementeze:

* Constructori **și Destructor**
* Operatorii:
* +
* -
* \*
* /
* % - pentru restul impartirii dintre doua numere mari
* ! - pentru factorialul numarului
* <<
* >>
* Orice alte metode necesare pentru a sprijini funcționarea internă a clasei (de exemplu, comparatori, metode private de manipulare a datelor etc.)

In programul principal, dintr-un fisier text sa se citeasca un sir de n numere mari si sa se afiseze intr-un fisier binar rezultatul operatiei:

Ex fisier:

3

6655512677417499441

4858538463887799363

155663335859663252556

1. Implementati o clasa *CutiePraline*, care va conține câmpurile private *aroma* (String) și *origine* (String). Clasa va avea, de asemenea:

* un constructor fără parametri
* un constructor ce va inițializa toate campurile
* o metoda de tip float getVolume(), care va intoarce valoarea 0;

Din aceasta clasa derivați clasele *Lindt, Baravelli, Raffaello.* Pentru un design interesant, cutiile vor avea forme diferite:

* Lindt va contine length, width, height (float);
* Baravelli va fi un cilindru. Acesta va conține un camp radius și unul height (float);
* *Raffaello*, fiind un cub, va conține un camp length (float);

Clasele vor avea:

* constructori fără parametri
* constructori care permit inițializarea membrilor. Identificați o modalitate de reutilizare a codului existent. Pentru fiecare tip de cutie veti initializa, in constructor, campurile flavor și origin cu tipul corespunzator

Suprascrieti metoda getVolume() pentru a intoarce volumul specific fiecarei cutii de bomboane, in functie de tipul sau.

Suprascrieti metoda toString() în clasele derivate, astfel încat aceasta să utilizeze implementarea metodei toString() din clasa de bază. Returnați un mesaj de forma “Cutia ” + *origine* + “ are aroma ” + flavor + “ si volumul ” + volume;