**Lista temelor**, cu toate cerintele pentru **a doua lucrare practica**.

* Cerinte comune tuturor temelor:

1. toate clase vor conține obligatoriu constructori de inițializare, parametrizati și de copiere, destructor, iar operatorii >>,<<,= să fie supraincarcati.
2. ilustrarea conceptelor de moștenire și functii virtuale (pure – unde se considera mai natural)
3. citirea informațiilor complete a n obiecte, memorarea și afisarea acestora

* cerinte specifice fiecarei teme:

**Tema 1.** liste de numere intregi, reprezentate ca tablouri uni- si bidimensionale:

Se dau urmatoarele clase:

- Vector (int dim, int \*a)

- Matrice (Vector \*v) – clasa abstracta

- Matrice\_oarecare (int lin):Matrice

- Matrice\_patratica (int dim): Matrice – nr de linii (dim) trebuie sa

coincida cu dimensiunea oricarui vector component

Clasele derivate trebuie sa contina cel puțin constructori parametrizati prin

care sa se evidentieze transmiterea parametrilor catre constructorul din clasa

de baza, destructori si o metoda care sa verifice daca o matrice triunghiulara

este diagonala. Pentru matricile patratice, functia de afisare sa conțină și

determinantul acestora.

**Tema 2**. vectori de numere complexe

Se dau urmatoarele clase:

- Complex (float re,im)

- Vector (int dim, Complex \*v).

- Stiva : Vector - cu operatiile de adaugare și stergere modificate corespunzator.

- Coada: Vector - cu operatiile de adaugare și stergere modificate corespunzator.

Clasele derivate trebuie sa contina cel puțin constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza, destructor si o metoda prin care sa verifice dacă stiva / coada este pur "imaginara".

**Tema 3.** Cozi de caractere (implementate dinamic):

Se dau urmatoarele clase:

- Nod (char info, nod\*next)

- Coada (nod \* prim, nod \* ultim, int dim\_max);

- DEQUE:Coada - cu operatiile de adaugare și stergere modificate corespunzator.

- DEQUE\_MARCAJ:Coada - cu operatiile de adaugare și stergere modificate corespunzator.

Clasele derivate trebuie sa contina cel puțin constructori parametrizati prin

care sa se evidentieze transmiterea parametrilor catre constructorul din clasa

de baza si destructor.

**Tema 4.** liste de numere întregi (implementate dinamic)

Se dau urmatoarele clase:

- Nod (char info, nod\*next)

- Nod\_dublu:Nod {nod \* ante;};

- Nod\_prioritate:Nod\_dublu {int prio;}

Să se implementeze clasa Coada\_prioritati care sa aibă elemente de tip

Nod\_prioritate. Clasele derivate trebuie sa contina cel puțin constructori

parametrizati prin care sa se evidentieze transmiterea parametrilor către

constructorul din clasa de baza si destructorI.

**Tema 5.** multimi finite de numere întregi reprezentate ca tablouri unidimensionale

Se dau urmatoarele clase:

- Clasa Pereche(int prim,doi)

- Multime\_pereche ( int n, Pereche \* p)

- Stiva\_pereche : Multime\_pereche – cu operatiile de adaugare și stergere modificate corespunzător;

- Coada\_pereche : Multime\_pereche – cu operatiile de adaugare și stergere modificate corespunzător;

Clasele derivate trebuie sa contina cel puțin constructori parametrizati prin

care sa se evidentieze transmiterea parametrilor catre constructorul din clasa

de baza si destructori. Să se implementeze o functie prin care se simuleaza

operatiile pe o stiva folosind 2 cozi.

**Tema 6.** polinoame reprezentate ca tablouri unidimensionale (prin gradul polinomului si vectorul coeficientilor.

Se dau urmatoarele clase:

- Clasa Monom(int grad, float coef)

- Clasa Polinom(int nr\_monoame, Monom \*m)

- Polinom\_ireductibil : Polinom.

Dacă vi se pare mai natural, puteți sa aveți clasa Polinom ca Și clasa de

baza pentru Polinom\_Ireductibil și pentru Polinom\_reductibil.

- Clasa Polinom(int nr\_monoame, Monom \*m) – clasa abstracta

- Polinom\_ireductibil : Polinom.

- Polinom reductibil:Polinom.

Afisarea unui polinom reductibil să fie făcută și ca produs de 2

polinoame.

Clasa derivata trebuie sa contina constructor parametrizat prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza, destructor si o metoda care sa aplice criteriul lui Eisenstein de verificare a ireductibilitatii polinoamelor.

(Webografie ajutatoare: [**http://www.profesoronline.ro/teorie-3140-1.html**](http://www.profesoronline.ro/teorie-3140-1.html)**)**

**Tema 7.** matrice de numere complexe reprezentate ca tablouri bidimensionale

Se dau urmatoarele clase:

- Clasa Complex(float re,im)

- Matrice( Complex \*\*v) – clasa abstracta

- Matrice\_oarecare (int lin, int col) : Matrice

- Matrice\_patratica (int dim): Matrice

Clasele derivate trebuie sa contina cel puțin constructori parametrizati prin

care sa se evidentieze transmiterea parametrilor catre constructorul din clasa

de baza, destructori si o metoda care sa verifice daca o matrice triunghiulara

este diagonala. Pentru matricile patratice, functia de afisare sa conțină și

determinantul acestora.

**Tema 8.** matrice de numere complexe reprezentate ca structuri inlantuite (ca matrici rare si nu neaparat patratice)

* Aceleași cerinte ca la Tema 7.

**Tema 9.** arbori binari de cautare in reprezentare inlantuita:

Se dau urmatoarele clase:

- Nod (int info, nod\*st, nod \*dr)

- Nod\_rosu\_negru:Nod {char culoare[]};

- Arbore(int nr\_noduri) – clasa abstracta

- ABC (Nod \*rad):Arbore

- Arbore\_bicolor (Nod\_rosu\_negru \*rad) : Arbore

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza, destructori si o functie virtuala prin care inaltimea arborelui bicolor sa fie identificata prin Adancimea neagra a radacinii.

Webografie: <https://writer.zoho.com/public/miha_cio/Arbori-bicolori1/noband>

**Tema 10.** arbori binari de cautare AVL in reprezentare inlantuita:

Se dau urmatoarele clase:

- Nod (int info, nod\*st, nod \*dr)

- Nod\_AVL:Nod {int echilibru};

- Arbore(int nr\_noduri) – clasa abstracta

- ABC (Nod \*rad):Arbore

- Arbore\_AVL(Nod\_AVL \*rad) : Arbore

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza, destructori si o functie virtuala prin care să se afiseze factorii de echilibru pentru fiecare nod.

**Tema 11.** arbori oarecare, in reprezentare inlantuita, prin legaturile fiu (catre fiul cel mai din stanga) si frate (catre urmatorul fiu al tatalui, in ordinea fiilor tatalui de la stanga la dreapta).

Se dau urmatoarele clase:

- Nod (int info)

- Nod\_ABC: Nod( Nod\*st, Nod\*dr)

- Nod\_fiu\_frate:Nod {int copii, Nod \* leg[10]};

- Arbore(int nr\_noduri) – clasa abstracta

- ABC (Nod \*rad):Arbore

- AB\_oarecare(Nod\_fiu\_frate \*rad) : Arbore

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza, destructori si o functie virtuala prin care afisarea elementelor unui arbore binar oarecare sa fie data de parcurgerea în latime, cu mentiunea listei fiilor pentru fiecare nod, iar afisarea unui ABC sa conțină cele 3 parcurgeri simultan.

**Tema 12.** grafuri

Se dau urmatoarele clase:

- Matrice (int \*\*a) – matrice de adiacenta

- Vector (int \*v, int dim)

- Lista (Vector \*l) – lista de adiacenta

- Graf (int nr\_noduri) – clasa abstracta

- Graf\_Neorientat(Lista L) : Graf

- Graf\_Orientat (Matrice A) :Graf

Clasele derivate trebuie sa contina constructor parametrizat prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza, destructor si o metoda care sa afiseze vectorul de tati, dacă pentru un Graf Orientat se verifica conexitatea lui și se da un nod de plecare pe post de rădăcina.

**Tema 13.**  grafuri orientate,

Se dau urmatoarele clase:

- Matrice (int \*\*a) – matrice de adiacenta

- Graf (int nr\_noduri) – clasa abstracta

- Graf\_complet (int nr\_muchii) : Graf

- Graf\_antisimetric( Matrice M) : Graf

- Graf\_turneu: Graf\_complet, Graf\_antisimetric

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa/clasele de baza, destructori si o metoda virtuala care sa afiseze arcele unui Graf\_antisiemtric, sau a unui Graf Turneu.

Obs: graf turneu – orientat complet si antisimetric. Un graf orientat se numeşte antisimetric dacă pentru oricare două vârfuri din graf x şi y dacă există arcul (x,y), atunci nu există arcul (y,x). Un graf orientat sau neorientat se numeşte complet dacă oricare două vârfuri din graf sunt adiacente.

**Tema 14:** Din clasa Nr\_Natural\_Mare să se deriveze clasa

Se dau urmatoarele clase:

- Vector (int \*a)

- Nr\_Natural\_Mare(int\_nrcif, Vector V)

- Numar\_Intreg\_mare(char semn):Nr\_Natural\_Mare

Să se implementeze operatiile uzuale. Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza si destructori.

**Tema 15.**

Se dau urmatoarele clase:

- Vector (int \*a)

- Nr\_Natural\_Mare(int\_nrcif, Vector V)

- Numar\_Fibonacci\_mare:Nr\_Natural\_Mare

Sa se implementeze ierarhia de clase mai sus mentionata, iar clasa derivata trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa baza, destructor si o functie virtuala care sa afiseze numarul Fibonacci, precum si descompunerea lui in suma de numere Fibonacci.

**Obs: Autorul Temelor 16 – final este Lect. Dr. Mihai Gabroveanu (**[**http://inf.ucv.ro/~mihaiug/courses/poo/labs/Probleme%20C++.pdf**](http://inf.ucv.ro/~mihaiug/courses/poo/labs/Probleme%20C++.pdf)**)**

**Tema 16.**

Se consideră un program care descrie organizarea personalului unei instituţii folosid clase derivate. O clasă numită Angajat deţine date şi funcţii referitoare la un angajat al instituţiei:

class Angajat{

char \* nume;

float salariu;

public:

Angajat();

Angajat(char \*n, int s, float sal);

Angajat(Angajat& r);

void display();

float getSalariu();

void setSalariu(float s);

};

Diferite categorii de angajaţi necesită date suplimentare faţă de cele definite în clasa Angajat, corespunzătoare postului pe care îl deţin. De exemplu, un şef de secţie (administator) este un angajat (deci sunt necesare toate datele care descriu această calitate) dar mai sunt necesare şi alte informaţii, de exemplu precizare secţiei pe care o conduce. Un administrator este un angajat, de aceea clasa Administrator se poate construi prin derivare din clasa Angajat astfel:

class Administrator : public Angajat {

int sectie;

public:

Administrator(const char \*n, float sal, int sec);

Administrator(Administrator& r);

~Administrator();

};

Sa se implementeze aceasta ierarhie de clase.

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza si destructori

**Tema 17.** Sa se implementeze un vector(tablou) in care se pot pastra obiecte de tipuri diferite definite de utilizator. Pentru aceasta se defineşte clasa Object ca o clasă de bază pentru toate tipurile derivate:

class Object{

public:

Object() {};

virtual ~Object() {};

virtual void display() = 0;

};

Clasa ObArray, defineşte un vector de pointeri de tip Object\* . Nu este necesar să fie limitată dimensiunea vectorului deoarece se asigură creşterea dimensiunii acestuia atunci când este necesar.

class ObArray : public Object {

Object \*\*p; // vector de pointeri

int size; // numar de elemente la un moment dat

int grows; // increment de creştere a dimensiunii

int dimens; // dimensiune vector

public:ObArray(int size=0,int grows, int dimens);//Constructor

~ObArray();//Destructor

void RemoveAll();//Elimina toate obiectele din vector

int getSize(); // Intoarce numarul de elemente din vector.

int add(Object\* x); //Adauga un element la vector.

int insertAt(int i, Object \*x);//Insereaza un element pe // o pozitie data

int removeAt(int i); //Elimina elementul de pe pozitia i

Object\* getAt(int i); //Intoarce elementul de la pozitia i

void display();// Afisare elementelor din tablou.

};

Sa se utilizeze aceste clase pentru a memora un tablou de Puncte si un tablou de numere Complexe.

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza si destructori

**Tema 18.**

Se dau urmatoarele clase:

- Clasa Persoana(id, nume)

- Clasa Abonat:Persoana(nr\_telefon)

- Clasa Abonat\_Skype: Abonat (id\_skype)

- Clasa Abonat\_Skype\_Romania (adresa\_mail) : Abonat\_Skype

- Clasa Abonat\_Skype\_Extern(tara) : Abonat\_Skype

Sa se construiasca clasa Agenda ce contina o lista de abonati si sa se supraincarce operatorul [ ](indexare)care returneaza abonatul cu numele precizat .

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza si destructori

**Tema 19.** Implementati urmatoarea ierarhie de clase:

class Localitate{

protected:

char \*denumire;

int cod;

long nr\_locuitori;

public:

......

virtual void display();

};

class Oras :public Localitate{

protected :

int nrBlocuri;

public:

........

};

class Capitala :public Oras{

protected:

char \*numePrefect;

public:

......

};

class Judet {

Localitate \*p;

int nrLoc;

public:

..........

};

Clasele derivate trebuie sa contina constructori parametrizati prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza si destructori.