Clase. Obiecte. Membri statici. Functii si clase friend. Probleme

Obiective:
-membri statici, exemple
-functii membre statice
-functii friend
-clase friend
-utilizare notiuni invatate pana acum

Membri statici ai claselor – a se vedea exemplele 3.1 si 3.2 din laboratorul 4

Functii membre statice pot accesa numai alti membri statici ai clasei, nu au pointerul this, nu pot supraincarca o functie statica cu o functie non-statica sau invers.

Exemplu:

```
class Operatii
{
static int a;
int b;
public:
      void setare(int i,int j)
      {
      a=i;
      b=j;
      }
      static void afisare();

/* int afisare() //error cannot overload static and non-static
                  //member functions with the same parameter types
      {
      return a+b;
      }  */
      int afisare(int i,int j)
      {          //am voie sa supraincarc daca am
                  // numar diferit de parametri
      return i+j;
      }
};
```

```
int Operatii::a;  //Defineste variabila globala
void Operatii::afisare()
{
cout<<"Membru static "<<a<<endl;
// cout<<b; //error illegal reference to non-static member
}

int main()
{
 Operatii ob1,ob2;
 ob1.setare(10,10);
 ob2.setare(20,20);
 Operatii::afisare();
 ob2.afisare();
 ob1.afisare();
 cout<<ob1.afisare(3,3);
}
```

Functii friend sunt acele functii care nu sunt membre ale aceleiasi clase si ele au acces la membrii privati declarati in acea clasa. Ele nu sunt membre ale clasei pentru care ele sunt friend, pot fi membre ale altei clase sau pot fi functii globale.

Sintaxa**:** friend tipul_functiei nume_functie(lista_argumente);

Exemplu:

```
class Problema
{
int a,b;
public:
      friend int suma(Problema x);
      explicit Problema(int i,int j);
};

Problema::Problema(int i,int j)
{
a=i;
b=j;
}
```

```
int suma(Problema obiect)
{
//aici avem acces la membri privati ai clasei Problema
     //deoarece a fost declarata functia ca fiind friend
return obiect.a+obiect.b;
}

int main()
{
Problema exemplu(3,4);
cout<<"aduna 3 cu 4 este "<<suma(exemplu);
}
```

Clase friend Spunem ca o clasa X este friend cu o clasa Y daca toti membrii clasei X sunt functii friend ale clasei Y. (adica toti membrii clasei X au acces la membrii din clasa Y).

Sintaxa: friend class X;

Exemplu:

```
class X;

class Y
{
    int a, b;
  public:
        Y(int i,int j)
        {
        a=i;
        b=j;
        }
    int f1()
      {
            cout<<"Mesaj: ";
            return a*b;
      }
     void f2(); //trebuie definita
     void convert (X x);
};
```

```
class X
{
    int c;
  public:
    void set (int f)
      {
            c=f;
       }
    friend class Y;
};

void Y::convert (X x) {
  a = x.c;
  b = x.c;
}

int main () {
  X obj1;
  Y obj2(2,5);
  cout<<obj2.f1()<<endl;
  obj1.set(4);
  obj2.convert(obj1);
  cout << obj2.f1();
  return 0;
}
```

Probleme:

1. **Definiti si implementati clasa Dreptunghi,** avand ca membri :

a) **Varianta 1:** Lungimea si Latimea si ca functii membre : setLungime, setLatime, getLungine, getLatime, Arie si Perimetru;

b) **Varianta2:** Lungimea si Latimea, un constructor implicit, un constructor explicit, un constructor de copiere, destructor, o metoda pentru calcularea **Ariei** unui dreptunghi, o metoda pentru calcularea **Perimetrului** unui dreptunghi. Folositi si pointerul *this*.

c) **Exemplificati pentru cel putin 3 obiecte.**

d) Implementati **o clasa friend Patrat** si calculati perimetrul patratului si diagonala principala a patratului, considerand ca latura a patratului Lungimea dreptunghiului**.**

e) Folosind un membru static **afisati numarul de obiecte** de tipul Dreptunghi si de tipul Patrat initializate**.**

2. **Definiti si implementati clasa Calendar** avand ca date membre: An, Luna, Zi, NumeZi (enumerarea de Luni pana Duminica) folosind

enum Zile{luni,marti,miercuri,joi,vineri};
Zile zi;

si ca functii membre :
-un constructor implicit, explicit, de copiere
-obtinerea datei curente
-modificarea datei curente
-incrementarea datei curente
-afisarea datei curente

Se va defini o functie prietena pentru citirea datei curente.

TEMA:

1. Proiectati si implementati o clasa Triunghi care sa permita efectuarea unor calcule si anume (cazurile triunghiurilor-oarecare, echilateral, dreptunghic, cate grade au unghiurile in fiecare caz, aplicarea a cel putin 3 teoreme (Pitagora, Thales, Teorema catetei, Teorema inaltimii, ...). Utilizati: constructor implicit, explicit, de copiere, destructor, functii statice si friend, membri statici si friend, const, mutable, pointerul *this*.

Folositi-va imaginatia si creati o functie friend la clasa Triunghi care sa foloseasca cativa membri ai clasei Triunghi.