Tema 1

Termen de predare: laboratorul din săptămâna a III-a

Obiective:

- Încapsularea (izolarea detaliilor implementării): *private*, *protected*, *public*
- Atribute şi metode: la nivel de instanţă (pointerul *this*)
- Atribute şi metode: la nivel de clasă
- Metode constante
- Apelul şi returul prin referinţă
- Controlul iniţializării: constructori şi destructori
- Alocarea de memorie: operatorii *new* şi *delete*
- Spaţii de nume
- Fluxuri de intrare/iesire: utilizarea consolei
- Polimorfism static: supraîncărcarea funcţiilor
- Polimorfism static: supraîncărcarea operatorilor (=, ==, !=, +=, -=, *=)
- Utilizare STL: *vector*

Cerinţe:

- Problemele trebuie prezentate pâna în săptămâna a 3-a inclusiv
- Fiecare problemă este notată cu maxim 10 puncte
- Toate problemele sunt obligatorii

Probleme:

1. Se dă următoarea clasă:

```
// file: complex.h

#ifndef COMPLEX
#define COMPLEX

class Complex
{
        double re, im;                          // partea reala si partea imaginara

public:
        Complex(double = 0, double = 0)         // constructor

        double real() const;                    // returneaza partea reala
        double imag() const;                    // returneaza partea imaginara
        double modul() const;                   // returneaza modulul (distanta fata de origine)

        void modificaRe(double = 0);            // modifica partea reala
        void modificaIm(double = 0);            // modifica partea imaginara
        void modifica(double = 0, double = 0);  // modifica partea reala si pe cea imaginara

        // returneaza distanta dintre doua numere complexe
        static double distanta(const Complex&, const Complex& );

        // returneaza distanta dintre obiectul curent si argument
        double distanta(const Complex&) const;

        // operatori supraincarcati
        Complex& operator+=(const Complex&);
        Complex& operator+=(double);
```

```
        Complex& operator*=(const Complex&);
        Complex& operator*=(double);
}

// operatori supraincarcati

ostream& operator<<(ostream& , const Complex& );
istream& operator>>(istream&,  Complex&);

const Complex operator+(const Complex&, const Complex&);
const Complex operator*(const Complex&, double);
const Complex operator*(double, const Complex&);

bool  operator==(const Complex&, const Complex&);
bool  operator!=(const Complex&, const Complex&);

#endif
```

Implementaţi metodele propuse şi testaţi-le intr-un program!

2. Se dă următoarea clasă:

```
// file punct.h

#ifndef PUNCT
#define PUNCT

class Punct
{
        int X, Y;                       // coordonatele punctului

public:

        static const Punct Origine;
        static const Punct UnuZero;
        static const Punct ZeroUnu;

        Punct(int X = 0, int Y = 0);    // constructor
        Punct(){}                       // destructor

        int  GetX() const;              // returneaza coordonata pe orizontala
        int  GetY() const;              // returneaza coordonata pe verticala

        void MutaX (int x = 1);         // deplaseaza punctul pe orizontala
        void MutaY (int y = 1);         // deplaseaza punctul pe verticala
        void MutaXY(int x = 1, int y = 1);  // deplaseaza punctul pe orizontala si verticala
};

#endif


// file punct.cpp
#include "punct.h"

const Punct Punct::Origine;
const Punct Punct::UnuZero(1);
const Punct Punct::ZeroUnu(0,1);
```

Adăugaţi acestei clase metode/operatori pentru:

- citirea şi afişarea unui punct;
- testarea dacă punctul curent este localizat in originea axelor;
- testarea dacă punctul curent are/nu are aceeaşi pozitie ca un punct primit ca parametru;
- testarea dacă două puncte sunt/nu sunt “egale”;

- interschimbarea coordonatelor punctului curent cu cele ale unui punct recepţionat ca parametru;
- calcularea distanţei de la punctul curent la un alt punct primit ca parametru;
- determinarea celorlalte două colţuri ale dreptunghiului identificat prin două puncte primite ca parametri;

Testaţi aceste metode într-un program!

3. Definiţi o clasă Triunghi având ca date membru trei elemente de tip Punct*. Definiţi metode/operatori pentru:

   - constructori, destructor;
   - citirea şi afişarea unui Triunghi;
   - calculul lungimii laturilor şi al perimetrului;
   - calculul ariei;
   - testarea dacă triunghiul curent este dreptunghic;
   - testarea daca două triunghiuri sunt asemenea;

   Testaţi aceste metode într-un program!

4. Definiţi o clasă Poligon având ca date membru un număr variabil de elemente de tip Punct , memorate cu ajutorul unui std::vector. Definiţi metode/operatori pentru:

   - citirea şi afişarea unui Poligon;
   - verificarea convexităţii poligonului;
   - calculul ariei poligonului;

   Testaţi aceste metode într-un program!

5. Definiţi o clasă Mulţime care să permită manipularea mulţimilor de numere întregi. Clasa va fi implementată prin intermediul unui std::vector. Definiţi operatori pentru:

   - adăugarea unui nou element intr-o multime (operator +=);
   - ştergerea unui element dintr-o multime (operator -=);
   - reuniunea a două multimi (operator+);
   - afişarea unei multimi (operator <<);
   - testarea egalităţii a două mulţimi;

   Testaţi aceşti operatori într-un program!