

Laboratoire n°6 en C++ : Héritage simple

Nom et prénom :

Réaliser une classe point utilisable pour manipuler des points. Pour la créer, utiliser 3 fichiers :

- un fichier d'en-tête contenant la déclaration de la classe
- un fichier d'implémentation contenant les définitions de la classe
- un fichier d'utilisation contenant le programme principal.

La classe point contient :

- 2 données membres privées réelles, les coordonnées du point
- un constructeur initialisant les données membres et leur donnant comme valeur par défaut 0.0
- une surcharge de l'opérateur >> permettant d'entrer les coordonnées d'un point
- une surcharge de l'opérateur << permettant d'afficher les coordonnées d'un point
- une fonction translater qui permet à partir de 2 paramètres réels de déplacer un point et qui retourne un point (le point déplacé)
- une fonction distance qui permet de retourner la distance séparant un point à un second point passé en paramètre
- une surcharge de l'opérateur + qui permet d'additionner 2 points et qui retourne un nouveau point (la somme des 2 points passés en paramètres).

Réaliser une classe triangle qui dérive publiquement de la classe point. Pour la créer, utiliser les 3 fichiers créés pour la classe point. :

La classe triangle contient :

- 3 données membres de type point qui correspondent aux sommets du triangle
- une fonction saisir qui permet la saisie des coordonnées des 3 sommets du triangle (elle utilise la surcharge >> de point)
- une fonction afficher qui permet d'afficher les 3 sommets du triangle (elle utilise la surcharge << de point)
- une fonction translater qui permet à partir de 2 paramètres réels de déplacer les 3 sommets du triangle (elle utilise la fonction translater de point)
- une fonction longueur_cotes qui retourne un pointeur sur des doubles contenant la longueur des 3 côtés du triangle
- une fonction equilateral qui utilise la fonction longueur_cotes pour tester que les 3 côtés du triangle sont identiques et qui retourne true si les 3 côtés sont égaux sinon elle retourne false
- une fonction perimetre qui utilise la fonction longueur_cotes pour calculer le périmètre du triangle et qui retourne un double (le périmètre du triangle).

Réaliser un programme principal qui crée 2 points et qui teste les différentes fonctions de la classe point . Il crée également un triangle et teste les différentes fonctions de la classe triangle.

```

#include<iostream.h>
#include<math.h>

class point{
protected:
    double x,y;
public:
    point(double xx=0.0,double yy=0.0);
    friend ostream& operator<<(ostream& o,point p);
    friend istream& operator>>(istream& i,point& p);
    point translater(int x_pos,int y_pos);
    double distance(point p);//longueur!!!
    friend point operator+(point p1,point p2);
    ~point();
};

class triangle:public point{
protected:
    point a;
    point b;
    point c;
public:
    triangle();
    void saisir();
    void afficher();
    triangle translater(int x_pos,int y_pos);
    double* longueur_cotes();
    bool equilateral();
    double perimetre();
    ~triangle();
};

```

```

//définitions

#include"labo1.h"
#include<math.h>

point::point(double xx,double yy){
    x=xx;
    y=yy;
}
istream& operator>>(istream& i,point &p){
    cout<<"entrez x:";
    i>>p.x;
    cout<<"entrez y:";
    i>>p.y;
    return i;
}
ostream& operator<<(ostream& o,point p){
    cout<<"voici les coordonnees:";
    o<<"("<<p.x<<","<<p.y<<")";
    o<<endl;
    return o;
}
point point::translater(int x_pos,int y_pos){
    point res;
    res.x=x+x_pos;
    res.y=y+y_pos;
    cout<<res;
    return res;
}
double point::distance(point p){
    double base,racine;
    base=pow((x-p.x),2)+pow((y-p.y),2);
    racine=sqrt(base);
    return racine;
}
point operator+(point p1,point p2){
    point res;
    res.x=p1.x+p2.x;
    res.y=p1.y+p2.y;
    return res;
}
point::~~point(){
}

/////fin declaration classe point/////

/*debut declaration classe triangle*/
triangle::triangle(){
}

```

```

void triangle::saisir()
{
    cout<<"\nentrez le sommet a\n";
    cin>>a;
    cout<<"\nentrez le sommet b\n";
    cin>>b;
    cout<<"\nentrez le sommet c\n";
    cin>>c;
}
void triangle::afficher(){
    cout<<a;
    cout<<b;
    cout<<c;
}
triangle triangle::translater(int x_pos,int y_pos){
    a.translater(x_pos,y_pos);
    b.translater(x_pos,y_pos);
    c.translater(x_pos,y_pos);
    return *this;
}
double* triangle::longueur_cotes(){
    double* ptr = new double[3];
    ptr[0]= a.distance(b);
    ptr[1]= b.distance(c);
    ptr[2]= c.distance(a);
    return ptr;
    delete ptr;
}
bool triangle::equilateral(){
    double* ptr= longueur_cotes();
    if ((ptr[0] == ptr[1]) && (ptr[0] == ptr[2])) return true;
    else return false;
}
double triangle::perimetre(){
    double *ptr= longueur_cotes();
    return (ptr[0] + ptr[1] + ptr[2]);
}
triangle::~triangle(){
}

```

```

//programme principal

#include"labo1.h"

void main()
{
    //programme fonctionnel sur la classe point
    point res;

    point p1;
    cout<<"entrez les coordonnees du point:\n";
    cin>>p1;
    cout<<p1<<endl;

    point p2;
    cout<<"entrez les coordonnees du point:\n";
    cin>>p2;
    cout<<p2<<endl;

    cout<<"translation:"<<endl;
    p1.translater(1,0);
    p2.translater(0,1);
    cout<<endl;

    cout<<"distance entre 2 points:"<<p1.distance(p2)<<endl;
    cout<<endl;
    res=p1+p2;
    cout<<"somme des 2 points => "<<res;

    triangle z;

    z.saisir();
    z.afficher();
    cout<<"perimetre="<<z.perimetre()<<endl;
    cout<<"Apres translation:\n";
    z.translater(3,2);
    cout<<endl;
    z.afficher();
    cout<<"perimetre du triangle="<<z.perimetre()<<endl;

    if(z.equilateral()==true)
    {
        cout<<"c'est un triangle equilateral"<<endl;
    }
    else{cout<<"ce n'est pas un triangle equilateral"<<endl;}
}

```