Programmation Orientée Objets L3 Miage – Info M1 IMDS

Exercices sur le cours 2

Cercle

On a déjà la classe point suivante représentant un point dans le plan (reprenez les définitions faites en TD) :

```
class point {
public:
  point(double x, double y);
  point();
  double x() const;
  double y() const;
  double distance(const point& p) const;
  void print(ostream& ost) const;
  void read(istream& ist);
  void move(double dx, double dy);
private:
  double d_x, d_y;
};

istream& operator>>(istream& ist, point& p);
ostream& operator<<(ostream& ost, const point& p);
```

On veut créer une classe représentant un cercle dans le plan.

Écrivez la déclaration de la classe `circle` :

- un cercle est construit à partir de son centre et de son rayon, à partir des coordonnées de son centre et de son rayon et par défaut (cercle de centre (0,0) et de rayon 0) ;
- on veut connaître son centre et son rayon : méthodes `center()` et `radius()` ;
- elle stocke son centre `d_center` et son rayon `d_radius`

Écrivez la procédure de test `testCircle1` qui demande à l'utilisateur un centre et un rayon, crée le cercle correspondant et affiche son centre et son rayon. Compilez et testez.

On veut afficher un cercle sur un flot de sortie sous la forme `[(x,y),r]` et lire un cercle depuis un flot d'entrée sous la même forme (méthodes `print(ost)`, `read(ist))` et pouvoir utiliser les surcharges des opérateurs d'écriture et de lecture sur les flots. **Ajoutez** ces méthodes et les surcharges des opérateurs.

Écrivez la procédure de test `testCircle2` qui demande un cercle et le ré-affiche.

On veut savoir si un cercle contient un point. **Ajoutez** la méthode `contains(p)` qui renvoie vrai si le cercle contient le point `p` (la distance du centre au point est inférieure au rayon).

Écrivez la procédure de test `testContains` qui crée le cercle de centre (10,10) et de rayon 5 et l'affiche, demande à l'utilisateur un point et affiche si ce point est dans le cercle ou non. Compilez et testez.

Enfin on veut déplacer un cercle en translation. **Ajoutez** la méthode `move(dx,dy)` qui déplace le cercle en translation de (dx,dy).

Solution cercle

Déclaration circle :

```
class circle {
public:
  circle(const point& center, double radius);
  circle(double x, double y, double radius);
  circle();
  point  center() const;
  double radius() const;
  bool   contains(const point& p) const;
  void move(double dx, double dy);
  void print(ostream& ost) const;
  void read(istream& ist);
private:
  point d_center;
  double d_radius;
};

ostream& operator<<(ostream& ost, const circle& c);
istream& operator>>(istream& ist, circle& c);
```

Définition classe :

```
circle::circle(const point& center, double radius):
d_center{center}, d_radius{radius}
{}

circle::circle(double x, double y, double radius):
d_center{x,y}, d_radius{radius}
{}

circle::circle(): circle{0.0, 0.0, 0.0}
{}

point  circle::center() const
{
  return d_center;
}

double circle::radius() const
{
  return d_radius;
}

bool circle::contains(const point& p) const
{
  return d_center.distance(p)<d_radius;
}

void circle::move(double dx, double dy)
{
  d_center.move(dx, dy);
}

void circle::print(ostream& ost) const
{
  ost<<'['<<d_center<<','<<d_radius<<']';
}

void circle::read(istream& ist)
{
  point center;
  double radius;
  char ponct;
  ist>>ponct>>center>>ponct>>radius>>ponct;
  if (ist.good())
  {
    d_center = center;
    d_radius = radius;
  }
}

ostream& operator<<(ostream& ost, const circle& c)
{
  c.print(ost);
  return ost;
}

istream& operator>>(istream& ist, circle& c)
{
  c.read(ist);
  return ist;
}
```

Programme de test

```
void testCircle1()
{
  point center;
  double radius;
```

```
  cout<<"le centre et le rayon : ";
  cin>>center>>radius;
  circle c {center,radius};
  cout<<"cercle de centre c="<<c.center()<<" et de rayon
r="<<c.radius()<<endl;
}

void testCircle2()
{
  circle c;
  cout<<"Entrez un cercle :";
  cin>>c;
  cout<<"Vous avez entré : "<<c<<endl;
}

void testContains()
{
  circle c{10,10,5};
  cout<<"cercle "<<c<<endl;
  cout<<"Entrez un point :";
  point p;
  cin>>p;
  if (c.contains(p))
  {
    cout<<"le point est dans le cercle"<<endl;
  }
  else
  {
    cout<<"le point n'est pas dans le cercle"<<endl;
  }
}
```