TP POO C++ : la surcharge des opérateurs

Sommaire

Les objectifs de ce tp sont de découvrir la programmation orientée objet en C++.

Rappels

La programmation orientée objet consiste à **définir des objets logiciels et à les faire interagir entre eux**.

Concrètement, un **objet** est une **structure de données** (**ses attributs** c.-à-d. des variables) qui définit son **état** et une **interface** (**ses méthodes** c.-à-d. des fonctions) qui définit son **comportement**.

Une **classe déclare des propriétés communes** à un ensemble d'objets.

Un **objet** est créé à partir d'un **modèle** appelé **classe**. Chaque objet créé à partir de cette classe est une **instance** de la classe en question.

Travail demandé

Nous allons découvrir la surcharge d'opérateurs de flux « et » et des opérateurs arithmétiques.

On vous fournit un programme `testPoint.cpp` où vous devez décommenter progressivement les parties de code source correspondant aux questions posées.

La surcharge des opérateurs de flux « et »

Rappels : Un **flot** est un **canal recevant (flot d'« entrée ») ou fournissant (flot de « sortie ») de l'information**. Ce canal est associé à un périphérique ou à un fichier.

Un **flot d'entrée** est un objet de type `istream` tandis qu'un **flot de sortie** est un objet de type `ostream`.

Le flot `cout` est un flot de sortie prédéfini connecté à la sortie standard `stdout`. De même, le flot `cin` est un flot d'entrée prédéfini connecté à l'entrée standard `stdin`.

On surchargera les opérateurs de flux « et » pour une classe quelconque, sous forme de **fonctions amies**, en utilisant ces « canevas » :

```
ostream & operator << (ostream & sortie, const type_classe & objet1)
{
   // Envoi sur le flot sortie des membres de objet en utilisant
   // les possibilités classiques de << pour les types de base
   // c'est-à-dire des instructions de la forme :
   // sortie << ..... ;

   return sortie ;
}

istream & operator >> (istream & entree, type_de_base & objet)
{
   // Lecture des informations correspondant aux différents membres de objet
   // en utilisant les possibilités classiques de >> pour les types de base
   // c'est-à-dire des instructions de la forme :
   // entree >> ..... ;

   return entree ;
}
```


Question 1. Compléter les fichiers `Point.cpp` et `Point.h` fournis afin d'implémenter la surcharge de l'opérateurs de flux de sortie « pour qu'il affiche un objet `Point` de la manière suivante : `<x,y>`. Décommenter les parties de code de cette question dans le fichier `testPoint.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
Point p0, p1(4, 0.0), p2(2.5, 2.5);

cout << "P0 = " << p0 << endl;
cout << "P1 = " << p1 << endl;
cout << "P2 = " << p2 << endl;
```

Ce qui donnera :

```
P0 = <0,0>
P1 = <4,0>
P2 = <2.5,2.5>
```

Question 2. Compléter les fichiers `Point.cpp` et `Point.h` fournis afin d'implémenter la surcharge de l'opérateurs de flux d'entrée » pour qu'il réalise la saisie d'un objet `Point` de la manière suivante : `<x,y>`. Décommenter les parties de code de cette question dans le fichier `testPoint.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
cout << "Entrez un point : ";
cin >> p0;

if (! cin)
{
   cout << "ERREUR de lecture !\n";
   return -1;
}

cout << "P0 = " << p0 << endl;
```

Ce qui donnera :

```
Entrez un point : <2,6>
P0 = <2,6>

Entrez un point : <s,k>
ERREUR de lecture !
```

La surcharge des opérateurs arithmétiques

Question 3. Compléter les fichiers `Point.cpp` et `Point.h` fournis afin d'implémenter la surcharge des opérateurs utilisés dans le code source fourni. Décommenter les parties de code de cette question dans le fichier `testPoint.cpp` et tester. Vérifier que vous obtenez les résultats ci-dessous.

```
cout << "P0 * 2 = " << p0 * 2 << endl;
cout << "2 * P0 = " << 2 * p0 << endl;
cout << "-P0 = " << -p0 << endl;
cout << "P0 + P1 = ";
cout << (p0 + p1) << endl;
cout << "P0 * P1 = ";
cout << (p0 * p1) << endl;
```


Ce qui doit donner avec `P0 = <1,1>` :

```
P0 * 2 = <2,2>
2 * P0 = <2,2>
-P0 = <-1,-1>
P0 + P1 = <5,1>
P0 * P1 = 4
```

