TD3 : Héritage, Surcharge, Classes Abstraites

Exercice 1 : Points Colorés

Créer une classe **PointColor** qui représente un point coloré. La classe peut hériter de la classe **Point** que vous avez définie précédemment. Définir les constructeurs et fonctions nécessaires dans **PointColor**.

Exercice 2 : Formes

On veut définir des différentes formes comme des **segments**, des **triangles**, des **rectangles** et des **cercles**. Pour tous les objets, on veut pouvoir les afficher et les déplacer. On doit aussi pouvoir regrouper des objets en un ensemble afin de leur faire subir un traitement global (par exemple, déplacer ensemble un cercle et un rectangle). Proposer une hiérarchie de classes et l'implémenter (*pensez à définir une classe abstraite pour représenter une forme abstraite*).

Exercice 3 : Piles et Files

Soit le programme suivant permettant de gérer des files (First In First Out) et des piles (Last In First Out) :

```
void main(){
   CPile pile;
   CFile file;

   CList* ptList = &file;
   * ptList < 0 < 1;  //empiler deux valeurs dans la file
   cout << * ptList;
   int i;
   * ptList > i;      //récupérer une valeur de la file dans i
   cout << * ptList << " i=" << i;

   ptList = &pile;
   * ptList < 0 < 1;  //empiler deux valeurs dans la pile
   cout << *ptPile;
   * ptList > i;      //récupérer une valeur de la pile dans i
   cout << * ptList << " i=" << i;
}
```

Dans ce programme, les opérations communes aux piles et aux files sont regroupées dans une classe de base **CList**, puis cette classe est spécialisée en une classe permettant de gérer une pile **CPile**, et une autre classe permettant de gérer une file **CFile**. Pour représenter une liste en mémoire, vous devez utilisez une liste chaînée.