L3 MIASHS
TP C++ N° 2

Exercice 1 : les classes

Dans cet exercice, vous allez implémenter la version 'classe' (`CPersonne`) de la structure (`TPersonne`) que vous avez définie lors du TP N° 1 :

1. Définissez la classe `CPersonne` et ses attributs privés, du type `string*`
2. Ajoutez les constructeurs et fonctions membres publics suivants :
   - `CPersonne` : initialise les attributs avec `nullptr`
   - `CPersonne(string,string,string,string)` : alloue une position de mémoire (`new`) pour chaque attribut, en stockant la valeur correspondante envoyée comme paramètre
   - `void set_[attribut](string)` : attribue une valeur à l'attribut correspondant (définissez une fonction `set` par attribut et utilisez le pointer `this`)
   - `string get_[attribut]()` : renvoie la valeur d'un attribut (définissez une fonction `get` par attribut et utilisez le pointer `this`). Indiquez explicitement que ces fonctions ne sont pas autorisées à modifier l'état de l'objet
   - `~CPersonne` : libère les positions de mémoire allouées lors de la création de l'objet
3. Surchargez l'opérateur `==` afin de pouvoir comparer deux objets de la classe `CPersonne`. Vous êtes libre pour choisir la sémantique de la comparaison
4. Surchargez l'opérateur `«` de façon à pouvoir utiliser *cout* sur un objet du type `CPersonne` : *cout << personne*
5. Modifiez votre classe de façon que le code suivant puisse être lancé sans erreurs d'exécution (et sans erreurs de conception !) :

```
CPersonne p1 {"1","Trojahn","Cassia","cassia.trojahn@irit.fr"};
CPersonne p2 = p1;
CPersonne p3;
p3 = p1;
CPersonne p4;
p4 = CPersonne("2","Clemente","Gilles","gilles@cc.fr");
```

La déclaration de la structure et des fonctions se fera dans un fichier .hpp et les définitions dans un fichier .cpp.

Exercice 2 : une liste doublement chaînée homogène

Vous allez fournir une version 'classe' de la liste doublement chaînée définie lors du TP N° 1. Pour cela, vous allez créer la classe `CListe`, ayant deux pointeurs `CNoeud*` :

```
class CNoeud {
      private :
        CPersonne* pers;
        CNoeud* pred;
        CNoeud* suiv;
      public :
```

```
        CNoeud();
        CNoeud(CPersonne*,CNoeud*,CNoeud*);
        CPersonne* get_pers() const;
        CNoeud* get_pred() const;
        CNoeud* get_suiv() const;
         ~CNoeud();
};

class CListe {
       CNoeud* tete;
       CNoeud* queue;
       public :
         CListe();
         CListe(CNoeud*,CNoeud*);
         bool ajoute(CPersonne*);
         CPersonne* recherche(string) const;
         CPersonne* supprime(string);
         bool vide() const;
         void affiche() const;
         CNoeud* get_tete() const;
         CNoeud* get_queue() const;
         ~CListe();
};
```

Pour faciliter la réutilisation du code développé lors du TP précèdent, les classes peuvent être définies comme *amies*.

Exercice 3 : une liste doublement chaînée hétérogène

L'objectif de cet exercice est d'implémenter une liste doublement chaînée **hétérogène**, afin de pouvoir stocker plusieurs types de personnes. Pour cela, il faut définir les classes `CDeveloppeur` et `CManager`, ayant toutes les deux la classe `CPersonne` comme base :

- `CDeveloppeur` contient comme attributs `projet_en_cours`, du type `string` et `niveau`, du type `short`
- `CManager` contient comme attributs `liste_developpeurs`, un `vector<CPersonne*>` stockant des objets du type `CPersonne*` (la classe `vector` sera vue plus tard, pour l'instant, vous pouvez utiliser l'initialisateur {} pour initialiser un objet du type `vector`, la méthode `at(i)` pour récupérer l'élément dans la position i, et la méthode `size` pour récupérer la taille du vecteur)

1. Définissez les constructeurs et le destructeur pour chaque classe dérivée – ces constructeurs doivent faire appel au constructeur de la classe de base
2. Définissez les fonctions membres *get*/*set* pour chaque classe dérivée
3. Construisez une liste chaînée de type `CListe` hétérogène contenant des développeurs et des managers
4. Affichez la liste créée. Attention : il faut afficher les attributs des classes dérivées ! Modifiez la définition de la classe `CPersonne` si besoin.

Exercice 4 (supplémentaire)

Remplacez le type `vector<CPersonne*>` par une liste du type `CListe`.