E4

Dans cet exercice nous allons successivement concevoir une classe String, une classe Compte (compte bancaire), puis modifier la classe Compte pour modéliser une liste de comptes.

Fonctions à utiliser pour la classe String :

```
<cstring> :
      int strlen(const char *s);
      // length of a string, not including the '\0' terminator
      char* strcpy(char* dest, const char* src);
      // copy src in dest
      int strcmp(const char *c1t, const char *c2);
      // compare c1 and c2, return 0 if the strings are equal
<cctype> :
      char toupper (char c);
      // convert c to its upper case equivalent
```

Partie I – classe String

Ecrire la classe String permet de stocker une chaîne de caractères sous forme de mémoire allouée dynamiquement.

```
class String
{
public :
      ...
private :
      char* ch;  // zone de mémoire allouée dynamiquement (new)
                 // dans laquelle on stocke la chaine de caractères
};
```

1) Ecrire un constructeur qui initialise l'objet à partir d'une chaîne (const char*) reçue en paramètre : il alloue pour l'objet une zone de mémoire de taille suffisante puis copie la chaîne reçue.

Lorsque le compilateur détruit une variable de type String, n'y a-t-il pas une action supplémentaire à faire, à la charge du programmeur ?

2) Ecrire une fonction d'affichage.

3) Ecrire une fonction qui change en majuscule chaque caractère de la chaîne.

Faire l'essai suivant montrant que le constructeur par copie créé par le compilateur ne convient pas :

- créer un objet String s1 et construire s2 par copie de s1; puis changer s2 en majuscules et afficher s2 et s1;
- constater le problème : s1 est aussi changé en majuscules.

4) Ecrire un constructeur par copie approprié et constater que le programme fonctionne alors correctement.

5) Ecrire une fonction membre,

```
bool estEgal(const String& str2)const
```

qui indique si la chaîne de l'objet courant est égale à celle de str2.

Vérifier qu'on peut aussi passer en paramètre de cette fonction des chaînes de type char* (conversion automatique par le constructeur).

Partie II – classe Compte

On désire maintenant écrire une classe Compte représentant des comptes rapportant des intérêts suivant un taux qui est le même pour tous les comptes.

Les données de chaque compte sont le nom (un objet String) et le montant et elles seront mises en partie privée.

Le taux sera également mis en partie privée et une fonction publique permettra de le modifier.
De quelle nature le taux et sa fonction de modification doivent-ils être ?

L'interface publique de la classe Compte comprendra aussi :
- un constructeur recevant le nom (const char*) et un montant initial (avec une valeur par défaut de 0),
- une fonction permettant de faire un versement,
- une fonction actualisant le montant en lui ajoutant les intérêts suivant le taux (montant courant + montant courant fois taux),
- une fonction affichant les informations du compte.

1) Créer la classe Compte

Partie III – mise à jour classe Compte

On désire chaîner les objets Compte entre eux sans écrire de nouvelle classe, en modifiant la classe existante. Pour cela on veut faire pointer chaque compte vers le compte suivant dans la liste. On souhaite mémoriser la tête de liste grâce à une donnée membre statique.

A la construction d'un nouvel objet Compte, l'objet est inséré (dans le constructeur) en tête de liste. Lorsque l'objet est détruit, il est supprimé de la liste (pour cela, le destructeur devra parcourir la liste jusqu'à trouver l'objet et le supprimer de la liste).

1) Ajouter dans la classe Compte les données privées *Compte* suiv*, pointeur sur le compte suivant dans la liste, et *Compte* tete*, tête de liste (statique). Mettre à jour le constructeur et créer le destructeur.

2) Ajouter une fonction qui affiche tous les comptes de la liste et une autre qui les actualise.

Dans le programme principal, pour tester l'insertion et la suppression dans la liste, créer des objets Compte par new pour pouvoir choisir par delete le moment de leur destruction.