TP C++ 5 : Fonctions amies et surdéfinition

M1 Mathématiques Appliquées

2019-2020

Remarques préliminaires :

- Il est fortement recommandé de valider les questions de manière incrémentale à l'aide d'un programme principal.
- Comme dans le TP4, on codera les classes dans des fichiers à part.

Exercice 1 : Ensemble d'entiers

On considère une classe `Ensemble` permettant de coder un tableau d'entiers dont la taille maximale sera fixée à la création de l'objet. La déclaration de la classe débutera comme suit :

```
class Ensemble{
  // Tableau d'entiers
  int *tab;
  // Taille max du tableau
  int nmax;
  // Nombre d'elements actuellement dans le tableau
  int nelts;

  // ...
};
```

1. Écrire des fonctions membres `taillemax` et `cardinal` renvoyant respectivement le nombre maximal d'éléments du tableau et le nombre actuel d'éléments dans le tableau.
2. Écrire un constructeur pour la classe `Ensemble` qui prenne en argument un entier représentant la taille maximale du tableau, alloue dynamiquement le tableau associé et initialise le nombre d'éléments du tableau à 0. Écrire le destructeur correspondant.
3. Écrire une fonction membre `afficher` qui affiche les éléments actuellement présents dans le tableau.
4. Écrire une fonction membre `ajouter` permettant d'ajouter un entier au tableau et renvoyant un booléen indiquant si l'ajout s'est fait ou non.

5. On considère la fonction suivante, externe à la classe `Ensemble`

```
int somme(Ensemble e);
```

dont le but est de faire la somme des éléments du tableau d'entiers contenu dans un objet de la classe `Ensemble`.

   (a) Définir cette fonction en la déclarant comme fonction amie de la classe `Ensemble`.

   (b) Tester cette fonction en appelant `afficher` avant et après son exécution. Quel est l'inconvénient de cette fonction ? Introduire un constructeur de recopie dans la classe `Ensemble` pour résoudre ce problème.

Exercice 2 : Couleurs en mode RGB

On considère la classe `CouleurRGB` définie comme suit :

```
class CouleurRGB{
  int valR,valG,valB;
  public:
    CouleurRGB(int,int,int);
}
```

Pour être conforme aux codes RGB des couleurs, on souhaitera que les valeurs prises par les variables `valR,valG,valB` soient comprises entre 0 et 255.

1. Définir le constructeur déclaré dans la classe. Si les entiers donnés en argument sont supérieurs à 255 (respectivement inférieurs à 0), leur valeur sera remplacée par 255 (respectivement 0).
   NB: La bibliothèque `iostream` contient les fonctions `min` et `max`, applicables aux entiers.

2. Déclarer et définir une fonction membre `afficher` qui affiche les valeurs des trois membres données.

3. Surdéfinir l'opérateur `+` en tant que fonction membre de la classe `CouleurRGB`. La "somme" de deux couleurs renverra la moyenne des deux couleurs en termes de niveaux de rouge, vert et bleu :

```
CouleurRGB c1 = CouleurRGB(0,120,255);
CouleurRGB c2 = CouleurRGB(120,120,145);
CouleurRGB c3 = c1+c2; // c3 contient (60,120,200)
```

4. Surdéfinir l'opérateur `==` en tant que fonction amie de la classe `CouleurRGB`. On dira que deux objets sont égaux si leurs trois membres données sont égaux.