TD N°3

Fonctions et classes amie, Surdéfinition des opérateurs,

Exercice 1 :

Suite de la classe ***SetEntiers (exercice 3 TD 2)***

On considère la fonction ***int somme (setEntiers e);*** dont le but est de faire la somme des éléments du tableau d'entiers contenu dans un objet de la classe ***setEntiers.***

1. Définir cette fonction en la déclarant comme fonction amie de la classe ***setEntiers***
2. Tester cette fonction.

Suite de la classe ***Matrices (exercice 4 TD 2)***

On considère la fonction ***Matrice& multiplication(Matrice , Matrice);*** dont le but est de faire la multiplication des deux matrices.

1. Définir cette fonction en la déclarant comme fonction amie de la classe ***Matrice***
2. Tester cette fonction.

Exercice 2 :

On considère la classe ***Complexe,*** qui possède deux membres données notés par ***reel***, et ***img***, qui correspondent respectivement à la partie réelle et imaginaire d'un nombre complexe.

1)

- Définir un constructeur par défaut qui initialise les membres de la classe à 0.
- Définir un constructeur de recopie : `Complexe (const Complexe &).`
- Définir une Fonction d'affichage : `void afficher().`

  Fonction qui retourne le module sachant que : $|z| = \sqrt{a^2 + b^2}$.: `double module();`
- Fonction qui retourne le conjugué sachant que : $\bar{z} = a - bi$ `Complexe conjugue();`
- Surdéfinir l'opérateur =, dont le prototype est : `Complexe & operator=(const Complexe &)`
- Surdéfinir l'opérateur +, dont le prototype est :`Complexe operator+(Complexe);`
- Surdéfinir l'opérateur -, dont le prototype est :`Complexe operator-(Complexe);`
- Surdéfinir l'opérateur *, dont le prototype est :`Complexe operator*(Complexe);`
- Surdéfinir l'opérateur +=, dont le prototype est : `Complexe & operator+=(const Complexe &)`
- Surdéfinir l'opérateur -=, dont le prototype est : `Complexe & operator-=(const Complexe &)`
- Surdéfinir l'opérateur *=, dont le prototype est :`Complexe & operator*=(const Complexe &)`
- Surdéfinir l'opérateur /=, dont le prototype est : `Complexe & operator/=(const Complexe &)`
- Surdéfinir l'opérateur ==, dont le prototype est `bool operator==(const Complexe &)`

- Les accesseurs qui permettent d'accéder aux membres donnés.
- Les manipulateurs qui permettent de modifier les membres donnés.

2) Définir les fonctions amie à la classe Complexe:

```
friend Complexe operator+(double, Complexe);
friend Complexe operator+(Complexe, double);
friend Complexe operator-(double, Complexe);
friend Complexe operator-(Complexe, double);
friend Complexe operator*(Complexe, double);
friend Complexe operator*(double, Complexe);
```

Exercice 3 :

Réaliser une classe String pour la gestion des chaînes de caractères, sa déclaration est la suivante :

```
//Quelques « define » utiles.
#define UNIT unsigned int
#define NULL 0
class String{
     char*chaine ;  // la chaîne en elle-même
     UINT taille ;  // la longueur de la chaîne
     public :
          String() ;
          String(char*) ;
          String (cost String &) ;
          ~String() ;
          String & operator=(const String &) ;
          Bool operator == (const String &) ;
          String & operator+=(const String &) ;
          String & operator+=(const char*) ;
          String & operator+=(const char) ;
          String operator+(const String&) ;
          Char& operator[](UNIT);
          bool isEmpty(); //si la chaine est vide
          void Empty() ;//détruire la chaîne
          UNIT getSize() ; // retourne la taille de la chaine.
 } ;
```