TD 3: Programmation en C++

Exercice 1 :

Un polynôme est un objet mathématique de la forme suivante :

$$c_n \times x^n + c_{n-1} \times x^{n-1} + \cdots + c_0 \times x^0$$

Les c_i sont les coefficients et x est la variable tous du type réel. n est le degré du polynôme, du type entier.

```
class polynome{
        int deg;
        float* coeffs;
public :
        };
```

1. Ecrire la définition des ***constructeurs*** de la classe polynome. Le constructeur ***polynome(int n)*** construit un polynôme de degré n dont les coefficients sont nuls ainsi que le constructeur de recopie.

   Ecrire les méthodes suivantes :

2. L'opérateur d'affectation ***operator=***
3. L'opérateur d'addition ***operator+***
4. Fournir une méthode ***polynome operator * (float k)*** qui retourne le produit d'un polynôme par une constante réelle k.
5. Ecrire la définition de la méthode ***Calcul*** qui évalue le polynôme pour la variable x.
6. Écrire une fonction amie de saisie ***creerPoly*** des coefficients et des degrés des différents monômes d'un polynôme, qui retourne le polynôme ainsi créé.
7. Écrire un programme principal int main() qui lit un polynôme ***P***, crée un polynome ***Q*** identique à ***P*** et affiche leur somme. Ensuite calcule le produit de ***P*** avec un réel ***val*** lu au clavier, enfin calcule la valeur du polynome pour ***val***.

Rappel*: L'addition de deux polynômes p1 et p2 est un polynôme dont le coefficient de rang i est la somme des coefficients de rang i de p1 et p2.*

Exercice 2 :

Une matrice peut être considérée comme une agrégation d'un vecteur. En effet, une matrice possède un indice de plus qu'un vecteur et si l'on fixe le premier indice on obtient un vecteur. Cette représentation permet de réutiliser le code de vecteur dans la définition de matrice. Dans cette perspective, vous pouvez à partir du code du programme qui est présenté en Annexe.

La classe ***Matrice*** est définie (comme en annexe) en utilisant la classe ***Vecteur***.

1. Définissez la définition de constructeur de la classe comme définie dans le résultat de l'exécution du programme défini en annexe.

2. Définissez une surcharge de l'opérateur d'indexation "[ ]" pour la classe Matrice dont le prototype est le suivant:

   Vecteur& operator[](const int)

3. Définissez une surcharge de l'opérateur de sortie **"<<** "pour la classe Matrice en faisant appel à l'opérateurs "**<<** " de la classe Vecteur.

La surcharge de l'opérateur doit se faire de telle façon que le programme principal présenté en annexe donne le résultat affiché aussi en annexe.

ANNEXE

```
#include <iostream.h>

// définition de la classe Vecteur

class Vecteur {
private:
                    int         dim;
                    int * tab;
public:
        Vecteur(int  );
        ~Vecteur() ;
        int & operator[](int);

};
```

```
// Définition de la classe Matrice

class  Matrice{
private:
         int lig;
         int col;
          Vecteur** mat;
public:
         Matrice(int , int);
        ~Matrice();
        Vecteur& operator[](int);
}
```

```
// Programme principal
 int main() {
    Vecteur V(5);
    cout << " V = " ;
    cout << V ;
    cout << endl << " Lecture de V"<< endl ;
    cin >> V ;
    cout << V ;
    Matrice M(2,3);
    cout << endl << " M = " << endl;
    cout << M ;
    cout << endl << " Lecture de M " << endl ;
    cin >> M ;
    cout <<  M ;
    return 0;
 }
 // Résultat d'exécution:

 V = (0  1  2  3  4)
  Lecture de V
  terme n° 0 = 2
  terme n° 1 = 4
  terme n° 2 = 6
  terme n° 3 = 8
  terme n° 4 = 10
 (2  4  6  8  10)
  M =
 (0  1  2)
 (0  1  2)
  Lecture de M
  Terme [ 0, 0 ] = 1
  Terme [ 0, 1 ] = 2
  Terme [ 0, 2 ] = 3
  Terme [ 1, 0 ] = 4
  Terme [ 1, 1 ] = 5
  Terme [ 1, 2 ] = 6
 (1  2  3)
 (4  5  6)
```