TD n° 1 et 2
La classe polynôme

Eric REMY
eric.remy@univ-amu.fr

Version du 27 février 2025

Cette planche d'exercices propose d'étudier l'implantation d'une classe Polynome permettant de représenter n'importe quel polynôme à coefficients réels. Les polynômes ayant de très nombreuses propriétés, ils seront une belle illustration de ce que la surcharge d'opérateurs permet de faire en langage C++.

Exercice

On rappelle qu'un *polynôme* $P(x)$ de degré n est une fonction de la forme :

$$P(x) = \sum_{i=0}^{n} a_i \cdot x^i = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \cdots + a_2 \cdot x^2 + a_1 \cdot x + a_0$$

où $a_i \in \mathbb{R}$ est le coefficient du *monôme* de degré i.

L'implantation devra permettre de manipuler la classe d'une façon illustrée par l'exemple suivant (programme source et sortie à l'écran). Vous prendrez garde à trouver tous les opérateurs mis en œuvre par l'exemple.

```
1  const double coef_p1[]={3,-1.414,3,6,0,1};
2  Polynome p1(5,coef_p1); // p1(x) = x^5 + 6x^3 + 3x^2 - √2x + 3
3  cout << "p1(x)=" << p1 << " et son degré est " << p1.getDegre() << "." << endl;
4
5  const double coef_p2[]={4,-2,3,1,-2,-1};
6  Polynome p2(5,coef_p2); // p2(x) = -x^5 - 2x^4 + x^3 + 3x^2 - 2x + 4
7  cout << "p2(x)=" << p2 << " et son degré est " << p2.getDegre() << "." << endl;
8
9  cout << "Construction de p3 comme copie de p2." << endl;
10 Polynome p3(p2);
11 if(p3==p2)
12   cout << "p3 est égal à p2... Ce qui est rassurant pour une copie !" << endl;
13
14 cout << "Affectation de p1 à p3." << endl;
15 p3 = p1; // On affecte sans construire puisque p1 existe déjà.
16
17 cout << "On affecte le coefficient du monôme de degré 1 de p3 à la constante de p3." << endl;
18 p3[0]=p3[1]; // On affecte le coefficient du monôme de degré 1 au monôme de degré 0...
19 cout << "p3 est maintenant égal à p3(x)=" << p3 << endl;
20 if(p2==p3)
21   cout << "p2 est égal à p3... ce qui ne devrait pas arriver !" << endl;
22 else
23   cout << "p2 est maintenant différent de p3 !" << endl;
24
25 cout << "(p1+p2)(x)=" << p1+p2 << endl;
26
27 // Polynome p4; // Vous devez rendre cette instruction impossible.
28
29 cout << "Sachant que p1(x)="  << p1
30      << ", on peut calculer ses valeurs en fonction de x : " << endl;
31 for(double x=-2; x<=2; x+=1)
32   cout << "Pour p1(" << x << ")=" << p1(x) << endl; // p1 est un foncteur!
33
34 cout << "Le polynôme p2(x)=" << p2 ;
35 Polynome p2prime( p2.deriver() ); // Dérivation de p2
36 cout << " a pour dérivée p2'(x)=" << p2prime;
37
38 return 0;
```

```
p1(x)=x^5+6x^3+3x^2-1.414x+3 et son degré est 5.
p2(x)=-x^5-2x^4+x^3+3x^2-2x+4 et son degré est 5.
Construction de p3 comme copie de p2.
p3 est égal à p2... Ce qui est rassurant pour une copie !
Affectation de p1 à p3.
On affecte le coefficient du monôme de degré 1 de p3 à la constante de p3.
p3 est maintenant égal à p3(x)=x^5+6x^3+3x^2-1.414x-1.414
p2 est maintenant différent de p3 !
(p1+p2)(x)=-2x^4+7x^3+6x^2-3.414x+7
Sachant que p1(x)=x^5+6x^3+3x^2-1.414x+3, on peut calculer ses valeurs en fonction de x :
Pour p1(-2)=-62.172
Pour p1(-1)=0.414
Pour p1(0)=3
Pour p1(1)=11.586
Pour p1(2)=92.172
Le polynôme p2(x)=-x^5-2x^4+x^3+3x^2-2x+4 a pour dérivée p2'(x)=-5x^4-8x^3+3x^2+6x-2
```

1. Donnez en C++ la déclaration de la classe Polynome.
2. Donnez en C++ la déclaration et la définition du constructeur de la classe Polynome recevant en paramètre le degré et l'adresse du tableau constant de doubles contenant les coefficients du polynôme. On rappelle qu'un polynôme de degré n comporte $n+1$ coefficients.
3. On veut empêcher l'utilisateur de la classe Polynome de créer des polynômes « vides », c'est-à-dire sans qu'il indique comment les initialiser. Savez-vous interdire à un utilisateur de la classe, la construction d'une instance sans qu'il ne précise de paramètre ? En d'autres mots, savez-vous comment interdire l'utilisation du *constructeur par défaut* ?
4. On s'intéresse maintenant au constructeur de copie. À quelle(s) ligne(s) de l'exemple est-il appelé ? Donnez son implantation en évitant d'écrire du code en double.
5. On s'intéresse au destructeur. À quelle(s) ligne(s) de l'exemple est-il appelé ? Combien de fois ? Pourquoi ? Donnez son implantation.
6. Donnez l'implantation du « getter » permettant de consulter le degré d'un polynôme.
7. Donnez l'implantation de la méthode Polynome Polynome::deriver() const qui calcule la dérivée du polynôme sur lequel elle est déclenchée. On rappelle que la dérivée $P'(x)$ de $P(x)$ se calcule par :

$$P'(x) = \left(\sum_{i=0}^{n} a_i \cdot x^i\right)' = \sum_{i=0}^{n} \left(a_i \cdot x^i\right)' = \sum_{i=0}^{n} a_i \cdot \left(x^i\right)' = \sum_{i=1}^{n} a_i \cdot i \cdot x^{i-1}$$

8. Donnez la déclaration et la définition de l'opérateur « = ». À quelle ligne de l'exemple est-il déclenché ? En quoi est-il différent du constructeur de copie ?
9. Donnez la définition de l'opérateur « == ». On rappelle que deux polynômes sont égaux si et seulement si leurs degrés sont égaux et leurs coefficients sont égaux.
10. Donnez la définition de l'opérateur « + » qui réalise l'addition de deux polynômes. Pensez au cas où les deux polynômes n'ont pas le même degré. Est-il nécessaire de simplifier le résultat après le calcul (regardez la ligne 9 de la sortie écran) ? Si oui, comment ?
    En TP, vous réaliserez la soustraction sur le même principe. Facultativement, vous pourrez également réaliser la multiplication, la division (qui donne le polynôme quotient) et le modulo (qui donne le polynôme reste)... mais c'est plus difficile à comprendre mathématiquement.
11. Donnez une définition de l'opérateur « << » qui permet d'écrire la formule du polynôme. Vous commencerez par une méthode simple et systématique. Facultativement, vous pourrez améliorer votre algorithme pour :
    — ne pas afficher le '+' devant un monôme si le coefficient est négatif ;
    — ne pas afficher du tout les monômes dont le coefficient est nul ;
    — ne pas afficher le coefficient s'il vaut +1 ;
    — n'afficher que « − » à la place du coefficient lorsqu'il vaut -1 ;
    — ne pas afficher « x^0 » pour le monôme constant, etc. ;
    — quand le polynôme est constant (c'est-à-dire de degré 0), afficher seulement la constante.
12. Donnez la définition de deux opérateurs « [ ] » paramétrés par un degré i et qui permettent respectivement la consultation du coefficient du monôme de degré i ou bien sa modification. La fonction doit contrôler que le degré i est compatible avec le polynôme courant ou sinon lancer une exception « std::out_of_range ».
13. Donnez la définition de l'opérateur « ( ) » qui reçoit en paramètre une valeur pour x et effectue le calcul de la valeur de $P(x)$. Une fois cet opérateur implanté, votre classe constitue maintenant un *foncteur* : chacune de ses instances peut être utilisée comme une fonction $P(x) : \mathbb{R} \to \mathbb{R}$.