Examen de langage C++ et calcul scientifique

DEA de mathématiques appliquées, Grenoble

Pierre Saramito

1 mars 2004

Le sujet d'examen comporte deux parties indépendantes. De plus, la plupart des questions sont indépendantes les unes des autres, bien que chaque partie ait une unité de thème.

Les documents sont autorisés.

Nous utiliserons la classe `valarray` de la librairie standard, qui présente un interface du type :

```
template <class T>
class valarray {
    public:
        valarray (int n);
        valarray (const T& initval, int n);
        int size() const;
        void resize(int n, const T& initval = T());
        const T& operator[] (int i) const;
        T& operator[] (int i);
        // ...
};
```

Partie 1 : matrices denses

Dans cette première partie, nous nous proposons d'étudier la programmation la plus simple possible de matrices *denses* rectangulaires en C++. Ces matrices ne font pas partie de la librairie standard actuelle. Cependant, différents projets récents de standardisation[1] proposent un interface du type :

```
template <class T>
class matrice {
    public:
        matrice (size_t n1 = 0, size_t n2 = 0);
        size_t nrow () const;
        size_t ncol () const;
        T operator() (size_t i, size_t j) const;
        T& operator() (size_t i, size_t j);
    protected:
        size_t d1, d2;
        valarray<T> m;
};
```

[1] lapack++, boost/ublas, etc.

Question 1 : Le tableau de type `valarray` dans la zone de données contient les `d1*d2` coefficients de la matrice, rangés colonne par colonne. Écrire le code du constructeur ainsi que des quatres accesseurs.

Question 2 : Écrire le code du produit matrice-vecteur suivant :

```
valarray<T> operator* (const matrice<T>& a, const valarray<T>& x);
```

Partie 2 : polynômes

Dans cette partie, nous nous proposons d'étudier la conception d'un système de calcul formel[2] en `C++` par la programmation d'une classe de polynômes.

Un polynôme est représenté ses coefficients $(c_i)_{0 \leqslant i \leqslant n-1}$:

$$p(X) = \sum_{k=0}^{n-1} c_k X^k, \quad c_{n-1} \neq 0.$$

Le degré du polynôme est alors $n-1$. Le polynôme nul correspond à $n = 0$ et est alors par convention de degré égal à -1. L'indéterminée `X` peut être définie simplement par une chaîne de caractères :

```
class indeterminee : public string {
    public:
        indeterminee (const string& s = "X") : string(s) {}
};
```

La classe `string` est prédéfinie dans la librairie standard du `C++`. La classe `polynome` contient alors l'indéterminée ainsi que le tableau des coefficients :

```
template <class T>
class polynome : public valarray<T> {
    public:
        // ...
    protected:
        indeterminee X;
};
```

Question 1 : Écrire un constructeur qui initialise un polynôme constant $p(X) = c_0 = c_0 \times X^0$. Par exemple, cela donnera :

```
int main () {
    polynome<double> p = 3;
}
```

Question 2 : Afin de pouvoir définir $p(X) = X$, **écrire** un second constructeur qui prend en argument l'indéterminée. Une utilisation de ce constructeur sera par exemple :

```
int main () {
    indeterminee X;
    polynome<double> p = X;
}
```

[2] Voir par exemple pour compléments : M. Bronstein, Symbolic Integration, Springer, 1997.

Question 3 : Définir le constructeur de copie et l'opérateur d'affectation :

```
template <class T>
class polynome : public valarray<T> {
    public:
       // ...
       polynome (const polynome<T>&);
       polynome<T>& operator= (const polynome<T>&);
       // ...
};
```

Question 4 : Définir la fonction membre `degre` qui renvoie le degré d'un polynôme.

Question 5 : Définir la somme et la différence de deux polynômes.

Note : on prendra soin de s'assurer que $p-p$ est un polynome nul (de degré égal à -1 par convention).

Question 6 : Définir le produit $c * p$ d'une constante par un polynôme, le produit $X * p$ d'une indéterminée par une constante, ainsi que le produit $p * q$ de deux polynômes entre eux.

Question 7 : écrire une fonction d'impression d'un polynôme, telle que le code :

```
int main () {
    indeterminee X;
    polynome<double> p = X;
    cout << (p+1)*(p+2) << endl;
}
```

imprime :

```
X^2 + 3*X + 2
```

Question 8 : Les polynômes de Legendre sont donnés par la formule de récurence :

$$\begin{aligned} P_0(X) &= 1 \\ P_1(X) &= X \\ (n+1)P_{n+1}(X) &= (2n+1)XP_n(X) - nP_{n-1}, \quad n \geqslant 1 \end{aligned}$$

Écrire un programme qui imprime les dix premiers polynômes de Legendre.

Question 9 : Définir les fonctions membre `derive` et `integre` de la classe `polynome`, qui renvoient respectivement la dérivée et la primitive d'un polynôme.

Question 10 : Modifier le programme précédent, pour imprimer aussi les dérivées des dix premiers polynômes de Legendre.

Question 11 : Écrire la fonction d'évaluation $p(a)$ d'un polynôme, paramétrée par le type `Valeur` de la valeur $'a'$ associé à l'indéterminée X :

```
template <class T>
class polynome {
    public:
        // ...
        template <class Valeur>
        Valeur operator() (const Valeur& Xval) const;
};
```

Le type `Valeur` est ici générique : il peut coïncider avec le type `T` des coefficients, ou bien être un objet plus complexe : par exemple un type matrice. On supposera pour cela que les types

`T` et `Valeur` sont compatibles, aus sens où l'élevation à la puissance par un entier naturel et la multiplication par un coefficient de type `T` sont correctement définis :

$$\texttt{Valeur} \longleftarrow \texttt{pow}(\texttt{Valeur}, \texttt{int})$$
$$\texttt{Valeur} \longleftarrow \texttt{T} * \texttt{Valeur}$$

Question 12 :

Pour deux polynomes a et b, avec $b \neq 0$, on note le q et r le quotient et le reste, respectivement, de la division euclidienne, tels que $a = b*q+r$, avec soit $r = 0$ soit $degre(r) < degre(b)$. L'algorithme suivant donne q et r :

$q := 0$
$r := a$
tanque $r \neq 0$ et $d := degre(r) - degre(b) \geqslant 0$ faire
  $t := lc(r)/lc(b) * X^d$
  $q := q + t$
  $r := r - b * t$
fin

où la notation $lc(p)$ représente le coefficient du terme de plus haut degré du polynôme p.

Écrire cet algorithme en `C++` en utilisant la classe `polynome`. Par commodité, on écrira aussi une fonction annexe

```
template <class T>
polynome<T> pow(symbole X, int d);
```

qui renvoie le monôme X^d.

Donner un exemple de programme d'appel qui imprime le quotient et le reste de la division de $a = 3X^3 + X^2 + X + 5$ par $b = 5X^2 - 3X + 1$.