TD_6 – Polynômes

1 Introduction

Le but de ce TD est de définir un certain nombre d'opérations sur les polynômes. Par souci de simplification, on traitera de polynômes de degré 3, sachant que l'extension à un degré n quelconque est immédiate. On définira ainsi une constante `DEGRE`, fixée à 3.

Un polynôme sera représenté par un tableau contenant ses coefficients. Pour alléger l'écriture du programme, on définira :

```
type poly = array[0..DEGRE] of real;
```

2 Fonctions de base

Ecrire les fonctions de base suivantes, dont on donne les prototypes :

1. Saisie d'un polynôme : `procedure readPoly(var P : poly) ;`
2. Affichage d'un polynôme : `procedure writePoly(P : poly) ;`
3. Addition de polynômes : `procedure sommePoly(P,Q : poly ; var S : poly) ;`
4. Multiplication par un réel : `procedure lambdaPoly(lambda : real ; P : poly ; var R : poly) ;`
5. Dérivation formelle : `procedure derivePoly(P : poly ; var P1 : poly) ;`
6. Primitivation (primitive de constante nulle) : `primitivePoly(P : poly ; var P1 : poly) ;`
7. Détermination du degré effectif : `function degrePoly(P : poly) : integer ;`
8. Multiplication de polynômes : `procedure multPoly(P, Q : poly ; var R : poly) ;`.
   On utilisera la formule :

$$\left(\sum_{i=0}^{n} a_i X^i\right)\left(\sum_{j=0}^{p} b_j X^j\right) = \sum_{k=0}^{n+p}\left(\sum_{i=0}^{k} a_i b_{k-i}\right) X^k$$

3 Evaluation de polynôme

3.1 Méthode naïve

Ecrire une fonction d'évaluation d'un polynôme P au point x_0. On élèvera x_0 à chacune des puissances contenues dans le polynôme, et on fera les produits et la somme nécessaire.

Evaluer le nombre d'opérations (additions ou multiplications) effectuées, pour un polynôme de degré n.

3.2 Méthode de Hörner

On remarque qu'on peut réécrire un polynôme sous la forme :

$$a_0 + a_1X + a_2X^2 + a_3X^3 + \cdots = a_0 + X(a_1 + X(a_2 + X(a_3 + \cdots)))$$

On parle de *schéma de Hörner*.

Réécrire une fonction d'évaluation d'un polynôme, basée sur cette écriture.

Evaluer le nombre d'opérations de cette dernière fonction, et comparer avec la fonction précédente.

3.3 Application

Avec la fonction `primitivePoly` et la fonction d'évaluation, écrire une fonction calculant l'intégrale d'un polynôme sur un intervalle réel $[a; b]$.