TP8 (3 séances)

Objectif : programmer une classe « `polynôme` » permettant de manipuler (addition, multiplication, sauvegarde dans un fichier, …) des polynômes à une variable et à coefficients entiers.

Un polynôme a une variable (ou indéterminée) est une expression de la forme :

$$P1(x)= a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1 x^1 + a_0$$

Où les a_i sont les coefficients du polynôme, ils sont des entiers relatifs (c.a.d. Ils sont des entiers positifs ou négatifs).

Rappel, via des exemples, de la définition des polynômes :

$0, -7, 11, 45, 132, 56789$ sont des polynômes de degré 0 ayant 1 coefficient non nul.
$9x, -54x, 89x, 637x, 2134x$ sont des polynômes de degré 1 ayant 1 coefficient non nul.
$4x^5, 87x^5, -597x^5, 712x^5$ sont des polynômes de degré 5 ayant 1 coefficient non nul.

$P1(x)= 7x^7 + 6x^6 + 4x^4 + 2x^2 = 7x^7+6x^6+0x^5 +4x^4+0x^3+2x^2+0x^1+0$
$P2(x)= 8x^8 + 5x^4 + 3x^3 = 8x^8 +0x^7+0x^6+0x^5 +5x^4+3x^3+0x^2+0x^1+0$
$P1(x)$ a 4 coefficients non nuls et $P2(x)$ a 3 coefficients non nuls. $P1(x) + P2(x)$ a 10 coefficients non nuls.

$P1(x) + 0 = P1(x)$
$P1(x) + 9x = 7x^7 + 6x^6 + 4x^4 + 2x^2 + 9x$
$P1(x) + 4x^6 = 7x^7 + 10x^6 + 4x^4 + 2x^2$
$P1(x) + 11x^{11} = 11x^{11} + 7x^7 + 6x^6 + 4x^4 + 2x^2$
$P1(x) + 0x^{45} = P1(x)$

$P1(x) + P2(x) = 8x^8 + 7x^7 + 6x^6 + 9x^4 + 3x^3 + 2x^2$

$P1(x) * 0 = 0$
$P1(x) * 9x = 63x^8 + 54x^7 + 36x^5 + 18x^3$
$P1(x) * 4x^6 = 28x^{13} + 24x^{12} + 16x^{10} + 8x^8$
$P1(x) * 11x^{11} = 77x^{18} + 66x^{17} + 44x^{15} + 22x^{13}$
$P1(x) * 0x^{45} = 0$

$$P1(x)*P2(x) = 56x^{15} + 48x^{14} + 32x^{12} + 16x^{10} + 35x^{11} + 30x^{10} + 20x^8 + 10x^6 + 21x^{10} + 18x^9 + 12x^7 + 6x^5$$

$$P1(x)*P2(x) = 56x^{15} + 48x^{14} + 32x^{12} + 35x^{11} + 67x^{10} + 18x^9 + 20x^8 + 12x^7 + 10x^6 + 6x^5$$

$(x-1)^2 = x^2-2x+1$ $(x-1)(x+1) = x^2-1$ $(x-1)^3 = x^3-3x^2+3x-1$
$(x-3)^2 = x^2-6x+9$ $(x-3)(x+3) = x^2-9$ $(x-2)^3 = x^3-6x^2+12x-8$

Algorithmique

1. Proposez une structure statique permettant de manipuler un polynôme à une variable ayant au plus MaxC coefficients non nuls.
2. Ecrivez un algorithme affichant sur la sortie standard « clairement » le polynôme stocké.
3. Ecrivez l'algorithme calculant la somme de deux polynômes.
4. Ecrivez l'algorithme calculant le produit de deux polynômes.
5. Ecrivez l'algorithme calculant le carré d'un polynôme.
6. Ecrivez l'algorithme calculant le cube d'un polynôme.
7. Proposez un format de stockage d'un polynôme dans un fichier.
    a. Ecrivez l'algorithme de sauvegarde d'un polynôme dans un fichier.
    b. Ecrivez l'algorithme récupérant dans un fichier un polynôme.

Programmation

1. Ecrivez le fichier d'entête « `polynome.h` » contenant les déclarations des constructeurs, des destructeurs, méthodes de la classe `polynome`, ainsi que l'ensemble de ces membres.

2. Ecrivez le fichier « `polynome.c` » contenant la définition des divers constructeurs, destructeurs et méthodes de la classe `polynome`.

Dossier à rendre début janvier 2009 :

1. Présentation de votre structure de données, permettant de manipuler un polynôme.
2. L'algorithme calculant la somme de deux polynômes, indiquez sa complexité en nombre de comparaison sen fonction de la taille des polynômes à additionner
3. L'algorithme calculant le produit de deux polynômes. Indiquez sa complexité en nombre de comparaison en fonction de la taille des polynômes à multiplier.
4. Le code facilement lisible (attention aux noms des méthodes, noms des variables, aux commentaires sur les méthodes, ...).
5. Jeu de tests que vous avez utilisé pour tester et valider votre code.
6. Un état honnête d'avancement du projet (précisant les fonctionnalités qui n'ont pas été implémentées, si vous avez détecté des bugs, ...).

Optionnel :

1. Structure dynamique- Concevez et programmez la classe « `DynamiquePolynome` » permettant de manipuler des polynômes de degré quelconque. Vous pouvez, si vous le souhaitez, utiliser la STL (standard library of C++), mais cela n'est pas obligatoire.

2. Ajoutez à votre classe la méthode « `puissance(n : entier)` » qui calcule le polynôme à la puissance `n`.

```
fonction puissanceOpt (px : polynome, puissance : entier) : polynome
// Calcule px^b par multiplications en minimisant nb. opérations
var résidu, résultat : polynome
    exposant : entier
début
    //On utilise le fait que px^(2k+1) = (px * (px)^2k  et que px^2k = (px*px)^k
   résultat ← px ; résidu ← 1 ; exposant ← puissance ;
   tant Que exposant > 1 faire début
           si exposant mod 2 = 1 alors résidu ← résidu*résultat ;
           résultat ← résultat * résultat ; exposant ← exposant div 2 ;
   fin
   résultat ← résultat * résidu ; retourner(résultat) ;
fin
```