Les polynômes attaquent

(ou : un projet sur les polynômes)

DUT Informatique
IUT de Vélizy

1 But du projet

Le but de ce projet est de pouvoir effectuer des calculs sur des polynômes à coefficients entiers à une variable, on voudrait par exemple que le programme puisse nous dire que

$$(X^2+3X)^5-(X-1)^2 = X^{10}+15X^9+90X^8+270X^7+405X^6+243X^5-X^2+2X-1.$$

2 Rappels de mathématiques

2.1 Notion de polynôme

On appelle *polynôme à coefficients entiers* une expression mathématique de la forme

$$P = a_nX^n + a_{n-1}X^{n-1} + \cdots + a_1X + a_0$$

où $a_i \in \mathbb{Z}$ pour $0 \leq i \leq n$, et X est une indéterminée.

2.2 Opérations sur les polynômes

On peut additionner et soustraire les polynômes entre eux en procédant coefficient par coefficient, et on peut multiplier les polynômes suivant les règles de distributivité habituelles. On rappelle que

$$aX^s \times bX^t = (a \cdot b)X^{s+t},$$

par exemple

$$3X^5 \times -2X^8 = -6X^{13}.$$

2.3 Notion de monôme et de degré

Une expression de la forme aX^k avec $a \neq 0$ est appelée **un monôme de degré** k.
Le **degré d'un polynôme** est par définition le plus grand degré de ses monômes.
Exemple :

$$P = 5X^3 - X + 1$$

- P est constitué de trois monômes: $5X^3$, $-X$ et 1, de degrés respectifs 3, 1 et 0
- Le degré de P est 3 car le plus grand degré parmi ses monômes est 3.

3 Stocker des polynômes en mémoire

3.1 Structure pour les monômes

On stockera un polynôme comme une structure contenant un tableau de monômes.
Un monôme est défini ainsi :

```
typedef struct monome {
   int coeff;
   int degre;
   } Monome;
```

Cette structure contient le degré (un entier *positif ou nul*) du monôme, ainsi que son coefficient (le nombre devant le X) qui est un entier positif ou négatif.
Dans ce projet, on "réduira" les polynômes de sorte à éliminer les monômes dont le coefficient est 0, de sorte que *seuls les monômes de coefficient non nuls* soient conservés (définir un degré pour les monômes nuls est problématique).

3.2 Struct pour les polynômes

Pour stocker un polynôme, on stocke l'ensemble de ses monômes dans un tableau de longueur 50 (modifiable, devrait suffire), tous regroupés au début du tableau et **dans l'ordre décroissant des degrés**. On devra aussi se rappeler du nombre de monômes qui sont stockés dans ce tableau.

```
typedef struct polynome {
    Monome      tab_monomes[50]; // les monomes du polynome
    int         nb_monomes;      // le nombre de monomes
    } Polynome;
```

Attention, les indices du tableau `tab_monomes` n'ont pas de sens particulier lié au degré ou au coefficient, on se contente de ranger les monômes à la suite dans ce tableau.

Par la suite, on dira qu'un polynôme est

- **réduit**, si l'on a regroupé les monômes de même degré en en faisant la somme (par exemple, le tableau ne contient pas le monôme $2X^5$ suivi du monôme $4X^5$, il contient à la place le monôme $6X^5$) et il ne contient pas de monôme nul (coefficient zéro, ce qui peut arriver après une addition par exemple)
- **trié**, si les monômes sont bien par ordre décroissant de degré dans le tableau.

3.3 Exemple d'ordre général

Si le polynôme

$$P_1 = 5X^4 - X + 1$$

est stocké dans une variable déclarée comme `Polynome p1 ;` on devra avoir `p1.nb_monomes` qui vaut 3, et le tableau `p1.tab_monomes` sera tel que

```
p1.tab_monomes[0].coeff = 5     p1.tab_monomes[0].degre = 4
p1.tab_monomes[1].coeff = -1    p1.tab_monomes[1].degre = 1
p1.tab_monomes[2].coeff = 1     p1.tab_monomes[2].degre = 0
```

Peu importe ce que contiennent les 47 autres monômes du tableau, puisqu'on ne va pas les consulter (de la même façon que pour une chaîne de caractères stockée dans un tableau, peu importe ce qui se trouve après le caractère nul.)

3.4 Exemples plus particuliers

Mentionnons le cas particulier du polynôme nul :

$$P_2 = 0.$$

Pour ce polynôme, on aura directement `p1.nb_monomes` qui vaut 0, puisque le polynôme ne contient aucun monômes de *coefficient non nul*. Pour ce polynôme le degré n'est pas défini, on convient parfois qu'il vaut $-\infty$; nous dirons pour simplifier que son degré est -1.
Ne pas confondre le polynôme nul avec un polynôme de degré 0, comme par exemple

$$P_3 = -4.$$

Ce polynôme contient un unique monôme, et donc `p3.nb_monomes` vaut 1. On aura alors dans la structure

```
p1.tab_monomes[0].coeff = -4    p1.tab_monomes[0].degre = 0
```

4 Le projet

4.1 Les fonctions mises à votre disposition

Vous sont fournis des fichiers `polynome.h` et `polynomes.c` qui contiennent la définition des structures ainsi que plusieurs fonctions de base vous permettant de manipuler des polynômes:

- `initPolynome` : initialise un polynôme passé en paramètre (met simplement son nombre de monômes à zéro) ;
- `str2Polynome`: reçoit une chaîne de caractères contenant le texte d'un polynôme, et construit la structure `Polynome` associée. Cette fonction fait appel à `standardiseDescription` qui fait briller votre polynôme: elle retire les caractères non reconnus dans le texte, et change les $'x'$ en $'X'$, remplace les $'-X'$ par des $'+-1X'$ pour mieux découper la chaîne, entre autres.
  Au final, `str2Polynome` est sympa: elle accepte un format d'entrée très souple pour vos monômes: − `5x^3` ou `-5x3` ou `-5X3` ou `+-5x3` ou même `- 5 X3`. Soyez quand même gentils avec elle.

Attention: `str2Polynome` ne réduit pas et ne trie pas le polynôme saisi. Vous devrez donc ne saisir que des polynômes réduits et triés par degrés décroissants... tant que vous n'aurez pas écrit les fonctions effectuant ce travail.

4.2 Votre travail

Vous devez écrire les fonctions qui suivent, par ordre de difficulté croissante. Les étoiles devant chaque fonction indiquent le niveau de difficulté.

4.2.1 * affichePolynome

Cette fonction affiche un polynôme passé en paramètre. Dans les fichier fournis, nous avons écrit le prototype de cette fonction que vous devez donc compléter. Une fois écrite, vous pouvez compiler le programme de base que nous vous fournissons et qui affiche un polynôme initialement passé sous forme de texte. Amusez-vous à modifier le polynôme.

Votre affichage doit respecter les règles qui suivent...

Règle	**Correct**	**Incorrect**
Un ^ pour les puissances	`3X^5`	`3X5`
Pas de puissance affichée si cette puissance vaut 1	`3X`	`3X1` ou `3X^1`
Pas de puissance ni de X si la puissance est nulle	`5`	`5X0` ou `5X^0`
Pas de coefficient affiché si ce coefficient vaut 1	`X^2`, $-$`X`	`1X^2`, $-$`1X`
Pas de + inutiles	`3X^5` $-$ `2X^3`	+`3X^5` + $-$`2X^3`
Polynôme nul affiché correctement	`0`	
Exemple	`3X^2` $-$ `X` + `5`	+`3X2` + $-$`1X` + `5X0`

4.2.2 * multiplieMonomePolynome

Cette fonction et toutes les suivantes qui le nécessitent reçoivent dans leurs paramètres un polynôme dont le contenu sera écrasé pour y écrire le résultat des calculs.
Cette fonction multipie un polynôme donné par un monôme donné. Remarquez que, si le polynôme fourni est bien trié et réduit, la multiplication par un monôme produit un polynôme également trié et réduit. Cette fonction devra donc être capable d'effectuer, par exemple, le calcul

$$-2X^2 \times (X^5 - 3X + 2) = -2X^7 + 6X^3 - 4X^2$$

4.2.3 ** ajouteMonomePolynome

Cette fonction ajoute un monôme donné à un polynôme donné. Deux cas sont à considérer afin que le polynôme produit demeure réduit et trié:

- soit le degré du monôme à ajouter existe déjà dans le polynôme, auquel cas il faut sommer les coefficients concernés
- soit ce degré n'apparaît pas, auquel cas il faut insérer le monôme au bon endroit dans le polynôme afin de bien conserver l'ordre des degrés décroissants.

4.2.4 ** ajoutePolynomePolynome1

Cette fonction ajoute un polynôme donné à un autre polynôme donné. Pour ce faire, vous appellerez simplement `ajouteMonomePolynome` avec chacun des monômes d'un des deux polynômes.
Quelle est la complexité de ce nouvel algorithme ? Précisez le dans la spécification de la fonction, en fonction des longueurs m, n des deux polynômes.

4.2.5 *** ajoutePolynomePolynome2

Cette fonction permet d'améliorer la complexité de la fonction précédente: considérant toujours que les deux polynômes sont triés, vous devez appliquer un algorithme similaire à la fusion croissante (voir TP d'algo sur le tableaux, ex 8): on parcourt simultanément les deux polynômes afin de construire le nouveau polynôme avec les monômes de plus grands degrés qu'on rencontre.
Quelle est la complexité de ce nouvel algorithme ? Précisez le dans la spécification de la fonction, en fonction des longueurs m, n des deux polynômes.

4.2.6 *** multipliePolynomePolynome

Cette fonction multipie un polynôme donné par un autre polynôme donné. Pour ce faire, vous devez pour chaque monôme du premier polynôme :

- le multiplier par l'autre polynôme à l'aide de `multiplieMonomePolynome`
- ajouter le résultat dans le polynôme résultat à l'aide de `ajouterPolynomePolynome`

En respectant ces directives, le polynôme obtenu est trié et réduit.

4.2.7 ** puissancePolynome

Cette fonction se base sur `multiplePolynomePolynome` pour calculer les puissances d'un polynôme, comme par exemple $(X^3 - X + 2)^4$. Attention lors de vos tests car le nombre de monômes peut rapidement dépasser 50. (**votre programme doit bien entendu être sécurisé pour éviter ce genre de déconvenues! Monteriez-vous dans un avion... etc**)

4.2.8 *** triPolynome

Écrivez cette fonction de tri qui permettra d'ordonner correctement les monômes d'un polynôme (réduit ou non, peu importe). Cette fonction sera par exemple utilisée après la saisie d'un polynôme par l'utilisateur, afin de trier ce polynôme. L'algorithme à utiliser peut être celui de votre choix, comme par exemple le tri par sélection vu en cours. La fonction reçoit un polynome qu'elle modifie directement.

4.2.9 *** reduitPolynomeTrie

L'utilisateur peut saisir des polynômes qui ne sont pas réduits: il peut exister plusieurs monômes de même degré. Cette fonction, recevant un **polynôme préalablement trié** va réduire ce polynôme. Employez l'algorithme de votre choix: il est possible d'écrire un algorithme *en place* (sans créer de polynôme supplémentaire) ou non, mais il doit être linéaire.

4.3 Interface

Écrivez un menu simple pour votre programme, qui comportera une section "démonstrations" où votre programme présente au choix quelques calculs complexes (sommes, multiplications, puissances) afin de prouver qu'il fonctionne correctement.
Une fois tout ceci effectué, vous pouvez vous employer à améliorer l'interface de votre programme afin de le rendre interactif. Vous pouvez par exemple dans votre menu créer un mode "calculatrice" invitant l'utilisateur à saisir des polynômes, à choisir les opérations à effectuer sur ces polynômes, etc. Votre imagination est la seule limite.

4.4 Compétences

Ce projet recouvre des compétences algorithmes et de programmation en C. A priori, vous serez évalués sur des compétences à choisir parmi les suivantes:

- En programmation:
  - Savoir écrire et comprendre des boucles imbriquées
  - Être à l'aise avec les pointeurs (&, *, passages de paramètres)
  - Accès dans les structures et tableaux imbriqués
  - Commenter son code, nommer correctement ses fonctions et variables
  - Considérer tous les cas particuliers d'un programme
  - Rendre votre programme esthétique, intéressant, attrayant, jouable (concerne les projets qui auront travaillé sur la partie Interface)
  - Incorporer des solutions techniques innovantes (concerne l'interface)
- En algorithmique:
  - Connaître les algorithmes de tri
  - Utiliser des tableaux croissants
  - parcours simple d'un tableau 1D
  - parcours avancé d'un tableau 1D