Mini-projet : calculatrice à polynômes

Dans ce projet, vous allez programmer en langage C une mini-calculatrice qui effectue des opérations élémentaires sur les polynômes : additions, soustractions, multiplications. Le sujet est composé de trois parties : en premier lieu, des rappels mathématiques sur les polynômes, la deuxième partie donne le cahier des charges du projet et la troisième partie fournit des indications et aides techniques pour vous guider dans le développement.

Dans tout le sujet, nous travaillerons uniquement avec des polynômes à coefficients entiers, appartenant à $\mathbb{Z}$.

1 Rappels mathématiques sur les polynômes

Définition (Polynôme)**.** On appelle *polynôme* à coefficients entiers, une expression algébrique de la forme :

$$P(X) = \sum_{k=0}^{n} a_k X^k, \qquad \text{où} \quad \begin{cases} n \in \mathbb{N} \\ \forall k \in [\![0, n]\!], a_k \in \mathbb{Z} \\ a_n \neq 0 \end{cases}$$

L'entier n est apppelé *degré* du polynôme P et les entiers $(a_k)_{0 \leq k \leq n}$ sont les *coefficients* de P. Pour chaque $k \in [\![0, n]\!]$, on dit que $a_k X^k$ est le *monôme de degré* k.

Remarque (Le polynôme nul)**.** Le polynôme dont tous les coefficients sont égaux à zéro est appelé le *polynôme nul*. Par convention, il est de degré $-\infty$. Ce cas particulier ne sera pas traité dans le cadre de ce projet.

Définition. L'ensemble des polynômes à coefficients dans $\mathbb{Z}$ est noté $\mathbb{Z}[X]$. Si $n \in \mathbb{N}$, l'ensemble des polynômes de degré inférieur ou égal à n est noté $\mathbb{Z}_n[X]$.

Dans toute la suite, on se donne deux polynômes P, Q de degrés respectifs $n, m \in \mathbb{N}$, avec :

$$P(X) = \sum_{k=0}^{n} a_k X^k \qquad Q(X) = \sum_{k=0}^{m} b_k X^k$$

Pour simplifier les notations, on suppose que $m \geq n$, quitte à échanger les rôles de P et Q.

Définition (Opposé d'un polynôme)**.** On appelle *opposé* du polynôme $P(X)$, le polynôme noté $-P(X)$ défini par :

$$-P(X) = \sum_{k=0}^{n} -a_k X^k$$

En termes simples, le polynôme $-P(X)$ est celui obtenu en prenant l'opposé de tous les coefficients de P.

Définition. On appelle *somme* de P et Q et on note $(P+Q)(X)$, le polynôme défini par :

$$(P+Q)(X) = \sum_{k=0}^{n} (a_k + b_k) X^k + \sum_{k=n+1}^{m} b_k X^k$$

Remarque. Derrière cette définition un peu lourde, l'idée est en fait très simple : la somme de $P(X)$ et $Q(X)$ est obtenue en effectuant la somme coefficient par coefficient. Mais comme les polynômes ne sont pas nécessairement de même degré ($m \geq n$), la définition doit prendre en compte les indices $k \in [\![n+1, m]\!]$ de manière séparée.

Définition (Différence de deux polynômes)**.** On appelle *différence* de P et Q le polynôme noté $(P-Q)(X)$, défini par $(P-Q)(X) = (P+(-Q))(X)$.

Il nous reste à définir le *produit* de deux polynômes. La définition est un peu plus délicate, car il nous faut prendre en compte le fait que la multiplication est distributive sur l'addition.

Définition (Produit de deux polynômes)**.** On appelle *produit* de $P(X)$ et $Q(X)$, le polynôme noté $(P \cdot Q)(X)$ et défini par :

$$(P \cdot Q)(X) = \sum_{k=0}^{m+n} c_k X^k, \quad \text{où} \quad \forall k \in [\![0, m+n]\!], \; c_k = \sum_{\ell=0}^{k} a_\ell b_{k-\ell}$$

avec

$$\begin{cases} a_\ell = 0 \text{ si } \ell \geq n \\ b_{\ell-k} = 0 \text{ si } k - \ell \geq m \end{cases}$$

Théorème. Les opérations précédemment définies ont les mêmes propriétés que celles définies sur $\mathbb{Z}$.

Étudiez des exemples !

Les définitions données ici sont abstraites et théoriques. Pourtant, elles contiennent toutes les informations dont vous aurez besoin pour programmer votre calculatrice.

Pour vous forger une bonne image mentale et comprendre comment fonctionnent ces définitions, il est conseillé de les appliquer sur des exemples. Choisissez deux polynômes $P(X)$ et $Q(X)$ ayant des coefficients entiers simples, et calculez leur somme, leur différence et leur produit en vous appuyant sur les définitions.

2 Cahier des charges

Votre calculatrice devra consister en un programme C qui s'exécute dans la console. Il s'agira d'une calculatrice simplifiée, qui effectuera des opérations sur seulement deux polynômes, que l'on notera P et Q.

Chaque polynôme sera saisi au clavier au moyen de ses coefficients. Une fois les deux polynômes créés, on pourra choisir entre les trois opérations au moyen d'un menu. Votre programme proposera un menu principal, dans lequel on pourra choisir entre plusieurs actions :

1) Initialiser ou modifier le plynôme P.

2) Initialiser ou modifier le plynôme Q.

3) Calculer $P + Q$.

4) Calculer $P - Q$.

5) Calculer $P \cdot Q$.

6) Quitter.

À la page 6 en annexes, vous trouverez un exemple d'exécution du programme. Ce n'est évidemment pas la seule présentation possible, elle est proposée pour vous guider.

Votre code source devra être organisé de façon modulaire, découpé en fonctions indépendantes les unes des autres. Vous veillerez à correctement nommer vos variables et vos fonctions.

Travail à rendre

Vous devrez rendre :

- l'ensemble du projet `Code::Blocks`, contenant notamment votre code source correctement indenté et commenté ;
- Un rapide compte-rendu (pas plus de 3 pages) *au format PDF*, analysant les difficultés rencontrées et faisant le résumé des éventuelles choses que vous n'auriez pas réussi à programmer.

L'ensemble de votre travail est à déposer sur EPREL dans une archive ZIP nommée `R101_miniProjet_<NOMDEFAMILLE>.zip`. Bien sûr, le code fourni devra compiler sans erreurs ni warnings.

3 Cahier technique

Cette section vous propose quelques pistes et indications pour vous guider dans la programmation.

3.1 Représenter un polynôme en C

Nous proposons de représenter un polynôme par un tableau. Si $n \in \mathbb{N}$ est le degré d'un polynôme $P(X) = \sum_{k=0}^{n} a_k X^k$, le code pourra contenir une variable `int degP = n` contenant le degré de P. On pourra aussi créer un tableau `int[] P` de taille `n+1`. La case `P[i]` contiendra le coefficient du monôme de degré `i`.

Exemple. Le polynôme $P(X) = 7x^2 + 8x - 1$ sera représenté par la variable `int degP = 2` et le tableau `P[] = {-1, 8, 7}`. Le poynôme $Q(X) = 8x^3 + 5x^2 - 3x - 2$ pourra être représenté par la variable `int degQ = 3` et le tableau `int Q[] = {-2, -3, 5, 8}`.

Comme le degré des polynômes P et Q ne sera pas connu lors de la compilation, mais uniquement lors de l'exécution, *les tableaux devront être alloués de manière dynamique* en utilisant la fonction `malloc`. Vous demanderez au clavier le degré souhaité, puis vous allouerez dynamiquement de la mémoire en quantité suffisante, et en dernier lieu vous demanderez de saisir les coefficients.

3.2 Bien s'organiser

Il est important de procéder de manière méthodique, en commençant par programmer les fonctionnalités les plus simples : affichage d'un polynôme, puis initialisation d'un polynôme, puis les opérations demandées.

Lorque vous développez une fonctionnalité, développez-la entièrement avant de passer à la suite. N'hésitez pas à faire plusieurs tests. Veillez à découper votre code en fonctions simples : chaque fonction doit être *atomique* : elle avoir une seule chose à faire et la faire correctement.

Il ne faut pas hésiter à compiler votre code très souvent et *à lire attentivement les messages d'erreurs* lorsque la compilation échoue.

3.3 Structuration du code source

Nous vous proposons d'organiser votre code source en trois parties.

Le fichier `main.c` contiendra les variables principales du programmme (les tableaux pour les polynômes et les variables pour les degrés).

Les fichiers `polynome.h` et `polynome.c` contiendront toutes les fonctions utiles à la manipulation des polynômes : création, modification, somme, différence, produit et éventuelles fonctions annexes.

Les fichiers `interface.h` et `interface.c` contiendront les fonctions gérant les menus et les affichages dans la console.

Le fichier `main.c` ne doit contenir que le strict minimum et uniquement la fonction `int main(void)` ; les fonctionnalités principales seront développées dans `polynome.c`.

3.4 Quelques prototypes de fonctions à programmer

Cette partie vous donne une proposition fichier `polynome.h` (éventuellement à compléter si besoin) avec des fonctions à programmer et leur prototype. *Ce ne sont que des propositions données à titre informatif*, il n'est pas obligatoire de les suivre. Vous pouvez procéder différemment, d'autres solutions sont possibles. En tout état de cause, vous êtes libres de créer toutes les fonctions auxiliaires que vous estimerez utiles.

polynome.h

```c
/* Fonctions d'affichage et d'initialisation */
void affichePolynome(int deg, int* P);
int initDegrePolynome();
int* initPolynome(int deg);

/* Opérations arithmétiques usuelles (+ - *) */
int* opposePolynome(int deg, int* P);
int* sommePolynomes(int degP, int* P, int degQ, int* Q, int* degS);
int* differencePolynomes(int degP, int* P, int degQ, int* Q, int* degD); // P-Q
int* produitPolynomes(int degP, int* P, int degQ, int* Q, int* degPdt);
```

Rappel sur les tableaux et pointeurs

On rappelle qu'en langage C, un tableau est en fait un *pointeur* vers l'adresse du premier élément. Ainsi, le paramètre `int* P` dans le prototype de la fonction `void opposePolynome(int deg, int* P)` peut se comprendre comme l'adresse une variable de type `int` *et* comme l'adresse du premier élément d'un tableau.

Dans notre cas, c'est la deuxième interprétation qui est la bonne : `P` est l'adresse du premier élément d'un tableau contenant les coeffcients du polynôme dont on souhaite calculer l'opposé. Le paramètre `int deg` est le degré du polynôme, donc le nombre d'éléments du tableau `P` est égal à `n+1`.

On donne ci-dessous les spécifications de chaque fonction.

- `void affiche(int deg, int* P);` — Affiche le polynôme `int* P` de degré `int deg`.
- `int initDegrePolynome();` — Demande de saisir au clavier le degré du polynôme que l'on souhaite initialiser et le renvoie en valeur de retour.
- `int initPolynome(int deg);` — Reçoit l'entier `int deg` en paramètre, alloue de la mémoire pour créer un polynôme de degré `deg`, demande la saisie des coefficients et renvoie l'adresse du polynôme créé.
- `int* opposePolynome(int deg, int* P);` — Reçoit un polynôme et son degré en paramètre, initialise le polynôme opposé et renvoie un pointeur vers celui-ci.
- `int* sommePolynomes(int degP, int* P, int degQ, int* Q, int* degS);` — Reçoit les polynômes `P` et `Q` en paramètre avec leurs degrés respectifs, calcule leur somme et renvoie un pointeur vers le polynôme calculé. Le degré de la somme est modifié par référence au pointeur `int* degS`.
- `int* differencePolynomes(int degP, int* P, int degQ, int* Q, int* degD);` — Même chose que pour la somme, mais calcule cette fois la différence.
- `int* produitPolynomes(int degP, int* P, int degQ, int* Q, int* degPdt);` — Même chose que pour la somme, mais calcule cette fois le produit.

Les prototypes indiqués ne sont que ceux des fonctions les plus importantes, vous pouvez créer des fonctions auxiliaires en fonction de vos besoins. Il n'est pas obligatoire de respecter les prototypes fournis.

Le mot de la fin

Ce travail fait l'objet d'une évaluation. Le bon fonctionnement du programme est évidemment un critère, mais ce n'est pas le seul ! L'organisation de votre code, le respect des principes de nommage des variables et fonctions, le bon usage des commentaires sont également des items importants de la note.

Les choix de programmation, l'utilisation logique des boucles et des conditions, la bonne manipulation des tableaux et la qualité de structuration de votre code source sont importants.

Il n'est pas utile, et encore moins souhaitable, de chercher à « masquer » les difficultés rencontrées. Il est préférable d'adopter une démarche scientifique et analytique en effectuant un bilan, dans votre compte-rendu, des problèmes rencontrés, des pistes évoquées pour les résoudre — que ces pistes aient été fructueuses ou non.

Annexes

polynomes.exe

```
=== MENU PRINCIPAL ===
Que voulez-vous faire ?
1. Initialiser ou réinitialiser le polynôme P
2. Initialiser ou réinitialiser le polynôme Q
3. Calculer P+Q
4. Calculer P-Q
5. Calculer P×Q
6. Sortir
Choix : 1

==== DEGRÉ DU POLYNÔME P ====
Saisir le degré du polynôme : 2
Saisir le monôme de degré 0 : -2
Saisir le monôme de degré 1 : 4
Saisir le monôme de degré 2 : 3
P[X] = 3×X^2 + 4×X^1 + -2

=== MENU PRINCIPAL ===
Que voulez-vous faire ?
1. Initialiser ou réinitialiser le polynôme P
2. Initialiser ou réinitialiser le polynôme Q
3. Calculer P+Q
4. Calculer P-Q
5. Calculer P×Q
6. Sortir
Choix : 2

==== DEGRÉ DU POLYNÔME Q ====
Saisir le degré du polynôme : 1
Saisir le monôme de degré 0 : -2
Saisir le monôme de degré 1 : 6
Q[X] = 6×X^1 + -2

=== MENU PRINCIPAL ===
Que voulez-vous faire ?
1. Initialiser ou réinitialiser le polynôme P
2. Initialiser ou réinitialiser le polynôme Q
3. Calculer P+Q
4. Calculer P-Q
5. Calculer P×Q
6. Sortir
Choix : 5
(P×Q)[X] = 20×X^3 + 2×X^2 + -16×X^1 + 4

=== MENU PRINCIPAL ===
Que voulez-vous faire ?
1. Initialiser ou réinitialiser le polynôme P
2. Initialiser ou réinitialiser le polynôme Q
3. Calculer P+Q
4. Calculer P-Q
5. Calculer P×Q
6. Sortir
Choix : 6
```