CALCULATRICE DES POLYNOMES

ZAKARIA EL HAJJAM
ALI HARIT

Encadré par:
P.Dargham

ENSA KHOURIBGA

SOMMAIRE

INTRODUCTION

le projet est une calculatrice de polynomes interactive implémentée en langage C. Cette application offre une gamme complète de fonctionnalités mathématiques, permettant à l'utilisateur de manipuler et d'effectuer des opérations sur des polynômes. Ces opérations incluent l'addition, la soustraction, la multiplication, la dérivation, l'intégration, l'évaluation, ainsi que la sauvegarde et la récupération de polynômes. Le programme utilise des structures de données pour représenter les fractions et les polynômes, optimisant ainsi la gestion des calculs. L'interface utilisateur, fonctionnant en mode console, propose des commandes intuitives pour exécuter ces opérations, offrant ainsi une expérience interactive avancée. Le projet met en avant l'application de concepts de programmation avancés en C, tels que les allocations dynamiques de mémoire, les structures et les fonctions récursives, pour créer une calculatrice polynomiale fonctionnelle et polyvalente.

LES STRUCTURES DES DONNÉES

La manipulation de ce projet nécessite l'utilisation de structures de données, parmi lesquelles deux structures sont cruciales pour bien gérer les opérations sur les polynômes :

1. Fraction : Cette structure de données vise à définir le coefficient d'un monôme à travers un numérateur et un dénominateur.
2. Polynôme : Cette structure utilise le type fraction ainsi qu'un exposant. De plus, elle est chaînée avec les monômes suivants grâce au type struct next.

```
typedef struct fraction
{
    int num;
    int denom;
} fraction;
typedef struct polynome
{
     fraction coefficient;
     int exposant;
     struct polynome *next;
} polynome;
```

LES FONCTIONS PRÉDÉFINIS

fgets()

La fonction fgets (file gets) est utilisée pour lire une chaîne de caractères depuis l'entrée standard jusqu'à ce qu'un caractère de fin chaine (**'\0'**) soit rencontré ou que la taille maximale spécifiée soit atteinte.

Extraction du code : `fgets(command, 100, stdin);`

saisie de la commande en utilisant fgets qui permet de saisir une chaine de caractere sans depasser la taille de 100 et stdin pour lire a partir du clavier ainsi que le contenue lue est stocké sur la variable command.

strcmp ()

La fonction strcmp (string compare) est utilisée pour comparer deux chaînes de caractères.

Extraction du code : `if (strcmp(cutString(command, 0, 4), "EXIT") == 0){}`

La fonction renvoie un entier négatif si str1 = **cutString(command, 0, 4)** est inférieur à str2 = **"EXIT"**, zéro si les deux chaînes sont égales, et un entier positif si **str1** est supérieur à **str2**.

sscanf()

La fonction **sscanf** en C (String Scan Formatted) est utilisée pour analyser une chaîne de caractères (string) en fonction d'un format spécifié, tout comme **scanf**, mais au lieu de lire à partir de la console, elle lit à partir d'une chaîne de caractères.

Extraction du code : `if (sscanf(cutString(command, 6, 1), " %d", &n) == 1){}`

- prend cette sous-chaîne comme premier argument, et le deuxième argument est le format attendu. Dans ce cas, le format est " **%d**". Le **%d** indique que la fonction doit s'attendre à lire un entier, et l'espace avant **%d** est utilisé pour gérer les espaces blancs éventuels dans la chaîne.
- Le troisième argument de **sscanf** est l'adresse de la variable dans laquelle stocker la valeur lue, ici **&n**.

isspace()

isspace est une fonction utilisée pour vérifier si un caractère est un espace (espace, tabulation, retour à la ligne, retour de chariot, etc.).

Extraction du code : `while (isspace(input[i])){}`

isspace(input[i]) vérifie si le caractère à la position i dans la chaîne input est un espace blanc. Si c'est le cas, la boucle while continue

isdigit()

isdigit est une fonction utilisée pour vérifier si un caractère est un chiffre décimal (0-9).

Extraction du code : `while (isdigit(input[i])){}`

isdigit(input[i]) vérifie si le caractère à la position i dans la chaîne input est un chiffre. Si c'est le cas, la boucle while continue

LES FONCTIONS

monome()

la fonction **monome** est utilisée pour créer un monôme avec un coefficient fractionnaire et un exposant donnés, et renvoie un pointeur vers cette structure polynome.

cutString()

la fonction **cutstring** prend en paramètres une chaîne de caractères source, un indice de début startIndex, et une longueur length. Elle crée et retourne une nouvelle chaîne de caractères (substring) extraite de la chaîne source, en commençant à l'indice startIndex et en copiant length caractères.

Somme()

la fonction samme() prend en paramètres deux fractions (f1 et f2) ainsi qu'une variable entière op pour indiquer l'opération à effectuer (0 pour l'addition, autre chose pour la soustraction). Elle retourne une fraction résultante de l'opération spécifiée.

division()

La fonction division prend en paramètre une fraction f1 et un entier exp. Elle retourne une nouvelle fraction où le numérateur est le même que celui de f1, mais le dénominateur est multiplié par l'entier exp.

produit()

La fonction produit prend en paramètre deux fractions (f1 et f2). Elle retourne une nouvelle fraction résultant du produit des deux fractions.

LET()

fonctionnement général

la fonction **LET** analyse une chaîne de caractères input qui représente un polynôme et construit une liste chaînée de monômes (termes du polynôme) triée par ordre décroissant des exposants. Elle utilise la fonction monome pour créer des monômes à partir des coefficients et exposants extraits de la chaîne. La fonction prend en compte les signes, les coefficients fractionnaires, les exposants et les espaces éventuels dans la chaîne d'entrée.

Prototype

La boucle principale parcourt la chaîne caractère par caractère et effectue les actions suivantes :

- Elle détecte le signe du monôme (+ ou -), le numérateur et le dénominateur du coefficient, ainsi que l'exposant.
- En fonction des informations extraites, elle crée un monôme à l'aide de la fonction monome.
- Elle insère le monôme dans la liste chaînée résultante à la position appropriée en fonction de l'exposant. Si un monôme avec le même exposant existe déjà dans la liste, elle effectue la somme des coefficients.
- Elle continue le processus jusqu'à la fin de la chaîne, en ignorant les espaces.

La fonction retourne un pointeur vers le début de la liste chaînée, qui représente le polynôme construit à partir de la chaîne d'entrée.

SET()

La fonction **SET** prend en paramètre un pointeur vers un pointeur de polynôme poly et une chaîne de caractères miseajour. Elle libère d'abord la mémoire occupée par le polynôme existant en parcourant la liste chaînée et en utilisant la fonction free pour chaque monôme. Ensuite, elle met à jour le polynôme en appelant la fonction LET avec la chaîne de caractères miseajour. Ainsi, la fonction SET permet de remplacer le polynôme existant par un nouveau polynôme construit à partir de la chaîne de caractères fournie.

EVAL()

La fonction EVAL prend en entrée un polynôme nommé poly et une valeur entière a. Elle permet de calculer la valeur du polynôme pour la variable a en utilisant une approche récursive. Si le polynôme est vide (c'est-à-dire qu'il n'a pas de terme), la fonction renvoie 0.0. Dans le cas contraire, elle calcule la valeur du terme actuel du polynôme pour la variable a en utilisant la fonction pow de la bibliothèque math.h pour effectuer l'exponentiation. Ensuite, elle ajoute ce terme à la valeur du polynôme pour le terme suivant, calculé récursivement en appelant la fonction EVAL avec le polynôme suivant (poly->next) et la même valeur de a. Ce processus se répète jusqu'à ce que tous les termes du polynôme aient été évalués. En fin de compte, la fonction renvoie la somme totale des termes évalués, représentant ainsi la valeur du polynôme pour la valeur spécifiée de la variable

copierPolynome()

La fonction **copierPolynome** crée une copie identique d'un polynôme existant. Elle vérifie d'abord si le polynôme original est vide, auquel cas elle retourne NULL. Si le polynôme original n'est pas vide, la fonction parcourt chaque monôme du polynôme, crée des copies correspondantes, et les relie pour former un nouveau polynôme, qui est ensuite renvoyé. Ainsi, la fonction assure une duplication complète du polynôme original tout en préservant l'indépendance des deux structures de données.

INT()

La fonction **INT** prend en entrée un polynôme nommé monPoly et renvoie un nouveau polynôme représentant l'intégrale du polynôme d'origine. Pour calculer cette intégrale, la fonction utilise une méthode simple. Elle parcourt chaque terme du polynôme initial, augmente son exposant d'une unité, puis divise son coefficient par le nouvel exposant. Ce processus est répété pour chaque terme du polynôme d'origine. En fin de compte, la fonction renvoie le nouveau polynôme obtenu après cette opération d'intégration.

ajouterMonome()

fonctionnement général

La fonction **ajouterMonome** permet d'ajouter un monôme dans un polynôme tout en maintenant l'ordre décroissant des exposants. Le paramètre op indique si l'opération à effectuer est une somme (0) ou une soustraction (1).

Prototype

- Si le polynôme initial (**monPoly**) est vide (c'est-à-dire **NULL**), la fonction crée un monôme avec les coefficients (**coef**) et l'exposant (**exp**) fournis et le renvoie.
- Si l'exposant du monôme à ajouter est supérieur ou différent de l'exposant du premier monôme du polynôme, le nouveau monôme est créé et inséré au début du polynôme. Le signe du coefficient est ajusté en cas de soustraction (**op == 1**).
- Si les exposants sont égaux, la fonction parcourt récursivement le reste du polynôme pour trouver la position appropriée pour insérer le monôme.
- Si l'opération (**op**) est 0 (addition), la fonction effectue la somme des coefficients des monômes ayant le même exposant.
- Si l'opération est 1 (soustraction), elle effectue la soustraction des coefficients.
- Le résultat est mis à jour dans le polynôme initial.
- À chaque étape, le polynôme modifié est retourné

Utilisation

on a utiliser cette fonction dans la fonction **ADD**() qui fait l'addition de deux polynomes , et **MUL**() qui fait la multiplication de deux polynomes

MUL()

La fonction MUL permet de calculer le produit de deux polynômes, monPoly1 et monPoly2. Elle crée un nouveau polynôme vide (mul) et utilise deux boucles pour parcourir chaque terme des deux polynômes d'origine. Pour chaque paire de termes rencontrés, elle multiplie leurs coefficients pour obtenir un nouveau coefficient, et elle additionne leurs exposants pour obtenir le nouvel exposant du terme résultant. Ensuite, elle utilise la fonction ajouterMonome pour ajouter ce terme au polynôme résultat mul.

DER()

La fonction **DER** permet de calculer la dérivée d'un polynôme. Si le polynôme initial est vide, la fonction retourne NULL. Sinon, elle crée un polynôme temporaire représentant le polynôme à dériver. Pour chaque terme de ce polynôme temporaire, elle multiplie le coefficient par l'exposant et décrémente l'exposant. Ensuite, elle vérifie si le terme suivant est nul ou de degré zéro. Si c'est le cas, elle le supprime. Sinon, elle passe au terme suivant de manière récursive.

ADD()

La fonction **ADD** permet d'effectuer la somme ou la soustraction de deux polynômes. Elle prend en paramètre deux polynômes P et Q, ainsi qu'un indicateur op pour spécifier l'opération à effectuer (0 pour la somme, 1 pour la soustraction). La fonction crée un polynôme temporaire contenant le premier polynôme (somme) et un autre contenant le deuxième polynôme (R). Ensuite, elle parcourt le deuxième polynôme et utilise la fonction ajouterMonome pour ajouter chaque monôme au polynôme temporaire de la somme en respectant l'opération spécifiée. Finalement, la fonction retourne le polynôme résultant de la somme ou de la soustraction.

POW()

La fonction **POW** permet de calculer la puissance n d'un polynôme. Si la puissance est égale à zéro, elle renvoie un polynôme contenant un seul monôme avec un coefficient de 1 et un exposant de 0. Si la puissance est égale à un, elle renvoie simplement le polynôme d'origine. Pour toute autre puissance n, elle utilise une approche récursive en multipliant le polynôme par lui-même (n-1) fois en appelant la fonction MUL. Finalement, elle renvoie le résultat de cette multiplication, représentant la puissance n du polynôme initial.

AFFECT()

La fonction **AFFECT** permet de copier un polynôme source dans un autre polynôme destination. Elle commence par libérer la mémoire du polynôme de destination en parcourant chaque terme et en utilisant la fonction free. Ensuite, elle crée un nouveau polynôme de destination (dest) et un pointeur courant (current) pour le parcourir.
Ensuite, elle parcourt le polynôme source, copie chaque monôme, et l'ajoute au polynôme de destination. Si le polynôme destination est vide, elle ajoute le monôme au début. Sinon, elle l'ajoute à la fin. Enfin, elle met à jour le pointeur courant et passe au monôme suivant du polynôme source.
À la fin de la fonction, le polynôme destination est une copie complète du polynôme source.

DISPLAY()

fonctionnement général

La fonction **DISPLAY** permet d'afficher un polynôme. Elle prend en paramètre le polynôme à afficher (poly) et une variable ` idcsSign ` pour gérer le cas du premier monôme.

Prototype
Cette fonction procède comme suite :

- Elle commence par vérifier si le polynôme est vide. Si c'est le cas, elle affiche une ligne vide et retourne. Ensuite, elle vérifie si c'est le premier monôme. Si non, elle affiche le signe approprié en fonction du signe du coefficient.
- Ensuite, elle examine l'exposant du monôme actuel. Si l'exposant est zéro, elle affiche simplement le coefficient. Si l'exposant est différent de zéro, elle affiche le monôme complet avec le coefficient, le "x", et l'exposant. Elle gère également différents cas selon la valeur du coefficient et du dénominateur.
- Enfin, elle incrémente le test pour indiquer que ce n'est plus le premier monôme, passe au monôme suivant, et rappelle la fonction récursivement pour afficher le reste du polynôme.

MAIN FILE

Ce programme en langage C est une calculatrice en ligne de commande pour les opérations sur les polynômes. Il utilise une liste chaînée pour représenter un polynôme, où chaque nœud représente un terme du polynôme. La fonction principale lit les commandes de l'utilisateur et effectue l'opération demandée sur les polynômes P et Q.

Voici une explication des opérations
EXIT : Libère la mémoire allouée pour les polynômes P et Q et quitte le programme.
LET : Crée un nouveau polynôme P ou Q à partir de l'entrée de l'utilisateur s'il n'existe pas déjà.
SET : Modifie un polynôme existant P ou Q avec l'entrée de l'utilisateur.
INT : Calcule l'intégrale du polynôme P ou Q et l'affiche.
AFFECT : Copie le polynôme de P à Q ou de Q à P.
DER : Calcule la dérivée du polynôme P ou Q et l'affiche.
POW : Élève le polynôme P ou Q à la puissance d'un entier spécifié par l'utilisateur et affiche le résultat.
MUL : Multiplie les polynômes P et Q et affiche le résultat.
ADD : Additionne les polynômes P et Q et affiche le résultat.
SOUS : Soustrait le polynôme Q de P ou P de Q et affiche le résultat.
DISPLAY : Affiche le polynôme P ou Q.
EVAL : Évalue le polynôme P ou Q à une valeur spécifiée par l'utilisateur.

La fonction cutString est utilisée pour extraire des parties de la commande de l'utilisateur pour le traitement. La fonction strcmp est utilisée pour comparer ces parties aux commandes connues. La fonction fgets est utilisée pour lire la commande de l'utilisateur à partir de l'entrée standard.

Le programme utilise une boucle while pour lire et traiter continuellement les commandes jusqu'à ce que l'utilisateur entre la commande EXIT. Les instructions if et else if sont utilisées pour déterminer quelle opération effectuer en fonction de la commande de l'utilisateur.

Les fonctions LET, SET, INT, AFFECT, DER, POW, MUL, ADD, DISPLAY et EVAL sont probablement définies dans le fichier "function.c" qui est inclus en haut du programme. Ces fonctions effectuent les opérations réelles sur les polynômes.

EXECUTION

Ce screenshot illustre le processus d'enregistrement de deux polynômes, P et Q, avec leur affichage initial. Ensuite, il démontre la mise à jour du polynôme Q à l'aide d'un nouveau polynôme.

```
>> LET P=3X^7-1/2X^13
>> LET Q=4X^8
>> DISPLAY P
P=-1/2x^13 +3x^7
>> DISPLAY Q
Q=4x^8
>> SET Q=5X^8
>> DISPLAY Q
Q=5x^8
```

```
>> MUL P,Q
15x^15 -5/2x^21
>> POW P,2
9x^14 -12/4x^20 +1/4x^26
>> POW Q,3
125x^24
>> DER Q
40x^7
>> INT P
-1/28x^14 +3/8x^8
```

Ainsi ce screen montre un exemple d'exécution des fonctions multiplication et puissance et intégrale entre deux polynomes.

Cela démontre le processus d'assignation d'un polynôme à un autre et la réinitialisation d'un polynôme à l'aide de la fonction set.

```
>> AFFECT P,Q
P=5x^8
Q=5x^8
>> SET P=2X^7
>> AFFECT Q,P
P=2x^7
Q=2x^7
```

```
>> ADD Q,P
P + Q = 4x^7
>> SET P= 4X^8
>> ADD P,Q
P + Q = 4x^8 +2x^7
>> SOUS Q,P
Q - P = -4x^8 +2x^7
>> SOUS P,Q
P - Q = 4x^8 -2x^7
```

Cet exemple d'exécution représente une situation où l'utilisateur effectue une addition ou une soustraction de deux polynômes. Peu importe si l'utilisateur entre "P+Q" ou "Q+P", le résultat est le même pour l'addition, tout comme pour la soustraction.

UTILITÉ

La calculatrice polynomiale développée en langage C revêt une utilité significative en offrant une plateforme avancée pour la manipulation et l'analyse de polynômes. Au-delà des opérations de base, elle permet des fonctionnalités sophistiquées telles que la factorisation, la simplification, et la recherche de racines. En introduisant une composante de visualisation graphique, l'outil fournit une compréhension visuelle du comportement des polynômes. Une extension majeure repose sur la capacité d'effectuer l'interpolation polynomiale, utile dans des domaines tels que l'analyse de données expérimentales.

CONCLUSION

En conclusion, ce projet de calculatrice polynomiale en langage C représente un outil mathématique complet et convivial. Avec ses fonctionnalités avancées, il permet la manipulation, l'analyse et la visualisation efficaces des polynômes. Sa polyvalence, associée à une interface utilisateur intuitive, en fait une ressource précieuse pour un large éventail d'utilisateurs, des étudiants aux professionnels en passant par les chercheurs. En résumé, cette calculatrice offre une solution puissante et accessible pour explorer les aspects complexes des polynômes, renforçant ainsi la compréhension des concepts mathématiques.