#include<stdio.h>

#include<stdlib.h>

//la structure de base est un monome défini par un coefficient et un exposant

struct monome {float coef; int exposant; struct monome\* suiv;};

typedef struct monome m;

//fonction de création du polynome, les termes sont saisis par l'utilisateur

m\* Creer\_Pol(){

//d est le degré du polynome

int d;

printf("entrer le degré de votre polynome : ");

scanf("%d", &d);

m \*p0; //pointeur sur le premier élément

m \*p; //pour alimenter la chaine au fur et à mesure

p0= (m\*)malloc(sizeof(m)); //premier élément

p=p0;

for(int k=d;k>=0;k--){

p->exposant=k;

printf("\nEntrer le coefficient du terme x^%d: ",k);

scanf("%f", &p->coef);

if(k==0) p->suiv=NULL;

else

{p->suiv=(m\*)malloc(sizeof(m));

p=p->suiv;

}

}

return p0;

}

//Fonction d'affichage

void Affiche(m \*p0){

int count = 1;

m \*p;

p = p0;

count = p->coef

printf("\n");

do

{

if(p->exposant==0)

printf (" %.2f + ", p->coef);

else

printf (" %.2fx^%d + ", p->coef, p->exposant);

count--;

p = p->suiv;

}

while (p != NULL);

}

//dérivation d'un polynome

m\* DerivPol(m \*\*p0){

m \*p;

m \*q;

p=\*p0;

q=\*p0; //nouvelle tête du polynome dérivé

while(p!=NULL){

if(p->exposant !=0)

{

q=p->suiv;

free(p);

p=q;

}

else

{

p->coef\*=p->exposant;

p->exposant-=1;

p=p->suiv;

}

}

return q;

}

//calcul de la primitive d'un polynome

void PrimPol(m \*\*p0){

m \*p;

p=\*p0;

while(p!=NULL){

p->coef/=p->exposant;

p->exposant+=1;

p=p->suiv;

}

}

//ajout d'un monome, les valeurs sont entrées par l'utilisateur

void AjoutMon(m\*\* p0)

{

m \*p, \*q;

p=\*p0;

m\* new\_p=(m\*)malloc(sizeof(m));

printf("\n Entrer l'exposant du monome à ajouter :");

scanf("%d",&new\_p->exposant);

printf("\n Entrer le coefficient correspondant :");

scanf("%f",&new\_p->coef);

while(p->exposant<new\_p->exposant){

printf("here \n");

q=p;

p=p->suiv;

if (p==NULL) break;

}

if(p==NULL)

//fin de la liste chainée atteinte, degré du monome supérieur au degré du polynome

{

q->suiv=(m\*)malloc(sizeof(m));

q->suiv=new\_p;

new\_p->suiv=NULL;

}

else {

if (p->exposant==new\_p->exposant)

//degré du monome à ajouter est égal à celui d'un terme du polynome

{

p->coef+=new\_p->coef;

}

else // p->exposant > ajout->exposant

{

//on insère le monome entre deux termes du polynome

q->suiv=new\_p;

new\_p->suiv=(m\*)malloc(sizeof(m));

new\_p->suiv=p;

}

}

}

//version pour ajouter un monome en paramètre

void AjoutMon2(m\*\* p0,m\* ajout)

{

m \*p, \*q;

p=\*p0;

while(p->exposant<ajout->exposant){

q=p;

p=p->suiv;

if (p==NULL) break;

}

if(p==NULL) //fin atteinte, degré du monome supérieur au degré du polynome

{

q->suiv=(m\*)malloc(sizeof(m));

q->suiv=ajout;

ajout->suiv=NULL;

}

else {

if (p->exposant==ajout->exposant)

{

p->coef+=ajout->coef;

}

else // p->exposant > ajout->exposant

{

q->suiv=ajout;

ajout->suiv=(m\*)malloc(sizeof(m));

ajout->suiv=p;

}

}

}

void AddPol(m\*\* p1,m p2){

m\* p,\*q;

q=\*p1;

p=&p2;

while(p!=NULL){

AjoutMon2(&q,p);

p=p->suiv;

}

}

int main(){

printf("Création du premier polynome ...\n");

m\* p=Creer\_Pol();

printf("\nAffichage du polynome 1 :\n");

Affiche(p);

printf("\nCréation du deuxiéme polynome ...\n");

m\* pp=Creer\_Pol();

printf("\nAffichage du polynome 2 :\n");

Affiche(pp);

printf("Somme des deux polynomes :\n");

AddPol(&p,\*pp);

printf("Affichage du polynome résultant :\n");

Affiche(p);

printf("Ajout d'un monome au polynome précedent :\n");

AjoutMon(&p);

printf("Affichage du polynome résultant :\n");

Affiche(p);

printf("\n Calcul de la dérivée du polynome résultant :\n");

DerivPol(&p);

Affiche(p);

return 0;

}