Polynomial Addition:

#include <stdio.h>

#include <stdlib.h>

struct poly

{

int coeff;

int pow;

struct poly \*Next;

};

typedef struct poly Poly;

void Create(Poly \*List);

void Display(Poly \*List);

void Addition(Poly \*Poly1, Poly \*Poly2, Poly \*Result);

int main()

{

Poly \*Poly1 = malloc(sizeof(Poly));

Poly \*Poly2 = malloc(sizeof(Poly));

Poly \*Result = malloc(sizeof(Poly));

Poly1->Next = NULL;

Poly2->Next = NULL;

printf("Enter the values for first polynomial :\n");

Create(Poly1);

printf("The polynomial equation is : ");

Display(Poly1);

printf("\nEnter the values for second polynomial :\n");

Create(Poly2);

printf("The polynomial equation is : ");

Display(Poly2);

Addition(Poly1, Poly2, Result);

printf("\nThe polynomial equation addition result is : ");

Display(Result);

return 0;

}

void Create(Poly \*List)

{

int choice;

Poly \*Position, \*NewNode;

Position = List;

do

{

NewNode = malloc(sizeof(Poly));

printf("Enter the coefficient : ");

scanf("%d", &NewNode->coeff);

printf("Enter the power : ");

scanf("%d", &NewNode->pow);

NewNode->Next = NULL;

Position->Next = NewNode;

Position = NewNode;

printf("Enter 1 to continue : ");

scanf("%d", &choice);

} while(choice == 1);

}

void Display(Poly \*List)

{

Poly \*Position;

Position = List->Next;

while(Position != NULL)

{

printf("%dx^%d", Position->coeff, Position->pow);

Position = Position->Next;

if(Position != NULL && Position->coeff > 0)

{

printf("+");

}

}

}

void Addition(Poly \*Poly1, Poly \*Poly2, Poly \*Result)

{

Poly \*Position;

Poly \*NewNode;

Poly1 = Poly1->Next;

Poly2 = Poly2->Next;

Result->Next = NULL;

Position = Result;

while(Poly1 != NULL && Poly2 != NULL)

{

NewNode = malloc(sizeof(Poly));

if(Poly1->pow == Poly2->pow)

{

NewNode->coeff = Poly1->coeff + Poly2->coeff;

NewNode->pow = Poly1->pow;

Poly1 = Poly1->Next;

Poly2 = Poly2->Next;

}

else if(Poly1->pow > Poly2->pow)

{

NewNode->coeff = Poly1->coeff;

NewNode->pow = Poly1->pow;

Poly1 = Poly1->Next;

}

else if(Poly1->pow < Poly2->pow)

{

NewNode->coeff = Poly2->coeff;

NewNode->pow = Poly2->pow;

Poly2 = Poly2->Next;

}

NewNode->Next = NULL;

Position->Next = NewNode;

Position = NewNode;

}

while(Poly1 != NULL || Poly2 != NULL)

{

NewNode = malloc(sizeof(Poly));

if(Poly1 != NULL)

{

NewNode->coeff = Poly1->coeff;

NewNode->pow = Poly1->pow;

Poly1 = Poly1->Next;

}

if(Poly2 != NULL)

{

NewNode->coeff = Poly2->coeff;

NewNode->pow = Poly2->pow;

Poly2 = Poly2->Next;

}

NewNode->Next = NULL;

Position->Next = NewNode;

Position = NewNode;

}

}

Polynomial Subtraction:

#include <stdio.h>

#include <stdlib.h>

struct poly

{

int coeff;

int pow;

struct poly \*Next;

};

typedef struct poly Poly;

void Create(Poly \*List);

void Display(Poly \*List);

void Subtraction(Poly \*Poly1, Poly \*Poly2, Poly \*Result);

int main()

{

Poly \*Poly1 = malloc(sizeof(Poly));

Poly \*Poly2 = malloc(sizeof(Poly));

Poly \*Result = malloc(sizeof(Poly));

Poly1->Next = NULL;

Poly2->Next = NULL;

printf("Enter the values for first polynomial :\n");

Create(Poly1);

printf("The polynomial equation is : ");

Display(Poly1);

printf("\nEnter the values for second polynomial :\n");

Create(Poly2);

printf("The polynomial equation is : ");

Display(Poly2);

Subtraction(Poly1, Poly2, Result);

printf("\nThe polynomial equation subtraction result is : ");

Display(Result);

return 0;

}

void Create(Poly \*List)

{

int choice;

Poly \*Position, \*NewNode;

Position = List;

do

{

NewNode = malloc(sizeof(Poly));

printf("Enter the coefficient : ");

scanf("%d", &NewNode->coeff);

printf("Enter the power : ");

scanf("%d", &NewNode->pow);

NewNode->Next = NULL;

Position->Next = NewNode;

Position = NewNode;

printf("Enter 1 to continue : ");

scanf("%d", &choice);

} while(choice == 1);

}

void Display(Poly \*List)

{

Poly \*Position;

Position = List->Next;

while(Position != NULL)

{

printf("%dx^%d", Position->coeff, Position->pow);

Position = Position->Next;

if(Position != NULL && Position->coeff > 0)

{

printf("+");

}

}

}

void Subtraction(Poly \*Poly1, Poly \*Poly2, Poly \*Result)

{

Poly \*Position;

Poly \*NewNode;

Poly1 = Poly1->Next;

Poly2 = Poly2->Next;

Result->Next = NULL;

Position = Result;

while(Poly1 != NULL && Poly2 != NULL)

{

NewNode = malloc(sizeof(Poly));

if(Poly1->pow == Poly2->pow)

{

NewNode->coeff = Poly1->coeff - Poly2->coeff;

NewNode->pow = Poly1->pow;

Poly1 = Poly1->Next;

Poly2 = Poly2->Next;

}

else if(Poly1->pow > Poly2->pow)

{

NewNode->coeff = Poly1->coeff;

NewNode->pow = Poly1->pow;

Poly1 = Poly1->Next;

}

else if(Poly1->pow < Poly2->pow)

{

NewNode->coeff = -(Poly2->coeff);

NewNode->pow = Poly2->pow;

Poly2 = Poly2->Next;

}

NewNode->Next = NULL;

Position->Next = NewNode;

Position = NewNode;

}

while(Poly1 != NULL || Poly2 != NULL)

{

NewNode = malloc(sizeof(Poly));

if(Poly1 != NULL)

{

NewNode->coeff = Poly1->coeff;

NewNode->pow = Poly1->pow;

Poly1 = Poly1->Next;

}

if(Poly2 != NULL)

{

NewNode->coeff = -(Poly2->coeff);

NewNode->pow = Poly2->pow;

Poly2 = Poly2->Next;

}

NewNode->Next = NULL;

Position->Next = NewNode;

Position = NewNode;

}

}