**Polynomial manipulation**

**Aim:**

To implement polynomial manipulation in c, namely , polynomial addition , polynomial subtraction and polynomial multiplication using single linked list.

**Algorithm:**

1. **Start:** Begin the program.
2. **Declare Structures and Function Prototypes:** Define a structure for polynomial terms (struct Term) and declare function prototypes for creating terms, inserting terms into polynomials, displaying polynomials, and performing polynomial operations (addition, subtraction, multiplication).
3. **Main Function:** Start the main function.
4. **Initialize Variables:** Declare variables for the choice of operation (choice), coefficients, exponents, and polynomial pointers (poly1, poly2, result).
5. **Operation Menu:** Display a menu for the user to choose the operation they want to perform (addition, subtraction, multiplication, or exit).
6. **Input Polynomials:** Prompt the user to input coefficients and exponents for the two polynomials based on the chosen operation.
7. **Perform Operation:** Depending on the user’s choice, call the corresponding function (addPolynomials, subtractPolynomials, multiplyPolynomials) to perform the operation on the input polynomials.
8. **Display Result:** Display the resultant polynomial after the operation.
9. **Free Memory:** Free memory allocated for the polynomials after each operation to prevent memory leaks.
10. **Repeat or Exit:** Ask the user if they want to continue with another operation or exit the program. If they choose to continue, repeat steps 5 to 9. If they choose to exit, end the program.

**Code:**

#include <stdio.h>

#include <stdlib.h>

// Define a structure for the polynomial term

Struct Term {

Int coefficient;

Int exponent;

Struct Term\* next;

};

// Function prototypes

Struct Term\* createTerm(int coefficient, int exponent);

Void insertTerm(struct Term\*\* poly, int coefficient, int exponent);

Void displayPolynomial(struct Term\* poly);

Struct Term\* addPolynomials(struct Term\* poly1, struct Term\* poly2);

Struct Term\* subtractPolynomials(struct Term\* poly1, struct Term\* poly2);

Struct Term\* multiplyPolynomials(struct Term\* poly1, struct Term\* poly2);

Void freePolynomial(struct Term\* poly);

Int main() {

Struct Term \*poly1 = NULL, \*poly2 = NULL, \*result = NULL;

Int choice, coefficient, exponent;

While (1) {

Printf(“\n1. Add polynomials\n2. Subtract polynomials\n3. Multiply polynomials\n4. Exit\nEnter your choice: “);

Scanf(“%d”, &choice);

Switch (choice) {

Case 1:

Printf(“Enter the first polynomial (coefficient followed by exponent, enter 0 0 to finish):\n”);

While (1) {

Scanf(“%d %d”, &coefficient, &exponent);

If (coefficient == 0 && exponent == 0)

Break;

insertTerm(&poly1, coefficient, exponent);

}

Printf(“Enter the second polynomial (coefficient followed by exponent, enter 0 0 to finish):\n”);

While (1) {

Scanf(“%d %d”, &coefficient, &exponent);

If (coefficient == 0 && exponent == 0)

Break;

insertTerm(&poly2, coefficient, exponent);

}

Printf(“Resultant polynomial after addition: “);

Result = addPolynomials(poly1, poly2);

displayPolynomial(result);

freePolynomial(result);

break;

case 2:

printf(“Enter the first polynomial (coefficient followed by exponent, enter 0 0 to finish):\n”);

while (1) {

scanf(“%d %d”, &coefficient, &exponent);

if (coefficient == 0 && exponent == 0)

break;

insertTerm(&poly1, coefficient, exponent);

}

Printf(“Enter the second polynomial (coefficient followed by exponent, enter 0 0 to finish):\n”);

While (1) {

Scanf(“%d %d”, &coefficient, &exponent);

If (coefficient == 0 && exponent == 0)

Break;

insertTerm(&poly2, coefficient, exponent);

}

Printf(“Resultant polynomial after subtraction: “);

Result = subtractPolynomials(poly1, poly2);

displayPolynomial(result);

freePolynomial(result);

break;

case 3:

printf(“Enter the first polynomial (coefficient followed by exponent, enter 0 0 to finish):\n”);

while (1) {

scanf(“%d %d”, &coefficient, &exponent);

if (coefficient == 0 && exponent == 0)

break;

insertTerm(&poly1, coefficient, exponent);

}

Printf(“Enter the second polynomial (coefficient followed by exponent, enter 0 0 to finish):\n”);

While (1) {

Scanf(“%d %d”, &coefficient, &exponent);

If (coefficient == 0 && exponent == 0)

Break;

insertTerm(&poly2, coefficient, exponent);

}

Printf(“Resultant polynomial after multiplication: “);

Result = multiplyPolynomials(poly1, poly2);

displayPolynomial(result);

freePolynomial(result);

break;

case 4:

exit(0);

default:

printf(“Invalid choice!\n”);

}

// Clear the polynomials after each operation

freePolynomial(poly1);

freePolynomial(poly2);

poly1 = NULL;

poly2 = NULL;

}

Return 0;

}

// Function to create a polynomial term

Struct Term\* createTerm(int coefficient, int exponent) {

Struct Term\* newTerm = (struct Term\*)malloc(sizeof(struct Term));

If (!newTerm) {

Printf(“Memory allocation failed.\n”);

Exit(1);

}

newTerm->coefficient = coefficient;

newTerm->exponent = exponent;

newTerm->next = NULL;

return newTerm;

}

// Function to insert a term into a polynomial in sorted order of exponents

Void insertTerm(struct Term\*\* poly, int coefficient, int exponent) {

Struct Term\* newTerm = createTerm(coefficient, exponent);

If (\*poly == NULL || exponent > (\*poly)->exponent) {

newTerm->next = \*poly;

\*poly = newTerm;

} else {

Struct Term\* temp = \*poly;

While (temp->next != NULL && temp->next->exponent > exponent) {

Temp = temp->next;

}

newTerm->next = temp->next;

temp->next = newTerm;

}

}

// Function to display a polynomial

Void displayPolynomial(struct Term\* poly) {

While (poly != NULL) {

Printf(“%dx^%d “, poly->coefficient, poly->exponent);

If (poly->next != NULL)

Printf(“+ “);

Poly = poly->next;

}

Printf(“\n”);

}

// Function to add two polynomials

Struct Term\* addPolynomials(struct Term\* poly1, struct Term\* poly2) {

Struct Term\* result = NULL;

While (poly1 != NULL && poly2 != NULL) {

If (poly1->exponent > poly2->exponent) {

insertTerm(&result, poly1->coefficient, poly1->exponent);

poly1 = poly1->next;

} else if (poly1->exponent < poly2->exponent) {

insertTerm(&result, poly2->coefficient, poly2->exponent);

poly2 = poly2->next;

} else {

insertTerm(&result, poly1->coefficient + poly2->coefficient, poly1->exponent);

poly1 = poly1->next;

poly2 = poly2->next;

}

}

While (poly1 != NULL) {

insertTerm(&result, poly1->coefficient, poly1->exponent);

poly1 = poly1->next;

}

While (poly2 != NULL) {

insertTerm(&result, poly2->coefficient, poly2->exponent);

poly2 = poly2->next;

}

Return result;

}

// Function to subtract two polynomials

Struct Term\* subtractPolynomials(struct Term\* poly1, struct Term\* poly2) {

Struct Term\* result = NULL;

While (poly1 != NULL && poly2 != NULL) {

If (poly1->exponent > poly2->exponent) {

insertTerm(&result, poly1->coefficient, poly1->exponent);

poly1 = poly1->next;

} else if (poly1->exponent < poly2->exponent) {

insertTerm(&result, -(poly2->coefficient), poly2->exponent);

poly2 = poly2->next;

} else {

insertTerm(&result, poly1->coefficient – poly2->coefficient, poly1->exponent);

poly1 = poly1->next;

poly2 = poly2->next;

}

}

While (poly1 != NULL) {

insertTerm(&result, poly1->coefficient, poly1->exponent);

poly1 = poly1->next;

}

While (poly2 != NULL) {

insertTerm(&result, -(poly2->coefficient), poly2->exponent);

poly2 = poly2->next;

}

Return result;

}

// Function to multiply two polynomials

Struct Term\* multiplyPolynomials(struct Term\* poly1, struct Term\* poly2) {

Struct Term\* result = NULL;

Struct Term\* temp1 = poly1;

While (temp1 != NULL) {

Struct Term\* temp2 = poly2;

While (temp2 != NULL) {

insertTerm(&result, temp1->coefficient \* temp2->coefficient, temp1->exponent + temp2->exponent);

temp2 = temp2->next;

}

Temp1 = temp1->next;

}

Return result;

}

// Function to free memory allocated for polynomial

Void freePolynomial(struct Term\* poly) {

Struct Term\* temp;

While (poly != NULL) {

Temp = poly;

Poly = poly->next;

Free(temp);

}

}

**Output:**

**1.Add polynomials**

**2. Subtract polynomials**

**3. Multiply polynomials**

**4. Exit**

**Enter your choice: 1**

**Enter the first polynomial (coefficient followed by exponent, enter 0 0 to finish):**

**2 3 4 2 1 1 0 0**

**Enter the second polynomial (coefficient followed by exponent, enter 0 0 to finish):**

**3 3 1 2 5 1 0 0**

**Resultant polynomial after addition : 5x^3 + 5x^2 + 6x^1**

**1. Add polynomials**

**2. Subtract polynomials**

**3. Multiply polynomials**

**4. Exit**

**Enter your choice: 2**

**Enter the first polynomial (coefficient followed by exponent, enter 0 0 to finish):**

**7 3 5 1 0 0**

**Enter the second polynomial ( coefficient followed by exponent, enter 0 0 to finish):**

**8 3 2 2 3 1 0 0**

**﻿Resultant polynomial after subtract ion: -1x^3 + -2x^2 + 2x^1**

**1.Add polynomials**

**2. Subtract polynomials**

**3. Multiply polynomials**

**4. Exit**

**Enter your choice: 3**

**Enter the first polynomial (coefficient followed by exponent, enter 0 0 to finish):**

**2 3 2 2 1 1**

**0 0**

**Enter the second polynomial ( coefficient followed by exponent, enter 0 0 to finish):**

**3 2 1 3 0 0**

**Resultant polynomial after multiplication: 2x^6 + 2x^5 + 6x^5 + 1 x^4 + 6x^4 + 3x^3**

**Result:**

Thus, the program was implemented successfully using Singly linked list .