

|  |
| --- |
| **Title:** Implementation of polynomials operations (addition, subtraction ) using link list. |

**Objective:** To understand an application of linear linked list.

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **2** | Use linear and non-linear data structure in domain like compiler  construction, DBMS etc |

**Books/ Journals/ Websites referred:** Data Structures using C (Second Edition) by Reema Thareja.

**Abstract**:

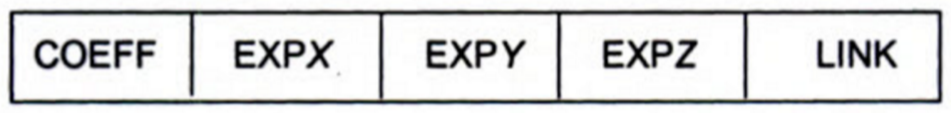
Polynomials are used in both scientific and business oriented problems. General-Purpose programming languages like do not have built-in data types or functions to Manipulate polynomials directly. Rather, it is common to represent polynomials using Arrays or linked lists. For any polynomial operation linked list representation is most suitable to deal with.

**Related Theory: -**

**Linked list representation of polynomial of single variables:**

Each polynomial can be represented as a list with linked list representation. The first node in the list represents first term in the polynomial; the second node represents the second term, and so on. Each node contains five fields. The first field is the coefficient. The second field is the index of x, the third field is the index of y, the fourth field is the index of z and the fifth field is a pointer to the next term.

**Diagram for linked representation of polynomial of single variables:**



Therefore while taking input of every term of the polynomial the user will type in exponents appropriately. For example, if we have 3x^2 + 5y then first input would be 3 2 0 0 Meaning, 3 = coefficient, 2 = index of x, 0 = index of y, 0 = index of z and second input would be 5 0 1 0.**Implementation Details:**

**1. Enlist all the Steps followed and various options explored**

**2. Explain your program logic, classes and methods used.**

**3. Explain the Importance of the approach followed by you**

Algorithm (Pseudo Code):

Step 1: Suppose there are 2 linked lists storing each term of two different polynomials as a node in the format: "coeff(c) Indexofx(x) Indexofy(y) Indexofz(z)" with s1 and s2 as start pointers to the 2 lists

Step 2: t1=NULL, t3=s2

Step 3: t1->next=s1

Step 4: while t3 is not equal to NULL

{

t1=NULL

t2=t1->next

While t2 is not equal to NULL

{

If t2->x=t3->x and t2->y=t3->y and t2->z=t3->z

{

t3->c=t3->c+t2->c

if t2=s1

{

s1=s1->next

}

t2=t2->next

t1->next=t2

Break

}

Else

{

t1=t1->next

t2=t2->next

}

}

t4=t3

t3=t3->next

}

t4->next=s1

Step 5: Set t=s2;

While t is not equal to NULL

{

Print t->c, t->x, t->y, t->z

t=t->next;

}

**Conclusion:** Polynomial addition has been successfully implemented by using linked lists to represent the polynomial. The expected result was outputted.