**Aim:** Demonstrate application of linked list

i) Polynomial addition

ii) Sparse matrix

**Introduction:**

A linked list is a linear data structure that consists of a sequence of nodes, where each node contains a data element and a reference (link) to the next node in the sequence. The first node is called the head, and the last node points to null. Linked lists can be used to implement dynamic data structures, where the size of the data structure can change during runtime, unlike arrays, which have a fixed size. Linked lists are useful for many applications, such as implementing stacks, queues, and memory management.

**i) Polynomial addition**

Polynomial addition is the process of combining two or more polynomials to form a new polynomial. It is done by combining like terms, which are terms having the same degree. The sum of two polynomials is obtained by adding the corresponding coefficients of each term. For example, the sum of the polynomials 2x^2 + 3x + 4 and x^2 + 5x + 2 is 3x^2 + 8x + 6.

#include <iostream>

using namespace std;

class Node{

public:

int num;

string coeff;

Node \*next = nullptr;

Node(int num, string coeff): num(num), coeff(coeff){}

};

class List{

public:

Node \*head = nullptr;

void add(int num, string coeff){

Node \*newNode = new Node(num, coeff);

if(head == nullptr){

head = newNode;

return;

}Node \*temp = head;

while(temp->next != nullptr)

temp = temp->next;

temp->next = newNode;

}

void print(){

Node \*temp = head;

while (temp != nullptr){

cout << temp->num << "" << temp->coeff;

if(temp->next != nullptr) cout << " + ";

temp = temp->next;

} cout << endl;

}

};

void add(List \*l1, List \*l2){

Node \*temp1 = l1->head;

Node \*temp2 = l2->head;

List \*ans = new List();

while(temp1 != nullptr){

ans->add(temp1->num + temp2->num, temp1->coeff);

temp1 = temp1->next;

temp2 = temp2->next;

}

l1->print();

l2->print();

cout << "----------------------" << endl;

ans->print();

}

int main(int argc, char const \*argv[]){

List \*l1 = new List();

List \*l2 = new List();

List \*arr[] = {l1, l2};

int n, num;

string coeff;

cout << "Enter Number Of co-efficient your equation have : ";

cin >> n;

for(int i = 0; i < 2; i++){

cout << "Enter Equation " << i+1<<endl;

for(int j = 0; j < n; j++){

cout << "enter number : ";

cin >> num;

cout << "enter coefficient "<< j+1 <<": ";

cin >> coeff;

arr[i]->add(num, coeff);

}

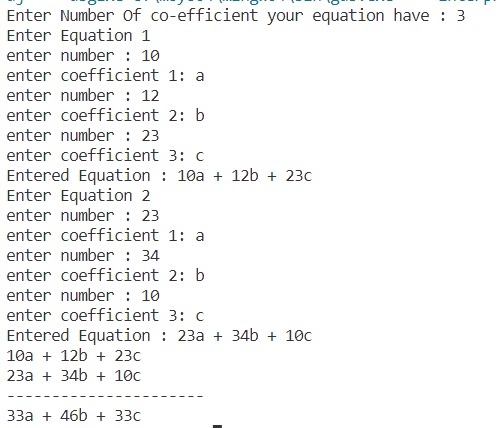
cout << "Entered Equation : ";

arr[i]->print();

} add(l1,l2);

return 0;

}



**ii) Sparse matrix**

A sparse matrix is a matrix in which most of the elements are zero. Instead of storing the entire matrix in memory, it is more efficient to store only the non-zero elements.

**Application of sparse matrix**

* Compressed Image and Video Storage: In image and video compression, a lot of zero elements are generated. Storing these elements in a sparse matrix significantly reduces the memory space required.
* Graph Representation: In graph theory, a sparse matrix is used to represent the adjacency matrix of a graph. This significantly reduces the memory requirement for large graphs.
* Scientific Computing: In many scientific applications, matrices are used to store large amounts of data. A sparse matrix is a more efficient representation of these matrices, reducing memory and computational requirements.

#include <iostream>

using namespace std;

class Node{

public:

int row, col, data;

Node \*next = nullptr;

Node(int row, int col, int data):row(row), col(col), data(data){}

};

class List{

public:

Node \*head = nullptr;

void add(int row, int col, int data){

Node \*newNode = new Node(row, col, data);

if(head == nullptr){

head = newNode;

return;

}

Node \*temp = head;

while(temp->next != nullptr)

temp = temp->next;

temp->next = newNode;

}

};

void printMatrix(List \*list, int m, int n){

cout << "SPARSE MATRIX"<<endl;

Node \*temp = list->head;

for(int i = 0; i <= m; i++){

for(int j = 0; j <= n; j++){

if(temp != nullptr && temp->row == i && temp->col == j){

cout << " "<< temp->data<<" ";

temp = temp->next;

}

else cout << " 0 ";

}

cout << endl;

}

}

int main(int argc, char const \*argv[])

{

List \*list = new List();

int nonZero, rowIndex, colIndex, data, maxColumn = 0, maxRow = 0;

cout << "Enter how many non-zero value your sparse matrix have : ";

cin >> nonZero;

for(int i = 0; i < nonZero; i++){

cout << "non-zero position "<<i+1<<endl;

cout << "enter row number : ";

cin >> rowIndex;

cout << "enter col number : ";

cin >> colIndex;

cout << "enter position value : ";

cin >> data;

list->add(rowIndex, colIndex, data);

if(rowIndex > maxRow) maxRow = rowIndex;

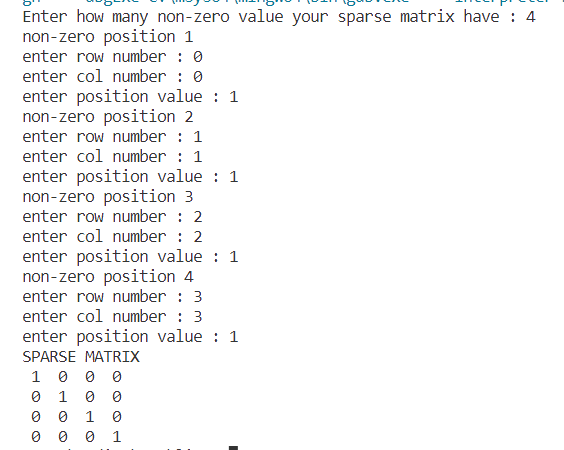
if(colIndex > maxColumn) maxColumn = colIndex;

}

printMatrix(list, maxRow, maxColumn);

return 0;

}



**Conclusion:** I have learned about application of linked list.