## **LEC-3 Single Linked List**

**Drawback of Array Data Structures:**

1. Fixed Size Initialization( we have to tell the array size )
2. Cannot add single block in existing array
3. Cannot remove single block from existing array
4. Wastage of memory
5. Contiguous Allocation

0 1 2 3

Types:

1. Singly Linked List
2. Doubly Linked List
3. Circular Link List

## **Linked List Implementation in C++**

L21-6225 Muhammad Abdullah Atif

#include <iostream>

using namespace std;

class LinkedList;

class Node {

    friend class LinkedList;

    int data;

    Node\* next;//end of class

};

class LinkedList{

private:

    Node\* head;

public:

    LinkedList() {

        head = nullptr;

    }

    bool insertAtStart(int val) {

        Node\* temp = new Node;

        temp->data = val;

        temp->next = head;

        head = temp;

        return 1;

    }//end of insertAtStart

    bool deleteFromStart() {

        if (head == nullptr) {

            return 0;

        }

        else {

            Node\* curr = head;

            head = curr->next;

            //head=head->next;

            delete curr;

            return 1;

        }

    }//end of deleteFromStart

void Display() {

        Node\* curr = head;

        while (curr != nullptr) {

            cout << curr->data << " ";

            curr = curr->next;

        }

    }//end of Display

    bool insertAtEnd(int val) {

        Node\* temp = new Node;

        temp->data = val;

        temp->next = nullptr;

        Node\* prev = 0, \*curr = head;

        while (curr != nullptr) {

            prev = curr;

            curr = curr->next;

        }

        //if element is first in list

        if (prev == 0) {

            head = temp;

            return 1;

        }

        else {

            prev->next = temp;

            return 1;

        }

    }//end of insertAtEnd

bool deleteFromEnd()

    {

        if (head == 0) {

            cout << "List is empty \n ";

            return 0;

        }

        else

        {

            Node\* curr = head, \* last = 0, \* secondLast = 0;

            while (curr != NULL) {

                secondLast = last;

                last = curr->next;

            }

            if (secondLast == 0) {

                head = NULL;

                delete last;

                last = 0;

                cout << "Last Block deleted \n";

                return 1;

            }

            else

            {

                secondLast->next = NULL;

                delete last;

                last = 0;

                cout << "Last Block deleted \n";

                return 1;

            }

        }

    }

    ~LinkedList() {

        while (head != NULL) {

            Node\* curr = head;

            head = curr->next;

            delete curr;

            curr = head;

        }

    }

};

int main() {

    LinkedList l1;

    l1.insertAtStart(6);

    l1.insertAtEnd(7);

    l1.deleteFromStart();

    l1.deleteFromEnd();

    l1.Display();

    system("pause");

    return 0;

}

## **SortedInsert and Quiz**

bool sortedInsert(int val)

    {

        Node\* temp = new Node;

        temp->data = val;

        Node\* curr = head, \*prev = 0;

        while (curr != NULL && curr->data<val)

        {

            prev = curr;

            curr = curr->next;

        }

        if (prev == 0)

        {

            head = temp;

        }

        else

        {

            prev->next = temp;

        }

        temp->next = curr;

        cout << "Element is inserted in Sorted Linked List...!\n";

        return 1;

    }