**USN:- 1NT21IS039**

## Aim:- IMPLEMETATION OF SINGLY LINKED LIST

## [GitHub-Link](https://github.com/logan-14/Implementation-of-LinkedList)

### **Algorithm:-**

1. Create a class Node which has two attributes: data and next. Next is a pointer to the next node in the list.
2. Create another class which has two attributes: head and tail.
3. addNode() will add a new node to the list:
   1. Create a new node.
   2. It first checks, whether the head is equal to null which means the list is empty.
   3. If the list is empty, both head and tail will point to the newly added node.
   4. If the list is not empty, the new node will be added to end of the list such that tail's next will point to a newly added node. This new node will become the new tail of the list.
4. countNodes() will count the nodes present in the list:
   1. Define a node current which will initially point to the head of the list.
   2. Declare and initialize a variable count to 0.
   3. Traverse through the list till current point to null.
   4. Increment the value of count by 1 for each node encountered in the list.
5. display() will display the nodes present in the list:
   1. Define a node current which will initially point to the head of the list.
   2. Traverse through the list till current points to null.
   3. Display each node by making current to point to node next to it in each iteration.

**Code:-**

// C program for the all operations in

// the Singly Linked List

#include <stdio.h>

#include <stdlib.h>

// Linked List Node

struct node

{

    int info;

    struct node \*link;

};

struct node \*start = NULL;

// Function to create list with n nodes initially

void createList()

{

    if (start == NULL)

    {

        int n;

        printf("\nEnter the number of nodes: ");

        scanf("%d", &n);

        if (n != 0)

        {

            int data;

            struct node \*newnode;

            struct node \*temp;

            newnode = malloc(sizeof(struct node));

            start = newnode;

            temp = start;

            printf("\nEnter number to"

                   " be inserted : ");

            scanf("%d", &data);

            start->info = data;

            for (int i = 2; i <= n; i++)

            {

                newnode = malloc(sizeof(struct node));

                temp->link = newnode;

                printf("\nEnter number to"

                       " be inserted : ");

                scanf("%d", &data);

                newnode->info = data;

                temp = temp->link;

            }

        }

        printf("\nThe list is created\n");

    }

    else

        printf("\nThe list is already created\n");

}

// Function to traverse the linked list

void traverse()

{

    struct node \*temp;

    // List is empty

    if (start == NULL)

        printf("\nList is empty\n");

    // Else print the LL

    else

    {

        temp = start;

        while (temp != NULL)

        {

            printf("Data = %d\n", temp->info);

            temp = temp->link;

        }

    }

}

// Function to insert at the front

// of the linked list

void insertAtFront()

{

    int data;

    struct node \*temp;

    temp = malloc(sizeof(struct node));

    printf("\nEnter number to"

           " be inserted : ");

    scanf("%d", &data);

    temp->info = data;

    // Pointer of temp will be

    // assigned to start

    temp->link = start;

    start = temp;

}

// Function to insert at the end of

// the linked list

void insertAtEnd()

{

    int data;

    struct node \*temp, \*head;

    temp = malloc(sizeof(struct node));

    // Enter the number

    printf("\nEnter number to"

           " be inserted : ");

    scanf("%d", &data);

    // Changes links

    temp->link = 0;

    temp->info = data;

    head = start;

    while (head->link != NULL)

    {

        head = head->link;

    }

    head->link = temp;

}

// Function to insert at any specified

// position in the linked list

void insertAtPosition()

{

    struct node \*temp, \*newnode;

    int pos, data, i = 1;

    newnode = malloc(sizeof(struct node));

    // Enter the position and data

    printf("\nEnter position and data :");

    scanf("%d %d", &pos, &data);

    // Change Links

    temp = start;

    newnode->info = data;

    newnode->link = 0;

    while (i < pos - 1)

    {

        temp = temp->link;

        i++;

    }

    newnode->link = temp->link;

    temp->link = newnode;

}

// Function to delete from the front

// of the linked list

void deleteFirst()

{

    struct node \*temp;

    if (start == NULL)

        printf("\nList is empty\n");

    else

    {

        temp = start;

        start = start->link;

        free(temp);

    }

}

// Function to delete from the end

// of the linked list

void deleteEnd()

{

    struct node \*temp, \*prevnode;

    if (start == NULL)

        printf("\nList is Empty\n");

    else

    {

        temp = start;

        while (temp->link != 0)

        {

            prevnode = temp;

            temp = temp->link;

        }

        free(temp);

        prevnode->link = 0;

    }

}

// Function to delete from any specified

// position from the linked list

void deletePosition()

{

    struct node \*temp, \*position;

    int i = 1, pos;

    // If LL is empty

    if (start == NULL)

        printf("\nList is empty\n");

    // Otherwise

    else

    {

        printf("\nEnter index : ");

        // Position to be deleted

        scanf("%d", &pos);

        position = malloc(sizeof(struct node));

        temp = start;

        // Traverse till position

        while (i < pos - 1)

        {

            temp = temp->link;

            i++;

        }

        // Change Links

        position = temp->link;

        temp->link = position->link;

        // Free memory

        free(position);

    }

}

// Driver Code

int main()

{

    int choice;

    while (1)

    {

        printf("\n\t1 To see list\n");

        printf("\t2 For insertion at"

               " starting\n");

        printf("\t3 For insertion at"

               " end\n");

        printf("\t4 For insertion at "

               "any position\n");

        printf("\t5 For deletion of "

               "first element\n");

        printf("\t6 For deletion of "

               "last element\n");

        printf("\t7 For deletion of "

               "element at any position\n");

        printf("\t8 To exit\n");

        printf("\nEnter Choice :\n");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            traverse();

            break;

        case 2:

            insertAtFront();

            break;

        case 3:

            insertAtEnd();

            break;

        case 4:

            insertAtPosition();

            break;

        case 5:

            deleteFirst();

            break;

        case 6:

            deleteEnd();

            break;

        case 7:

            deletePosition();

            break;

        case 8:

            exit(1);

            break;

        default:

            printf("Incorrect Choice\n");

        }

    }

    return 0;

}

**Output:-**

  1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :

2

Enter number to be inserted : 10

        1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :

3

Enter number to be inserted : 20

        1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :

4

Enter position and data :2

40

        1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :

1

Data = 10

Data = 40

Data = 20

        1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :

1

Data = 10

Data = 40

Data = 20

        1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :

5

        1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :

6

        1 To see list

        2 For insertion at starting

        3 For insertion at end

        4 For insertion at any position

        5 For deletion of first element

        6 For deletion of last element

        7 For deletion of element at any position

        8 To exit

Enter Choice :