**AIM:**  To implement single linked list.

There are two cases in singly linked list. They are

1. Insertion
2. Insertion in beginning
3. Insertion in middle
4. Insertion in the end
5. Deletion
6. Deletion in beginning
7. Deletion in middle
8. Deletion in the end

**CASE 1:**

**ALGORITHM:**

1. InsertBeg(List, e)

**Step 1 :** Start.

**Step 2 :** Set NewNode = addressof(Node).

**Step 3 :** Set NewNode -> Element = e.

**Step 4 :** If List = NULL, then goto Step 5 else goto Step 6.

**Step 5 :** Set NewNode -> Next = NULL and goto Step 7.

**Step 6 :** Set NewNode -> Next = List -> Next.

**Step 7 :** Set List -> Next = NewNode.

**Step 8:** Stop.

1. InsertMid(List, p, e)

**Step 1 :** Start.

**Step 2 :** Set NewNode = addressof(Node).

**Step 3 :** Set Position = Find(List, p).

**Step 4 :** Set NewNode -> Element = e.

**Step 5 :** Set NewNode -> Next = Position-> Next.

**Step 6 :** Set Position -> Next = NewNode.

**Step 7 :** Stop.

1. InsertLast(List, e)

**Step 1 :** Start.

**Step 2 :** Set NewNode = addressof(Node).

**Step 3 :** Set NewNode -> Element = e.

**Step 4 :** Set NewNode -> Next = NULL.

**Step 5 :** If List = NULL, then goto Step 6 else goto Step 7.

**Step 6 :** Set List -> Next = NewNode and goto Step 11.

**Step 7 :** Set Position = List.

**Step 8 :** Repeat the Step 9 until Position -> Next != NULL.

**Step 9 :** Set Position = Position -> Next.

**Step 10:** Set Position -> Next = NewNode.

**Step 11:** Stop.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

typedef struct node Node;

int isempty(Node\*);

void insertbeginning(Node\*);

void insertmiddle(Node\*);

void insertlast(Node\*);

void traverse(Node\*);

int main()

{

Node \*head=malloc(sizeof(Node));

int choice;

do{

printf("\n1.Insert beginning node \n2.Insert middle node \n3.Insert last node \n4.traverse ");

printf("Enter your choice ");

scanf("%d", &choice);

switch(choice){

case 1:

insertbeginning(head);

traverse(head->next);

break;

case 2:

insertmiddle(head);

traverse(head->next);

break;

case 3:

insertlast(head);

traverse(head->next);

break;

case 4:

traverse(head->next);

break;

}

}while(choice<=4);

return 0;

}

int isempty(Node \*temp)

{

if(temp->next==NULL)

return 1;

else

return 0;

}

void insertbeginning(Node\*temp){

Node \*newnode=malloc(sizeof(Node));

printf("Enter the data ");

scanf("%d", &newnode->data);

if(isempty(temp))

{

temp->next=newnode;

newnode->next=NULL;

}

else

{

newnode->next=temp->next;

temp->next=newnode;

}

}

void insertmiddle(Node \*temp){

Node \*newnode=malloc(sizeof(Node));

printf("Enter the data ");

scanf("%d", &newnode->data);

int key;

printf("Enter the data after which you want to add the new node ");

scanf("%d", &key);

while(temp->next!=NULL&&temp->data!=key){

temp=temp->next;

}

newnode->next=temp->next;

temp->next=newnode;

}

void insertlast(Node \*temp){

Node \*newnode=malloc(sizeof(Node));

printf("Enter the data ");

scanf("%d", &newnode->data);

while(temp->next!=NULL){

temp=temp->next;

}

newnode->next=NULL;

temp->next=newnode;

}

void traverse(Node \*temp){

while(temp!=NULL){

printf("%d->", temp->data);

temp=temp->next;

}

printf("NULL");

}

**OUTPUT FOR THE ABOVE PROGRAM:**

1.Insert beginning node

2.Insert middle node

3.Insert last node

4.traverse Enter your choice 1

Enter the data 22

22->NULL

1.Insert beginning node

2.Insert middle node

3.Insert last node

4.traverse Enter your choice 1

Enter the data 33

33->22->NULL

1.Insert beginning node

2.Insert middle node

3.Insert last node

4.traverse Enter your choice 1

Enter the data 44

44->33->22->NULL

1.Insert beginning node

2.Insert middle node

3.Insert last node

4.traverse Enter your choice 2

Enter the data 55

Enter the data after which you want to add the new node 33

44->33->55->22->NULL

1.Insert beginning node

2.Insert middle node

3.Insert last node

4.traverse Enter your choice 4

44->33->55->22->NULL

**CASE-2:**

**ALGORITHM:**

DeleteBeg(List, e)

**Step 1 :** Start.

**Step 2 :** If !IsEmpty = True, then goto Step 3 else goto Step 7.

**Step 3 :** Set TempNode = List -> Next.

**Step 4 :** Set List -> Next = TempNode -> Next.

**Step 5 :** Display the TempNode -> Element.

**Step 6 :** Delete TempNode and goto Step 8.

**Step 7 :** Display “List is Empty”.

**Step 8:** Stop.

DeleteMid(List, e)

**Step 1 :** Start.

**Step 2 :** If !IsEmpty = True, then goto Step 3 else goto Step 9.

**Step 3 :** Set Position = FindPrevious(List, e).

**Step 4 :** If !Islast(Position) = True, then goto Step 5 else goto Step 10.

**Step 5 :** Set TempNode = Position -> Next.

**Step 6 :** Set Position -> Next = TempNode -> Next.

**Step 7 :** Display the TempNode -> Element.

**Step 8 :** Delete TempNode and goto Step 10.

**Step 9 :** Display “List is Empty”.

**Step 10:** Stop.

DeleteEnd(List)

**Step 1 :** Start.

**Step 2 :** If !IsEmpty = True, then goto Step 3 else goto Step 10.

**Step 3 :** Set Position = List.

**Step 4 :** Repeat the Step 5 until Position◊Next != NULL.

**Step 5 :** Set Position = Position◊Next.

**Step 6 :** Set TempNode = Position◊Next.

**Step 7 :** Set Position◊Next = NULL.

**Step 8 :** Display TempNode◊Element.

**Step 9 :** Delete TempNode and goto Step 11.

**Step 10:** Display “List is Empty”.

**Step 11:** Stop.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

typedef struct node Node;

int isempty(Node\*);

void create(Node\*);

void deletebeginning(Node\*);

void deletemiddle(Node\*);

void deletelast(Node\*);

void traverse(Node\*);

int main()

{

Node \*head=malloc(sizeof(Node));

int choice;

do{

printf("\n1. Creation\n2.Delete beginning node \n3.Delete middle node \n4.Delete last node \n5.traverse \n");

printf("Enter your choice ");

scanf("%d", &choice);

switch(choice){

case 1:

create(head);

traverse(head->next);

break;

case 2:

deletebeginning(head);

traverse(head->next);

break;

case 3:

deletemiddle(head);

traverse(head->next);

break;

case 4:

deletelast(head);

traverse(head->next);

break;

case 5:

traverse(head->next);

break;

}

}while(choice<=4);

return 0;

}

int isempty(Node \*temp)

{

if(temp->next==NULL)

return 1;

else

return 0;

}

void create(Node\*temp){

Node \*newnode=malloc(sizeof(Node));

printf("Enter the data ");

scanf("%d", &newnode->data);

if(temp->next==NULL)

newnode->next=NULL;

else

newnode->next=temp->next;

temp->next=newnode;

}

void deletebeginning(Node \*head){

Node \*temp;

temp=head->next;

head->next=temp->next;

printf("The deleted node is %d\n", temp->data);

free(temp);

}

void deletemiddle(Node \*temp){

Node \*prev;

int key;

printf("Enter the key ");

scanf("%d", &key);

while(temp->next!=NULL&&temp->data!=key){

prev=temp;

temp=temp->next;

}

prev->next=temp->next;

temp->next=NULL;

free(temp);

}

void deletelast(Node \*temp){

Node \*prev;

while(temp->next!=NULL){

prev=temp;

temp=temp->next;

}

prev->next=NULL;

free(temp);

}

void traverse(Node \*temp){

while(temp!=NULL){

printf("%d->", temp->data);

temp=temp->next;

}

printf("NULL");

}

**OUTPUT FOR THE ABOVE PROGRAM:**

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 1

Enter the data 22

22->NULL

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 1

Enter the data 33

33->22->NULL

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 1

Enter the data 44

44->33->22->NULL

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 2

The deleted node is 44

33->22->NULL

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 3

Enter the key 22

33->NULL

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 1

Enter the data 55

55->33->NULL

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 4

55->NULL

1. Creation

2.Delete beginning node

3.Delete middle node

4.Delete last node

5.traverse

Enter your choice 5

55->NULL

**RESULT:** Hence, it is implemented using singly linked list.