**Single linked list**

**Aim**:

The aim of the program is single linked list is excuted

**Algorithm**:

1. Start

2. Create a structure and functions for each operations

3. Display the main menu

4. Read user choice

5. Execute choice operation

6. Display operation completion

7. Back to main menu

8. Check for exit, if no Execute the operation for the given choice

9. Otherwise end

**Program**:

#include<stdio.h>

#include<stdlib.h>

Struct node{

Int data;

Struct node \*next;

};

Void insert\_begin(struct node\*L,int x){

Struct node \*new=(struct node\*)malloc(sizeof(struct node));

If(new!=NULL){

New->data=x;

New->next=L->next;

L->next=new;

}

Else{

Printf(“Memory not allocated”);

}

}

Void insert\_after\_p(struct node\*L,int x,int pos){

Struct node \*new=(struct node\*)malloc(sizeof(struct node));

If(new!=NULL){

New->data=x;

Struct node \*p;

P=L->next;

Int i=1;

While(p!=NULL&&i<pos)

{

P=p->next;

I++;

}

New->next=p->next;

p->next=new;

}

Else{

Printf(“Menory not allocated”);

}

}

Void insert\_end(struct node \*L,int x)

{

Struct node \*new=(struct node\*)malloc(sizeof(struct node));

If(new!=NULL){

New->data=x;

Struct node \*p;

P=L->next;

While(p->next!=NULL)

{

P=p->next;

}

New->next=NULL;

p->next=new;

}

}

Int find(struct node \*L,int x)

{

Struct node \*p;

P=L->next;int i=1;

While(p!=NULL&&p->data!=x)

{

P=p->next;

I++;

}

If(p!=NULL)

Printf(“Element %d is found at position %d”,x,i);

Else

Printf(“element not found”);

}

Int find\_next(struct node \*L,int x){

Struct node \*p=L->next;

While(p!=NULL&&p->data!=x){

P=p->next;

}

If(p!=NULL)

Printf(“Next element after %d is %d”,x,p->next->data);

Else

Printf(“NO next element”);

}

Int find\_prev(struct node\*L,int x)

{

Struct node \*p=L->next;

While(p!=NULL&&p->next->data!=x){

P=p->next;}

If(p!=NULL)

Printf(“Previous element before %d is %d”,x,p->data);

Else

Printf(“NO previous element”);

}

Int islast(struct node\*L,int x){

Struct node \*p=L->next;

While(p->data!=x&&p!=NULL){

P=p->next;}

If(p->next==NULL){

Return 1;

}

Else

Return 0;}

Int isempty(struct node \*L){

If(L->next==NULL){

Return 1;

}

Else{

Return 0;

}}

Void delete\_beginning(struct node \*L){

Struct node \*p=L->next;

L->next=p->next;

Free(p);}

Void delete\_after\_p(struct node\*L,int pos){

Struct node \*p;

Int i=0;

Struct node \*s=L;

While(i<=pos)

{

P=s;

S=s->next;

I++;

}

p->next=s->next;

free(s);

}

Void delete\_end(struct node \*L){

Struct node \*p;

P=L->next;

Struct node \*s=p->next;

While(s->next!=NULL){

P=p->next;

S=p->next;

}

p->next=NULL;

free(s);

}

Void delete\_list(struct node \*L)

{

Struct node \*p=L->next;

Struct node \*s=p->next;

While(p!=NULL)

{

S=p->next;

Free(p);

P=s;

}

L->next=NULL;}

Void display(struct node \*L){

Struct node \*temp=L->next;

While(temp!=NULL){

Printf(“%d”,temp->data);

Temp=temp->next;

}}

Int main(){

Int n,opt,data,position;

Printf(“Enter the no. Of nodes:”);

Scanf(“%d”,&n);

Struct node \*head=(struct node\*)malloc(sizeof(struct node));

Struct node\*p,\*temp=head;

For(int i=0;i<n;i++)

{

P=(struct node\*)malloc(sizeof(struct node));

Printf(“Enter the nodes:”);

Scanf(“%d”,&p->data);

p->next=NULL;

if(head==NULL){

head=p=temp;

}

Else{

Temp->next=p;

Temp=p;

}

}

Do{

Printf(“\n1.Insert at first\n2.Insert after p\n3.Insert at end\n4.Find an element\n5.find next element\n6.find previous element\n7.check whether last or not\n8.Check whether empty or not\n9.delete first node\n10.delete after p\n11.delete at the end\n12.delete list\n13.display\n14.exit\n”);

Printf(“Enter your option:”);

Scanf(“%d”,&opt);

Switch(opt)

{

Case 1:

Printf(“Enter data:”);

Scanf(“%d”,&data);

Insert\_begin(head,data);

Display(head);

Break;

Case 2:

Printf(“Enter data”);

Scanf(“%d”,&data);

Printf(“Enter position after which to insert:”);

Scanf(“%d”,&position);

Insert\_after\_p(head,data,position);

Display(head);

Break;

Case 3:

Printf(“Enter data:”);

Scanf(“%d”,&data);

Insert\_end(head,data);

Display(head);

Break;

Case 4:

Printf(“Enter element to be found:”);

Scanf(“%d”,&data);

Find(head,data);

Break;

Case 5:

Printf(“Enter element to be find next:”);

Scanf(“%d”,&data);

Find\_next(head,data);

Break;

Case 6:

Printf(“Enter element to find previous:”);

Scanf(“%d”,&data);

Find\_prev(head,data);

Break;

Case 7:

Printf(“enter the data to check if its last”);

Scanf(“%d”,&data);

Int a= islast(head,data);

If(a)

Printf(“%d is the last node”,data);

Else

Printf(“%d is not the last node.”,data);

Break;

Case 8:

If(isempty(head))

Printf(“The list is empty”);

Else

Printf(“The list is not empty”);

Break;

Case 9:

Delete\_beginning(head);

Display(head);

Break;

Case 10:

Printf(“enter position after which to delete a node:”);

Scanf(“%d”,&position);

Delete\_after\_p(head,position);

Display(head);

Break;

Case 11:

Delete\_end(head);

Display(head);

Break;

Case 12:

Delete\_list(head);

Display(head);

Break;

Case 13:

Display(head);

Break;

Case 14:

Printf(“\nExiting the program\n”);

Break;

Default:

Printf(“invalid option”);

Break;

}

}while(opt!=14);

Return 0;

}

**Output**:

Enter the no. Of nodes:2

Enter the nodes:1

Enter the nodes:3

1.Insert at first

2.Insert after p

3.Insert at end

4.Find an element

5.find next element

6.find previous element

7.check whether last or not

8.Check whether empty or not

9.delete first node

10.delete after p

11.delete at the end

12.delete list

13.display

14.exit

Enter your option:3

Enter data:1

131

1.Insert at first

2.Insert after p

3.Insert at end

4.Find an element

5.find next element

6.find previous element

7.check whether last or not

8.Check whether empty or not

9.delete first node

10.delete after p

11.delete at the end

12.delete list

13.display

14.exit

Enter your option:14

Exit

**Result:**

The program successfully implemented and excuted