**Program:**

**/\*Creation of a single linked list\*/**

#include<stdio.h>

#include<alloc.h>

void creation();

void traversal();

struct node

{

int data;

struct node \*link;

}\*ptr,\*header,\*new;

void main()

{

int ch;

clrscr();

header =NULL;

while(1)

{

printf("\n Enter the choice of operation 1.creation 2.traversal: ");

scanf("%d",&ch);

switch(ch)

{

case 1: creation();

break;

case 2: traversal();

break;

default: exit(0);

}

}

}

void creation()

{

int item,x,key,pos;

printf("enter the data value to insert");

scanf("%d",&x);

if(header->link==NULL)

{

header->link=new;

new->link=NULL

new->data=x;

}

else

{

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

}

ptr->link=new;

new->link=NULL

new->data=x;

}

}

void traversal()

{

printf("\nelements in the list are");

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

printf("\t%d",ptr->data);

}

}

Output:

enter the choice of operation 1.Creation 2.traversal: 1

enter the data value to insert 10

enter the choice of operation 1.Creation 2.traversal: 2

elements in the list are 10

enter the choice of operation 1.Creation 2.traversal: 1

enter the data value to insert 20

enter the choice of operation 1.Creation 2.traversal: 1

enter the data value to insert 30

enter the choice of operation 1.Creation 2.traversal: 2

elements in the list are 10 20 30

enter the choice of operation 1.Creation 2.traversal: 1

enter the data value to insert 40

enter the choice of operation 1.Creation 2.traversal: 2

elements in the list are 10 20 30 40

enter the choice of operation 1.Creation 2.traversal: 0

**Program:**

**/\*insertion of a single linked list\*/**

#include<stdio.h>

#include<alloc.h>

void insertion();

void traversal();

struct node

{

int data;

struct node \*link;

}\*ptr,\*header,\*new;

void main()

{

int ch;

clrscr();

header =NULL;

while(1)

{

printf("\nenter the choice of operation 1.insert 2.traversal: ");

scanf("%d",&ch);

switch(ch)

{

case 1:insertion();

break;

case 2:traversal();

break;

default:exit(0);

}

}

}

void insertion()

{

int item,x,key,pos;

printf("enter the data value to insert");

scanf("%d",&x);

new=malloc(sizeof(struct node));

printf("enter the position for insertion 1.begining 2.ending 3.At any position");

scanf("%d",&pos);

/\* insertion at beginning\*/

if(pos==1)

{

new->link= header->link;

header->link=new;

new->data=x;

}

/\* insertion at ending\*/

else if(pos==2)

{

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

}

ptr->link=new;

new->link=NULL;

new->data=x;

}

/\* insertion at any pos\*/

else if(pos==3)

{

printf("\nenter key value");

scanf("%d",&key);

ptr=header;

while(ptr->link!=NULL && ptr->data!=key)

{

ptr=ptr->link;

}

if(ptr->link==NULL)

{

/\* Special case i.e. insertion of a node at any position that leads to insertion at end\*/

if(ptr->data==key)

{

new->link=ptr->link;

ptr->link=new;

new->data=x;

}

else

{

printf("\n Key not available");

}

}

else

{

new->link=ptr->link;

ptr->link=new;

new->data=x;

}

}

}

void traversal()

{

printf("\n elements in the list are");

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

printf("\t%d",ptr->data);

}

}

Output:

enter the choice of operation 1.insert 2.traversal: 1

enter the data value to insert 10

enter the position for insertion 1.begining 2.ending 3.At any position1

enter the choice of operation 1.insert 2.traversal: 2

elements in the list are 10

enter the choice of operation 1.insert 2.traversal: 1

enter the data value to insert 20

enter the position for insertion 1.begining 2.ending 3.At any position2

enter the choice of operation 1.insert 2.traversal: 2

elements in the list are 10 20

enter the choice of operation 1.insert 2.traversal: 1

enter the data value to insert 30

enter the position for insertion 1.begining 2.ending 3.At any position 1

enter the choice of operation 1.insert 2.traversal: 2

elements in the list are 30 10 20

enter the choice of operation 1.insert 2.traversal: 1

enter the choice of operation 1.insert 2.traversal: 1

enter the data value to insert 40

enter the position for insertion 1.begining 2.ending 3.At any position 3

enter key value 10

enter the choice of operation 1.insert 2.traversal: 2

elements in the list are 30 10 40 20

enter the choice of operation 1.insert 2.traversal:0

**Program:**

**/\* SLL DELETION \*/**

#include<stdio.h>

#include<malloc.h>

void create();

void delete();

void traverse();

struct node

{

int data;

struct node \*link;

}\*header,\*new,\*ptr,\*ptr1;

void main()

{

int ch;

clrscr();

header=NULL;

while(1)

{

printf("\n\nEnter the choice of operation");

printf("\n1.Create \t 2.Delete \t 3. Traversal\n");

scanf("%d",&ch);

switch(ch)

{

case 1: create();

break;

case 2: delete();

break;

case 3: traverse();

break;

default:exit(0);

}

}

getch();

} void creation()

{

int item,x,key,pos;

printf("enter the data value to insert");

scanf("%d",&x);

if(header->link==NULL)

{

header->link=new;

new->link=NULL

new->data=x;

}

else

{

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

}

ptr->link=new;

new->link=NULL

new->data=x;

}

}

void delete()

{

int pos,x,key;

printf("\nEnter the position for deletion");

printf("\n 1.Begining 2.Ending\t3.At any Position\n");

scanf("%d",&pos);

if(pos==1) /\*Deletion at beginning\*/

{

ptr=header;

if(ptr->link==NULL)

{

printf("\n SLL is empty");

}

else

{

ptr1=ptr;

ptr=ptr->link;

ptr1->link=ptr->link; /\* Address of second node i.e link part of first node is copied to link part of header node\*/

printf("\nNode deleted is %d",ptr->data);

free(ptr);

}

}

else if (pos==2) /\* Deletion at Ending \*/

{

ptr=header;

if(ptr->link==NULL)

{

printf("\n SLL is empty, unable to perform deletion");

}

else

{

ptr1=ptr;

ptr=ptr->link;

while(ptr->link!=NULL)

{

ptr1=ptr;

ptr=ptr->link;

}

ptr1->link=NULL; /\* Last but one node link part is replaced with NULL \*/

printf("\nDeleted Node is %d",ptr->data);

free(ptr);

}

}

else if(pos==3) /\* Deletion at any position \*/

{

ptr=header;

if(ptr->link==NULL)

{

printf("\n SLL is empty, unable to perform deletion");

}

else

{

printf("\nEnter the data value to delete");

scanf("%d",&key);

while(ptr->data!=key && ptr->link!=NULL)

{

ptr1=ptr;

ptr=ptr->link; /\* Move to the next node\*/

}

if(ptr->link==NULL)

{

printf("\n Node with key was not found");

}

else

{

ptr1->link=ptr->link;

printf("\nDeleted node is %d",ptr->data);

free(ptr);

}

}

}

}

void traverse()

{

printf("\n Elements in the list are:\n");

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

printf("\t%d",ptr->data);

}

}

**Output:**

Enter the choice of operation

1.Create 2.Delete 3. Traversal

1

Enter the data value10

Enter the choice of operation

1.Create 2.Delete 3. Traversal

1

Enter the data value 20

Enter the choice of operation

1.Create 2.Delete 3. Traversal

1

Enter the data value 30

Enter the choice of operation

1.Create 2.Delete 3. Traversal

1

Enter the data value 40

Enter the choice of operation

1.Create 2.Delete 3. Traversal

1

Enter the data value 50

Enter the choice of operation

1.Create 2.Delete 3. Traversal

1

Enter the data value60

Enter the choice of operation

1.Create 2.Delete 3. Traversal

3

Elements in the list are:

10 20 30 40 50 60

Enter the choice of operation

1.Create 2.Delete 3. Traversal

2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

1

Node deleted is 10

Enter the choice of operation

1.Create 2.Delete 3. Traversal

3

Elements in the list are:

20 30 40 50 60

Enter the choice of operation

1.Create 2.Delete 3. Traversal

2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

2

Deleted Node is 60

Enter the choice of operation

1.Create 2.Delete 3. Traversal

3

Elements in the list are:

20 30 40 50

Enter the choice of operation

1.Create 2.Delete 3. Traversal

2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

3

Enter the data value to delete 30

Deleted node is 30

Enter the choice of operation

1.Create 2.Delete 3. Traversal

3

Elements in the list are:

20 40 50

Enter the choice of operation

1.Create 2.Delete 3. Traversal

2

Enter the position for deletion

1.Begining 2.Ending 3.At any Position

3

Enter the data value to delete65

Node with key was not found

Enter the choice of operation

1.Create 2.Delete 3. Traversal

0

**Program:**

**/\* Reversing a single linked list \*/**

#include<stdio.h>

#include<alloc.h>

void creation();

void rverse();

void traversal();

struct node

{

int data;

struct node \*link;

}\*ptr,\*header,\*new,\*current,\*next,\*prev;

void main()

{

int ch;

clrscr();

header =NULL;

while(1)

{

printf("\n Enter the choice of operation 1.creation 2.traversal: ");

scanf("%d",&ch);

switch(ch)

{

case 1: creation();

break;

case 2: reverse();

break;

case 3: traversal();

break;

default: exit(0);

}

}

}

void creation()

{

int item,x,key,pos;

printf("enter the data value to insert");

scanf("%d",&x);

if(header->link==NULL)

{

header->link=new;

new->link=NULL

new->data=x;

}

else

{

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

}

ptr->link=new;

new->link=NULL

new->data=x;

}

}

void traversal()

{

printf("\nelements in the list are");

ptr=header;

while(ptr->link!=NULL)

{

ptr=ptr->link;

printf("\t%d",ptr->data);

}

}

void reverse()

{

prev = NULL;

current = header;

next = NULL;

while (current != NULL) {

// Store next

next = current->next;

// Reverse current node's pointer

current->next = prev;

// Move pointers one position ahead.

prev = current;

current = next;

}

header = prev;

}