WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Deletion of first element, specified element and last element in the list. d) Display the contents of the linked list.

#include<stdlib.h>

#include <stdio.h>

void create();

void display();

void insert\_begin();

void insert\_end();

void insert\_pos();

void delete\_begin();

void delete\_end();

void delete\_pos();

struct node

{

int info;

struct node \*next;

};

struct node \*start=NULL;

int main()

{

int choice;

while(1){

printf("\n\*\*\*MENU\*\*\*\n");

printf("\n 1. Create a list");

printf("\n 2. Display the list");

printf("\n 3. Insert node at the beginning");

printf("\n 4. Insert node at the end");

printf("\n 5. Insert node at any specified position");

printf("\n 6. Delete node from the beginning");

printf("\n 7. Delete node from the end");

printf("\n 8. Delete node from any specified position");

printf("\n 9. Exit");

printf("\n Enter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

insert\_begin();

break;

case 4:

insert\_end();

break;

case 5:

insert\_pos();

break;

case 6:

delete\_begin();

break;

case 7:

delete\_end();

break;

case 8:

delete\_pos();

break;

case 9:

exit(0);

break;

default:

printf("\n Wrong Choice!\n");

break;

}

}

return 0;

}

void create()

{

struct node \*temp,\*ptr;

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

printf("\nEnter the value for the node: ");

scanf("%d",&temp->info);

temp->next=NULL;

if(start==NULL)

{

start=temp;

}

else

{

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

void display()

{

struct node \*ptr;

if(start==NULL)

{

printf("\nList is empty!\n");

return;

}

else

{

ptr=start;

printf("\n The List elements are:\n");

while(ptr!=NULL)

{

printf("%d ",ptr->info );

ptr=ptr->next ;

}

}

}

void insert\_begin()

{

struct node \*temp;

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

printf("\nEnter the value for the node: " );

scanf("%d",&temp->info);

temp->next =NULL;

if(start==NULL)

{

start=temp;

}

else

{

temp->next=start;

start=temp;

}

}

void insert\_end()

{

struct node \*temp,\*ptr;

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

printf("\nEnter the value for the node: " );

scanf("%d",&temp->info );

temp->next =NULL;

if(start==NULL)

{

start=temp;

}

else

{

ptr=start;

while(ptr->next !=NULL)

{

ptr=ptr->next ;

}

ptr->next =temp;

}

}

void insert\_pos()

{

struct node \*ptr,\*temp;

int i,pos;

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

printf("\nEnter the position for the new node to be inserted: ");

scanf("%d",&pos);

printf("\nEnter the value of the node: ");

scanf("%d",&temp->info) ;

temp->next=NULL;

if(pos==0)

{

temp->next=start;

start=temp;

}

else

{

for(i=0,ptr=start;i<pos-1;i++)

{

ptr=ptr->next;

}

temp->next =ptr->next ;

ptr->next=temp;

}

}

void delete\_begin()

{

struct node \*ptr;

if(start==NULL)

{

printf("\nList is Empty!\n");

return;

}

else

{

ptr=start;

start=start->next ;

printf("\nThe deleted element is : %d ",ptr->info);

free(ptr);

}

}

void delete\_end()

{

struct node \*temp,\*ptr;

if(start==NULL)

{

printf("\nList is Empty!\n");

exit(0);

}

else if(start->next ==NULL)

{

ptr=start;

start=NULL;

printf("\nThe deleted element is: %d ",ptr->info);

free(ptr);

}

else

{

ptr=start;

while(ptr->next!=NULL)

{

temp=ptr;

ptr=ptr->next;

}

temp->next=NULL;

printf("\nThe deleted element is: %d ",ptr->info);

free(ptr);

}

}

void delete\_pos()

{

int i,pos;

struct node \*temp,\*ptr;

if(start==NULL)

{

printf("\nThe List is Empty!\n");

exit(0);

}

else

{

printf("\nEnter the position of the node to be deleted: \n");

scanf("%d",&pos);

if(pos==0)

{

ptr=start;

start=start->next ;

printf("\nThe deleted element is: %d ",ptr->info );

free(ptr);

}

else

{

ptr=start;

for(i=0;i<pos;i++)

{

temp=ptr;

ptr=ptr->next ;

if(ptr==NULL)

{

printf("\nPosition not Found!\n");

return;

}

}

temp->next =ptr->next ;

printf("\nThe deleted element is: %d ",ptr->info );

free(ptr);

}

}

}

OUTPUT:

 

 

 

 