/\*

GOPIKRISHNA V

S3 CSE A

52

Menu Driven Doubly Linked List

\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node \*llink,\*rlink;

int data;

};

struct node \*head=NULL,\*ptr,\*temp;

void display()

{

ptr=head;

if(ptr==NULL)

printf("List Empty\n\n");

else

{

printf("List >> [");

while(ptr!=NULL)

{

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

ptr=ptr->rlink;

}

printf("\b]\n");

}

}

void insertF(int x)

{

struct node \*new;

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

new->data=x;

new->llink=NULL;

new->rlink=NULL;

if(head==NULL)

head=new;

else

{

new->rlink=head;

head->llink=new;

head=new;

}

printf("%d --> INSERTED\n\n",x);

}

void insertE(int x)

{

struct node \*new;

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

new->data=x;

new->llink=NULL;

new->rlink=NULL;

if(head==NULL)

head=new;

else

{

ptr=head;

while(ptr->rlink!=NULL)

{

ptr=ptr->rlink;

}

ptr->rlink=new;

new->llink=ptr;

}

printf("%d --> INSERTED\n\n",x);

}

void insertP(int x)

{

int key;

printf("Enter the Key (Element) = ");

scanf("%d",&key);

if(head==NULL)

printf("List Empty\n");

else

{

ptr=head;

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

ptr=ptr->rlink;

if(ptr->data != key)

printf("Key Not Found - Insertion Not Possible\n");

else

{

struct node \*new;

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

new->data=x;

new->llink=ptr;

new->rlink=ptr->rlink;

if(new->rlink !=NULL)

new->rlink->llink=new;

ptr->rlink=new;

printf("%d --> INSERTED\n\n",x);

}

}

}

void deleteF()

{

if(head==NULL)

printf("List Empty\n");

else

{

if(head->rlink==NULL)

{

temp=head;

head=NULL;

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

free(temp);

}

else

{

head=head->rlink;

printf("%d --> DELETED\n",head->rlink->data);

free(head->llink);

head->llink=NULL;

}

}

}

void deleteE()

{

if(head==NULL)

printf("List Empty\n");

else

{

if(head->rlink==NULL)

{

temp=head;

head=NULL;

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

free(temp);

}

else

{

ptr=head;

while(ptr->rlink!=NULL)

{

ptr=ptr->rlink;

}

ptr->llink->rlink=NULL;

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

free(ptr);

}

}

}

void deleteP()

{

int key;

printf("Enter the Key (Element) = ");

scanf("%d",&key);

if(head==NULL)

printf("List Empty\n");

else

{

if(head->rlink==NULL)

{

if(head->data==key)

{

temp=head;

head=NULL;

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

free(temp);

}

else

{

printf("Key Not Found - Deletion Not Possible\n");

}

}

else

{

if(head->data=key)

{

head=head->rlink;

printf("%d --> DELETED\n",head->llink->data);

free(head->llink);

head->llink=NULL;

}

else

{

ptr=head;

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

{

ptr=ptr->rlink;

}

if(ptr->data!=key)

printf("Key Not Found - Deletion Not Possible\n");

else

{

ptr->llink->rlink=ptr->rlink;

if(ptr->rlink!=NULL)

{

ptr->rlink->llink=ptr->llink;

}

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

free(ptr);

}

}

}

}

}

int get()

{

int x;

printf("Enter the Element = ");

scanf("%d",&x);

return x;

}

void main()

{

start:

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

printf("1.Display\n");

printf("2.Insert at Front\n");

printf("3.Insert at End\n");

printf("4.Insert at Key Position\n");

printf("5.Delete from Front\n");

printf("6.Delete from End\n");

printf("7.Delete from Key Position\n");

printf("0.Exit\n");

int ch;

printf("Enter the Choice = ");

scanf("%d",&ch);

switch(ch)

{

case 1:display();

break;

case 2:insertF(get());

break;

case 3:insertE(get());

break;

case 4:insertP(get());

break;

case 5:deleteF();

break;

case 6:deleteE();

break;

case 7:deleteP();

break;

case 0:exit(0);

break;

default:printf("Wrong Input\n");

}

goto start;

}

**OUTPUT**

 

 