|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No. 09**

|  |  |
| --- | --- |
| Semester | S.E. Semester III – Computer Engineering |
| Subject | Data Structures Lab (CSL301) |
| Subject Professor In-charge | Prof. Swapnil S. Sonawane |
| Assisting Teachers | Prof. Swapnil S. Sonawane |

|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**Title:**

Implement Circular Linked List ADT.

**Objective:**

Students will be able to explain various data structures, related terminologies and its types.

**Explanation:**

Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.

We traverse a circular singly linked list until we reach the same node where we started. The circular singly liked list has no beginning and no ending. There is no null value present in the next part of any of the nodes.

Circular linked list are mostly used in task maintenance in operating systems. There are many examples where circular linked list are being used in computer science including browser surfing where a record of pages visited in the past by the user, is maintained in the form of circular linked lists and can be accessed again on clicking the previous button.

**Program Code:**

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node \*next;

}node;

typedef struct LL

{

node \*last;

}LL;

void insertbegin(LL \*l,int x)

{

node \*newrec;

newrec=(node\*)malloc(sizeof(node));

newrec->data=x;

newrec->next=NULL;

if(l->last==NULL)

{

newrec->next=newrec;

l->last=newrec;

}

else

{

newrec->next=l->last->next;

l->last->next=newrec;

}

}

void insertend(LL \*l,int x)

{

node \*newrec;

newrec=(node\*)malloc(sizeof(node));

newrec->data=x;

newrec->next=NULL;

if(l->last==NULL)

{

newrec->next=newrec;

l->last=newrec;

}

else

{

newrec->next=l->last->next;

l->last->next=newrec;

l->last=newrec;

}

}

void deletebegin(LL \*l)

{

node \*p;

if(l->last==NULL)

{

printf("\nDeletion not possible...");

}

else

{

p=l->last->next;

if(p->next==p)//only one node

{

l->last=NULL;

}

else

{

l->last->next=p->next;

}

free(p);

}

}

void deleteend(LL \*l)

{

node \*p,\*q;

if(l->last==NULL)

{

printf("\nDeletion not possible...");

}

else

{

p=l->last;

if(p->next==p)//only one node

{

l->last=NULL;

}

else

{

q=l->last->next;

while(q->next!=l->last)

{

q=q->next;

}

q->next=p->next;

l->last=q;

}

free(p);

}

}

void display(LL \*l)

{

node \*p;

if(l->last==NULL)

{

printf("\nList is empty...");

}

else

{

p=l->last->next;

do

{

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

p=p->next;

}

while(p!=l->last->next);

}

}

int main()

{

int ch,x;

LL l;

l.last=NULL;

while(1)

{

printf("\nMenu:\n1-Insert at beginning\n2-Insert at end\n3-Delete at beginning\n4-Delete at end\n5-Display\n6-EXIT\nEnter Choice=");

scanf("%d",&ch);

if(ch==6)

break;

switch(ch)

{

case 1:

{

printf("\nEnter element to be inserted=");

scanf("%d",&x);

insertbegin(&l,x);

display(&l);

}

break;

case 2:

{

printf("\nEnter element to be inserted=");

scanf("%d",&x);

insertend(&l,x);

display(&l);

}

break;

case 3:

{

deletebegin(&l);

display(&l);

}

break;

case 4:

{

deleteend(&l);

display(&l);

}

break;

case 5:

{

display(&l);

}

break;

default:

{

printf("\nInvalid Choice...");

}

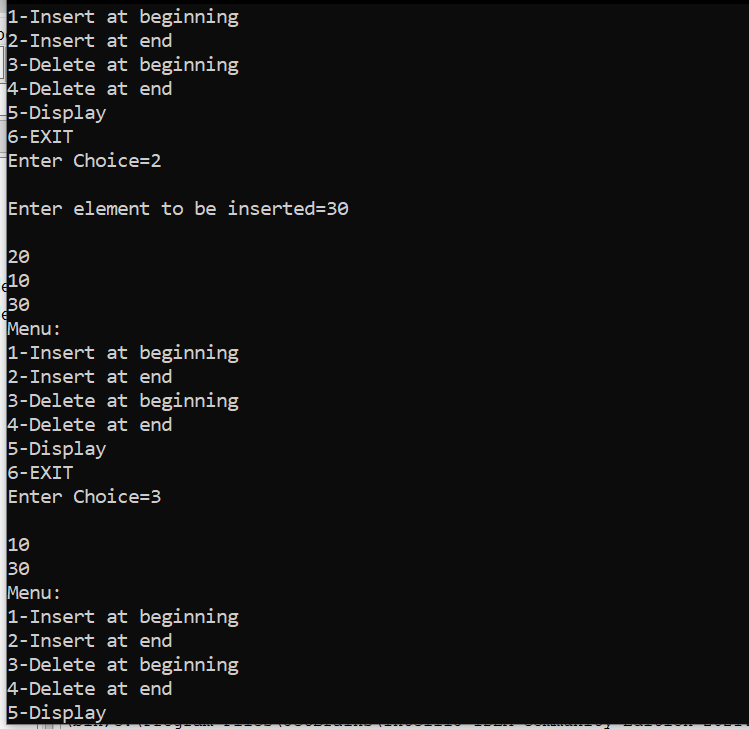
}

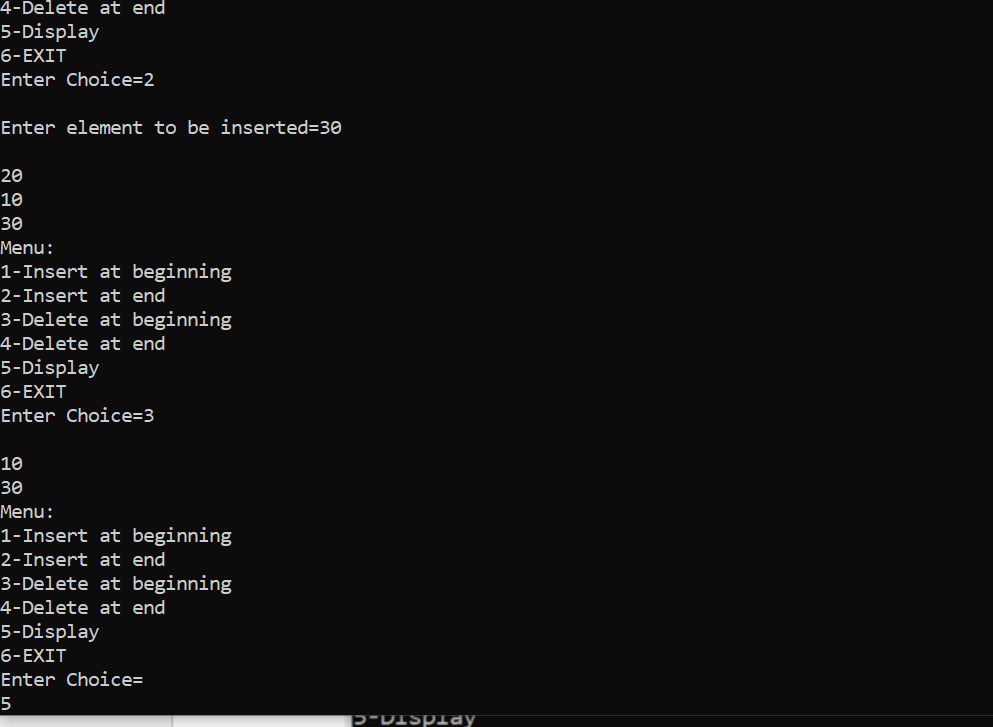
}

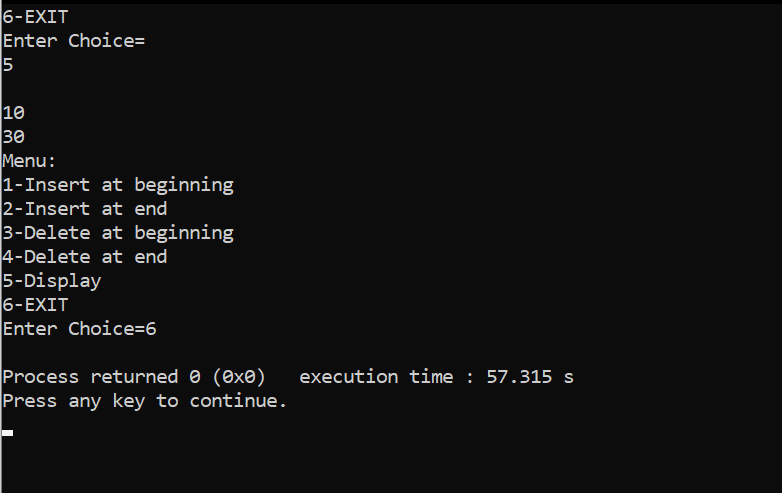
return 0;

}

**Output:**

****

****

****

**Conclusion:**

Through this experiment students were able to successfully implement a circular Linked List.