**TITLE 40**

Write a C program to implement a Circular-Queue, user must choose the following options:

a. Add an element to the Queue – EnQueue.

b. Remove an element from the Queue – DeQueue.

c. Display the elements of the Queue.

d. Terminate the program.

**OBJECTIVE:**

By the end of this problem we will be able to implement Circular-Queue

**PROBLEM STATEMENT:**

In this program we will be implementing Circular-Queue with the user choosing one of the above options in the question. Input from user:

Enter your choice:

Once the input is collected and stored, the output is displayed.

**ALGORITHM:**

START

Define variables: cqueue\_arr, item

INPUT: Read from the user

COMPUTATION: Computing the implementation of Circular-Queue

DISPLAY: Displaying the output according to the user’s choice

STOP

**PROGRAM:**

# include<stdio.h>  
# define MAX 5  
  
int cqueue\_arr[MAX];  
int front = -1;  
int rear = -1;  
  
/\*Begin of insert\*/  
void insert(int item)  
{  
if((front == 0 && rear == MAX-1) || (front == rear+1))  
{  
printf("Queue Overflow \n");  
return;  
}  
if (front == -1)  /\*If queue is empty \*/  
{  
front = 0;  
rear = 0;  
}  
else  
{  
if(rear == MAX-1) /\*rear is at last position of queue \*/  
rear = 0;  
else  
rear = rear+1;  
}  
cqueue\_arr[rear] = item ;  
}  
/\*End of insert\*/  
  
/\*Begin of del\*/  
void del()  
{  
if (front == -1)  
{  
printf("Queue Underflow\n");  
return ;  
}  
printf("Element deleted from queue is : %d\n",cqueue\_arr[front]);  
if(front == rear) /\* queue has only one element \*/  
{  
front = -1;  
rear=-1;  
}  
else  
{  
if(front == MAX-1)  
front = 0;  
else  
front = front+1;  
}  
}  
/\*End of del() \*/  
  
/\*Begin of display\*/  
void display()  
{  
int front\_pos = front,rear\_pos = rear;  
if(front == -1)  
{  
printf("Queue is empty\n");  
return;  
}  
printf("Queue elements :\n");  
if( front\_pos <= rear\_pos )  
while(front\_pos <= rear\_pos)  
{  
printf("%d ",cqueue\_arr[front\_pos]);  
front\_pos++;  
}  
else  
{  
while(front\_pos <= MAX-1)  
{  
printf("%d ",cqueue\_arr[front\_pos]);  
front\_pos++;  
}  
front\_pos = 0;  
while(front\_pos <= rear\_pos)  
{  
printf("%d ",cqueue\_arr[front\_pos]);  
front\_pos++;  
}  
}  
printf("\n");  
}  
/\*End of display\*/  
  
/\*Begin of main\*/  
int main()  
{  
int choice,item;  
do  
{  
printf("1.Insert\n");  
printf("2.Delete\n");  
printf("3.Display\n");  
printf("4.Quit\n");  
  
printf("Enter your choice : ");  
scanf("%d",&choice);  
  
switch(choice)  
{  
case 1 :  
printf("Input the element for insertion in queue : ");  
scanf("%d", &item);  
  
insert(item);  
break;  
case 2 :  
del();  
break;  
case 3:  
display();  
break;  
case 4:  
break;  
default:  
printf("Wrong choice\n");  
}  
}while(choice!=4);  
  
return 0;  
}

**CONCLUSION:**

The simulation of the above program helped me learn about how Circular-Queue is implemented according to the user’s choices.

**OUTPUT:**

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 1

Input the element for insertion in queue : 7

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 1

Input the element for insertion in queue : 8

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 1

Input the element for insertion in queue : 9

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 3

Queue elements :

7 8 9

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 4