INDEX

1. **Introduction to data structures**
2. **Array Implementation Of Stack**
3. **Application Of Stack – Conversion Of Infix To Postfix**
4. **Implementation Of Linear Queue Using Arrays**
5. **Array Implementation Of Circular Queue**
6.

Ex. No.:5

Date :

Implementation of Circular Queue Using Arrays

Aim

To write a C-program to implement circular queue data structure using arrays.

Operations on queue:

MakeEmpty(q): To make q as an empty queue
Enqueue(q, x): To insert an item x at the rear of the queue, this is also called by names add, insert.
Dequeue(q): To delete an item from the front of the queue q. This is also known as Delete, Remove.
IsFull(q): To check whether the queue q is full.
IsEmpty(q): To check whether the queue q is empty
Traverse (q): To read entire queue that is display the content of the queue.

Algorithm for insertion an item in circular queue:

```
1. This algorithm is assume that rear and front are initially set to -1
        if(front== (rear+1)%MAXSIZE)
            print "queue overflow" and return
        else
            set rear=(rear+1)%MAXSIZE
            cqueue[rear]=item
2. End.
```

Algorithm to delete an element from the circular queue:

```
1. if (front==-1)
        print "queue is empty" and return
   else
        item=cqueue[front++]

        If(front == rear) //check if the queue contains only one element
            Set front = rear=-1
        else
        front = front +1
2. end
```

Declaration of a Queue:

```
# define MAXQUEUE 100 /* size of the queue items*/
struct queue
{
      int front;
      int rear;
      int items[MAXQUEUE];
};
typedef struct queue qt;
```

Defining the operations of circular queue:

1. The MakeEmpty function:

```
void makeEmpty(qt  *q)
{
     q->rear=-1;
    q->front=-1;
}
```

Program

```
/*Array implementation of circular queue*/
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5

struct cqueue
{
int items[SIZE];
int front;
int rear;
};
typedef struct cqueue cq;

void make_empty(cq *q){
        q->front = -1;
        q->rear =-1;
}

int isFull(cq* q)
{
    if( (q->front ==(q->rear + 1)%SIZE))
        return 1;
    return 0;
}

int isEmpty(cq *q)
{
    if(q->front == -1)
        return 1;
    return 0;
```

```
}

void enQueue(cq *q, int element)
{
    if(isFull(q))
        printf("\n Queue is full!! \n");
    else
    {
        if(q->front == -1)
               q->front = 0;
        q->rear = (q->rear + 1) % SIZE;
        q->items[q->rear] = element;
        printf("\n Inserted -> %d \n", q->rear);
    }
}


int deQueue(cq *q)
{
    int element;
    if(isEmpty(q)) {
        printf("\n Queue is empty !! \n");
        return(-1);
    } else {
        element = q->items[q->front];
        if (q->front == q->rear){
            make_empty(q);
        } /* Q has only one element, so we reset the queue after dequeing it. ? */
        else {
            q->front = (q->front + 1) % SIZE;

        }
        printf("\n Deleted element -> %d \n", element);
        return(element);
    }
}

void display(cq *q)
{
    int i;
    if(isEmpty(q)) printf(" \n Empty Queue\n");
    else
    {

        printf("\n Items -> ");
        for( i = q->front; i!=q->rear; i=(i+1)%SIZE) {
            printf("%d ",q->items[i]);
        }
        printf("%d ",q->items[i]);
    }
}
void main()
{
    int choice,newElement;
    cq q;
    make_empty(&q);
    while (1)
    {
        printf("\n\n1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("enter new element");
                scanf("%d", &newElement);
                enQueue(&q,newElement);
                break;
            case 2:
                newElement = deQueue(&q);
                break;
            case 3:
                display(&q);
                 break;
            case 4:
            exit(1);
            default:
            printf("Wrong choice \n");
        } /*End of switch*/
    } /*End of while*/
} /*End of main()*/
```