Mechi Multiple Campus

(Tribhuvan University)
Bhadrapur, Jhapa

Lab Report of

Data Structures and Algorithm (CACS-201)

Implementation of Circular Queue

Faculty of Humanities & Social Sciences

Tribhuvan University

Kritipur, Nepal

Submitted By

Name: Santosh Bhandari

Roll No: 58

Submitted To

Mechi Multiple Campus

Department of Bachelor in Computer Application

Bhadrapur, Jhapa, Nepal

Introduction to Circular Queue

Circular Queue is also a linear data structure where last element of queue is connected to the first element, thus creating a circle. It is the one in which the insertion of a new element is done at very first location of the queue if the last location of the queue is full. A circular queue overcomes the problem of unutilized space in linear queue implementation as array.

Advantages of Circular Queue over linear Queue

i) Circular Queue requires less memory as compare to linear queue.

ii) Elements can be inserted easily if there are vacant locations.

iii) In circular queue, the rear and front end are not fixed so the insertion and deletion can be changed which is useful.

iv) In Circular Queue, element can be store if any location is vacant.

Algorithm to insert and delete data from the Circular Queue

Enqueue Operation Algorithm

```
① [check if the Queue is Full or Not]
        if REAR = MAX-1 and FRONT = 0
            then print OVERFLOW and Exit
        OR
        FRONT = REAR+1
            then print OVERFLOW and Exit
② [update the value of FRONT and REAR]
        if REAR = -1
            then Set REAR = 0 and FRONT = 0
        else if REAR = MAX-1
            then Set REAR = 0
        else
            REAR = REAR+1
③ [Insert Data]
        QUEUE[REAR] = new data
④ Exit
```

Dequeue Operation Algorithm

```
① [check if the Queue is empty or not]
        if REAR=-1
            then print UNDERFLOW and Exit
② [Delete Data]
        Delete QUEUE [FRONT]
③ [update the Value of FRONT and REAR]
        if REAR=FRONT
            then Set REAR=-1 and FRONT=-1
        else if FRONT=MAX-1
            then Set FRONT=0
        else
            FRONT=FRONT+1
④ Exit
```

Program Code

```
#include<stdio.h>
void enqueue();
void dequeue();
void display();
int queue[3],front=-1,rear=-1,max=3;
void main(){
        top:
        printf("\n\n***Option***\n1.Insert Data in Queue\n2.Remove Data From
Queue\n3.Display Data of Queue\n\nSelect Your Option(1,2,3): ");
        int n;
        scanf("%d",&n);
        switch(n){
                case 1:
                        enqueue();
                        goto top;
                case 2:
                        dequeue();
                        goto top;
                case 3:
                        display();
                        goto top;
                case 4:
                        exit(0);
                default:
                        printf("Wrong Entry.");
                        goto top;
        }
}
void enqueue(){
        if(rear==(max-1) && front==0 || front==rear+1)
                printf("OVERFLOW");
        else {
                if(rear==-1)
                        rear=front=0;
                else if(rear==(max-1))
                        rear=0;
                else
                        rear++;
                printf("Enter a Data: ");
                scanf("%d",&queue[rear]);
                printf("%d inserted in Queue.",queue[rear]);
        }
}
void dequeue(){
        if(rear==-1)
                printf("UNDERFLOW");
        else{
                printf("%d Deleted from Queue.",queue[front]);
                if(front==rear)
                        front=rear=-1;
                else
                        if(front==(max-1))
                                front=0;
                        else
                                front++;
        }
}
void display(){
        if(rear==-1)
                printf("Queue is Empty.");
        else if (rear>front){
                int i;
                printf("Data on Queue: ");
                for(i=front;i<=rear;i++)
                        printf("%d\t",queue[i]);
        }
        else{
                int i,j;
                printf("Data on Queue: ");
                for(i=front;i<max;i++)
                        printf("%d\t",queue[i]);
                for(j=0;j<=rear;j++)
                        printf("%d\t",queue[j]);
        }
}
```

Output of the Program

```
***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 1
Enter a Data: 10
10 inserted in Queue.

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 1
Enter a Data: 20
20 inserted in Queue.

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 1
Enter a Data: 30
30 inserted in Queue.
```

```
***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 3
Data on Queue: 10       20      30

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 1
OVERFLOW
***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 2
10 Deleted from Queue.

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 1
Enter a Data: 40
40 inserted in Queue.

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 1
OVERFLOW

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 2
20 Deleted from Queue.
```

```
***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 2
30 Deleted from Queue.

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 2
40 Deleted from Queue.

***Option***
1.Insert Data in Queue
2.Remove Data From Queue
3.Display Data of Queue

Select Your Option(1,2,3): 2
UNDERFLOW
```

Conclusion

Circular Queue is a linear data structure where last element of the Queue is connected to the first element, thus creating a circle. In this queue, the user can insert the data if any of the location of queue is empty.