Fundamentals of
Data Structures using C

Circular Queue

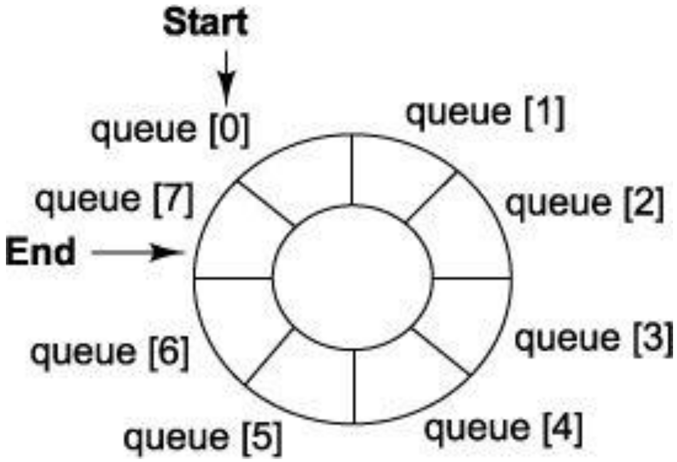
B.Bhuvaneswaran, AP (SG) / CSE

☎ 9791519152
✉ bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

Introduction

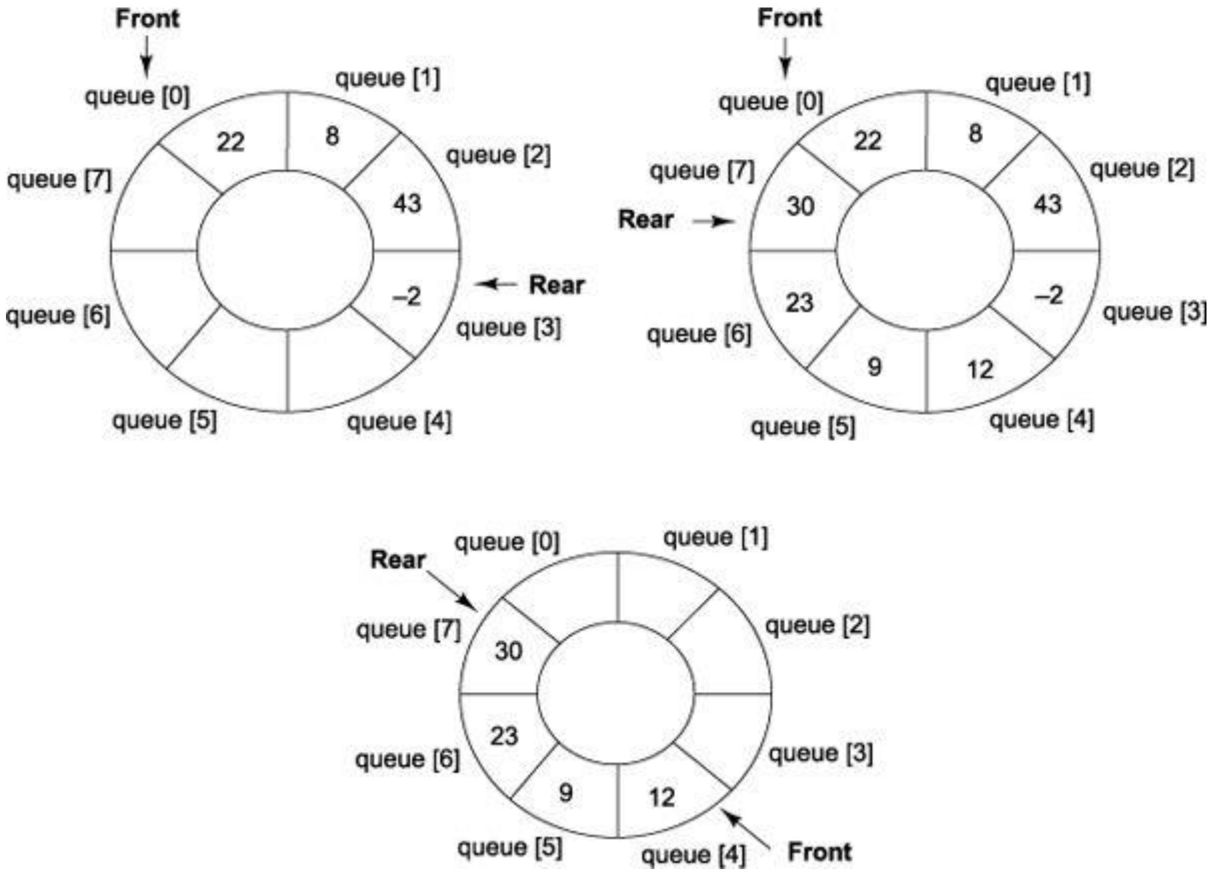
- A circular queue is a queue whose start and end locations are logically connected with each other.
- That means, the start location comes after the end location.
- If we continue to add elements in a circular queue till its end location, then after the end location has been filled, the next element will be added at the beginning of the queue.



Introduction

- As we can see in Fig., the start location of the queue comes after its end location.
- Thus, if the queue is filled till its capacity, i.e., the end location, then the start location will be checked for space, and if it is empty, the new element will be added there.

Introduction



Advantages

- Circular queues remove one of the main disadvantages of array implemented queues in which a lot of memory space is wasted due to inefficient utilization.

Check whether a Queue is Full

```
int IsFull()
{
        if(front == (rear + 1) % MAX)
                return 1;
        else
                return 0;
}
```

Check whether a Queue is Empty

```
int IsEmpty()
{
        if(front == -1)
                return 1;
        else
                return 0;
}
```

Enqueue an Element on to the Queue

```
void Enqueue(int ele)
{
        if(IsFull())
                printf("Queue is Overflow...!\n");
        else
        {
                rear = (rear + 1) % MAX;
                CQueue[rear] = ele;
                if(front == -1)
                        front = 0;
        }
}
```

Dequeue an Element from the Queue

```
void Dequeue()
{
        if(IsEmpty())
                printf("Queue is Underflow...!\n");
        else
        {
                printf("%d\n", CQueue[front]);
                if(front == rear)
                        front = rear = -1;
                else
                        front = (front + 1) % MAX;
        }
}
```

Display Queue Elements

```
void Display()
{
        int i;
        if(IsEmpty())
                printf("Queue is Underflow...!\n");
        else
        {
                for(i = front; i != rear; i = (i + 1) % MAX)
                        printf("%d\t", CQueue[i]);
                printf("%d\n", CQueue[i]);
        }
}
```

Queries?

Thank You!