Fundamentals of
Data Structures using C

Array Implementation of Queue

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

Introduction

- In this implementation each queue is associated with two pointers namely front and rear, which is -1 for an empty queue.

Enqueue

- To insert an element X onto the queue, the rear pointer is incremented by 1 and then set:

  Queue[rear] = X.

Dequeue

- To delete an element from the queue, the Queue[front] value is returned and the front pointer is incremented by 1.

Check whether a Queue is Full

```
int IsFull()
{
        if(rear == MAX - 1)
                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;
                Queue[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", Queue[front]);
                if(front == rear)
                        front = rear = -1;
                else
                        front = front + 1;
        }
}
```

Display Queue Elements

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

Queries?

Thank You!