College of computer science & mathematics

Dep. Of Computer Science

# DATA STRUCTURE
# هياكل بيانات

## Lecture 6 : Circular queue

Prepared & Presented by

Mohammed B. Omar

2023 -2024

# Definition of Circular queue

**Circular queue is a linear data structure. It follows FIFO principle. In circular queue the last node is connected back to the first node to make a circle. Elements are added at the rear end and the elements are deleted at front end of the queue.**

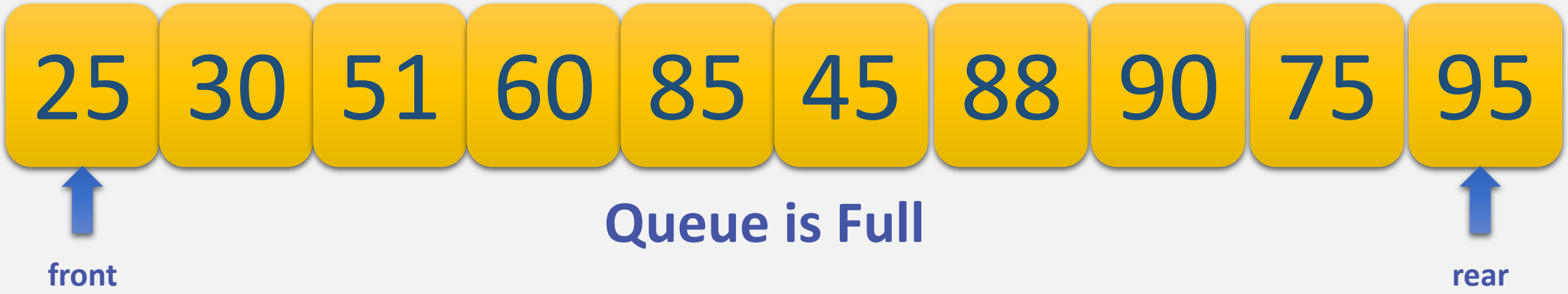
# Queue

**In a normal Queue Data Structure, we can insert elements until queue becomes full.**

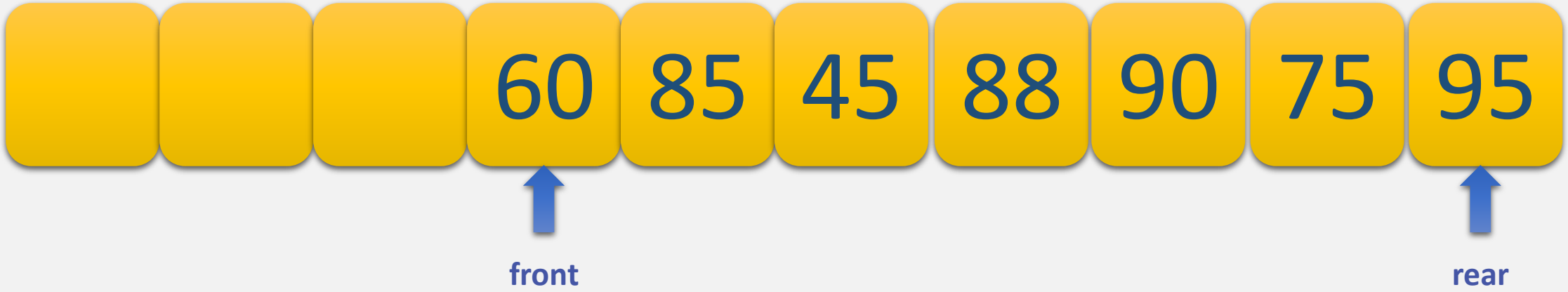
**But once if queue becomes full, we cannot insert the next element until all the elements are deleted from the queue.**

**For example consider the queue below...**

**After inserting all the elements into the queue.**



Now consider the following situation after deleting three elements from the queue...



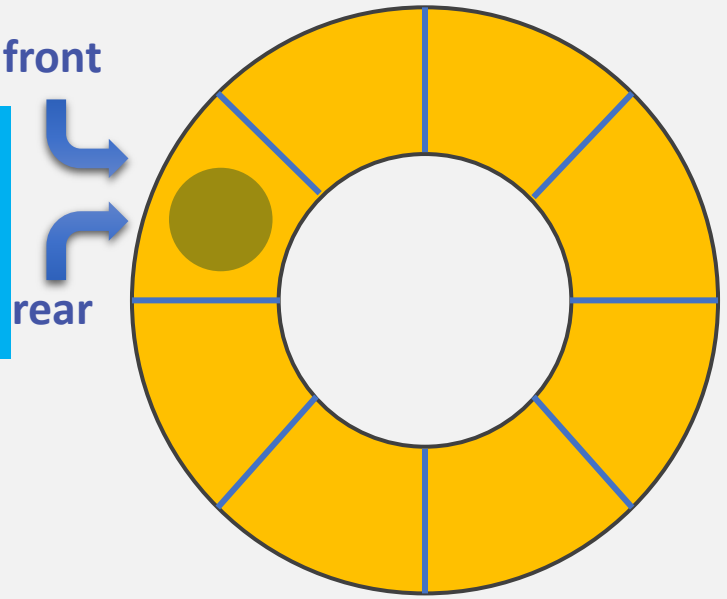

## Queue is Full (even three elements are deleted)

This situation also says that Queue is Full and we cannot insert the new element because, '**rear**' is still at last position. In above situation, even though we have empty positions in the queue we cannot make use of them to insert new element. This is the major problem in normal queue data structure. To overcome this problem we use circular queue data structure.

# What is Circular Queue?

A Circular Queue can be defined as follows... Circular Queue **is also a linear data structure, which follows the principle of FIFO(First In First Out), but instead of ending the queue at the last position, it again starts from the first position after the last, hence making the queue behave like a circular data structure.**

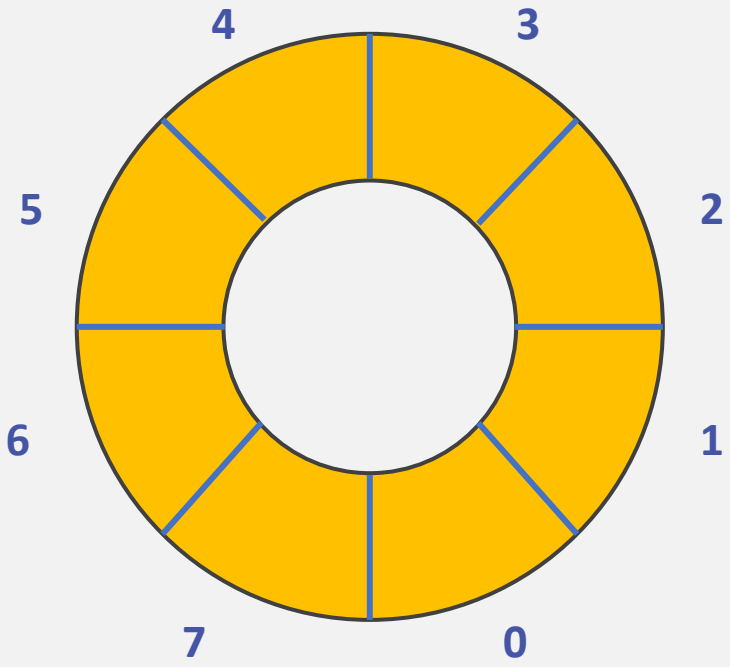
Graphical representation of a circular queue is as follows...




**When the queue is empty, REAR is initialized to (0,-1,maxsize) and FRONT is initialized to 0.**

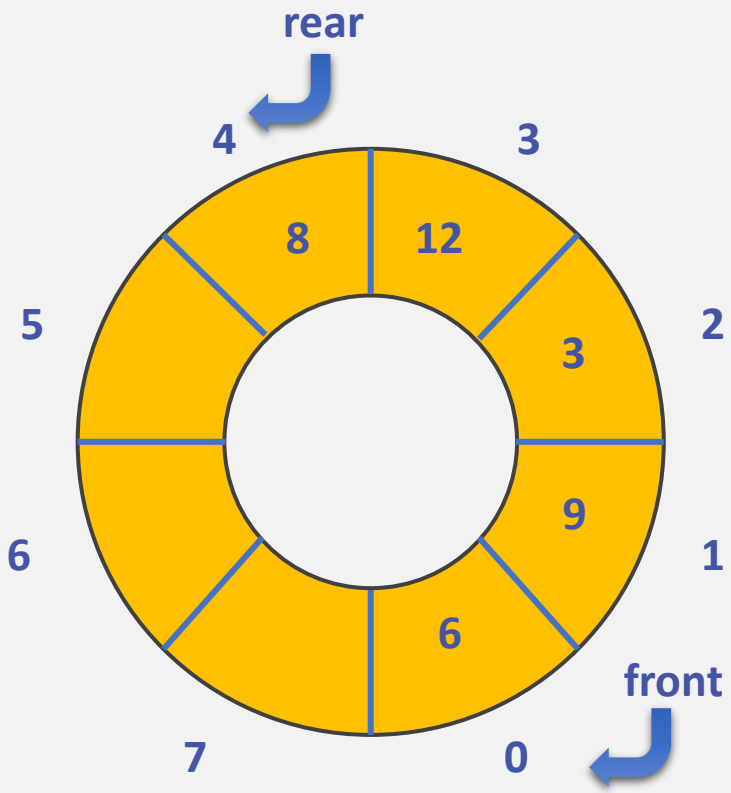
**Rear = (Rear+1)%MaxSize**

# Basic operations in Queue




Empty queue

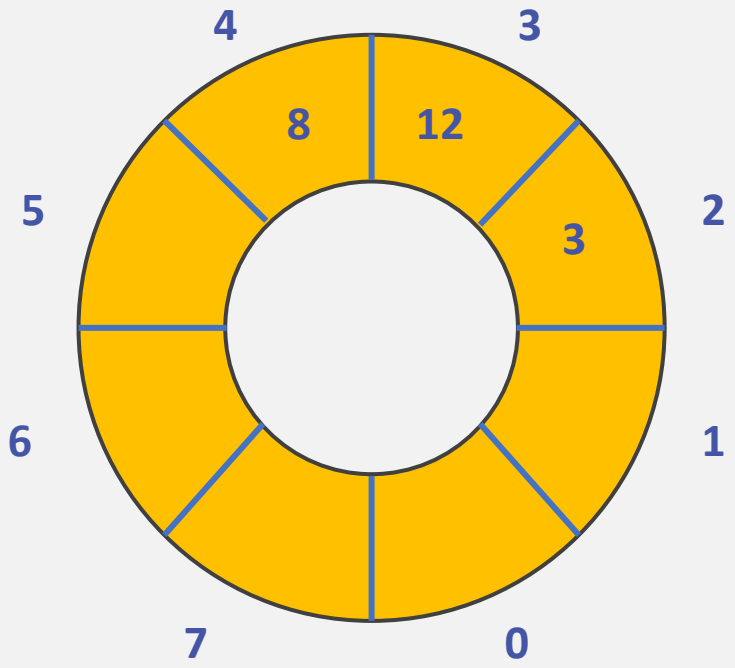
| Front | Rear |
|---|---|
| -1 | -1 |




Enqueuing five items

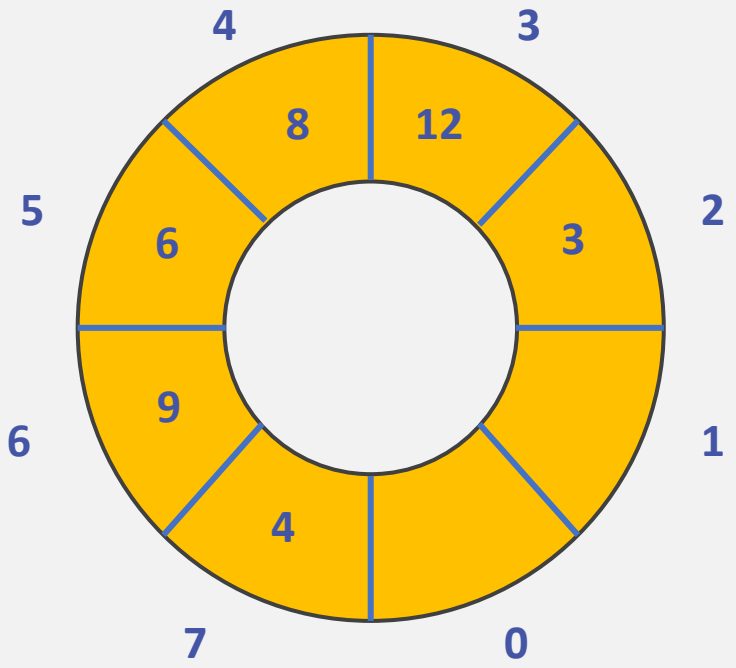
| Front | Rear |
|---|---|
| 0 | 4 |

# Basic operations in Queue




Dequeuing two items

Front 2 Rear 4




Enqueuing six items

Front 2 Rear 7

# Application of Circular Queue

Below we have some common real-world examples where circular queues are used:

1. Computer controlled Traffic Signal System uses circular queue.
2. CPU scheduling and Memory management.

# Comparison between Queue and Circular Queue:

| LINEAR QUEUE | CIRCULAR QUEUE |
|---|---|
| A linear data structure that stores data as a sequence of element similar to real world queue | A linear data structure in which the last item connects back to first item forming a circle. |
| Possible to enter new items from the rear end and remove the items from the front. | Possible to enter and remove elements from any position. |
| Requires more memory | Requires less memory |
| Less efficient | More efficient |

# Implementation of Circular Queue

To implement a circular queue data structure using array, we first perform the following steps before we implement actual operations.

- **Step 1: Include all the header files which are used in the program and define a constant 'SIZE' with specific value.**
- **Step 2: Declare all user defined functions used in circular queue implementation.**
- **Step 3: Create a one dimensional array with above defined SIZE (int cQueue[SIZE])**
- **Step 4: Define two integer variables 'front' and 'rear' and initialize both with '-1'. (int front = -1, rear = -1)**
- **Step 5: Implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on circular queue.**

# EnQueue(value) - Inserting value into the Circular Queue

**In a circular queue, enQueue() is a function which is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at rear position.**

**The enQueue() function takes one integer value as parameter and inserts that value into the circular queue. We can use the following steps to insert an element into the circular queue...**

# EnQueue Algorithm:

**Description: Here ITEM is an array with MaxQueue locations. FRONT and REAR points to the front and rear of the QUEUE. Data is the value to be inserted.**

1. If ( ( ( REAR +1 ) Mod MaxQueue ) = FRONT ) Then
2. Print: Overflow
3. Else

Begin

4. ITEM [REAR] = Data
5. Set REAR = ( REAR +1 ) Mod MaxQueue [Increment REAR by 1]
6. Print: Data inserted

End

7. Exit

# DeQueue() - Deleting a value from the Circular Queue

**In a circular queue, deQueue() is a function used to delete an element from the circular queue. In a circular queue, the element is always deleted from front position. The deQueue() function doesn't take any value as parameter. We can use the following steps to delete an element from the circular queue...**

# DeQueue Algorithm:

**Description: Here ITEM is an array with MaxQueue locations. FRONT and REAR points to the front and rear of the QUEUE. Data is the value to be inserted.**

1. If (FRONT = REAR) Then [Check for underflow]
2. Print: Underflow
3. Else

Begin

4. Data = ITEM [FRONT]
5. Set FRONT = ( FRONT +1 ) Mod MaxQueue [Increment FRONT by 1]
6. Print: Data deleted

End

7. Exit

# Queue creation:

**The Circular queue in the programming language, can be defined as a structure, struct to represent the Circular queue includes:**

**1. Data: items is an array used to store the elements of the queue.**

**2. Integer variables FRONT and REAR points to the front and rear of the queue.**

```
#define MaxQueue 100

struct CircularQueue{

int front , rear;

int items[MaxQueue];

};
```

Declared a variable of type struct CircularQueue to reserve space for the queue:

```
struct CircularQueue;
```

# Function Empty( )

```
int empty(struct CircularQueue cq )

{

if ( cq.front == cq.rear )

return 1;

else

return 0;

}
```

# Thank You
# &
# Good luck