Practical No -12

Title: Implement circular queue for pizza parlor

Aim: Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulate the system using circular queue using array.

Prerequisite:

- Basics understanding of Queue

Objectives:

- To understand implementation of Circular Queue

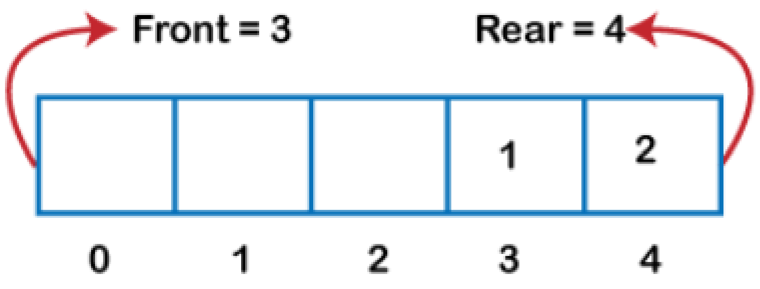
Input: Sequences of pizza order

Outcome:

- At end of this experiment, student will be able to illustrate the concept of circular queue in data structure.

Theory:

There was one limitation in the array implementation of Queue. If the rear reaches to the end position of the Queue then there might be possibility that some vacant spaces are left in the beginning which cannot be utilized. So, to overcome such limitations, the concept of the circular queue was introduced.

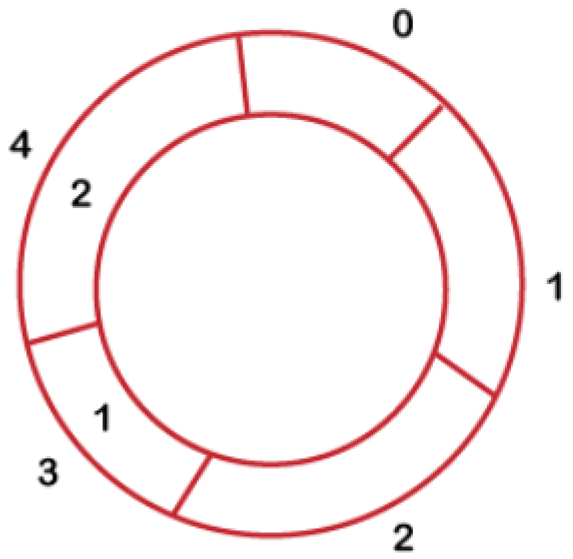


Here, indexes 0 and 1 and 2 can only be used after resetting the queue (deletion of all elements). This reduces the actual size of the queue.

Circular Queue:

A circular queue is the extended version of a regular queue where the last element is connected to the first element. Thus forming a circle-like structure.

Representation of Circular Queue:



Circular Queue works by the process of circular increment i.e. when we try to increment the pointer and we reach the end of the queue, we start from the beginning of the queue. Here, the circular increment is performed by modulo division with the queue size. That is,

if REAR + 1 == 5 (overflow!), REAR = (REAR + 1)%5 = 0 (start of queue)

Circular Queue Operations

The circular queue work as follows:

- two pointers FRONT and REAR
- FRONT track the first element of the queue
- REAR track the last elements of the queue
- initially, set value of FRONT and REAR to -1

1. Enqueue Operation

- check if the queue is full
- for the first element, set value of FRONT to 0
- circularly increase the REAR index by 1 (i.e. if the rear reaches the end, next it would be at the start of the queue)
- add the new element in the position pointed to by REAR

2. Dequeue Operation

- check if the queue is empty

- return the value pointed by FRONT
- circularly increase the FRONT index by 1
- for the last element, reset the values of FRONT and REAR to -1

In circular queue, we always **delete** (or access) data, pointed by **front** pointer and while inserting (or storing) data in queue we take help of **rear** pointer.

Conclusion:

Thus we have implemented C++ program of circular queue for pizza parlor.