+

Queue

bitCode technologies

learn.innovate.share

Queues

- A Queue is an ordered collection of items from which items may be deleted at one end called front of the queue, and into which items may be inserted at the other end called the rear of the queue.

- The first element inserted in the queue is first element deleted from the queue.

Operations on Queue

- Four primitive operations can be performed on the queue.
- Insert:

  The operation insert inserts the element at the rear of the queue.
- Delete:

  The operation delete deletes the front element from the queue.
- QEmpty:

  Returns true if the queue is empty.
- QFull:

  Returns true if the queue is full.

+ Queue

- Initially the front and rear is set to -1.
- The queue is empty whenever the rear < front.
- The number of elements in the queue at any time is equal to the value rear – front.
- In simple queues it is possible to reach the absurd situation where the queue is empty, yet no new element could be added.
- One solution is to modify the remove operation so that when an item is deleted the entire queue is shifted to the beginning of the array.

+ Queue

- The queue will no longer need a front field. Since the element at the position 0 is always the first element.
- The empty queue is represented by the condition if rear equals -1.
- This method is inefficient as every time an element is deleted all the elements need to be shifted.
- Another solution is building a queue as a linked list or as an circular queue.

bitCode technologies

+ Queue Implementation using array

```
class Queue {

  int * arr;
  int size, front, rear;

public:

  Queue( int size);
  void insert( int data );
  int deleteData ( );
  int queueEmpty ( );
  int queueFull ( );

};
```

bitCode technologies

Queue Implementation using Lists

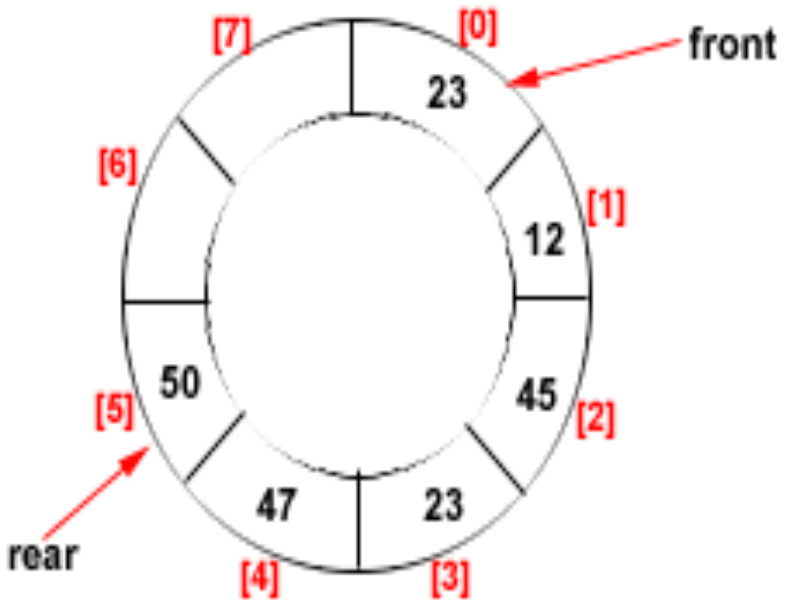
```
class Queue {

  Node * front;
  Node * rear;

public:
  Queue( );
  void insert( int data );
  int deleteData ( );
  int queueEmpty ( );
  int queueFull ( );

};
```

bitCode technologies

Circular Queue

- Static queues have a very big drawback that once the queue is FULL, even though we delete few elements from the "front" and relieve some occupied space, we are not able to add anymore elements, as the "rear" has already reached the Queue's rear most position.
- The solution lies in a queue in which the moment "rear" reaches the Queue's last position, the "first" element will become the queue's new "rear".
- As the name suggests, this Queue is not straight but circular, and so called as “Circular Queue”.

+ Struture



• Here in this queue as the rear will reach the last position, then it will be advanced to zero so that memory locations can be reused.

•The operations that can be performed on circular queue are

- Insert
- Delete
- QueueEmpty
- QueueFull
- Display

bitCode technologies

Is the queue Empty or Full

- It is difficult under this representation to determine when the queue is empty.
- Both an empty queue and a full queue would be indicated by having the head and tail point to the same element.
- There are two ways around this: either maintain a variable with the number of items in the queue, or create the array with one more element that you will actually need so that the queue is never full.

Other Types Of Queues

- **DQueue:**

  It is a linear list in which elements can be added or deleted at either ends of the queue.

- **Priority Queue:**

  It is a linear list in which elements are stored according to their priority of processing.

- **Bulk Move Queues:**

  Here make the array much bigger than the maximum number of items in the queue. When you run out of room at the end of the array, you move the entire contents of the queue back to the front of the array (this is the "bulk move" from which the structure gets its name).