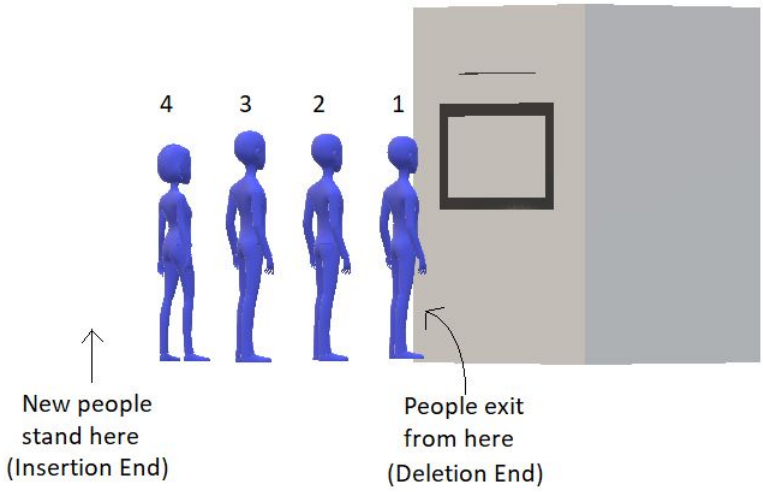
Queue Data Structure in Hindi

codewithharry.com/videos/data-structures-and-algorithms-in-hindi-38

In the last tutorial, we finished learning stacks. And today, we will start a new data structure named queue. Queue as an English word must be a well-known thing to you. We stand in a queue while waiting for our turn to come. Indian railway is one of the places where people stand in a long queue, waiting for their chance to buy a ticket. One important thing to observe, which is quite intuitive, is that your chance comes first when you come first in the queue. And the people standing last, who have joined the queue last, get to buy the ticket in the end.

Unlike stacks, where we followed LIFO(Last In First Out) discipline, here in the queue, we have FIFO(First In First Out). Follow the illustration below to get a visual understanding of a queue.



In stacks, we had to maintain just one end, *head,* where both insertion and deletion used to take place, and the other end was closed. But here, in queues, we have to maintain both the ends because we have insertion at one end and deletion from the other end.

Queue ADT

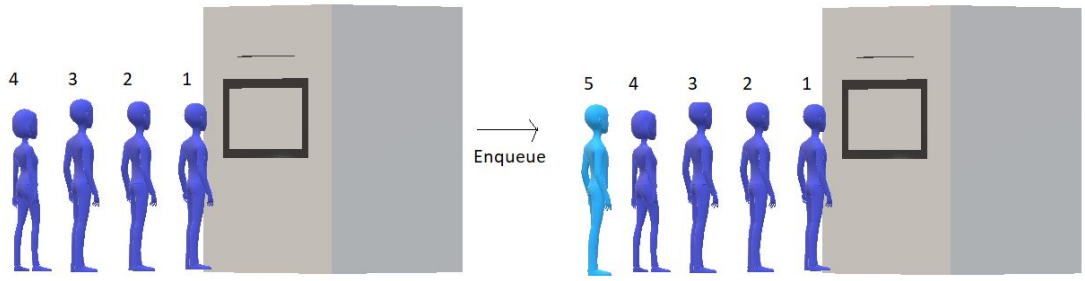
Data:

In order to create a queue, we need two pointers, one pointing to the insertion end, to gain knowledge about the address where the new element will be inserted to. And the other pointer pointing to the deletion end, which holds the address of the element which will be deleted first. Along with that, we need the storage to hold the element itself.

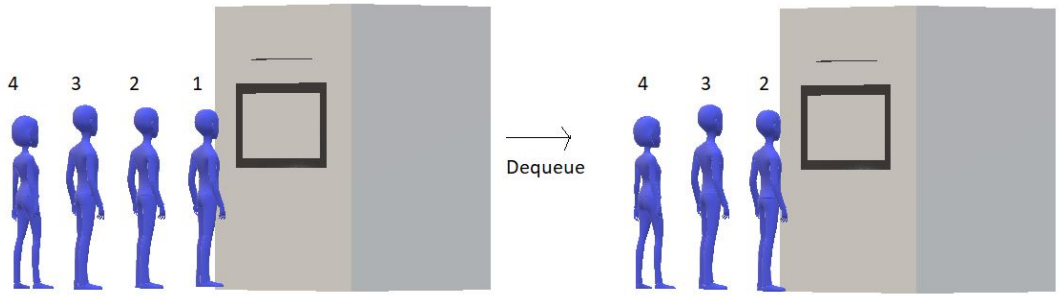
Methods:

Here are some of the basic methods we would want to have in queues:

1. enqueue() : to insert an element in a queue.



2. dequeue(): to remove an element from the queue



3. firstVal(): to return the value which is at the first position.

4. lastVal(): to return the value which is at the last position.

5. peek(position): to return the element at some specific position.

6. isempty() / isfull(): to determine whether the queue is empty or full, which helps us carry out efficient enqueue and dequeue operations.

This was our abstract data type, queue. We have in this what we thought would suffice our needs for now. The list could be longer, but in my opinion, this is sufficient.

A queue can be implemented in a number of ways. We can use both an array and a linked list and even a stack, and not just that, but by any ADT. We'll see all these methods in the coming tutorials. A queue is not limited to ticket counters or shops/malls, it has much wider applications, and you will yourself realize that while we proceed.

A queue is a collection of elements with certain operations following FIFO (First in First Out) discipline. We insert at one end and delete from the other. And this is what you have to keep in mind for now.

I appreciate your support throughout. I hope you enjoyed the tutorial. If you genuinely appreciate my work, please let your friends know about this course too. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial, where we'll implement queues using arrays. Till then, keep learning.