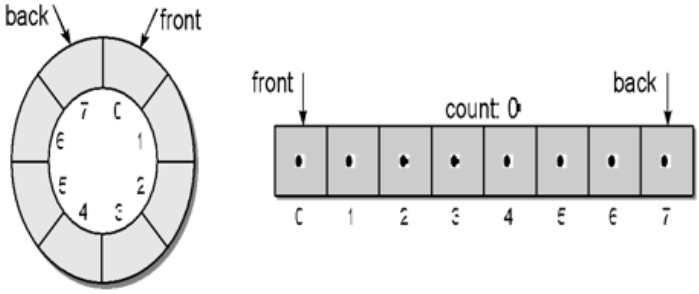
Introduction for Queue Data Structure

A queue is a **first in, first out (FIFO)** structure. This means that the first item to join the queue is the first to leave the queue. A queue can be implemented using an array or using OOP techniques.

Another type of queue is a circular queue. With this type of queue, if the end of the available spaces is reached, then the next item to be added uses any available spaces at the start of the queue. This is a more efficient use of space.

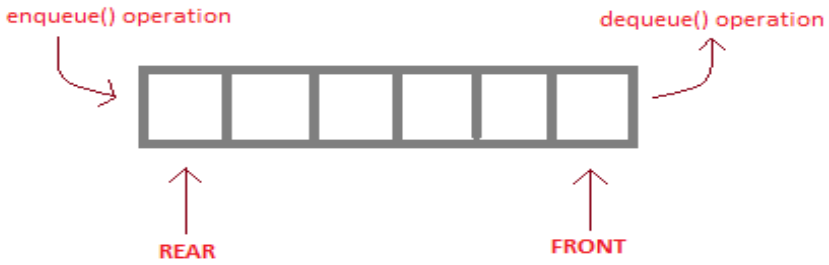


Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows **First-In-First-Out** methodology, i.e., the data item stored first will be accessed first.

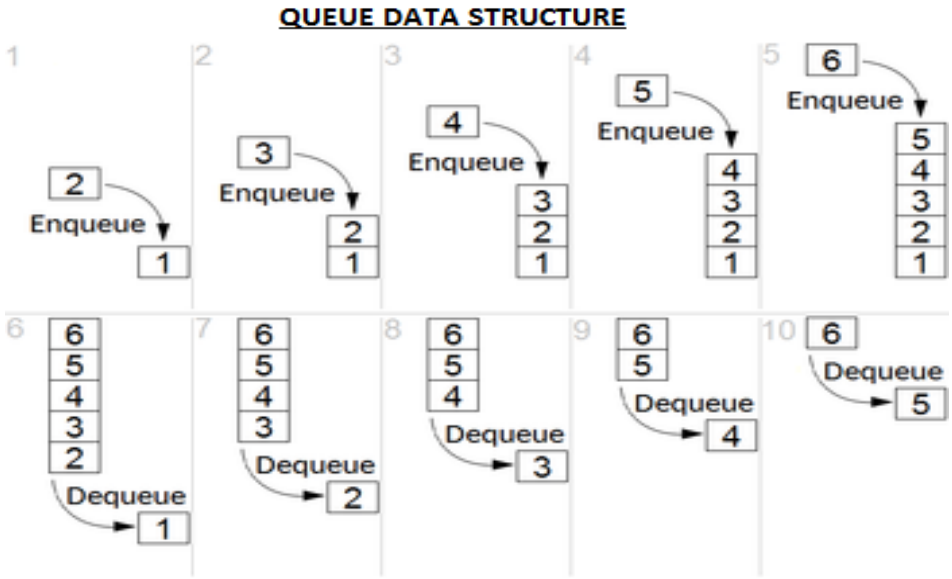
- ➔ A real-world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first. More real-world examples can be seen as queues at the ticket windows and bus-stops.
- ➔ Waiting for the doctors' clinic is a real-world example for the Queues.

The queue abstract data type is defined by the following structure and operations. A queue is structured, as described above, as an ordered collection of items which are added at one end, called the "**rear**," and removed from the other end, called the "**front**." Queues maintain a **FIFO** ordering property. The queue operations are given below.

Queue() creates a new queue that is empty. It needs no parameters and returns an empty queue.
enqueue(item) adds a new item to the rear of the queue. It needs the item and returns nothing.
dequeue() removes the front item from the queue. It needs no parameters and returns the item. The queue is modified.
isEmpty() tests to see whether the queue is empty. It needs no parameters and returns a Boolean value.
size() returns the number of items in the queue. It needs no parameters and returns an integer.



enqueue( ) is the operation for adding an element into Queue.
dequeue( ) is the operation for removing an element from Queue .



Question 1

In real life, a queue is a line of people waiting for something. In the following picture you can observe some patients are waiting for the doctor. “Maggie” is being treated by the doctor. “John”, “Kate” and “Shankar” are waiting for the doctor. Queueing policy is FIFO (First In First Out). You are required to provide programming solutions in C for the given problems according to this scenario.

a) Create a C program to implement abstract data type (ADT) "Queue" using array concept. Represent above scenario using above Queue.
b) Perform necessary operations due to following arrivals/leavings. Assume that only 03 seats are available in the queue.
   i) "John" is leaving.
   ii) "Zara" is coming.
   iii) "Jack" is coming.
   iv) "Kate" and "Shankar" are leaving one after the other.
c) How many patients are waiting for the doctor?

Question 2

a) Write pseudo code to sort a set of numbers in ascending order using Queue ADTs.

b) Convert the pseudo code to C. Display status of Queue in each computing step in the algorithm until it reaches the final solution.

c) Test following data set in your program.

40, 50, 20, 60, 10

Question 3

Write a C program to perform below functionalities. Given an array of non-negative integers. Find the largest number that can be formed from the given array elements.

For example, if the input array is {8, 1, 9}, the output should be "9 8 1", and if the input array is {8, 1, 7, 6, 0}, output should be "8 7 6 1 0"

Question 4

In the Josephus problem from antiquity, n people are in dire straits and agree to the following strategy to reduce the population. They arrange themselves in a circle (at positions numbered from 0 to n−1) and proceed around the circle, eliminating every m^{th} person until only one person is left. Legend has it that Josephus figured out where to sit to avoid being eliminated. Write a queue based C program that takes two integer command- line arguments m and n and prints the order in which people are eliminated (and thus would show Josephus where to sit in the circle).

Question 5

(K^{th} String from the End) write a Queue client k^{th} String that takes a user inputs. The Kth string from the end found on standard input, assuming that standard input has K or more strings.

Example:

K=9

Input String: it was the best of times it was the worst of times

Output: best

Submission

You should upload your answers as a single zip file to the LMS **on or before 23rd March 2018 at 11.55pm.** Make sure to rename your file as index_no.zip.