QUEUE

By
Ms. J. Joy Rose
Assistant Professor

INTRODUCTION

- Queue is an abstract data structure.
- 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 (FIFO) 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

Queue Representation

A queue can also be implemented using Arrays, Linked-lists, Pointers and Structures.

For the sake of simplicity, we shall implement queues using one-dimensional array.

Basic Operations

Queue operations may involve initializing or defining the queue, utilizing it, and then completely erasing it from the memory.

Here we shall try to understand the basic operations associated with queues −

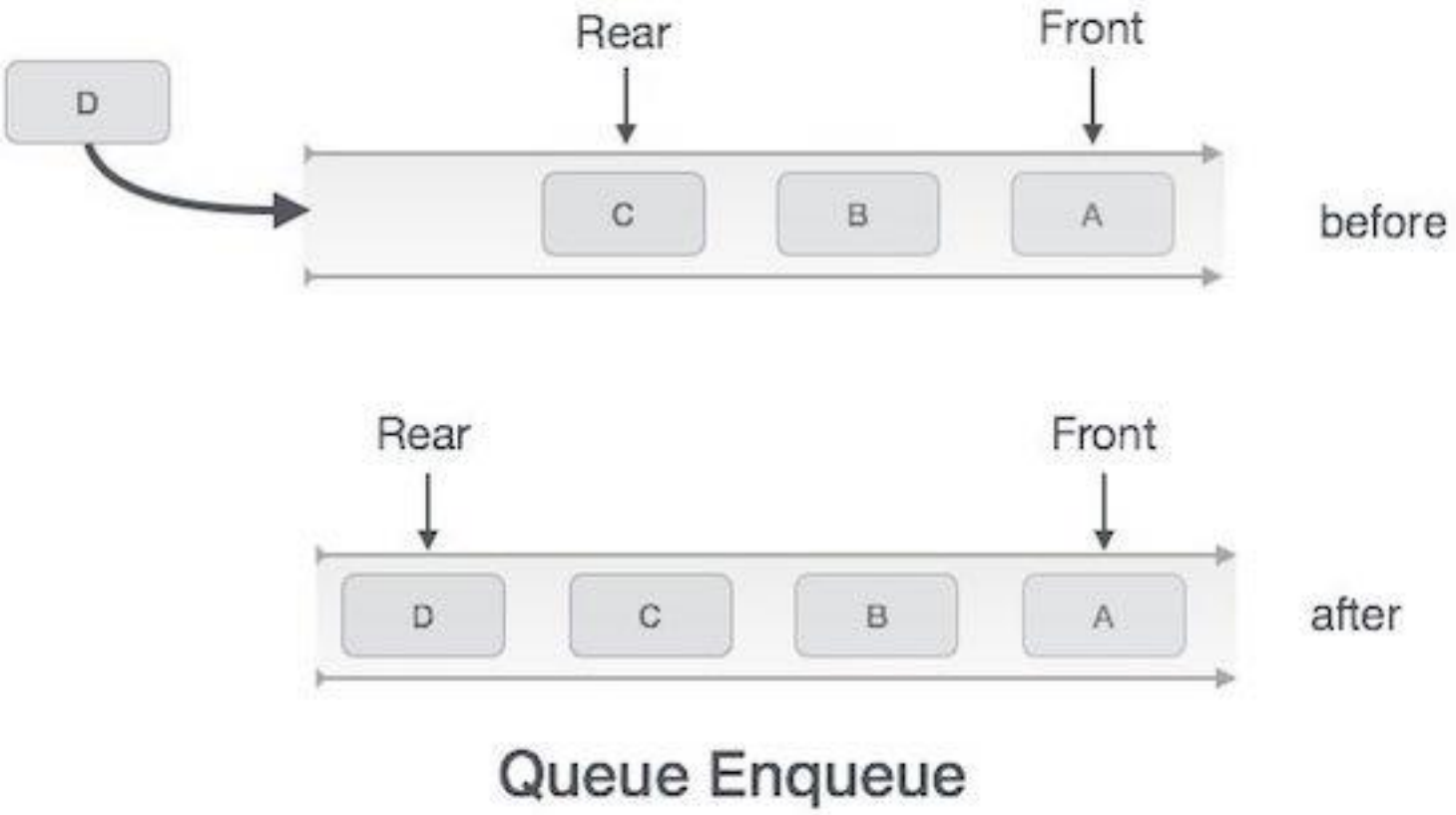
- **enqueue()** − add (store) an item to the queue.
- **dequeue()** − remove (access) an item from the queue.

Enqueue Operation:

Queues maintain two data pointers, **front** and **rear**. Therefore, its operations are comparatively difficult to implement than that of stacks.

The following steps should be taken to enqueue (insert) data into a queue −

- **Step 1** − Check if the queue is full.
- **Step 2** − If the queue is full, produce overflow error and exit.
- **Step 3** − If the queue is not full, increment **rear** pointer to point the next empty space.
- **Step 4** − Add data element to the queue location, where the rear is pointing.
- **Step 5** − return success.

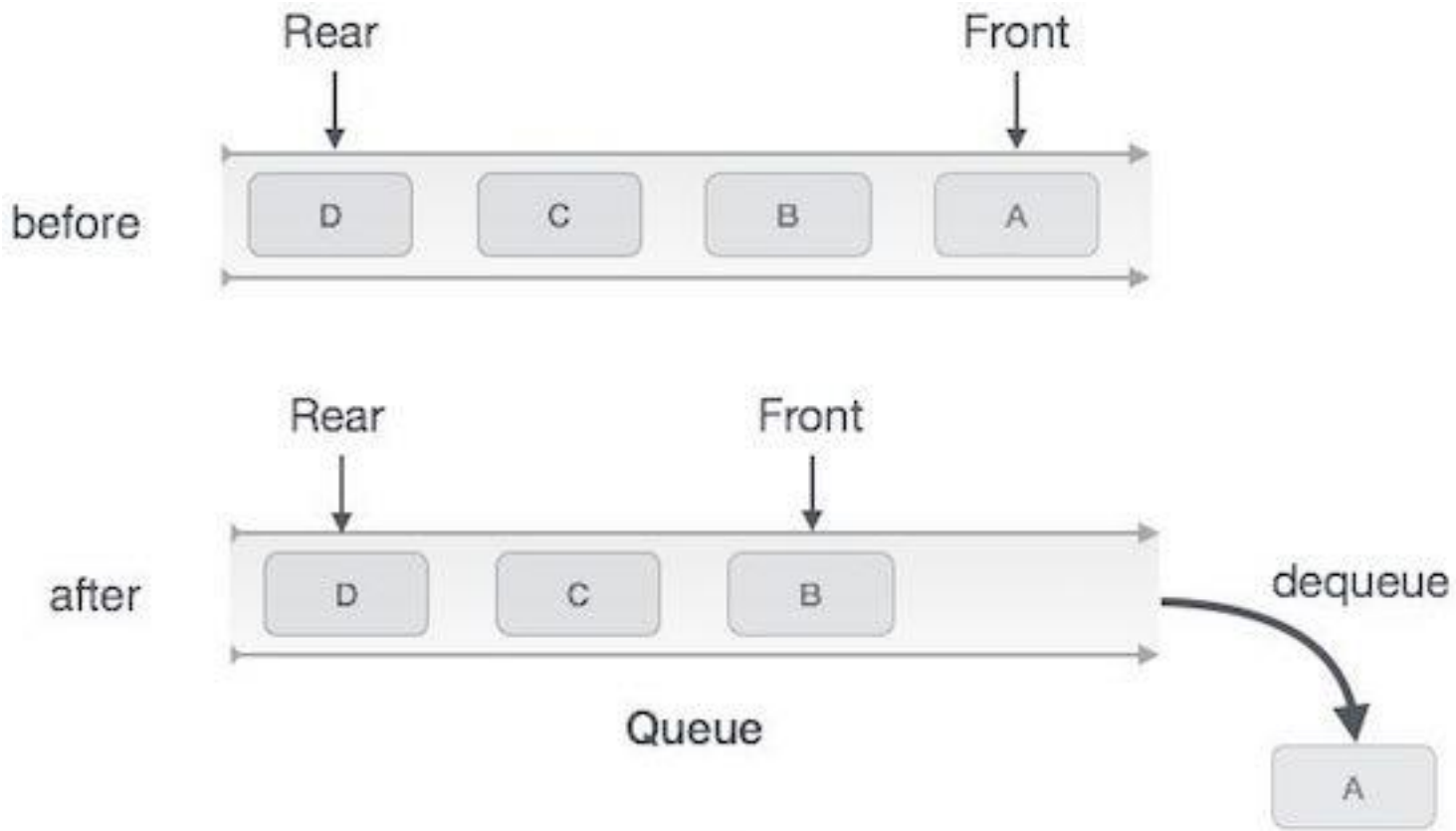


Queue Enqueue

Dequeue Operation:

Accessing data from the queue is a process of two tasks − access the data where **front** is pointing and remove the data after access. The following steps are taken to perform **dequeue** operation −

- **Step 1** − Check if the queue is empty.
- **Step 2** − If the queue is empty, produce underflow error and exit.
- **Step 3** − If the queue is not empty, access the data where **front** is pointing.
- **Step 4** − Increment **front** pointer to point to the next available data element.
- **Step 5** − Return success.



Queue Dequeue