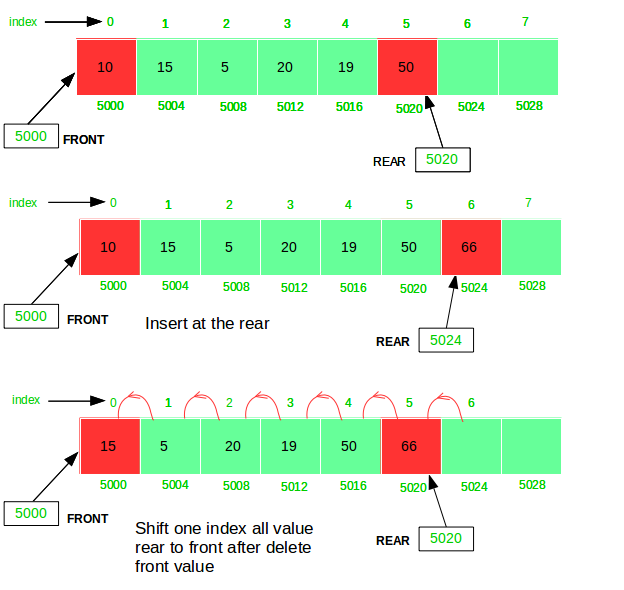
**Queue implementation using array**

To implement a [queue](https://www.geeksforgeeks.org/queue-data-structure/) using an array,

* [create an array](https://www.geeksforgeeks.org/arrays-in-c-cpp/) arr of size **n** and
* take two variables **front** and **rear** both of which will be initialized to 0 which means the queue is currently empty.
* Element
  + rear is the index up to which the elements are stored in the array and
  + front is the index of the first element of the array.

Now, some of the implementations of queue operations are as follows:

* ***Enqueue:*** *Addition of an element to the queue. Adding an element will be performed after checking whether the queue is full or not. If rear < n which indicates that the array is not full then store the element at arr[rear] and increment rear by 1 but if rear == n then it is said to be an Overflow condition as the array is full.*
* ***Dequeue:*** *Removal of an element from the queue. An element can only be deleted when there is at least an element to delete i.e. rear > 0. Now, the element at arr[front] can be deleted but all the remaining elements have to shift to the left by one position in order for the dequeue operation to delete the second element from the left on another dequeue operation.*
* ***Front:*** *Get the front element from the queue i.e. arr[front] if the queue is not empty.*
* ***Display:*** *Print all elements of the queue. If the queue is non-empty, traverse and print all the elements from the index front to rear.*

**

Below is the implementation of a queue using an array:

In the below code , we are initializing front and rear as 0, but in general we have to initialize it with -1.

If we assign rear as 0, rear will always point to next block of the end element, in fact , rear should point the index of last element,

eg. When we insert element in queue , it will add in the end i.e. after the current rear and then point the rear to the new element ,

According to the following code:

IN the first dry run, front=rear = 0;

in void queueEnqueue(int data)

else part will be executed,

so arr[rear] =data;// rear =0, rear pointing to the latest element

rear++; //now rear = 1, rear pointing to the next block after end element not the end element

//that’s against the original definition of rear

**Queue using Linked List**

*we maintain two pointers,* ***front****, and* ***rear****. The front**points to the first item of the queue and**rear**points to the last item.*

* ***enQueue():*** *This operation adds a new node after the rear**and moves the rear**to the next node.*
* ***deQueue():*** *This operation removes the front node and moves the front**to the next node.*

Follow the below steps to solve the problem:

* Create a class QNode with data members integer data and QNode\* next
  + A parameterized constructor that takes an integer x value as a parameter and sets data equal to xand next as NULL
* Create a class Queue with data members QNode frontand rear
* Enqueue Operation with parameter x:
  + Initialize QNode\* tempwith data = x
  + If the rear is set to NULL then set the front and rear to tempand return(Base Case)
  + Else set rear next totemp and then move rear to temp
* Dequeue Operation:
  + If the front is set to NULL return(Base Case)
  + Initialize QNode tempwith front and set front to its next
  + If the front is equal to NULLthen set the rear to NULL
  + Delete temp from the memory