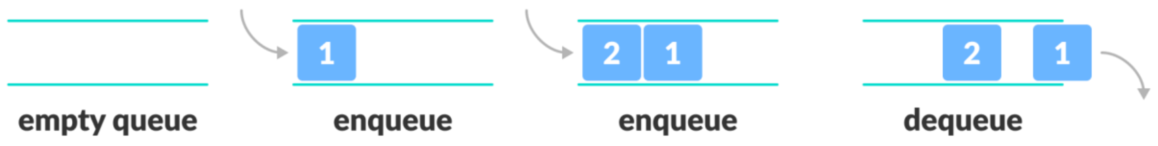
Queue

A queue is a **linear data structure** that is open at both ends and the operations are performed in **First In First Out (FIFO) order.**

FIFO means, jo phele andar jayega vo hi phele bahar aayega.



FIFO Representation of Queue

Putting items in the queue is called **enqueue**, and removing items from the queue is called **dequeue.**

Basic Operations of Queue:

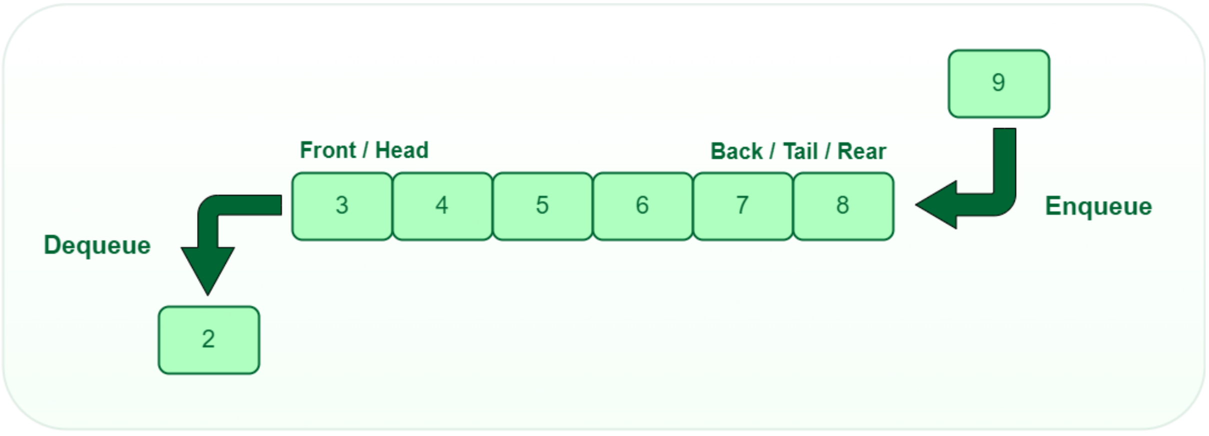
Enqueue: Add an element to the end of the queue

Dequeue: Remove an element from the front of the queue

IsEmpty: Check if the queue is empty

IsFull: Check if the queue is full

Peek: Get the value of the front of the queue without removing it



STL Implementation of Queue:

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    // using stl implementation of queue.
    queue<int> q;

    //pushing elements into queue
    q.push(1);
    q.push(14);
    q.push(18);

    // size of queue
    cout<<"Size of queue is: "<<q.size()<<endl; // output: 3

    // popping elements from queue
    q.pop(); // 1 is popped because queue is a FIFO data structire.

    cout<<"Size of queue is: "<<q.size()<<endl; // outpus: 2

    // check if queue is empty or not
    if(q.empty()){
        cout<<"Queue is empty\n";
    }
    else{
        cout<<"Queue is not empty\n";
    }

    // printing queue front ekement
    cout<<q.front()<<endl; //output : 14, because 1 is popped.
}
```

Queue Implementation:

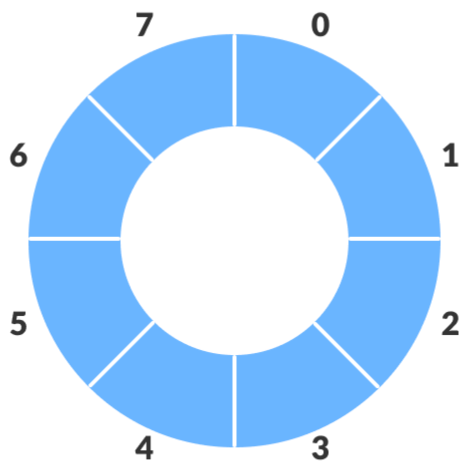
You can implement queue using,

1. Array/vector.
2. Linked List.

Circular Queue:

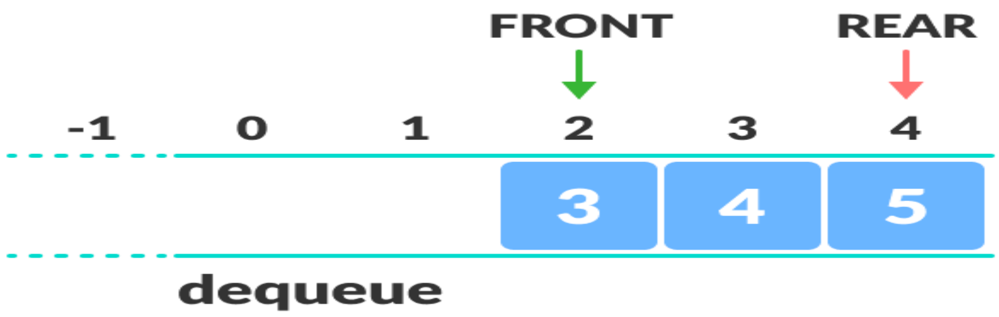
A circular queue is the extended version of a regular queue where the last element is connected to the first element.

Thus forming a circle-like structure.



Circular queue representation

The circular queue solves the major limitation of the normal queue. In a normal queue, after a bit of insertion and deletion, there will be non-usable empty space.



Limitation of the regular Queue

Here, indexes 0 and 1 can only be used after resetting the queue (deletion of all elements). This reduces the actual size of the queue.

How Circular Queue Works

Circular Queue works by the process of circular increment i.e. when we try to increment the pointer and we reach the end of the queue, we start from the beginning of the queue.

More types of Queues are: -

1. **Input restricted Queue: -** means you can insert element into queue only from one side i.e., rear and you can pop elements of queue from both sides.

   Operations:- push_back(), pop_front() and pop_back().

2. **Output restricted Queue: -** means you can insert element into queue from both side and you can pop elements of queue from one side i.e., from front.

   Operations:- push_front(), push_back() and pop_front().

3. **Doubly Ended Queue: -** means you can perform push and pop operation from both sides i.e., from front side and rear side.

   Operations:- push_front(), push_back(), pop_front() and pop_back().