**Basic Operations of Queue**

A queue is an object (an abstract data structure - ADT) that allows the following operations:

1. **Enqueue**: Add an element to the end of the queue
2. **Dequeue**: Remove an element from the front of the queue
3. **IsEmpty**: Check if the queue is empty
4. **IsFull**: Check if the queue is full
5. **Peek**: Get the value of the front of the queue without removing it

## Working of Queue

Queue operations work as follows:

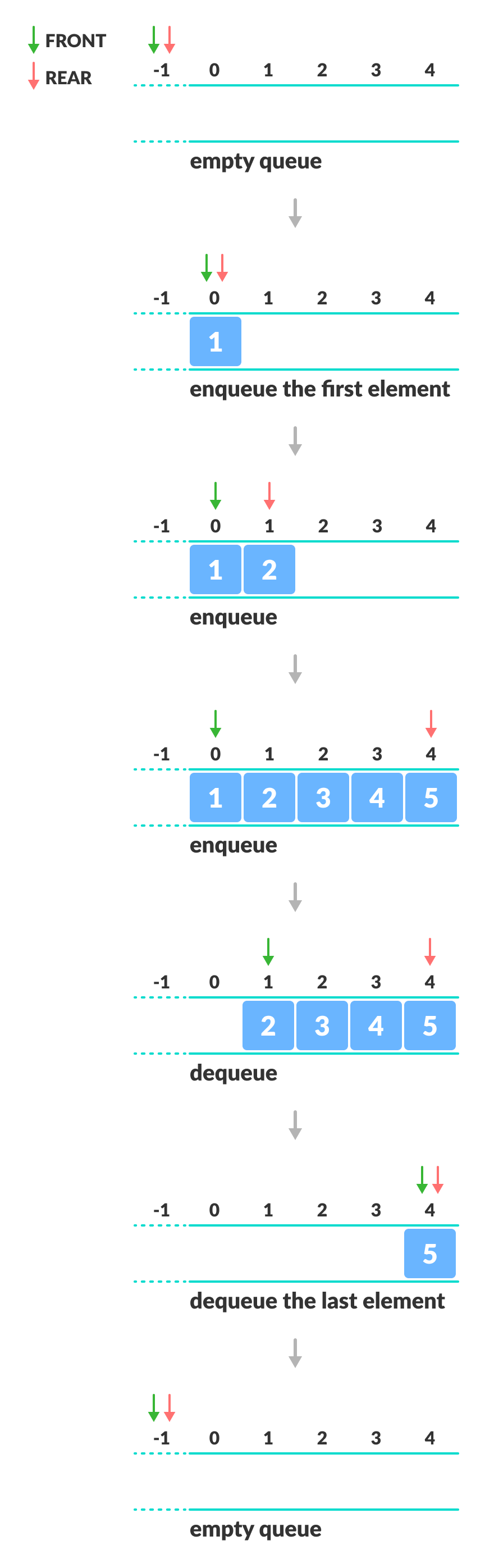
1. two pointers FRONT and REAR
2. FRONT track the first element of the queue
3. REAR track the last element of the queue
4. initially, set value of FRONT and REAR to -1

### Enqueue Operation

1. check if the queue is full
2. for the first element, set the value of FRONT to 0
3. increase the REAR index by 1
4. add the new element in the position pointed to by REAR

### Dequeue Operation

1. check if the queue is empty
2. return the value pointed by FRONT
3. increase the FRONT index by 1
4. element, reset the values of FRONT and REAR to -1

* for the la

**C program to implement EnQueue and DeQueue operation**

#include <stdio.h>

#define SIZE 5

void enQueue(int);

void deQueue();

void display();

int items[SIZE], front = -1, rear = -1;

void enQueue(int value) {

if (rear == SIZE - 1)

printf("\nQueue is Full!!");

else {

if (front == -1)

front = 0;

rear++;

items[rear] = value;

printf("\nInserted -> %d", value);

}

}

void deQueue() {

if (front == -1)

printf("\nQueue is Empty!!");

else {

printf("\nDeleted : %d", items[front]);

front++;

if (front > rear)

front = rear = -1;

}

}

// Function to print the queue

void display() {

if (rear == -1)

printf("\nQueue is Empty!!!");

else {

int i;

printf("\nQueue elements are:\n");

for (i = front; i <= rear; i++)

printf("%d ", items[i]);

}

printf("\n");

}

int main() {

//deQueue is not possible on empty queue

deQueue();

//enQueue 5 elements

enQueue(1);

enQueue(5);

enQueue(10);

enQueue(4);

enQueue(5);

// 6th element can't be added to because the queue is full

//enQueue(6);

display();

//deQueue removes element entered first i.e. 1

deQueue();

deQueue();

deQueue();

//Now we have just 4 elements

display();

return 0;

}