**USN:- 1NT21IS039**

## Aim:- ORDINARY QUEUE IMPLEMENTATION USING ARRAY

[GitHub-Link](https://github.com/logan-14/Queue-Using-LinkedList)

**ALGORITHIM:-**

**Insert operation**

**Step 1**: Allocate the space for the new node PTR

**Step 2**: SET PTR -> DATA = VAL

**Step 3**: IF FRONT = NULL

SET FRONT = REAR = PTR

SET FRONT -> NEXT = REAR -> NEXT = NULL

ELSE

SET REAR -> NEXT = PTR

SET REAR = PTR

SET REAR -> NEXT = NULL

[END OF IF]

**Step 4**: END

**Deletion**

**Step 1**: IF FRONT = NULL

Write " Underflow "

Go to Step 5

[END OF IF]

**Step 2**: SET PTR = FRONT

**Step 3**: SET FRONT = FRONT -> NEXT

**Step 4**: FREE PTR

**Step 5**: END

**Code:-**

// A C program to demonstrate linked list based

// implementation of queue

#include <stdio.h>

#include <stdlib.h>

// A linked list (LL) node to store a queue entry

struct QNode

{

    int key;

    struct QNode \*next;

};

// The queue, front stores the front node of LL and rear

// stores the last node of LL

struct Queue

{

    struct QNode \*front, \*rear;

};

// A utility function to create a new linked list node.

struct QNode \*newNode(int k)

{

    struct QNode \*temp = (struct QNode \*)malloc(sizeof(struct QNode));

    temp->key = k;

    temp->next = NULL;

    return temp;

}

// A utility function to create an empty queue

struct Queue \*createQueue()

{

    struct Queue \*q = (struct Queue \*)malloc(sizeof(struct Queue));

    q->front = q->rear = NULL;

    return q;

}

// The function to add a key k to q

void enQueue(struct Queue \*q, int k)

{

    // Create a new LL node

    struct QNode \*temp = newNode(k);

    // If queue is empty, then new node is front and rear

    // both

    if (q->rear == NULL)

    {

        q->front = q->rear = temp;

        return;

    }

    // Add the new node at the end of queue and change rear

    q->rear->next = temp;

    q->rear = temp;

}

// Function to remove a key from given queue q

void deQueue(struct Queue \*q)

{

    // If queue is empty, return NULL.

    if (q->front == NULL)

        return;

    // Store previous front and move front one node ahead

    struct QNode \*temp = q->front;

    q->front = q->front->next;

    // If front becomes NULL, then change rear also as NULL

    if (q->front == NULL)

        q->rear = NULL;

    free(temp);

}

// Driver code

int main()

{

    struct Queue \*q = createQueue();

    enQueue(q, 10);

    enQueue(q, 20);

    deQueue(q);

    deQueue(q);

    enQueue(q, 30);

    enQueue(q, 40);

    enQueue(q, 50);

    deQueue(q);

    printf("Queue Front : %d \n", q->front->key);

    printf("Queue Rear : %d", q->rear->key);

    return 0;

}

**Output:-**

Queue Front : 40

Queue Rear : 50