Circular Queue | Set 2 (Circular Linked List Implementation)

Prerequisite – Circular Singly Linked List

We have discussed basics and how to implement circular queue using array in set 1.
Circular Queue | Set 1 (Introduction and Array Implementation)

In this post another method of circular queue implementation is discussed, using Circular Singly Linked List.

Operations on Circular Queue:

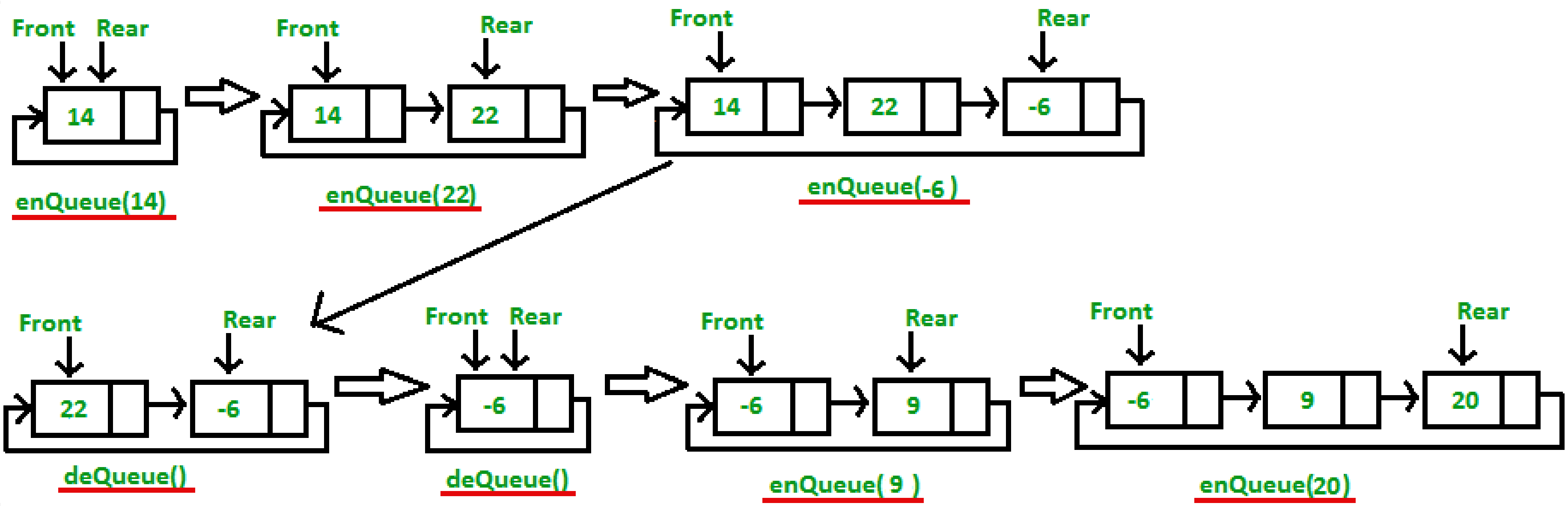
- **Front:**Get the front item from queue.
- **Rear:** Get the last item from queue.
- **enQueue(value)** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at Rear position.

  Steps:

  1. Create a new node dynamically and insert value into it.
  2. Check if front==NULL, if it is true then front = rear = (newly created node)
  3. If it is false then rare=(newly created node) and rear node always contains the address of the front node.
- **deQueue()** This function is used to delete an element from the circular queue. In a queue, the element is always deleted from front position.

  Steps:

  1. Check whether queue is empty or not means front == NULL.
  2. If it is empty then display Queue is empty. If queue is not empty then step 3



Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

```
// C or C++ program for insertion and
// deletion in Circular Queue
#include <bits/stdc++.h>
using namespace std;

// Structure of a Node
struct Node
{
    int data;
    struct Node* link;
};

struct Queue
{
    struct Node *front, *rear;
};

// Function to create Circular queue
void enQueue(Queue *q, int value)
{
    struct Node *temp = new Node;
```

```
    else
        q->rear->link = temp;

    q->rear = temp;
    q->rear->link = q->front;
}

// Function to delete element from Circular Queue
int deQueue(Queue *q)
{
    if (q->front == NULL)
    {
        printf ("Queue is empty");
        return INT_MIN;
    }

    // If this is the last node to be deleted
    int value; // Value to be dequeued
    if (q->front == q->rear)
    {
        value = q->front->data;
        free(q->front);
        q->front = NULL;
        q->rear = NULL;
    }
    else  // There are more than one nodes
    {
        struct Node *temp = q->front;
        value = temp->data;
        q->front = q->front->link;
        q->rear->link= q->front;
        free(temp);
    }

    return value ;
}

// Function displaying the elements of Circular Queue
void displayQueue(struct Queue *q)
{
    struct Node *temp = q->front;
    printf("\nElements in Circular Queue are: ");
    while (temp->link != q->front)
    {
        printf("%d ", temp->data);
        temp = temp->link;
    }
    printf("%d", temp->data);
}
```

```
{
    // Create a queue and initialize front and rear
    Queue *q = new Queue;
    q->front = q->rear = NULL;

    // Inserting elements in Circular Queue
    enQueue(q, 14);
    enQueue(q, 22);
    enQueue(q, 6);

    // Display elements present in Circular Queue
    displayQueue(q);

    // Deleting elements from Circular Queue
    printf("\nDeleted value = %d", deQueue(q));
    printf("\nDeleted value = %d", deQueue(q));

    // Remaining elements in Circular Queue
    displayQueue(q);

    enQueue(q, 9);
    enQueue(q, 20);
    displayQueue(q);

    return 0;
}
```

Run on IDE

Output:

```
Elements in Circular Queue are: 14 22 6
Deleted value = 14
Deleted value = 22
Elements in Circular Queue are: 6
Elements in Circular Queue are: 6 9 20
```

Time Complexity: Time complexity of enQueue(), deQueue() operation is O(1) as there is no loop in any of the operation.

Note : In case of linked list implementation, a queue can be easily implemented without being circular. However in case of array implementation, we need a circular queue to save space.

page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner   Company Wise Coding Practice

Linked Lists Queue

Recommended Posts:

Top 20 Linked List Interview Question
Write a program function to detect loop in a linked list
Given a linked list of line segments, remove middle points
Minimum time required to rot all oranges

(Login to Rate and Mark)

1 Average Difficulty : **1/5.0**
Based on **1** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

 Contact Us! About Us! Advertise with us! Privacy Policy