Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

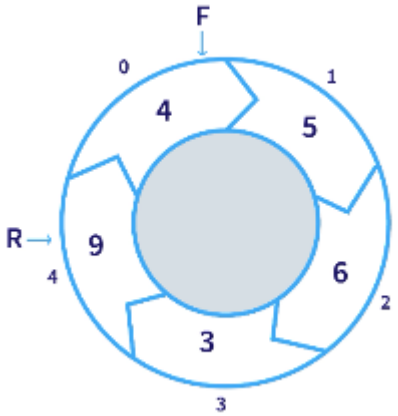
Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 05

AIM: Implementation of circular Queue.

Theory:

A circular queue is a linear queue (all the operations performed in FIFO pattern) in which the end position of the queue is connected to the start position to form a circular ring, thus also known as a Ring Buffer.



Now since we have a circular path the start index and the end index of a circular queue will be dynamic. The front pointer Front will mark the start of the queue instead of the 0th index and the rear pointer Rear will mark its end instead of the n-1th index (n = size of queue). As we do not have definite indices to mark the circular queue's start and end, the conditions for queue underflow(deletion of an element in an empty queue) and queue overflow(insertion of an element when no more element can be inserted as the rear pointer reached queue's end or the queue is full) will also change and is explained in Enqueue operation and Dequeue operation sections.

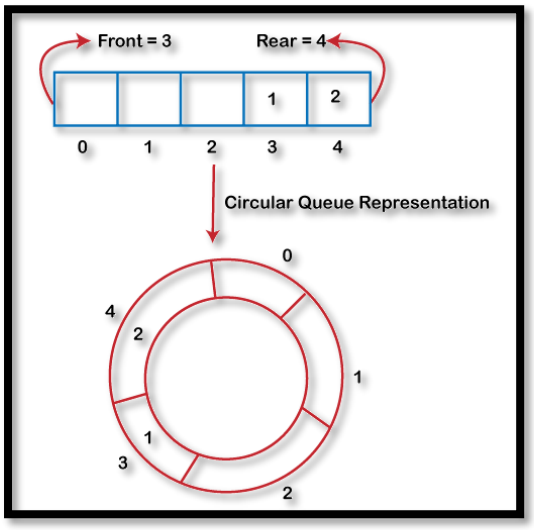
Algorithm:

```
Enqueue:
1.  IF (REAR+1)%MAX = FRONT
2.  Write " OVERFLOW "
3.  Goto step 4
4.  [End OF IF]
5.  IF FRONT = -1 and REAR = -1
6.  SET FRONT = REAR = 0
7.  ELSE IF REAR = MAX - 1 and FRONT ! = 0
8.  SET REAR = 0
9.  ELSE
10. SET REAR = (REAR + 1) % MAX
11. [END OF IF]
12. SET QUEUE[REAR] = VAL
13. EXIT

Dequeue:
1.  IF FRONT = -1
2.  Write " UNDERFLOW "
3.  Goto Step 4
```

```
4.  [END of IF]
5.  SET VAL = QUEUE[FRONT]
6.  IF FRONT = REAR
7.  SET FRONT = REAR = -1
8.  ELSE
9.  IF FRONT = MAX -1
10. SET FRONT = 0
11. ELSE
12. SET FRONT = FRONT + 1
13. [END of IF]
14. [END OF IF]
15. EXIT
```

Example:



Program:

```
#include <stdio.h>
#include <conio.h>
#define max 3
void insert(int a[max], int n, int *frontq, int *rearq);
void del(int a[max], int *frontq, int *rearq);
void print(int a[max], int frontq, int rearq);
void main()
{
    printf("Prerna Sunil Jadhav - 60004220127\n");
    int front = -1, rear = -1, a[max];
    int n, i, ch;
    printf("Enter %d number to be inserted: ", max);
    for (i = 0; i <= max - 1; i++)
    {
```

```
        scanf("%d", &n);
        insert(a, n, &front, &rear);
    }
    printf("\nElements in queue are: ");
    print(a, front, rear);
    A:printf("\nEnter The operation you want to perform \n1) Insertion \n2) Deletion \n3) Exit \n");
    scanf("%d", &ch);
    switch (ch)
    {
        case 1:
        {
            printf("\nEnter an element:");
            for (i = 0; i < 1; i++)
            {
                scanf("%d", &n);
                insert(a, n, &front, &rear);
            }
            printf("\nAfter inserting the queue is :");
            print(a, front, rear);
            goto A;
        }

        case 2:
        {
            del(a, &front, &rear);
            printf("\nAfter deletion queue is:");
            for (i = front; i <= rear; i++)
            {
                printf("\n %d", a[i]);
            }
            goto A;
        }
    }
    getch();
}
void insert(int a[max], int n, int *frontq, int *rearq)
{
    if (*rearq == max - 1 && frontq > 0)
    {
        if (*rearq + 1 == *frontq)
        {
            printf("\nQueue is full");
            return;
        }
        else
```

```
        {
            *rearq = 0;
            a[*rearq] = n;
        }
    }
    else
    {
        (*rearq)++;
        a[*rearq] = n;
    }
    if (*frontq == -1)
        (*frontq)++;
}
void del(int a[max], int *frontq, int *rearq)
{
    int i;
    if (*frontq == -1)
    {
        printf("\nQueue is empty");
        return;
    }
    i = a[*frontq];
    printf("\nElement deleted is : %d", i);
    (*frontq)++;
    if (*frontq == *rearq)
    {
        *frontq = -1;
        *rearq = -1;
    }
}
void print(int a[max], int frontq, int rearq)
{
    int i;
    if (frontq == -1)
    {
        printf("\nQueue is empty: ");
        return;
    }
    else if (rearq > frontq)
    {
        for (i = frontq; i <= rearq; i++)
        {
            printf("\n %d", a[i]);
        }
    }
    else
```

```
    {
        for (i = frontq; i < max; i++)
        {
            printf("\n %d", a[i]);
        }
        for (i = rearq; i < frontq; i++)
        {
            printf("\n %d", a[i]);
        }
    }
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter 3 number to be inserted: 89 670 543

Elements in queue are:
 89
 670
 543
Enter The operation you want to perform
1) Insertion
2) Deletion
3) Exit
1

Enter an element:342

After inserting the queue is :
 342
 670
 543
Enter The operation you want to perform
1) Insertion
2) Deletion
3) Exit
2

Element deleted is : 342
```

Conclusion:

- Circular queue is a queue with a ring-like structure where the queue end is connected to the queue start.
- The queue operations in circular queues are similar to basic queues with an advantage to move the queue pointers from queue end to queue start once the queue end is reached and the queue is not completely full.
- It can be implemented using an array or a circular linked list. The time complexity for each queue operation takes O(1).