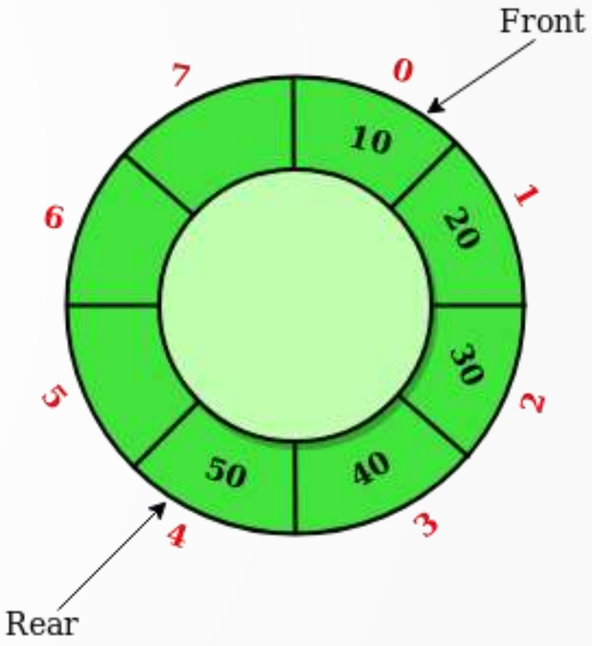
Data Structures : Circular Queue

B.Sc. 3rd Semester, Paper C5

Paulami Basu Ray
Assistant Professor
Department of Computer Science & Applications
Prabhat Kumar College, Contai

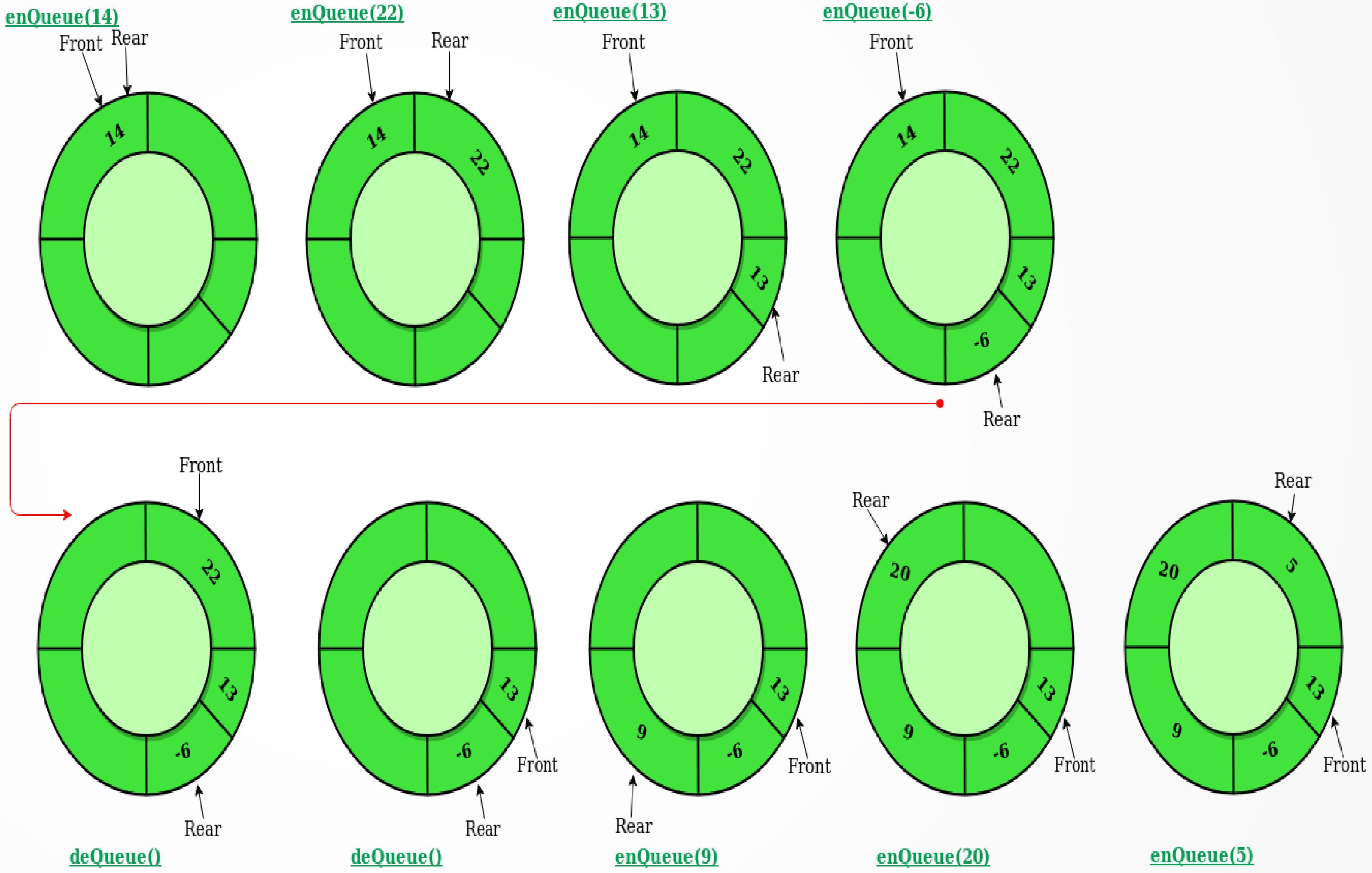
Circular Queue: Definition

- Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **‘Ring Buffer’**.



Queue Operations

- Enqueue
- Dequeue



C Program for Queue implementation

```
/*
  C Program to Implement a Queue using an Array
*/
#include <stdio.h>

#define MAX 15

void enqueue();
void dequeue();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
int size=0;
void main()
{
int choice;
while (1)
{
                printf("1.Insert element to queue \n");
                printf("2.Delete element from queue \n");
                printf("3.Display all elements of queue \n");
                printf("4.Quit \n");
                printf(“Enter your choice:”);
                scanf(“%d”,&choice);
switch (choice)
{
    case 1:
        enqueue();
        break;
    case 2:
        dequeue();
        break;
    case 3:
        display();
        break;
    case 4:
        return;
    default:
        printf("Wrong choice \n");
    } /* End of switch */
    } /* End of while */
} /* End of main() */
```

```
void enqueue()
{
        int add_item;
        if (size == MAX)
                printf("Queue Overflow \n");
        else
        {
                if (front == - 1)
                /*If queue is initially empty */
                front = 0;
                printf("Insert the element in queue : ");
                scanf("%d", &add_item);
                rear =(rear + 1)%MAX;
                queue_array[rear] = add_item;
                size++;
        }
} /* End of insert() */
```

```
void dequeue()
{
        if (size = = 0)
        {
                printf("Queue Underflow \n");
                return ;
        }
        else
        {
                printf("Element deleted from queue is : %d\n",
                queue_array[front]);
                front = (front + 1)%MAX;
                size--;
        }
} /* End of dequeue() */
```

```
void display()
{
    int i;
    if ( front > rear)
    {
        for(i=front; i<=MAX; i++)
            printf(“%d”,queue_array[i]);
        for(i=0; i<=rear; i++)
            printf(“%d”,queue_array[i]);
    }
    else
    {
        for(i=front; i<size; i++)
            printf(“%d”,queue_array[i]);
    }

} /* End of display() */
```