DATA STRUCTURES
USING "C"

(Queue)

For

BCA Part-II (Session 2018-21) Students

BY

ANANT KUMAR
MCA, M. Phil, M. Tech.
Faculty Member
Department of Computer Science
J. D. Women's College, Patna

Queue

A queue is another special kind of list, where items are inserted at one end called the rear and deleted at the other end called the front. Another name for a queue is a "FIFO" or "First-in-first-out"list.

The operations for a queue are analogues to those for a stack, the difference is that the insertions go at the end of the list, rather than the beginning. Following operations on queues:

- *enqueue*: which inserts an element at the end of the queue.
- *dequeue*: which deletes an element at the start of the queue.

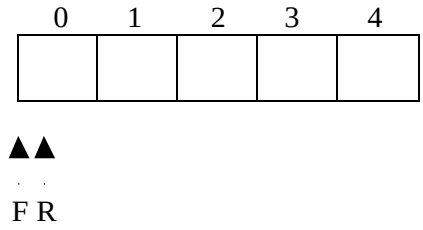
Representation of Queue

Queue is represented using:

- An Array
- Linked list

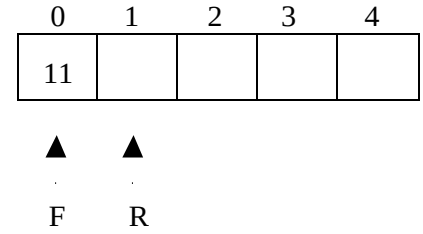
An Array **Representation**

Let us consider a queue, which can hold maximum of five elements. Initially the queue is empty.



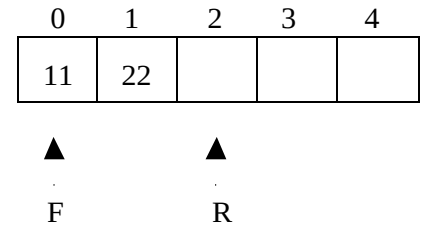
Queue Empty
F RO NT = RE A R = 0

Now, insert 11 to the queue. Then queue status will be:



RE A R = RE A R + 1 =
1 FRO NT = 0

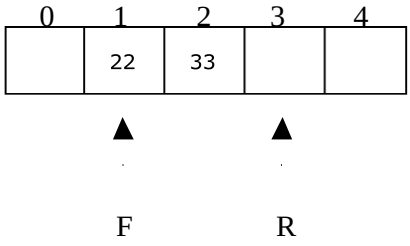
Next, insert 22 to the queue. Then the queue status is:



RE A R = RE A R + 1 =
2 FRO NT = 0

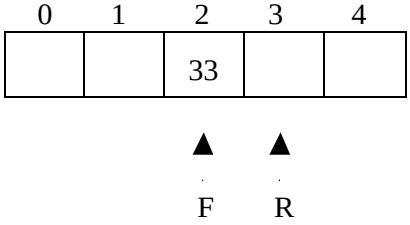
Again insert another element 33 to the queue. The status of the queue is:

Now, delete an element. The element deleted is the element at the front of the queue. So the status of the queue is:



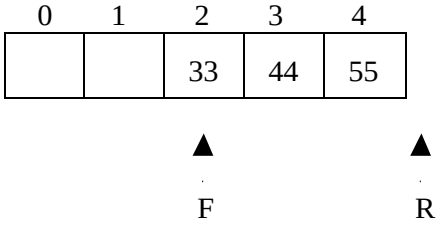
RE A R = 3
F RO NT = F R O NT + 1 =1

Again, delete an element. The element to be deleted is always pointed to by the FRONT pointer. So, 22 is deleted. The queue status is as follows:



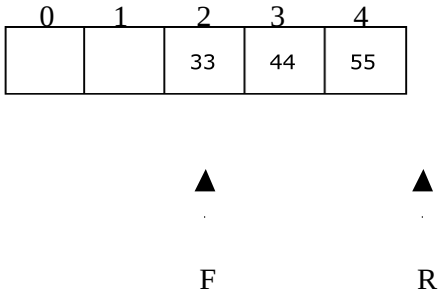
RE A R = 3
F RO NT = F R O NT + 1 =2

Now, insert new elements 44 and 55 into the queue. The queue status is:



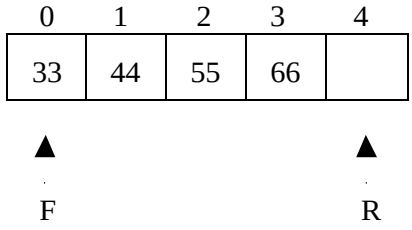
RE A R = 5 FRO
NT = 2

Next insert another element, say 66 to the queue. We cannot insert 66 to the queue as the rear crossed the maximum size of the queue (i.e., 5). There will be queue full signal. The queue status is as follows:



RE A R = 5
FRO NT = 2

Now it is not possible to insert an element 66 even though there are two vacant positions in the linear queue. To over come this problem the elements of the queue are to be shifted towards the beginning of the queue so that it creates vacant position at the rear end. Then the FRONT and REAR are to be adjusted properly. The element 66 can be inserted at the rear end. After this operation, the queue status is asfollows:



RE A R = 4
FRO NT = 0

This difficulty can overcome if we treat queue position with index 0 as a position that comes after position with index 4 i.e., we treat the queue as a **circular queue.**

Types of Queues in Data Structure

- Simple Queue
- Circular Queue
- Priority Queue
- Doubly Ended Queue (Dequeue)

Applications of Queue

1. It is used to schedule the jobs to be processed by the CPU.
2. When multiple users send print jobs to a printer, each printing job is kept in the printing queue. Then the printer prints those jobs according to first in first out (FIFO) basis.
3. Breadth first search uses a queue data structure to find an element from a graph.

Queue operations using array

1. insertQ(): inserts an element at the end of queueQ.
2. deleteQ(): deletes the first element of Q.
3. displayQ(): displays the elements in the queue.

```
# include <conio.h>
# define MAX 6
int Q[MAX];
int front=0,rear=0;
void insertQ();
void deleteQ();
void displayQ();

void main()
{
      int ch;
      while(1)
      {
clrscr();
printf("\n \tQueue operations using ARRAY..");
printf("\n----------**********            \n");
printf("\n 1. Insert ");
printf("\n 2. Delete ");
printf("\n 3. Display");
printf("\n 4. Quit ");
printf("\n Enter your choice: ");
scanf("%d", &ch);
switch(ch)
{
        case1:
                insertQ();
                break;
        case2:
                deleteQ();
                break;
        case3:
                displayQ();
                break;
        case4:

}
getch();
     }
}
void insertQ()
{
int data;
if(rear == MAX)
        printf("\n Linear Queue is full");
else
{
        printf("\n Enter data: ");
        scanf("%d", &data);
        Q[rear] = data;
```

```
        rear++;
        printf("\n Data Inserted in the Queue ");
}
}


void deleteQ()
{
if(rear == front)
        printf("\n\n Queue is Empty..");
else
{
        printf("\n Deleted element from Queue is %d", Q[front]);
        front++;
}
}

void displayQ()
{
int i;
if(front == rear)
        printf("\n\n\t Queue is Empty");
else
{
                printf("\n Elements in Queue are: ");
        for(i = front; i < rear; i++)
        {
                printf("%d\t", Q[i]);
        }
}
}
```