Experiment No 4: Implement of Linear Queue ADT using Array

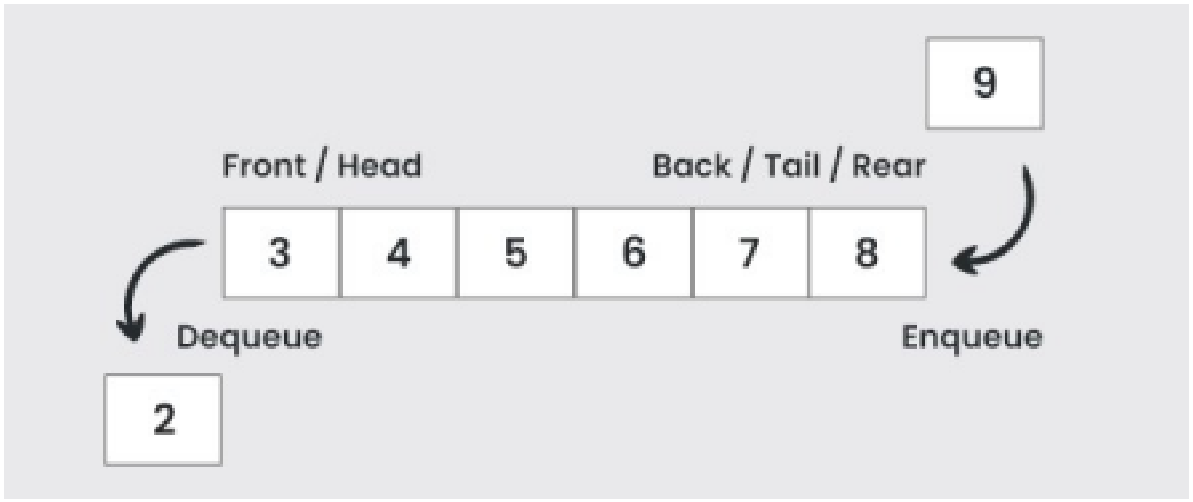
Aim: To implement a Queue using arrays.

Objective:

- Understand the Queue data structure and its basis operations.
- Understand the method of defining Queue ADT and its basic operations.
- Learn how to create objects from an ADT and member function are invoked.

Theory:

A Queue is an ordered collection of items from which items may be deleted at one end (called the front of the quete) and into which items may be inserted at the other end (the rear of the queue). Queues remember things in first-in-first-out (FIFO) order. The basic operations in a queue are: Enqueue-Adds an item to the end of queue. Dequeue - Removes an item from the front



A queue is implemented using a one dimensional array. FRONT is an integer value, which contains the array index of the front element of the array. REAR is an integer value, which contains the array index of the rear element of the array. When an element is deleted from the queue, the value of front is increased by one. When an element is inserted into the queue, the value of rear is increased by one.

Infix
Postfix

Algorithm:

```
ENQUEUE(item)

    1. If (queue is full)
         Print ”overflow”
    2. if (frist node insertion)
         Front++
    3. rear++
         Queue[rear]=value

DEQUEUE()

    1. If (queue is empty)
         printf “underflow”
    2. if (front=rear)
          Front=1 and rear=-1
    3. t=queue[front]
    4. front++
    5. return t

ISEMPTY()

    1. If(front=-1) then
                return 1
    2. return 0

ISFULL()

    1. If (rear=max)then
              return 1
    2. return 0
```

Code :

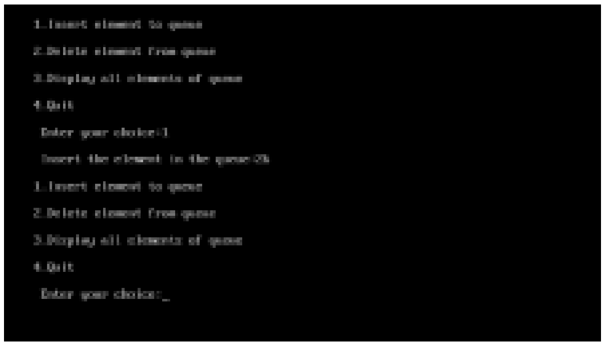
```
#include<stdio.h>

#include<stdlib.h>

#include<conio.h>
```

```
#define MAX 50
void insert();
void dequeue();
void display();
int queue_array[MAX];
int rear=-1;
int front=-1;
void main()
{
int choice;
clrscr();
while (1)
{
printf("\n1.Insert element to queue\n");
printf("\n2.Delete element from queue\n");
printf("\n3.Display all elements of queue\n");
printf("\n4.Quit\n");
printf("\n Enter your choice:");
scanf("\n%d",&choice);
switch (choice)
{
case 1:
insert();
break;
case 2:
dequeue();
break;
case 3:
```

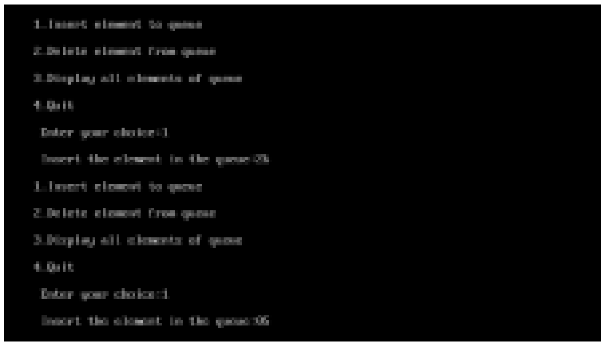
```
display();
break;
case 4:
exit(1);
default:
printf("\n Wrong choice \n");
}
}
}

void insert()
{
int item;
if (rear==MAX-1)
printf("\n Queue overflow \n");
else
{
if(front==-1)front=0;
printf("\n Insert the element in the queue:");
scanf("\n%d",&item);
rear=rear+1;
queue_array[rear]=item;
}
}
void dequeue()
{
if(front==-1||front>rear)
{
```

```
printf("\n Queue underflow\n");
return;
}
else
{
printf("\nElement deleted from queue is: %d\n",queue_array[front]);
front=front+1;
}
}
void display()
{
int i;
if(front==-1)
printf("\nQueue is empty\n");
else
{
printf("\nQueue is:\n");
for(i=front;i<=rear;i++)
printf("\n%d\t",queue_array[i]);
printf("\n");
}
}
```

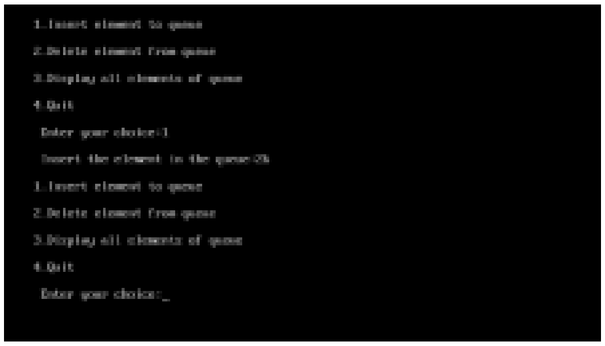
Output:



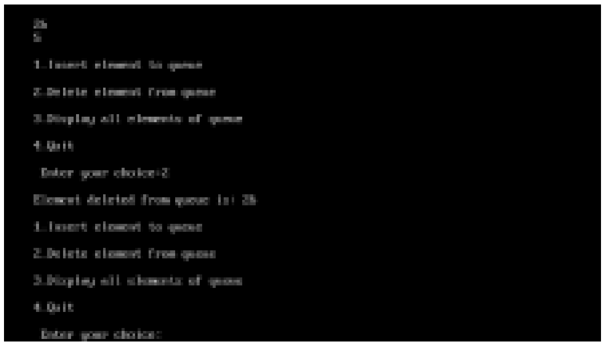
```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:1
 Insert the element in the queue:26
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:_
```



```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:1
 Insert the element in the queue:26
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:1
 Insert the element in the queue:05
```



```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:1
 Insert the element in the queue:26
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:_
```



```
26
5
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:2
Element deleted from queue is: 26
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
 Enter your choice:
```

Conclusion :

The queue data structure is a linear type of data structure that is used to store the elements. In this data structure elements are stored in the FIFO technique. A queue data structure used an array or linked list during its implementation.