Experiment No. 2

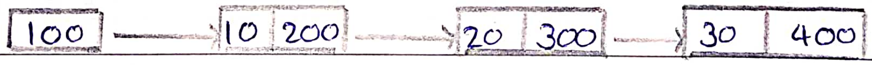
Aim : Implementation of queue using array for real-world application.

Objectives : 1) To introduce the concepts of data structure and analysis procedure.
2) To conceptualize linear data structures and its implementation for various real-world applications.

Theory :

1) Introduction to linear and non-linear data structure:

Linear Data Structure : Organize data elements in linear fashion and each element is attached one after other contiguous memory locations allocation.



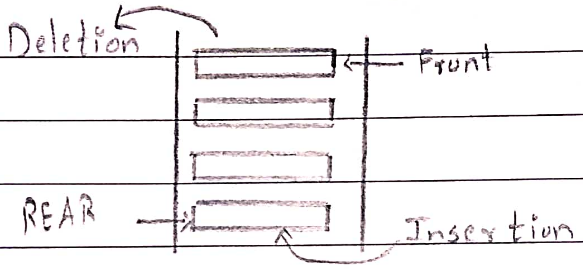
Examples :- Array, stack, Queue, Lists

Non-Linear Data Structure : Organization is not in a sequential fashion and its possible to attach a element to other ~~serv~~, several data elements multiple relationships among them.

Examples: Graphs, Tree

2) Introduction to Queue:

- Queue is a linear structure which follows a particular order in which the operation are performed. the ~~ed~~. order is First In First Out (FIFO).
- In a queue, new elements are added to queue from one end called REAR end & elements are always removed from other end called Front end.



Operation in Queue:

i) Inqueue : Adds an item in queue.
ii) Dequeue : Removes an item in queue.
iii) Front : Get the front item from queue.
iv) Rear : Get~~H~~ the rear item from queue.

3) Algorithm:

1) QINSERT (Q, F, R, N, Y) given F & R pointers to front and rear elements of queue Q having N elements, element Y insertion in queue Q.

```
i) If R ≥ N
       then write ('Overflow')
ii) [Increment rear pointer] R ← R + 1
iii) [Insert element] Q[R] ← Y
```

```
iv) [Is front pointer properly set ?]
        If    F=0
              then  F ← 1
                Return
```

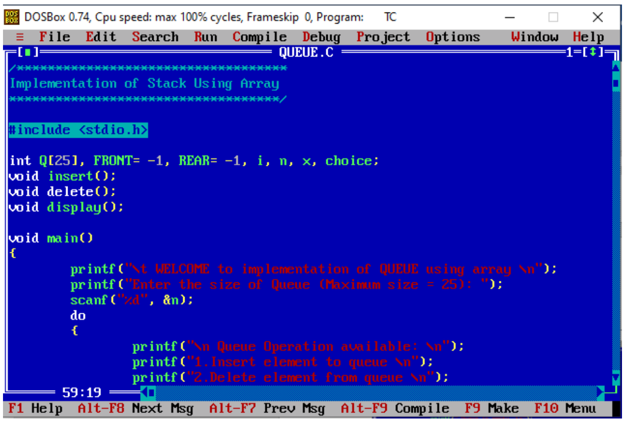
2) QDELETE (Q, F, R), given F & R pointers to front and rear elements of queue Q, element Y is to be deleted

```
i)  if   F = 0
          then write ('underflow')
               Return (0)
ii) [Delete element]    Y ← Q[F]
iii) [Queue  empty]
        if   F = R
             then  F ← R ← 0
             else   F ← F + 1   (increment front pointer)
iv) [Return  element]
              Return (Y)
```

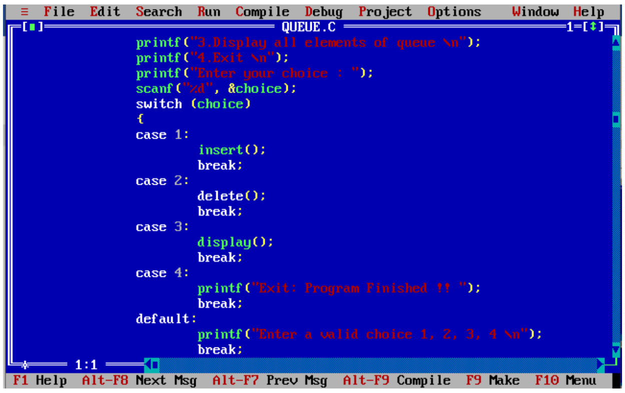
4) Example : At grocery store checkout counter people stands in queue for bill payment, as new person comes and stands at end of ~~row~~ queue and person after their bill payment is done get out of queue from front

5) Conclusion : From this experiment we learned how to implement queue using array and how to perform various queue operation ~~using algorithm~~ with the help of algorithm.

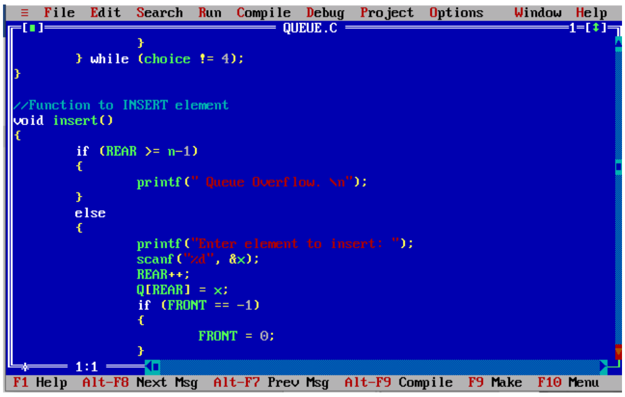
Outcome : Apply the concepts of queue for real-world application.


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
≡ File Edit Search Run Compile Debug Project Options Window Help
QUEUE.C
1
59:19
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

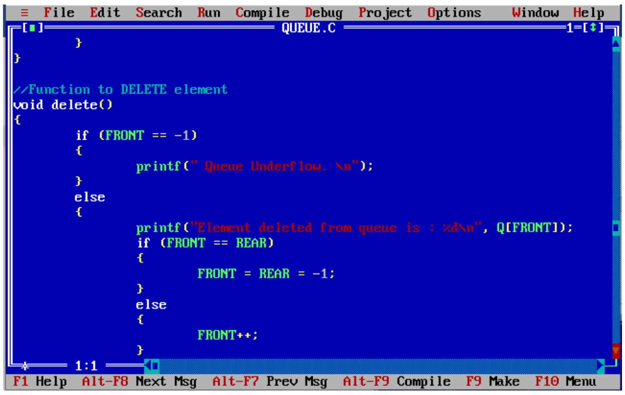
```
/***********************************
Implementation of Stack Using Array
***********************************/

#include <stdio.h>

int Q[25], FRONT= -1, REAR= -1, i, n, x, choice;
void insert();
void delete();
void display();

void main()
{
        printf("\t WELCOME to implementation of QUEUE using array \n");
        printf("Enter the size of Queue (Maximum size = 25): ");
        scanf("%d", &n);
        do
        {
                printf("\n Queue Operation available: \n");
                printf("1.Insert element to queue \n");
                printf("2.Delete element from queue \n");
```


≡ File Edit Search Run Compile Debug Project Options Window Help
QUEUE.C
1
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

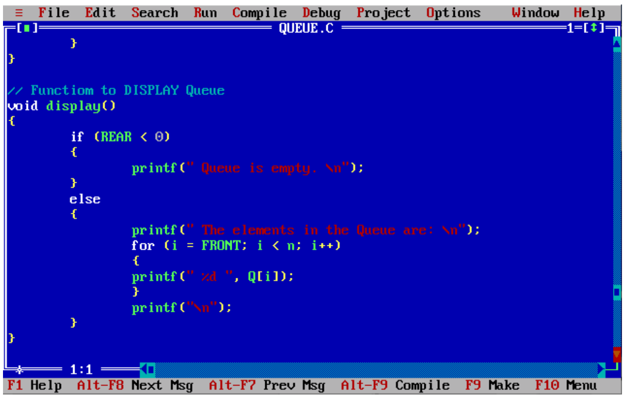
```
                printf("3.Display all elements of queue \n");
                printf("4.Exit \n");
                printf("Enter your choice : ");
                scanf("%d", &choice);
                switch (choice)
                {
                case 1:
                        insert();
                        break;
                case 2:
                        delete();
                        break;
                case 3:
                        display();
                        break;
                case 4:
                        printf("Exit: Program Finished !! ");
                        break;
                default:
                        printf("Enter a valid choice 1, 2, 3, 4 \n");
                        break;
```


≡ File Edit Search Run Compile Debug Project Options Window Help
QUEUE.C
1
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

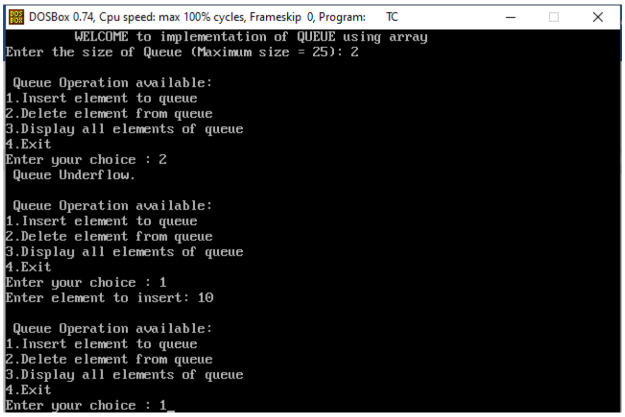
```
                }
        } while (choice != 4);
}

//Function to INSERT element
void insert()
{
        if (REAR >= n-1)
        {
                printf(" Queue Overflow. \n");
        }
        else
        {
                printf("Enter element to insert: ");
                scanf("%d", &x);
                REAR++;
                Q[REAR] = x;
                if (FRONT == -1)
                {
                        FRONT = 0;
                }
```


≡ File Edit Search Run Compile Debug Project Options Window Help
QUEUE.C
1
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

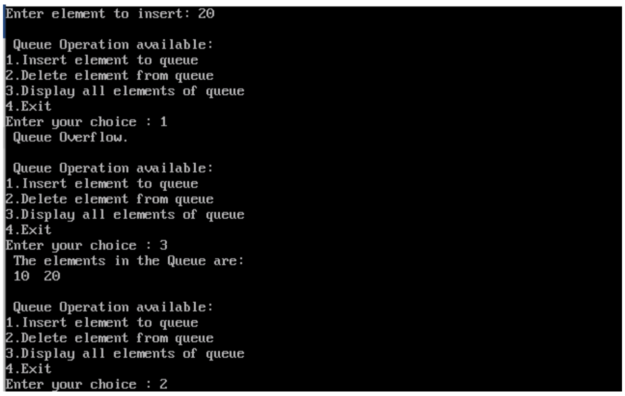
```
        }
}

//Function to DELETE element
void delete()
{
        if (FRONT == -1)
        {
                printf(" Queue Underflow. \n");
        }
        else
        {
                printf("Element deleted from queue is : %d\n", Q[FRONT]);
                if (FRONT == REAR)
                {
                        FRONT = REAR = -1;
                }
                else
                {
                        FRONT++;
                }
```


≡ File Edit Search Run Compile Debug Project Options Window Help
QUEUE.C
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

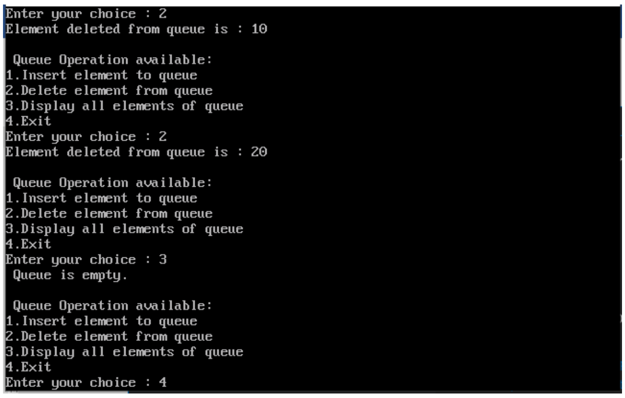
```
        }
}

// Function to DISPLAY Queue
void display()
{
        if (REAR < 0)
        {
                printf(" Queue is empty. \n");
        }
        else
        {
                printf(" The elements in the Queue are: \n");
                for (i = FRONT; i < n; i++)
                {
                printf(" %d ", Q[i]);
                }
                printf("\n");
        }
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
         WELCOME to implementation of QUEUE using array
Enter the size of Queue (Maximum size = 25): 2

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 2
 Queue Underflow.

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 1
Enter element to insert: 10

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 1_
```



```
Enter element to insert: 20

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 1
 Queue Overflow.

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 3
 The elements in the Queue are:
 10  20

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 2
```



```
Enter your choice : 2
Element deleted from queue is : 10

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 2
Element deleted from queue is : 20

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 3
 Queue is empty.

 Queue Operation available:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Exit
Enter your choice : 4
```