Data Structures Lab

Experiment 5: Queue Operations

Project By:

Mohammed Rabeeh
Roll No: 35
TVE18CS036

Contents

1 Aim **2**

2 Problem Description **2**

3 Algorithm **2**

3.1 Enqueue 2

3.2 Dequeue 2

3.3 Display 3

4 Program Code **3**

5 Output **5**

6 Result **5**

1 Aim

To implement the different operations like enqueue, dequeue and display of the queue data structure in C.

2 Problem Description

Queue is a very commonly used data structure. It's a First in First Out data structure (FIFO) meaning that the first element to enter the queue will leave first. Some of the popular stack operations are given below.

1. **Enqueue:** Inserts are element into the rear of the queue.
2. **Dequeue:** Remove the front inserted element. Returns the removed element.
3. **Display:** Prints the queue.

3 Algorithm

3.1 Enqueue

1. Read input element.
2. If rear is equal to size - 1, then print "Queue is full".
3. Else, rear = rear + 1
4. If front is equal to -1, assign front = 0
5. Assign queue[rear] = element

3.2 Dequeue

1. If front is equal to -1 or front ¿ rear, print "Queue is empty"
2. Else print queue[front]
3. front = front - 1

3.3 Display

1. If front is equal to -1, print "Queue is empty"
2. else, run a for loop from i = front to i = rear.
3. print queue[i].

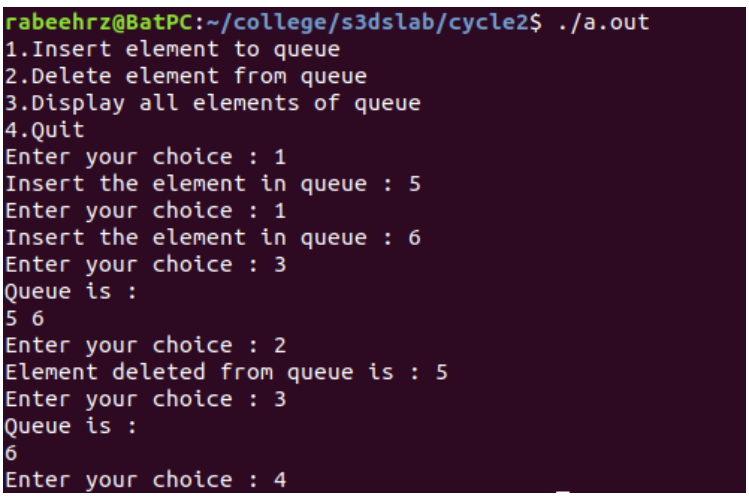
4 Program Code

```
#include<stdio.h>
#include<stdlib.h>
#define MAX_SIZE 500
void insert();
void delete();
void display();
int queue_array[MAX_SIZE], rear = -1, front = -1;

void main() {
   int choice;
    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");
    while (1) {
        printf("Enter your choice : ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insert();
                break;
            case 2:
                delete();
              break;
            case 3:
               display();
                break;
```

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

        }

    }

}

void insert() {
        int add_item;
        if (rear == MAX_SIZE - 1)
           printf("Queue Full \n");
        else {
            if (front == - 1)
               front = 0;
            printf("Insert the element in queue : ");
            scanf("%d", &add_item);
            rear = rear + 1;
            queue_array[rear] = add_item;
        }
}

void delete() {
        if (front == - 1 || front > rear) {
            printf("Queue Empty \n");
            return ;
        } else {
            printf("Element deleted from queue is : %d\n",
                queue_array[front]);
            front = front + 1;
        }

}

void display() {
        int i;
        if (front == - 1)
```

```
        printf("Queue is empty \n");
    else {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
        printf("\n");
    }
}
```

5 Output



```
rabeehrz@BatPC:~/college/s3dslab/cycle2$ ./a.out
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 queue : 5
Enter your choice : 1
Insert the element in queue : 6
Enter your choice : 3
Queue is :
5 6
Enter your choice : 2
Element deleted from queue is : 5
Enter your choice : 3
Queue is :
6
Enter your choice : 4
```

6 Result

The queue data structure and its functions were implemented in the C language and the output was verified.