



DATA STRUCTURE  
CIRCULAR QUEUE

ARİFE GÜL YALÇIN  
B1605.090054

.....	1
<b>SUMMARY .....</b>	<b>3</b>
<b>ABSTRACT .....</b>	<b>3</b>
<b>1. INTRODUCTION.....</b>	<b>3</b>
<b>2. BASIC INFORMATIONS .....</b>	<b>4</b>
<b>3. MAIN PART .....</b>	<b>4</b>
3.1. INSERT ( ENQUEUE() ).....	4
3.2. DELETE ( DEQUEUE() ) .....	5
3.3. DISPLAY ( DISPLAY() ) .....	5

## **SUMMARY**

*The Circular Queue is a linear data structure where the end position is connected back to the initial position to form a circle. There are 3 main operations for the Circular Queue.*

*These;*

1. *enqueue() Operation*
2. *dequeue() Operation*
3. *display() Operation*

## **ABSTRACT**

*First I should explain why a circular queue is needed when there is already a linear queue data structure.*

*In linear queue, when the queue is completely filled, it is not possible to add more values. Because when the values are deleted, we actually move the front of the tail forward. So we consider the queue smaller. It is necessary to reset the linear queue to add new value.*

*The Circular Queue is a linear data structure that follows the FIFO principle. FIFO stands for first-in-first-out. Circular Queue, on the other hand, restarts the queue from the first position after the end, rather than ending it at the last position.*

## **1. INTRODUCTION**

As seen in Figure 1, there are 3 different options and 1 exit option for Circular Queue operations. These transactions;

- 1) **INSERT**
- 2) **DELETE**
- 3) **DISPLAY**
- 4) **EXIT**

The user must first INSERT value to the queue.

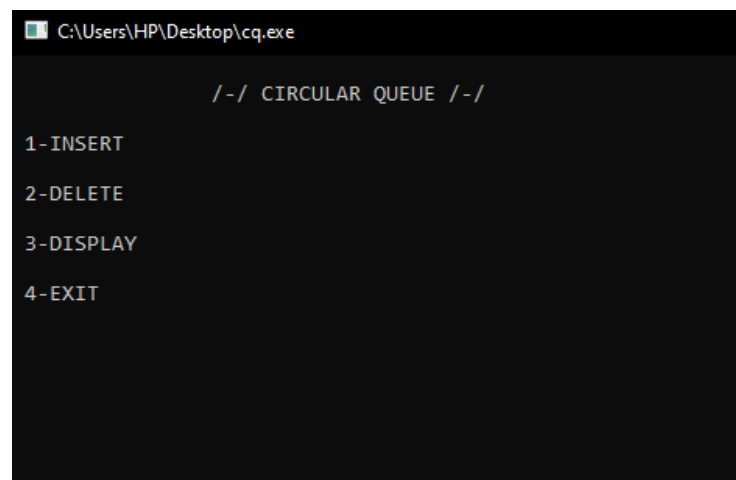


Figure 1 Circular Queue Operations

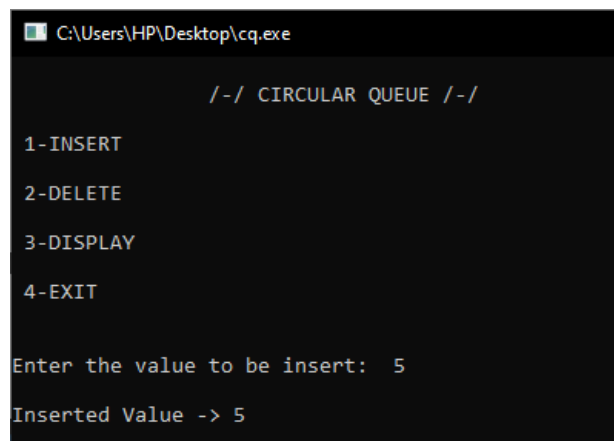
## 2. BASIC INFORMATION

The codes written within the scope of the project were written using the IDE named Dev-C++ and TDM-GCC Compiler was used as the compiler. Text file with .txt extension is used to write and read the data in the system.

## 3. MAIN PART

### 3.1. INSERT ( enqueue() )

- ✓ It is checked whether the queue is full.
- ✓ It is set as 0 FRONT value in the array for the first element.
- ✓ If rear = MAX-1 and front are not equal to 0, the array is declared empty.
- ✓ Each added value takes place in the array in order (the REAR index is incremented circularly by 1 each in the array order)



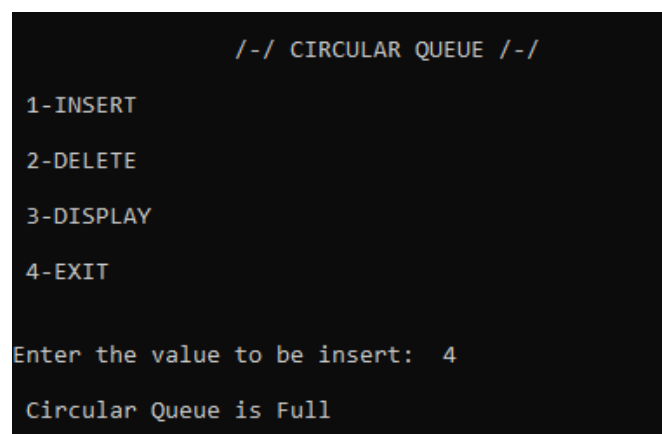
```
C:\Users\HP\Desktop\cq.exe

/-/ CIRCULAR QUEUE /-/

1-INSERT
2-DELETE
3-DISPLAY
4-EXIT

Enter the value to be insert: 5
Inserted Value -> 5
```

Figure 3.1. Insertion Value



```
C:\Users\HP\Desktop\cq.exe

/-/ CIRCULAR QUEUE /-/

1-INSERT
2-DELETE
3-DISPLAY
4-EXIT

Enter the value to be insert: 4
Circular Queue is Full
```

Figure 3.1.1. Circular Queue is Full

### 3.2. DELETE ( dequeue() )

- ✓ It is checked if there is a value in the row ( front = -1 and rear = -1, the array is empty and cannot be deleted ).
- ✓ Return the value indicated by front
- ✓ Increase front index by 1 circularly
- ✓ Reset the 'front' and 'rear' values to -1 for the last element.

```
        /-/ CIRCULAR QUEUE /-/  
  
1-INSERT  
2-DELETE  
3-DISPLAY  
4-EXIT  
  
Deleted Value : 5
```

Figure 3.2. Delete Value

### 3.3. DISPLAY ( display() )

- ✓ It is checked whether the queue is empty.
- ✓ If the row is not empty 'front' is assigned a value of i.
- ✓ If front <= rear, the value of i is displayed sequentially until rear.
- ✓ If i <= MAX-1, it is displayed sequentially until rear.

```
        /-/ CIRCULAR QUEUE /-/  
  
1-INSERT  
2-DELETE  
3-DISPLAY  
4-EXIT  
  
(  
Circular Queue Elements are :  
5      9      7      4      8      2
```

Figure 3.3. Before Delete Value

```

                                /-/ CIRCULAR QUEUE /-/

1-INSERT
2-DELETE
3-DISPLAY
4-EXIT

Circular Queue Elements are :
9      7      4      8      2      1
                                /-/ CIRCULAR QUEUE /-/

```

Figure 3.3.1. After Delete Value