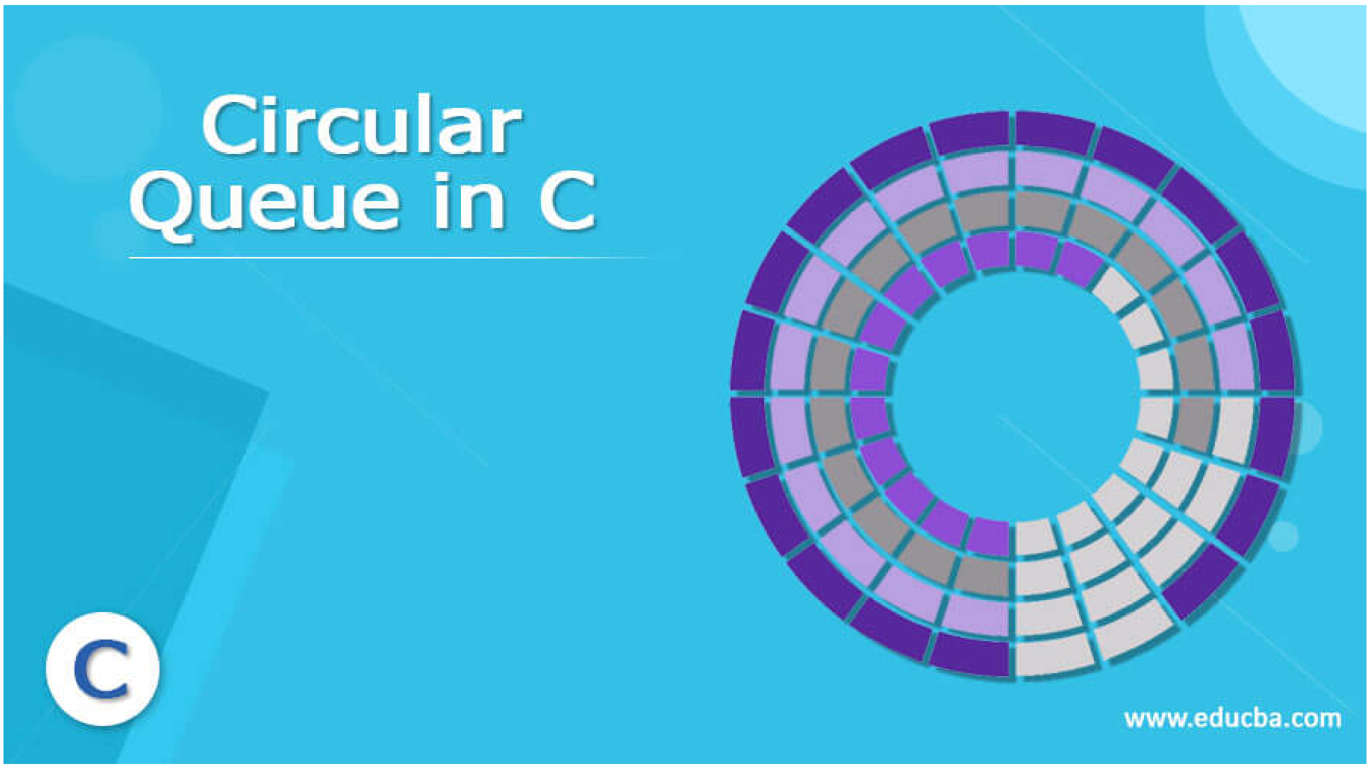
EDUCBA
(https://www.educba.com/software-development/)

←
(https://www.educba.com/queue-in-c/)

→
(https://www.educba.com/hexadecimal-in-c/)



Definition of Circular Queue in C

QUIZ

C Circular queue is defined as an implementation of the concept of the circular queue in programming language in a practical manner. Circular Queue is coined from the concept of a linear data structure that allows operations to be performed on the structure as FIFO (First In

EDUCBA
(https://www.educba.com/software-development/)

void spaces within it due to empty spaces at the front as FIFO technique is implemented.

Syntax:

In the syntax section, we will go through some syntaxes which are required for the implementation of the circular queue in C. The elements in this section of the article on circular queue don't necessarily depict the entire runnable code, but gives a sense of what are the different placeholders that are in place for the queue methodology to be able to execute successfully.

Declaring the header in C:

```
#include<stdio.h>
```

If statement in C:

```
if ( <variable name> <condition operator> <check value)> ){
< Statements >;
}


else{
< Statements >;
}
```

EDUCBA
(https://www.educba.com/software-development/)

```
< Function statements >;
}
```

Printing of statements in C:

```
printf(“ < Statements to be printed > ” );
```

How Circular queue works in C?

By now we pretty much have a fair idea of the structure of the circular queue, and it implements FIFO ordering. What FIFO essentially means is elements that go in first are removed the first. This structure is similar to a queue at an airport, wherein the person standing at the first gets to board the flight and so on till the last passenger boards the flight. But now what we talked about a queue at the airport, is like a linear queue. There is a limited number of seats in a flight and hence as the capacity gets full no one is allowed to be a part of the line or in other words doesn’t even get the ticket. Now assume a similar case where we have limited resources in a computer, and one is performing memory-intensive tasks all of which takes some time. Now the task which starts first followed by tasks that subsequently start coming consecutively starts getting piled up and now if the resource gets full, the other task is out into the waitlist. Assume that had the scenario been like the one we talked about in flights, any task in the waitlist will never execute as the line has completely ended. Now let us say we link the last element to the

first element saying that if the last element is filled check if the first element is free and QUIZ then start putting or inserting the task in resource management and keep the workflow continuous.

process is known as dequeuing). Apart from the elements we just talked about i.e. enqueuing, dequeuing, front (pointer), rear (pointer) we also have an operation isEmpty or isFull which does a check on the queue to understand if the queue is empty or full.

The circular increment is performed by modulo division, which gives the remainder of th division we perform by the corresponding numerator and denominator. Now to understand the working of modulo division, in case of enqueuing, the rear pointer is incremented by (rear + 1

(https://www.educba.com/software-development/)

replaced by N-1, we get ((N-1)+1)%N = N%N = 0. Thus the rear now again starts pointing to the start only on the condition that the queue is notFull. In case the queue is full one can easily throw out an exception saying that the queue is full and can't be loaded with more data. Let us now go through the process of enqueue and dequeue as pseudocode so that when we go through the code, it be super simple to grasp, and also the working of a circular queue is clearer.

Enqueue:

- Check if the Queue is full. If yes, throw an exception that nothing can be enqueued.
- The first element should contain FRONT as 0.
- Using modulo division increment the REAR index.
- Add the element to the new REAR index.

Dequeue:

- Check if the queue is empty. If yes, throw an exception that nothing can be dequeued.
- The value pointed by FRONT is returned.
- Using modulo division increment the FRONT index.
- In case of last element, we can forcefully set values of FRONT and REAR to -1.

Examples

Let us discuss examples of

Example #1

Implementation of circular Queue:

Syntax:

EDUCBA
(https://www.educba.com/software-development/)

```
// Is the queue full?
int checkFull() {
if ((front == rear + 1) || (front == 0 && rear == ARRSIZE - 1))
return 1;
return 0;
}
// Is the Queue Empty?
int checkEmpty() {
if (front == -1) return 1;
return 0;
}
// Element Adding
void enQueue(int ele) {
if (checkFull())
printf("\n Can't enter more. Queue Full \n");
else {
if (front == -1) front = 0;
rear = (rear + 1) % ARRSIZE;
array[rear] = ele;
printf("\n Pushed -> %d", ele);
}
}

// Element removing
int deQueue() {
int ele;
```

EDUCBA
(https://www.educba.com/software-development/)

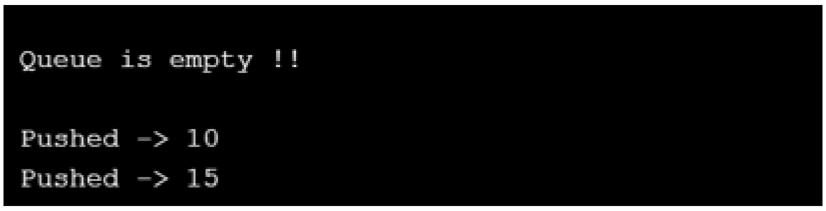
```
ele = array[front];
if (front == rear) {
front = -1;
rear = -1;
}
// Reset Queue after all elements are removed
else {
front = (front + 1) % ARRSIZE;
}
printf("\n Popped out -> %d \n", ele);
return (ele);
}
}
// Queue Display
void display() {
int i;
if (checkEmpty())
printf(" \n The queue is Empty\n");
else {
printf("\n Pointer for first element -> %d ", front);
printf("\n Items -> ");
for (i = front; i != rear; i = (i + 1) % ARRSIZE) {
printf("%d ", array[i]);

}
printf("%d ", array[i]);
printf("\n Pointer for Last element -> %d \n", rear);
```

QUIZ

```
deQueue();
enQueue(10);
enQueue(15);
enQueue(20);
enQueue(30);
enQueue(50);
enQueue(60);
// Will Fail inserting as the Queue is Full
enQueue(1);
display();
deQueue();
display();
// Will succeed as we removed one element using deQueue()
enQueue(2);
display();
// Will again Fail inserting as the Queue is Full
enQueue(100);
return 0;
}
```

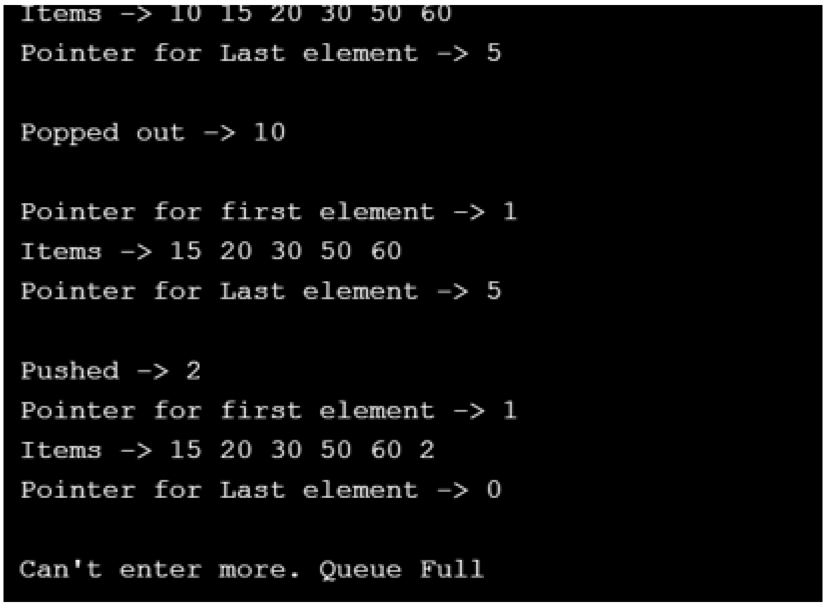
Output:



```
Queue is empty !!

Pushed -> 10
Pushed -> 15
```

(https://www.educba.com/software-development/)



```
Items -> 10 15 20 30 50 60
Pointer for Last element -> 5

Popped out -> 10

Pointer for first element -> 1
Items -> 15 20 30 50 60
Pointer for Last element -> 5

Pushed -> 2
Pointer for first element -> 1
Items -> 15 20 30 50 60 2
Pointer for Last element -> 0

Can't enter more. Queue Full
```

Conclusion

To conclude, in this article we have learned the working of the circular queue in C. Next, we encourage readers to try out switch case which will be a more user-friendly implementation of the circular queue as the entire flexibility will be on the user for pushing and popping of elements.

Recommended Articles

This is a guide to Circular Queue in C. Here we discuss definition, syntax, How Circular queue works in C? examples with code implementation. You may also have a look at the following articles to learn more –

1. Queue in Python (https://www.educba.com/queue-in-python/)
2. Priority Queue in C++ (https://www.educba.com/priority-queue-in-c-plus-plus/)
3. Queue in C (https://www.educba.com/queue-in-c/)
4. Java Queue Interface (https://www.educba.com/java-queue-interface/)

EDUCBA (https://www.educba.com/software-development/)

source=footer)

Privacy Policy (https://www.educba.com/privacy-policy/?source=footer)

Apps

iPhone & iPad (https://itunes.apple.com/in/app/educba-learning-app/id1341654580?mt=8)

Android (https://play.google.com/store/apps/details?id=com.educba.www)

Resources

Free Courses (https://www.educba.com/software-development/free-courses/?source=footer)

Java Tutorials (https://www.educba.com/software-development/software-development-tutorials/java-tutorial/?source=footer)

Python Tutorials (https://www.educba.com/software-development/software-development-tutorials/python-tutorial/?source=footer)

All Tutorials (https://www.educba.com/software-development/software-development-tutorials/?source=footer)

Certification Courses

All Courses (https://www.educba.com/software-development/courses/?source=footer)

Software Development Course - All in One Bundle (https://www.educba.com/software-development/courses/software-development-course/?source=footer)

Become a Python Developer (https://www.educba.com/software-development/courses/python-certification-course/?source=footer)

QUIZ

Java Course (https://www.educba.com/software-development/courses/java-course/?source=footer)

EDUCBA
(https://www.educba.com/software-development/)

VB.NET Course (https://www.educba.com/software-development/courses/vb-net-course/?source=footer)

PHP Course (https://www.educba.com/software-development/courses/php-course/?source=footer)

QUIZ