How To Implement Queue in C?

Last updated on Mar 29,2022 *98.8K Views*

Manthan Naik

A Queue is a linear data structure that stores a collection of elements. The queue operates on first in first out (FIFO) algorithm. This article will help you explore Queue In C

Following pointers will be covered in this article,

So let us get started then,

Analogy For Queue

You are visiting a doctor for a check-up. There are many people at the clinic. A lady is entering the names of all the people in a file. The person who comes first gets places first. When the doctor is free, he calls the first patient inside. This is a queue and follows a first in first out method as the first person to enter his name in the list gets treated first.

The people who are treated their names are removed from the list. This is how a queue works.

There are 2 pointers, the front is at the front of the queue and rear is at the back of the queue. We add elements from the back of the queue and remove them from the front of the queue.

Moving on with this article on Queue In C,

Operations On A Queue

- Enqueue- adding an element in the queue if there is space in the queue.
- Dequeue- Removing elements from a queue if there are any elements in the queue
- Front- get the first item from the queue.
- Rear- get the last item from the queue.
- isEmpty/isFull- checks if the queue is empty or full.

Applications

- A queue is used in scheduling processes in a CPU.
- It is used in data transfer between processes.
- It holds multiple jobs which are then called when needed.

Moving on with this article on Queue In C,

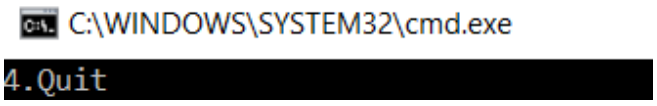
Java Certification Training Course

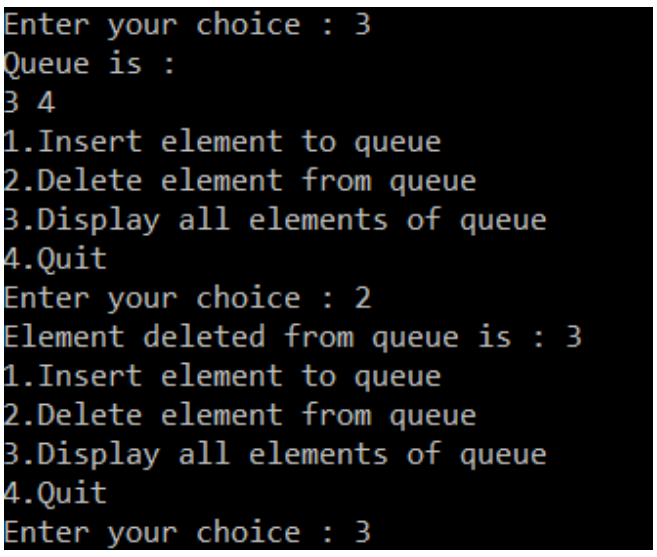
```
#include <stdio.h>
#include<stdlib.h>
#define MAX 50
void insert();
```



```
printf("4.Quit n");
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 item;
if(rear == MAX - 1)
printf("Queue Overflow n");
else
{
if(front== - 1)
front = 0;
printf("Inset the element in queue : ");
scanf("%d", &item);
rear = rear + 1;
queue_array[rear] = item;
}
}
void delete()
{
if(front == - 1 || front > rear)
{
printf("Queue Underflow n");
return;
}
else
{
printf("Element deleted from queue is : %dn", 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");
}
}
```

Output





Explanation

This code is a menu-driven implementation of a queue. First, define the size of MAX variable to be 50. Then, the array called queue_array is declared of size MAX. There are three functions that are to be declared. The functions are, insert, display and delete functions. A menu-driven main function is used. The user is asked to enter his choice and call the appropriate function to perform the task.

There are 2 pointers, the front is at the front of the queue and rear is at the back of the queue. We add elements from the back of the queue and remove them from the front of the queue.

Moving on with this article on Queue In C,

Insert Function

```
1  void insert()
2  {
3  int item;
4  if(rear == MAX - 1)
5  printf("Queue Overflow n");
6  else
7  {
8  if(front == - 1)
9  front = 0;
10 printf("Inset the element in queue : ");
11 scanf("%d", &item);
12 rear = rear + 1;
13 queue_array[rear] = item;
14 }
15 }
```

In the insertion part, First, declare an item which is to be added. The user will enter the item. Check if the queue is full, if yes give the overflow message or else check if the queue is empty. The rear is then incremented by one and the at the location rear add the new item.

Moving on with this article on Queue In C,

Delete Function

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

In the delete part, check again if the queue is empty. If yes, print underflow error. Otherwise, print the first element, that is the element that will be deleted and increment front. This is how the deletion takes place.

Moving on with this article on Queue In C,

Display Function

```
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");
}
}
```

Just display the queue like how an array. Check if the queue is empty here as well. A Queue has a complexity of O(1) as no loops are present in any operation.

Moving on with this article on Queue In C,

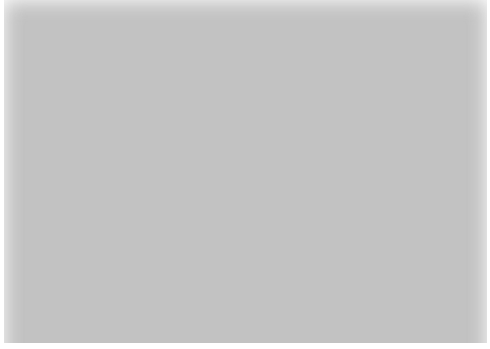
Limitations Of This Implementation

Consider a queue, with size 5. We have entered 5 elements but have later deleted first 2 elements. Now there is a problem. We have free space but, space won’t be used as we can’t traverse again. This problem is solved using the circular queue.

This brings us to the end of this article on Queue In C.

With this we come to the end of this article on 'Queue In C'. I hope you found this informative and helpful, stay tuned for more tutorials on similar topics.You may also checkout our training program to get in-depth knowledge on jQuery

Trending Courses in Programming & Frameworks

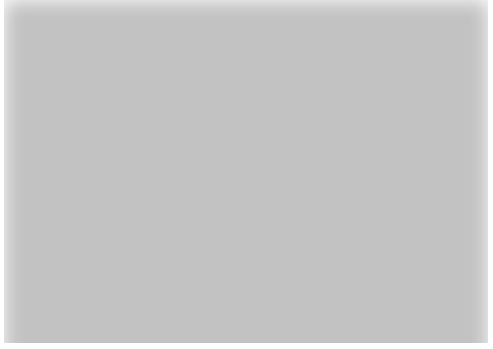


Full Stack Web Development Internship Program

- 3k Enrolled Learners
- Weekend/Weekday
- Live Class

Reviews

5 (850)

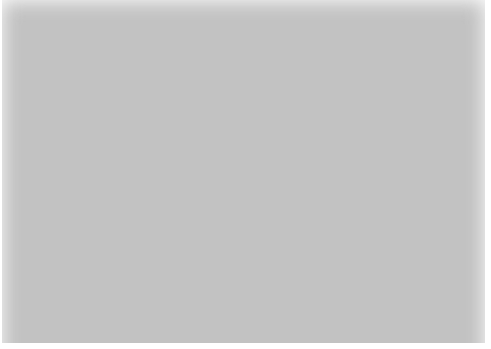


Java Certification Training Course

- 55k Enrolled Learners
- Weekend
- Live Class

Reviews

4 (21800)

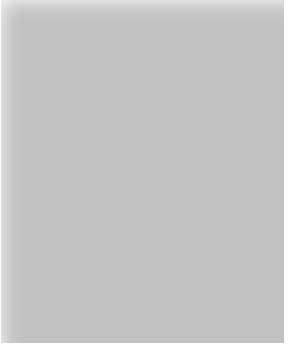


Python Scripting Certification Training

- 12k Enrolled Learners
- Weekend
- Self Paced

Reviews

5 (4600)



C Programming Certification Co

- 1k Enrolled Learner
- Weekend
- Self Paced

Reviews

5 (250

‹›

Browse Categories

Artificial Intelligence | BI and Visualization | Big Data | Blockchain | Cloud Computing | Cyber Security | Data Science | Data Warehousing and ETL | Databases | DevOps | Digital Marketing | Enterprise | Front End Web Development | Mobile Development | Operating Systems | Project Management and Methodologies | Robotic Process Automation | Software Testing | Systems & Architecture

edureka!

TRENDING CERTIFICATION COURSES

- DevOps Certification Training
- AWS Architect Certification Training
- Big Data Hadoop Certification Training
- Tableau Training & Certification
- Python Certification Training for Data Science
- Selenium Certification Training
- PMP® Certification Exam Training
- Robotic Process Automation Training using UiPath
- Apache Spark and Scala Certification Training
- Microsoft Power BI Training
- Online Java Course and Training
- Python Certification Course

TRENDING MASTERS COURSES

- Data Scientist Masters Program
- DevOps Engineer Masters Program
- Cloud Architect Masters Program
- Big Data Architect Masters Program
- Machine Learning Engineer Masters Program
- Full Stack Web Developer Masters Program
- Business Intelligence Masters Program
- Data Analyst Masters Program
- Test Automation Engineer Masters Program
- Post-Graduate Program in Artificial Intelligence & Machine Learning
- Post-Graduate Program in Big Data Engineering