

Programming Tutorials by SourceTricks

(<http://www.sourcetricks.com/>)

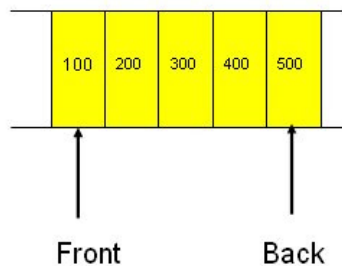
5:36:00 PM SourceTricks (<https://www.blogger.com/profile/17421383822093860960>) Data Structures (<http://www.sourcetricks.com/search/label/Data%20Structures>)
14 comments (<http://www.sourcetricks.com/2008/07/c-queues.html#comment-form>)

C++ Queues (<http://www.sourcetricks.com/2008/07/c-queues.html>)

What is a queue? How to implement queues using C++?

This article explains the queue data structure and demonstrates sample implementation using C++.

- Queue is a first-in, first-out (FIFO) data structure. i.e. the element added first to the queue will be the one to be removed first.
- Elements are always added to the back of the queue and removed from the front of the queue.
- Typical use cases of queues include:- (1) In Inter-Process Communication (IPC) message queues is a common mechanism for communication between processes.
- Some of the common terminology associated with queues include ADD/ PUSH and DELETE/ POP of elements to the queue.
- ADD/ PUSH refers to adding an element to the queue.
- DELETE/ POP refers to removing an element from the queue.
- Diagram below explains the queue.



(http://bp3.blogger.com/_QGv0qvtpT1Y/SHnxZkvqB_I/AAAAAAAAADQ/d_1ii7meAWII/s1600-h/q.bmp)

Demonstrate the implementation of a simple queue using arrays

```
#include <iostream>
#include <cstdlib>
using namespace std;

const int MAX_SIZE = 100;

class QueueOverflowException
{
public:
    QueueOverflowException()
    {
        cout << "Queue overflow" << endl;
    }
};

class QueueEmptyException
{
public:
    QueueEmptyException()
    {
        cout << "Queue empty" << endl;
    }
};

class ArrayQueue
{
private:
    int data[MAX_SIZE];
    int front;
    int rear;
public:
    ArrayQueue()
    {
        front = -1;
        rear = -1;
    }

    void Enqueue(int element)
    {
        // Don't allow the queue to grow more
        // than MAX_SIZE - 1
        if ( Size() == MAX_SIZE - 1 )
            throw new QueueOverflowException();

        data[rear] = element;

        // MOD is used so that rear indicator
        // can wrap around
        rear = ++rear % MAX_SIZE;
    }

    int Dequeue()
    {
        if ( isEmpty() )
            throw new QueueEmptyException();

        int ret = data[front];

        // MOD is used so that front indicator
        // can wrap around
        front = ++front % MAX_SIZE;

        return ret;
    }

    int Front()
    {
        if ( isEmpty() )
            throw new QueueEmptyException();

        return data[front];
    }

    int Size()
    {
        return abs(rear - front);
    }

    bool isEmpty()
    {
        return ( front == rear ) ? true : false;
    }
};

int main()
{
```

```
ArrayQueue q;
try {
    if ( q.isEmpty() )
    {
        cout << "Queue is empty" << endl;
    }

    // Enqueue elements
    q.Enqueue(100);
    q.Enqueue(200);
    q.Enqueue(300);

    // Size of queue
    cout << "Size of queue = " << q.Size() << endl;

    // Front element
    cout << q.Front() << endl;

    // Dequeue elements
    cout << q.Dequeue() << endl;
    cout << q.Dequeue() << endl;
    cout << q.Dequeue() << endl;
}
catch (...) {
    cout << "Some exception occurred" << endl;
}
}
```

OUTPUT:-

```
Queue is empty
Size of queue = 3
100
100
200
300
```

Demonstrate the implementation of a simple queue using linked lists

```

#include <iostream>
using namespace std;

class QueueEmptyException
{
public:
    QueueEmptyException()
    {
        cout << "Queue empty" << endl;
    }
};

class Node
{
public:
    int data;
    Node* next;
};

class ListQueue
{
private:
    Node* front;
    Node* rear;
    int count;

public:
    ListQueue()
    {
        front = NULL;
        rear = NULL;
        count = 0;
    }

    void Enqueue(int element)
    {
        // Create a new node
        Node* tmp = new Node();
        tmp->data = element;
        tmp->next = NULL;

        if ( isEmpty() ) {
            // Add the first element
            front = rear = tmp;
        }
        else {
            // Append to the list
            rear->next = tmp;
            rear = tmp;
        }

        count++;
    }

    int Dequeue()
    {
        if ( isEmpty() )
            throw new QueueEmptyException();

        int ret = front->data;
        Node* tmp = front;

        // Move the front pointer to next node
        front = front->next;

        count--;
        delete tmp;
        return ret;
    }

    int Front()
    {
        if ( isEmpty() )
            throw new QueueEmptyException();

        return front->data;
    }

    int Size()
    {
        return count;
    }

    bool isEmpty()
    {
        return count == 0 ? true : false;
    }
};

```

```
    }  
};  
  
int main()  
{  
    ListQueue q;  
    try {  
        if ( q.isEmpty() )  
        {  
            cout << "Queue is empty" << endl;  
        }  
  
        // Enqueue elements  
        q.Enqueue(100);  
        q.Enqueue(200);  
        q.Enqueue(300);  
  
        // Size of queue  
        cout << "Size of queue = " << q.Size() << endl;  
  
        // Front element  
        cout << q.Front() << endl;  
  
        // Dequeue elements  
        cout << q.Dequeue() << endl;  
        cout << q.Dequeue() << endl;  
        cout << q.Dequeue() << endl;  
    }  
    catch (...) {  
        cout << "Some exception occurred" << endl;  
    }  
}
```

OUTPUT:-

```
Queue is empty  
Size of queue = 3  
100  
100  
200  
300
```

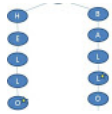
Demonstrate the queue library usage in standard template library (STL)

```
#include <iostream>  
#include <queue>  
using namespace std;  
  
int main()  
{  
    queue<int> q;  
    q.push(100);  
    q.push(200);  
    q.push(300);  
    q.push(400);  
  
    cout << "Size of the queue: " << q.size() << endl;  
  
    cout << q.front() << endl;  
    q.pop();  
  
    cout << q.front() << endl;  
    q.pop();  
  
    cout << q.front() << endl;  
    q.pop();  
  
    cout << q.front() << endl;  
    q.pop();  
  
    if ( q.empty() ) {  
        cout << "Queue is empty" << endl;  
    }  
}
```

OUTPUT:-

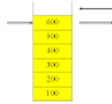
```
Size of the queue: 4
100
200
300
400
Queue is empty
```

Related Posts:



C++ Tries (<http://www.sourcetricks.com/2011/06/c-tries.html>)

What is a Trie? How to implement Tries in C++? This article explains the Trie data structure and provides a sample implementation in C++. Tries are used to i... Read More (<http://www.sourcetricks.com/2011/06/c-tries.html>)



C++ Stacks (<http://www.sourcetricks.com/2008/07/c-stacks.html>)

What is a stack? How to implement stacks using C++? This article explains about the basics of a stack data structure and demonstrates with simple examples imp... Read More (<http://www.sourcetricks.com/2008/07/c-stacks.html>)



C++ Heaps (<http://www.sourcetricks.com/2011/06/c-heaps.html>)

What is a heap data structure? How to implement in C++? This article explains the heap data structure and provides a sample implementation using C++. Heap ... Read More (<http://www.sourcetricks.com/2011/06/c-heaps.html>)



C++ Pre-Order, In-Order, Post-Order Traversal of Binary Search Trees (<http://www.sourcetricks.com/2011/05/c-pre-order-in-order-post-order.html>)

Pre-Order, In-Order, Post-Order traversal of Binary Search Trees (BST) This article explains the depth first search (DFS) traversal methods for binary search... Read More (<http://www.sourcetricks.com/2011/05/c-pre-order-in-order-post-order.html>)



Binary Search Trees in C++ (<http://www.sourcetricks.com/2011/06/binary-search-trees-in-c.html>)

What is binary search trees? How to implement in C++? This article explains the concept of binary search trees (BST) and provides a sample implementation in ... Read More (<http://www.sourcetricks.com/2011/06/binary-search-trees-in-c.html>)

← Newer Post (<http://www.sourcetricks.com/2008/07/c-singly-linked-lists.html>)

Older Post → (<http://www.sourcetricks.com/2008/07/c-stacks.html>)

14 comments :

Anonymous June 27, 2009 at 1:09 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1246088398711#c7276116569305769911>)

nice tutorial

Reply

Luke Spencer (<http://www.popssurvivalguide.com>) December 18, 2009 at 7:30 AM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1261101631576#c7047204252519844596>)

I took all this stuff a few years ago...thanks for the refresh.

Reply



John (<https://www.blogger.com/profile/15799594578657850460>) February 1, 2011 at 11:02 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1296581576378#c4120818922925338879>)

nice tutorial, thanks really needed this

Reply



Harpreet (<http://harpreetsingh.myopenid.com/>) March 20, 2011 at 2:18 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1300610922840#c7914351570104784542>)

Seems like a bug to me in the first piece of code,

if you add 80 integers to queue, then delete 50 of them,

the queue is left with 30 integers.

Assuming queue size is 100, the user should be able to add 70 more integers into the queue, but the program does not support this.

Reply

Replies



Manish Agrawal (<https://www.blogger.com/profile/14197431911448422786>) October 9, 2013 at 3:30 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1381312825061#c5316529461808769768>)

there is need slight modification in code. The size function has not been currently Implemented.

So here is the modified code:

```
#include
```

```
#include
```

```
using namespace std;
```

[Back to Top](#)

```

const int MAX_SIZE = 3;

class QueueOverflowException
{
public:
    QueueOverflowException()
    {
        cout << "Queue overflow" << endl;
    }
};

class QueueEmptyException
{
public:
    QueueEmptyException()
    {
        cout << "Queue empty" << endl;
    }
};

class ArrayQueue
{
private:
    int data[MAX_SIZE];
    int front;
    int rear;
public:
    ArrayQueue()
    {
        front = 0;
        rear = 0;
    }

    void Enqueue(int element)
    {
        // Don't allow the queue to grow more
        // than MAX_SIZE - 1
        if ( Size() == MAX_SIZE - 1 )
            throw new QueueOverflowException();
        //cout<<rear<<" "<<front<<endl;
        data[rear] = element;

        // MOD is used so that rear indicator
        // can wrap around
        rear = ++rear % MAX_SIZE;
    }

    int Dequeue()
    {
        if ( isEmpty() )
            throw new QueueEmptyException();

        int ret = data[front];

        // MOD is used so that front indicator
        // can wrap around
        front = ++front % MAX_SIZE;

        return ret;
    }

    int Front()
    {
        if ( isEmpty() )
            throw new QueueEmptyException();

        return data[front];
    }

    int Size()
    {
        if(front<rear) return (rear-front);
        return(rear-front+MAX_SIZE);
    }

    bool isEmpty()
    {
        return ( front == rear ) ? true : false;
    }
};

```

[Back to Top](#)

```

    }
};

int main()
{
    ArrayQueue q;
    try {
        if ( q.isEmpty() )
        {
            cout << "Queue is empty" << endl;
        }

        // Enqueue elements
        q.Enqueue(100);
        cout << q.Dequeue() << endl;
        q.Enqueue(200);
        cout << q.Dequeue() << endl;
        cout << q.Dequeue() << endl;
        q.Enqueue(300);
        q.Enqueue(200);
        //cout << q.Size() << endl;

        // Size of queue
        cout << "Size of queue = " << q.Size() << endl;
        q.Enqueue(200);

        // Front element
        //cout << q.Front() << endl;

        // Dequeue elements
        //cout << q.Dequeue() << endl;
    }
    catch (...) {
        cout << "Some exception occurred" << endl;
    }
}

```

Reply

Anonymous September 3, 2011 at 8:32 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1315062138150#c5407316357733481563>)

I know people freak out when you remind them about C++ standards/conventions (apparently "forum guys" are more intelligent than C++ standard committee members! -sarcasm) but in C++ all upper case names are -conventionally- reserved for macros. You are using a global variable (which I can't see why you are using it! You could EASILY avoid using that global variable) and you named it in upper case letters.

These are the things that confuses people who are new to C++.

Reply

Anonymous September 11, 2011 at 11:13 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1315762983467#c3720644896033607329>)

Ok this time I put it simple and clear, may be you publish my comment or change your code! There is absolutely no need for that global constant variable (with all capitol letters!). This variable is used inside ArrayQueue class, it is used only and only inside this class, therefore it should be an attribute of this class.

C++ coding standards are not just bells and whistles! if you can't follow them I suggest you move to C

Reply



Sven Larsson (<https://www.blogger.com/profile/15472987568473481602>) January 16, 2013 at 2:59 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1358328572881#c8256387622945110176>)

Hi coders! I found a bug in the array implementation :-(When rear wraps around (front has not) the Size() value is wrong e.g. $\text{abs}(0 - 99) = 99$. The size is really 1 not 99 ...

Cheers!

Sven

Reply



vee inno (<https://www.blogger.com/profile/07038533807487705852>) February 27, 2013 at 9:26 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1361980592075#c2223406224132275441>)

aoa. i want a simple push pop,full,empty implementation through stacks without using classes... plzzzzzz help me guys..
another favour i required.. i want to learn what is queue.i don't know even its basics

Reply

[Back to Top](#)



Cody Brown (<https://www.blogger.com/profile/09346785846001509555>) October 23, 2013 at 7:52 AM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1382494946978#c6169212379614675291>)

Regarding Array implementation:

Yeah you cant initialize rear and front as -1 if you do not increment them BEFORE the item insertion in enqueue or dequeue. You can initialize them as -1 if you increment before the item insertion. But it is best to simply initialize them as 0 and 0.

Say you enqueue the first item. You would be putting the first item into data[-1] , which isn't ok.

Another thing, when you dequeue, you should increment before you insert the data for good practice. For this code, the rear index doesnt actually point to the rear, it points to the array index after the rear, which can be confusing.

Reply



Gaia1956 (<https://www.blogger.com/profile/01001152173646370129>) October 10, 2016 at 8:03 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1476109981093#c8164825506890793446>)

This comment has been removed by a blog administrator.

Reply



Gaia1956 (<https://www.blogger.com/profile/01001152173646370129>) October 10, 2016 at 8:03 PM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1476109990215#c8426594116855914525>)

This comment has been removed by a blog administrator.

Reply



chenmeinvo (<https://www.blogger.com/profile/04850790224822937115>) December 29, 2016 at 11:57 AM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1482992876877#c671406355952894316>)

the north face outlet (<http://www.thenorthface.net.co>)

nfl jerseys cheap (<http://www.cheapnfljerseys-wholesale.us.com>)

canada goose sale (<http://www.outletcanadagoose.com.co>)

cheap ray ban sunglasses (<http://www.raybansonsale.com.co>)

michael kors outlet clearance (<http://www.michaelkorsonlineoutlet.us.org>)

burberry scarf (<http://www.burberyyoutlet-online.net.co>)

toms shoes outlet (<http://www.toms-shoes.cc>)

tod's shoes online (<http://www.todsoutlet.in.net>)

kate spade outlet (<http://www.katespadeoutlets.us.org>)

mbt shoes clearance (<http://www.mbt.us.com>)

2016.12.29xukaimin

Reply



Jian Zhuo (<https://www.blogger.com/profile/06468977099257039433>) January 11, 2017 at 9:02 AM (<http://www.sourcetricks.com/2008/07/c-queues.html?showComment=1484105524938#c8040222524398941428>)

designer handbags (<http://www.designerhandbags.net.co>)

toms shoes (<http://www.tomsshoes.us.com>)

louboutin uk (<http://www.christianlouboutinshoes.org.uk>)

true religion outlet (<http://www.coachoutletonlinecoachfactoryoutlet.us.com>)

coach outlet (<http://www.coachfactoryoutletsaleonline.us.com>)

longchamp bags (<http://www.longchampbags.uk>)

ugg sale (<http://www.uggsale.us.com>)

yeezy boost 350 (<http://www.yeezyboost-350.us.com>)

wholesale nike shoes (<http://www.discountnikeshoessc.org>)

coach factory outlet (<http://www.coachfactoryonlineoutlet.com.co>)

20170111

Reply

Enter your comment...

Comment as: Unknown (Goo ▼)

Publish Preview

Sign out

☐ Notify me


(<https://www.blogger.com/comment-iframe.g?blogID=7748177500667831327&postID=5599041188023099254&blogspotRpcToken=858827>)

Tutorial Pages

- Java Tutorials (<http://www.sourcetricks.com/p/java-tutorials.html>)
- Micro Services (<http://www.sourcetricks.com/p/microservices.html>)
- Java Interview Questions (<http://www.sourcetricks.com/2014/06/core-java-interview-questions.htm> l)
- Scala Tutorials (<http://www.sourcetricks.com/p/scala-tutorials.html>)
- Programming in C++ (<http://www.sourcetricks.com/p/programming-in-c.html>)
- Programming interview questions and answers in C++ (<http://www.sourcetricks.com/p/programming-interview-questions-and.ht> ml)
- Data Structures using C++ (<http://www.sourcetricks.com/p/data-structures-using-c.html>)
- Algorithms in C++ (<http://www.sourcetricks.com/p/algorithms-in-c.html>)
- Design Patterns using C++ (<http://www.sourcetricks.com/p/design-patterns-using-c.html>)
- Android (<http://www.sourcetricks.com/p/android.html>)
- HTML, CSS and Javascript (<http://www.sourcetricks.com/p/html-css-and-javascript.html>)
- UML Notations (<http://www.sourcetricks.com/2008/05/uml-generalization.html>)

Tag Cloud

Java (<http://www.sourcetricks.com/search/label/Java>) CPP
(<http://www.sourcetricks.com/search/label/Cpp>) Programming interview questions and
answers (<http://www.sourcetricks.com/search/label/Programming%20interview%20questions%20and%20answers>)
Design Patterns (<http://www.sourcetricks.com/search/label/Design%20Patterns>) Scala
(<http://www.sourcetricks.com/search/label/Scala>) Android (<http://www.sourcetricks.com/search/label/Android>) Algorithms
(<http://www.sourcetricks.com/search/label/Algorithms>) Data Structures (<http://www.sourcetricks.com/search/label/Data%20Structures>) Micro Services
(<http://www.sourcetricks.com/search/label/Micro%20Services>) JavaScript (<http://www.sourcetricks.com/search/label/JavaScript>) tools
(<http://www.sourcetricks.com/search/label/tools>) UML (<http://www.sourcetricks.com/search/label/UML>)html (<http://www.sourcetricks.com/search/label/html>)




Sourcetricks
572 likes

Like Page

Share

Be the first of your friends to like this



Follow @sourcetricks 52 followers

Popular Posts

C++ Pre-Order, In-Order, Post-Order Traversal of Binary Search Trees (<http://www.sourcetricks.com/2011/05/c-pre-order-in-order-post-order.html>)

Pre-Order, In-Order, Post-Order traversal of Binary Search Trees (BST) This article explains the depth first search (DFS) traversal meth...

C++ Singly Linked Lists (<http://www.sourcetricks.com/2008/07/c-singly-linked-lists.html>)

What is a singly linked list? How to implement singly linked lists in C++? This article explains the concept of singly linked list data ...

C++ Queues (<http://www.sourcetricks.com/2008/07/c-queues.html>)

What is a queue? How to implement queues using C++? This article explains the queue data structure and demonstrates sample implementati...


C++ Tries (<http://www.sourcetricks.com/2011/06/c-tries.html>)



What is a Trie? How to implement Tries in C++? This article explains the Trie data structure and provides a sample implementation in C++....

C++ Heaps (<http://www.sourcetricks.com/2011/06/c-heaps.html>)

What is a heap data structure? How to implement in C++? This article explains the heap data structure and provides a sample implementat...

Subscribe To

 **Posts** 

 **Comments** 

Contact Us (<http://www.sourcetricks.com/p/contact.html>)

Follow by Email

Email address...

Submit

Copyright © 2017 Programming Tutorials by SourceTricks (<http://www.sourcetricks.com/>)

[Back to Top](#)