TITLE

AUTHOR
Version
CREATEDATE

Table of Contents

Table of contents

Hierarchical Index

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

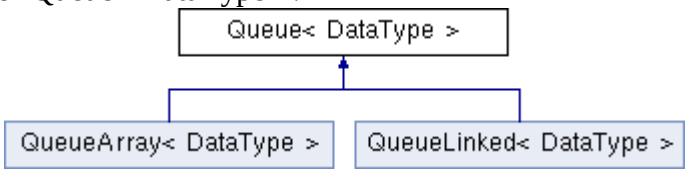
File List

Here is a list of all documented files with brief descriptions:

Class Documentation

Queue< DataType > Class Template Reference

Inheritance diagram for Queue< DataType >:



Public Member Functions

- virtual void **enqueue** (const DataType &newDataItem)=0 throw (logic_error)
- virtual DataType **dequeue** ()=0 throw (logic_error)
- virtual void **clear** ()=0
- virtual bool **isEmpty** () const =0
- virtual bool **isFull** () const =0
- virtual void **showStructure** () const =0

Static Public Attributes

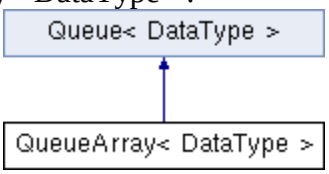
- static const int **MAX_QUEUE_SIZE** = 8

The documentation for this class was generated from the following files:

- Queue.h
- show7.cpp

QueueArray< DataType > Class Template Reference

Inheritance diagram for QueueArray< DataType >:



Public Member Functions

- **QueueArray** (int maxNumber=**Queue**< DataType >::MAX_QUEUE_SIZE)
- **QueueArray** (const **QueueArray** &other)
- **QueueArray** & **operator=** (const **QueueArray** &other)
- void **enqueue** (const DataType &newDataItem) throw (logic_error)
- DataType **dequeue** () throw (logic_error)
- void **clear** ()
- bool **isEmpty** () const
- bool **isFull** () const
- void **putFront** (const DataType &newDataItem) throw (logic_error)
- DataType **getRear** () throw (logic_error)
- int **getLength** () const
- void **showStructure** () const

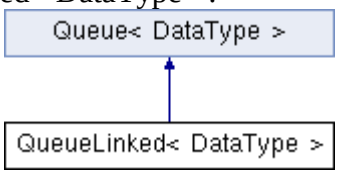
Additional Inherited Members

The documentation for this class was generated from the following files:

- QueueArray.h
- show7.cpp

QueueLinked< DataType > Class Template Reference

Inheritance diagram for QueueLinked< DataType >:



Public Member Functions

- **QueueLinked** (int maxNumber=**Queue**< DataType >::MAX_QUEUE_SIZE)
- **QueueLinked** (const **QueueLinked** &other)
- **QueueLinked** & **operator=** (const **QueueLinked** &other)
- **~QueueLinked** ()
- void **enqueue** (const DataType &newDataItem) throw (logic_error)
- DataType **dequeue** () throw (logic_error)
- void **clear** ()
- bool **isEmpty** () const
- bool **isFull** () const
- void **putFront** (const DataType &newDataItem) throw (logic_error)
- DataType **getRear** () throw (logic_error)
- int **getLength** () const
- void **showStructure** () const

Additional Inherited Members

Constructor & Destructor Documentation

template<class DataType > QueueLinked< DataType >::QueueLinked (int *maxNumber* = `Queue<DataType>::MAX_QUEUE_SIZE`)

Method implementations This constructor creates an empty queue.

Parameters:

int	Max Number of data items this queue can hold

Returns:

This constructor does not return anything

Precondition:

None

Postcondition:

Creates an empty queue. Allocates enough memory for the queue containing maxNumber data items if necessary)

template<class DataType > QueueLinked< DataType >::QueueLinked (const QueueLinked< DataType > & *other*)

This copy constructor will make a deep copy of another queue. Initializes front and back to NULL, then uses the enqueue method to add items from the other queue.

Parameters:

QueueLinked&	Linked **Queue** to copy

Returns:

This copy constructor does not return anything

Precondition:

None

Postcondition:

Initializes the queue to be equivalent to the other **Queue** object paramter

template<class DataType > QueueLinked< DataType >::~QueueLinked ()

Destructor. Deallocates the memory used to store the queue. Calls the clear method.

Returns:

This destructor does not return anything

Precondition:

None

Postcondition:

Deallocates the memory used to store the queue.

Member Function Documentation

template<class DataType > void QueueLinked< DataType >::clear () `[virtual]`

Deletes all items in the queue, freeing memory and making the queue empty.

Returns:

This method returns void.

Precondition:

None.

Postcondition:

This queue will be empty

Implements **Queue**< **DataType** > (*p.5*).

template<class DataType > DataType QueueLinked< DataType >::dequeue () throw logic_error) `[virtual]`

If the queue is not empty, remove the first item in the queue (from the front) and return it. Otherwise, an error is thrown.

Returns:

DataType The first item in the queue

Precondition:

Queue is not empty

Postcondition:

Removes the least recently added data item from the queue and returns it.

Implements **Queue**< **DataType** > (*p.5*).

template<class DataType > void QueueLinked< DataType >::enqueue (const DataType & *newDataItem*) throw logic_error) `[virtual]`

Inserts a new item to the back of the queue.

Parameters:

DataType&	Data to be inserted into the queue

Returns:

This function is void

Precondition:

Queue is not full

Postcondition:

Inserts newDataItem at the rear of the queue

Implements **Queue**< **DataType** > (*p.5*).

template<class DataType > int QueueLinked< DataType >::getLength () const

Counts and returns the number of items in the queue.

Returns:

int Returns the number of items in the queue

Precondition:

None

Postcondition:

The order and items in the queue will be unchanged.

template<class DataType > DataType QueueLinked< DataType >::getRear () throw logic_error)

Removes the last item from the queue and returns it.

Returns:

DataType Returns the last item in the queue

Precondition:

Queue is not empty.

Postcondition:

Removes the last item from the queue and returns it.

template<class DataType > bool QueueLinked< DataType >::isEmpty () const `[virtual]`

Returns true if the queue is empty. Otherwise, returns false.

Returns:

bool Returns true if the queue has no items in it.

Precondition:

None

Postcondition:

The queue will be unchanged

Implements **Queue**< **DataType** > (*p.5*).

template<class DataType > bool QueueLinked< DataType >::isFull () const `[virtual]`

Returns true if the queue is full. Otherwise, returns false.

Parameters:

	return bool Always returns false.

Precondition:

None.

Postcondition:

The queue will be unchanged

Implements **Queue**< **DataType** > (*p.5*).

template<class DataType > QueueLinked< DataType > & QueueLinked< DataType >::operator= (const QueueLinked< DataType > & *other*)

Overloaded assignment operator. Creates a deep copy of another linked queue by clearing the current list and then using the enqueue method to insert new items into the list.

Parameters:

QueueLinked&	Other linked queue that this queue will be a copy of

Returns:

A reference to the modified queue

Precondition:

None

Postcondition:

Sets the queue to be equivalent to the other **Queue** object parameters and returns a reference to the modified queue

template<class DataType > void QueueLinked< DataType >::putFront (const DataType & *newDataItem*) throw logic_error)

Inserts a new data item to the front of the queue without changing the order of the original queue.

Parameters:

DataType&	A new item to add to the queue

Returns:

This function returns void.

Precondition:

Queue is not full

Postcondition:

newDataItem will be inserted at the front of the queue

template<typename DataType > void QueueLinked< DataType >::showStructure () const `[virtual]`

Outputs the data items in the queue. If the queue is empty, outputs "Empty Queue". This operation is intended only for testing/debugging purposes, and only supports queue data items that are one of C++'s predefined data types (int, char, etc) or other data structures with an overriden ostream operator<<.

Returns:

This function returns void

Precondition:

None

Postcondition:

The order and items in the queue will be unchanged.

Implements **Queue**< **DataType** > (*p.5*).

The documentation for this class was generated from the following files:

- QueueLinked.h
- **QueueLinked.cpp**

File Documentation

QueueLinked.cpp File Reference

This program will implement a Linked **Queue**.
`#include "QueueLinked.h"`

Detailed Description

This program will implement a Linked **Queue**.

Author:
Tim Kwist

Version:
1.0

The specifications of this program are defined by C++ Data Structures: A Laboratory Course (3rd edition) by Brandle, JGeisler, Roberge, Whittington, exercise 7. This implementation only implements the Linked List version of the **Queue**.

Date:
Wednesday, September 10, 2014

Index

INDEX