Project 7 - Catherine Pollock

Generated by Doxygen 1.7.6.1

Tue Sep 9 2014 17:19:47

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Queue< DataType > . . . . . **??**
QueueLinked< DataType > . . . . . **??**

Chapter 2

Class Index

2.1 Class List

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

Queue< DataType > . . . . . ??
QueueLinked< DataType > . . . . . ??

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

config.h . . . . . ??
Queue.h . . . . . ??
QueueLinked.cpp . . . . . ??
QueueLinked.h . . . . . ??
show7.cpp . . . . . ??
storesim.cpp . . . . . ??
storesim.cs . . . . . ??
test7.cpp . . . . . ??
test7catherine.cpp . . . . . ??

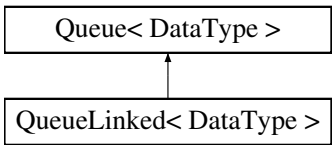
Chapter 4

Class Documentation

4.1 Queue< DataType > Class Template Reference

`#include <Queue.h>`

Inheritance diagram for Queue< DataType >:



Public Member Functions

- virtual ~Queue ()
- 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

template<typename DataType> class Queue< DataType >

4.1.1 Constructor & Destructor Documentation

4.1.1.1 template<typename DataType > Queue< DataType >::~Queue ( ) [virtual]

4.1.2 Member Function Documentation

4.1.2.1 template<typename DataType > virtual void Queue< DataType >::clear ( ) [pure virtual]

Implemented in QueueLinked< DataType >.

4.1.2.2 template<typename DataType > virtual DataType Queue< DataType >::dequeue ( ) throw (logic_error) [pure virtual]

Implemented in QueueLinked< DataType >.

4.1.2.3 template<typename DataType > virtual void Queue< DataType >::enqueue (const DataType & *newDataItem*) throw (logic_error) [pure virtual]

Implemented in QueueLinked< DataType >.

4.1.2.4 template<typename DataType > virtual bool Queue< DataType >::isEmpty ( ) const [pure virtual]

Implemented in QueueLinked< DataType >.

4.1.2.5 template<typename DataType > virtual bool Queue< DataType >::isFull ( ) const [pure virtual]

Implemented in QueueLinked< DataType >.

4.1.2.6 template<typename DataType > virtual void Queue< DataType >::showStructure ( ) const [pure virtual]

Implemented in QueueLinked< DataType >.

4.1.3 Member Data Documentation

4.1.3.1 template<typename DataType > const int Queue< DataType >::MAX_QUEUE_SIZE = 8 [static]

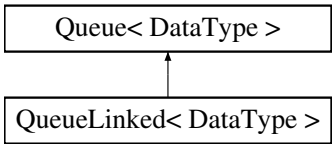
The documentation for this class was generated from the following file:

- Queue.h

4.2 QueueLinked< DataType > Class Template Reference

`#include <QueueLinked.h>`

Inheritance diagram for QueueLinked< DataType >:



Classes

- class **QueueNode**

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

template<typename DataType> class QueueLinked< DataType >

4.2.1 Constructor & Destructor Documentation

4.2.1.1 template<typename DataType > QueueLinked< DataType >::QueueLinked (int *maxNumber* = Queue<DataType>::MAX_QUEUE_SIZE)

Default constructor

Creates an empty queue. Sets front and back to null.

Parameters

given	a max number of data items allowed in queue, which is ignored (of type int)

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

Copy constructor

Initiates an QueueLinked object to be equivalent to the other Queue object parameter. It clears all values, sets tempOther to other's front, and makes a new node for this queue with tempOther's data and null for next. Then, it loops through until tempOther's next value is null and continues to set data values, copied from other.

Parameters

const	QueueLinked other (source for copying values)

Stores null for front and back

Declares temp values to hold

sets tempOther to other's front

sets front value for new queue

loops through other's values

makes a node after back

sets back to new node

advances tempOther

4.2.1.3 template<typename DataType > QueueLinked< DataType >::~QueueLinked ( )

Destructor

Deallocates (frees) the memory used to store the queue. Starts from the front and deletes nodes until the queue is empty. while queue has values, dequeue the values

set pointers to null

4.2.2 Member Function Documentation

4.2.2.1 template<typename DataType > void QueueLinked< DataType >::clear ( ) [virtual]

Clears the queue

Clears the queue of all values. While the queue has values, dequeue them. Then set front and back to null.

Returns

void

while queue has values, dequeue the values

set pointers to null

Implements Queue< DataType >.

4.2.2.2 template<typename DataType > DataType QueueLinked< DataType >::dequeue ( ) throw (logic_error) [virtual]

Dequeue (remove)

Removes the least recently added (front) data item from the queue and returns it. - Throws an exception if the queue is empty. Gets the data from front. If front is the only value in queue, it deletes the front node and sets front and back to null. Else, a temp QueueNode is declared and set to node after front. Then, front is deleted and temp is made the new front. Lastly, the data is returned that was stored in the original front.

Precondition

Queue is not empty

Exceptions

If	queue is empty

Returns

DataType removed from end of queue

If queue is empty, throw exception

gets value of front node

if value is only one in queue

deletes front node and sets front and back to null since it is now empty

delcares temp node to hold front's next value

sets temp to value after head

deletes value at head

makes front original front's next (temp's) node

return data from front

Implements Queue< DataType >.

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

Enqueue (insert)

Inserts newDataItem at the rear of the queue. Throws an exception if queue is full. It sets temp to a new QueueNode with the data newDataItem and the next value as null.

If the queue is empty, front and back are set to temp. Otherwise, back's next value is set to temp and back is advanced to the new node.

Parameters

const	DataType& newDataItem (value given to new node of data)

Precondition

Queue is not full

Exceptions

If	queue is full

Returns

void

If queue is full, throw exception

sets temp as node to be added

checks if queue is empty

sets first value to new node

sets value after the last

advances back

Implements Queue< DataType >.

4.2.2.4 template<typename DataType > int QueueLinked< DataType >::getLength ( ) const

Gets number of values in queue

Sets a counter to zero and a QueueNode temp to front. Advances temp and incriments counter until NULL is reached.

Precondition

queue is not empty

Exceptions

if	queue is empty

Returns

DataType (data value from end)

4.2.2.5 template<typename DataType > DataType QueueLinked< DataType >::getRear ( ) throw (logic_error)

Gets the data item from the end of the queue

Checks if the queue is empty. If it is, throw an exception. Else, save the data at rear. If it is the only value in queue, clear the queue. Otherwise, loop through from front until one before back. Delete back, set the temp to back and set back's next to null.

Precondition

queue is not empty

Exceptions

if	queue is empty

Returns

DataType (data value from end)

If empty, throw exception

set return value

If only one value, delete and set to null

Otherwise, advance temp to node before back, delete back and set back to temp

return back's data

4.2.2.6 template<typename DataType > bool QueueLinked< DataType >::isEmpty ( ) const [virtual]

Checks if queue is empty

Checks if the queue is empty. If it is, return true. If it has values in it, return false.

Returns

bool (that states if queue is empty)

Returns if the front equals null

Implements Queue< DataType >.

4.2.2.7 template<typename DataType > bool QueueLinked< DataType >::isFull ( ) const [virtual]

Checks if queue is full

Checks if the queue is full. If it is (which will not occur), true is returned. If it is not full, false is returned.

Returns

bool (that states if queue is full)

Returns false because queue cannot be full

Implements Queue< DataType >.

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

Overloaded assignment operator

Sets the queue to be eqivalent to the other Queue object parameter and returns a reference to the modified queue. If other and this are the same queue, the function does not copy. Otherwise, it clears all values, sets tempOther to other's front, and makes a new node for this queue with tempOther's data and null for next. Then, it loops through until tempOther's next value is null and continues to set data values, copied from other.

Parameters

const	QueueLinked other (source for copying values)

checks if they are not the same queue

Clears all values from queue

Declares temp values to hold

sets tempOther to other's front

sets front value for new queue

loops through other's values

returns this queue

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

Puts data item at front of queue

Checks if the queue is full. If it is, throw an exception. If it is not, check if queue is empty. If it is enqueue like normal. Otherwise, set temp to front, and make the new front a node with newDataItem and temp as next.

Parameters

const	DataType& newDataItem (data value to put at front of queue)

Precondition

queue is not full

Exceptions

if	queue is full

Returns

void

throws exception if queue is full

checks if queue is empty

otherwise, set temp to front and make a new front with data and temp as next

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

Outputs the data items in queue

Outputs the elements in a queue. If the queue is empty, outputs "Empty queue". This operation is intended for testing and debugging purposes only.

Returns

void

Iterates through the queue

Implements Queue< DataType >.

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

- QueueLinked.h
- QueueLinked.cpp
- show7.cpp

Chapter 5

File Documentation

5.1 config.h File Reference

Defines

- #define LAB7_TEST1 1
- #define LAB7_TEST2 1
- #define LAB7_TEST3 1

5.1.1 Define Documentation

5.1.1.1 #define LAB7_TEST1 1

Queue class (Lab 7) configuration file. Activate test #N by defining the corresponding LAB7_TESTN to have the value 1.

5.1.1.2 #define LAB7_TEST2 1

5.1.1.3 #define LAB7_TEST3 1

5.2 Queue.h File Reference

`#include <stdexcept> #include <iostream>`

Classes

- class Queue< DataType >

5.3 QueueLinked.cpp File Reference

```
#include <stdexcept> #include <iostream> #include "Queue-
Linked.h"
```

5.3.1 Detailed Description

Author

CatherinePollock

Date

9/8/14

This is the implementation file for the QueueLinked.h file.

5.4 QueueLinked.h File Reference

```
#include <stdexcept>   #include <iostream>   #include "-
Queue.h"
```

Classes

- class QueueLinked< DataType >
- class **QueueLinked< DataType >::QueueNode**

5.5 show7.cpp File Reference

5.6 storesim.cpp File Reference

```
#include <iostream> #include <iomanip> #include <cstdlib> ×
#include <ctime>   #include "config.h"   #include "Queue-
Linked.cpp"
```

Functions

- int main ()

5.6.1 Function Documentation

5.6.1.1 int main ( )

5.7 storesim.cs File Reference

```
#include <iostream> #include <iomanip> #include <cstdlib> ×
#include <ctime>   #include "config.h"   #include "Queue-
Linked.cpp"
```

Functions

- int main ()

5.7.1 Function Documentation

5.7.1.1 int main ( )

5.8 test7.cpp File Reference

```
#include <iostream>  #include "config.h"  #include "Queue-
Linked.cpp"
```

Functions

- void print_help ()
- template<typename DataType >
  void test_queue (Queue< DataType > &testQueue)
- int main ()

5.8.1 Function Documentation

5.8.1.1 int main ( )

5.8.1.2 void print_help ( )

5.8.1.3 template<typename DataType > void test_queue (Queue< DataType > & *testQueue*)

5.9 test7catherine.cpp File Reference

```
#include <iostream>  #include "config.h"  #include "Queue-
Linked.cpp"
```

Functions

- void print_help ()
- template<typename DataType >
  void test_queue (Queue< DataType > &testQueue)
- int main ()

5.9.1 Function Documentation

5.9.1.1 int main ( )

5.9.1.2 void print_help ( )

5.9.1.3 template<typename DataType > void test_queue (Queue< DataType > & *testQueue*)