Henry_Huffman_PA_02_Lab_7

Generated by Doxygen 1.7.6.1

Tue Sep 9 2014 21:38:55

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

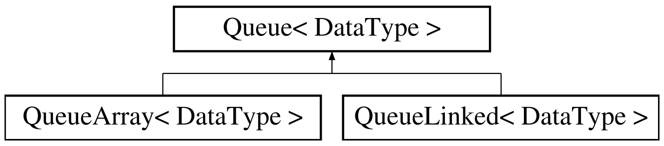
Chapter 4

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 QueueArray< DataType >, and QueueLinked< DataType >.

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

Implemented in QueueArray< DataType >, and QueueLinked< DataType >.

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

Implemented in QueueArray< DataType >, and QueueLinked< DataType >.

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

Implemented in QueueArray< DataType >, and QueueLinked< DataType >.

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

Implemented in QueueArray< DataType >, and QueueLinked< DataType >.

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

Implemented in QueueArray< DataType >, and QueueLinked< DataType >.

4.1.3 Member Data Documentation

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

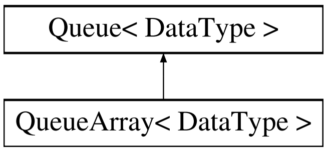
The documentation for this class was generated from the following file:

- Queue.h

4.2 QueueArray< DataType > Class Template Reference

`#include <QueueArray.h>`

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)
- ∼QueueArray ()
- 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 QueueArray< DataType >

4.2.1 Constructor & Destructor Documentation

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

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

4.2.1.3 template<typename DataType > QueueArray< DataType >::∼QueueArray ( )

4.2.2 Member Function Documentation

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

Implements Queue< DataType >.

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

Implements Queue< DataType >.

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

Implements Queue< DataType >.

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

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

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

Implements Queue< DataType >.

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

Implements Queue< DataType >.

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

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

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

Implements Queue< DataType >.

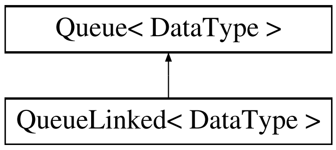
The documentation for this class was generated from the following files:

- QueueArray.h
- show7.cpp

4.3 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 >

4.3.1 Constructor & Destructor Documentation

4.3.1.1 template<typename DataType > QueueLinked< DataType >::QueueLinked (int *ignore* = Queue<DataType>::MAX_QUEUE_SIZE)

Default queue constructor

This function will initialize the queue. The queue will then be used throughout the rest of the program to hold data specified by the user. It will also be used to simulate people waiting in a line.

Parameters

ignore	- this is a variable that is not to be used in functions implementations

Returns

none

Exceptions

none	

Precondition

the queue will be uninitialized

Postcondition

the node will be initialized

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

Copy Constructor

This constructor is called when a user wants to copy another queue. Moreover, this copy constructor uses the overloaded assignement operator to copy the other queue.

Parameters

other	- this is the queue that is going to be copied

Returns

none

Exceptions

none	

Precondition

the queue will be uninitialized

Postcondition

the queue will be initialized with the identical data and structure as the other queue.

4.3.1.3 template<typename DataType > **QueueLinked**< DataType >::∼**QueueLinked** ( )

Deconstructor

The deconstructor deallocates space that was allocated to the queue. This deconstructor calls the clear function because its will remove all the nodes of the specified queue. It will then set the front and back pointers to null.

Parameters

none	

Returns

none

Exceptions

noen	

Precondition

there will be a queue with allocated memory

Postcondition

the memory of the queue will be deallocated and pointers will be set to null

4.3.2 Member Function Documentation

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

clear operator

This function is called when the user wished to get rid of all the data stored in current queue. If the queue is not empty, the clear function will move through each node and deallocate the memory in use. Finally, it will set front and back pointers to null.

Parameters

none	

Returns

none

Exceptions

none	

Precondition

a queue with or without data

Postcondition

a queue without any data

Implements Queue< DataType >.

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

dequeue function

This function is used to remove the least recently added item from the queue. In the line simulation, this function is used to remove the customer in line and retrieve their arrival time. In this function, if there is nothing to remove, a error message will be thrown. If the only item of a queue is removed, it will set the pointers to null; otherwise, only the first item in the queue will be removed.

Parameters

none	

Returns

DataType - this function will return whatever data is in the first node of the queue.

Exceptions

this	function will not work if the queue is empty

Precondition

a non-empty queue

Postcondition

a queue with one less data item in it, possibly empty. The data in removed item is returned.

Implements Queue< DataType >.

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

enqueue function

The purpose of this function is to add a node with new data to the back of the queue. It is used in the line simulation to add customers with their arrival time to the back of the line. This function will check to see if the queue is full. If full, it will throw an error message. Otherwise, The function will check to see if it is empty and add the data accordingly. Because this is a void function, nothing will be returned.

Parameters

newData-Item	- the data of the new node that is added to the back of the queue

Returns

none

Exceptions

queue	can not be full

Precondition

a queue with or without data in it

Postcondition

a queue with atleast one item of data in it or a new data item added to the back of the queue

Implements Queue< DataType >.

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

getLength function

This function determines the number of items in the queue.

Parameters

none	

Returns

int - an integer matching the number of items in the queue

Exceptions

none	

Precondition

a queue

Postcondition

the number of items in the queue is returned

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

getRear

This function will get the data from the most recently added item of the queue, then it will deallocate it. If the queue is full it will throw an error message.

Parameters

none	

Returns

DataType - returns the data that was in the last item in the queue

Exceptions

the	queue can not be empty

Precondition

a queue with atleast one data item in it

Postcondition

a queue with one less data item in it, and the value of the data item is returned

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

Empty function

This function checks to see if a queue is empty. It is essential becuase it is used frequently throughout the rest of the implementations. If there is data in the queue, it will return false. If empty it will return true.

Parameters

none	

Returns

bool - it returns a boolean with true if it is empty, and false if it has data

Exceptions

none	

Precondition

a queue that may or may not be empty

Postcondition

a boolean with a value that determines whether or not a queue is empty

Implements Queue< DataType >.

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

isFull function

This function checks to see if a queue is full or not. Because I currently do not run the possiblity of running out of memory, this funciton will always return false.

Parameters

none	

Returns

bool - a boolean with the value of false because I will not run out of memory while running this program.

Exceptions

none	

Precondition

a queue with or without data in it

Postcondition

a boolean set to false because the queue will never be full

Implements Queue< DataType >.

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

Overloaded assignment operator

This operator is called when a linked queue is being assigned to another linked queue. The purpose of this function is to get an identical copy of the assigned queue.

Parameters

other	- the queue that is to be copied

Returns

(∗this) - it returns the current queue (the new copy of the queue)

Exceptions

this	function will not work if a queue is trying to assign itself

Precondition

two queues will be initialized and one will be assigned to the other

Postcondition

the data and structure of the two queues will be identical

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

putFront function

This function will take new data and place it at the front of the queue. This function will add a node to the queue if it is empty. If the queue is not empty, a new node will be added to the beginning of current queue with changing the remainder of the queue

Parameters

newData-Item	- the data that will be placed in the new node

Returns

none

Exceptions

none	

Precondition

a queue with or without data

Postcondition

a queue with a new data item added to the beginning of it

4.3.2.10 template<typename DT > void QueueLinked< DT >::showStructure ( ) const [virtual]

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

test various capabilities of program changes the implementation from an array based queue to a linked queue also enables the putFront, getRear, and getLength functions

Defines

- #define LAB7_TEST1 1

  changed configuration to test full functionality of program
- #define LAB7_TEST2 1
- #define LAB7_TEST3 1

5.1.1 Detailed Description

test various capabilities of program changes the implementation from an array based queue to a linked queue also enables the putFront, getRear, and getLength functions Queue class (Lab 7) configuration file. Activate test #N by defining the corresponding LAB7_TESTN to have the value 1.

Version

1.1

Date

Tuesday, September 08, 2014

5.1.2 Define Documentation

5.1.2.1 #define LAB7_TEST1 1

changed configuration to test full functionality of program

5.1.2.2 #define LAB7_TEST2 1

5.1.2.3 #define LAB7_TEST3 1

5.2 Queue.h File Reference

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

Classes

- class Queue< DataType >

5.3 QueueArray.h File Reference

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

Classes

- class QueueArray< DataType >

5.4 QueueLinked.cpp File Reference

```
#include "QueueLinked.h"
```

5.5 QueueLinked.h File Reference

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

Classes

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

5.6 show7.cpp File Reference

5.7 storesim.cpp File Reference

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

5.7.1 Detailed Description

5.8 storesim.cs File Reference

`#include <iostream>` `#include <iomanip>` `#include <cstdlib>` ×
`#include <ctime>` `#include "QueueArray.cpp"`

Functions

- int main ()

5.8.1 Function Documentation

5.8.1.1 int main ( )

5.9 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 ()

5.9.1 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*)