PA09-L11 Catherine Pollock

Generated by Doxygen 1.7.6.1

Tue Nov 4 2014 22:50:03

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 Greater< KeyType > Class Template Reference

Public Member Functions

- bool operator() (const KeyType &a, const KeyType &b) const

template<typename KeyType = int> class Greater< KeyType >

4.1.1 Member Function Documentation

4.1.1.1 template<typename KeyType = int> bool Greater< KeyType >::operator() (const KeyType & *a*, const KeyType & *b*) const [inline]

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

- test11.cpp

4.2 Heap< DataType, KeyType, Comparator > Class Template - Reference

`#include <Heap.h>`

Public Member Functions

- Heap (int maxNumber=DEFAULT_MAX_HEAP_SIZE)
- Heap (const Heap &other)
- Heap & operator= (const Heap &other)
- ~Heap ()

- void insert (const DataType &newDataItem) throw (logic_error)
- DataType remove () throw (logic_error)
- void clear ()
- bool isEmpty () const
- bool isFull () const
- void showStructure () const
- void writeLevels () const

Static Public Attributes

- static const int DEFAULT_MAX_HEAP_SIZE = 10

Private Member Functions

- void showSubtree (int index, int level) const

Private Attributes

- int maxSize
- int size
- DataType * dataItems
- Comparator comparator

template<typename DataType, typename KeyType = int, typename Comparator = Less<KeyType>> class Heap< DataType, KeyType, Comparator >

4.2.1 Constructor & Destructor Documentation

4.2.1.1 template<typename DataType , typename KeyType , typename Comparator > Heap< DataType, KeyType, Comparator >::Heap (int *maxNumber* = DEFAULT_MAX_HEAP_SIZE)

default constructor

Creates an empty hash table of size maxNumber with data type DataType.

Parameters

maxNumber	int of table size

initialize variables/data

4.2.1.2 template<typename DataType , typename KeyType , typename Comparator > Heap< DataType, KeyType, Comparator >::Heap (const Heap< DataType, KeyType, Comparator > & *other*)

copy constructor

Initializes the heap to be equivalent to the other heap parameter.

Parameters

other	reference to a heap to be copied from

call operator =

4.2.1.3 template<typename DataType , typename KeyType , typename Comparator > Heap< DataType, KeyType, Comparator >::~Heap ( )

destructor

Dellocates (frees) the memory used to store the heap. calls clear to delete data in heap

4.2.2 Member Function Documentation

4.2.2.1 template<typename DataType , typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::clear ( )

clear

Removes all data items in the heap. while heap still has data

remove root value

4.2.2.2 template<typename DataType, typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::insert (const DataType & *newDataItem*) throw (logic_error)

insert

Inserts newDataItem into the heap. Inserts this data item as the bottom right most data item in the heap and moves it upward until the properties that define a heap are restored.

Parameters

newDataItem	reference to the data to be inserted

Precondition

heap is not full

Exceptions

logic_error	if heap is full

initialize variables

if the heap is full, throw exception

set the last item's data to passed data

incriment size of heap

until the children are no longer larger than parent

set temp's data

set parent to current child's data

set child to parent's (temp's) data

set a new index (parent) and repeat

4.2.2.3 template<typename DataType , typename KeyType , typename Comparator > bool Heap< DataType, KeyType, Comparator >::isEmpty ( ) const

isEmpty

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

Returns

bool if heap is empty or not

return if heap is empty

4.2.2.4 template<typename DataType , typename KeyType , typename Comparator > bool Heap< DataType, KeyType, Comparator >::isFull ( ) const

isFull

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

Returns

bool if heap is full or not

return if heap is full

4.2.2.5 template<typename DataType , typename KeyType , typename Comparator > Heap< DataType, KeyType, Comparator > & Heap< DataType, KeyType, Comparator >::operator= (const Heap< DataType, KeyType, Comparator > & *other*)

assignment operator

Sets the heap to be equivalent to the other heap parameter and returns a reference to this object.

Parameters

other	Heap reference to a heap to be copied from

Returns

Heap& reference to this heap

if the address of other is not equal to this

clear this

initialize variables/data

loop through each value and copy data from other

return dereferenced this

4.2.2.6 template<typename DataType , typename KeyType , typename Comparator > DataType Heap< DataType, KeyType, Comparator >::remove ( ) throw (logic_error)

remove

Removes the data item with the highest priority (the root) from the heap and returns it. Replaces the root data item with the bottom rightmost data item and moves this data item downward until the properties that define a heap are restored.

Precondition

heap is not empty

Exceptions

logic_error	if heap is empty

Returns

DataType containing data removed from root value

throw logic error if empty heap

initialize variables

decrement heap size

set data to be returned

set root to have bottom rightmost value

while parent index is greater than total number of data items

if two children to check

test if left larger and set largestIndex

test if right larger and set largestIndex

if a child was greater than parent

swap data

change parent index to check

if parent was largest

return stored data

if only left child to check

test if left larger

swap data

change parent index to check

if parent was largest

return stored data

if no children to check

return stored data

return stored data

4.2.2.7 template<typename DataType , typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::showStructure ( ) const

showStructure

Outputs the priorities of the data items in a heap in both array and tree form. If the heap is empty, outputs "Empty heap". This operation is intended for testing/debugging purposes only. Loop counter

Output array form

Output tree form

4.2.2.8 template<typename DataType , typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::showSubtree (int *index,* int *level*) const [private]

showSubtree

Helper function for the showStructure() function. Outputs the subtree (subheap) whose root is stored in dataItems[index]. Argument level is the level of this dataItems within the tree.

Parameters

index	(int) current index at
level	(int) current level at

Output right subtree

Tab over to level

Output dataItems's priority

Output "connector"

Output left subtree

4.2.2.9 template<typename DataType , typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::writeLevels ( ) const

writeLevels

Outputs the data items in a heap in level order, one level per line. Only outputs each data item's priority. If the heap is empty, then outputs "Empty heap". if heap is empty, print empty heap

otherwise loop though each data member and print

print endline if complete level has been printed

4.2.3 Member Data Documentation

4.2.3.1 template<typename DataType, typename KeyType = int, typename Comparator = Less<KeyType>> Comparator Heap< DataType, KeyType, Comparator >::comparator [private]

4.2.3.2 template<typename DataType, typename KeyType = int, typename Comparator = Less<KeyType>> DataType* Heap< DataType, KeyType, Comparator >::dataItems [private]

4.2.3.3 template<typename DataType, typename KeyType = int, typename Comparator = Less<KeyType>> const int Heap< DataType, KeyType, Comparator >::DEFAULT_MAX_HEAP_SIZE = 10 [static]

4.2.3.4 template<typename DataType, typename KeyType = int, typename Comparator = Less<KeyType>> int Heap< DataType, KeyType, Comparator >::maxSize [private]

4.2.3.5 template<typename DataType, typename KeyType = int, typename Comparator = Less<KeyType>> int Heap< DataType, KeyType, Comparator >::size [private]

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

- Heap.h
- Heap.cpp
- show11.cpp

4.3 Less< KeyType > Class Template Reference

`#include <Heap.h>`

Public Member Functions

- bool operator() (const KeyType &a, const KeyType &b) const

template<typename KeyType = int> class Less< KeyType >

4.3.1 Member Function Documentation

4.3.1.1 template<typename KeyType = int> bool Less< KeyType >::operator() (const KeyType & *a*, const KeyType & *b*) const [inline]

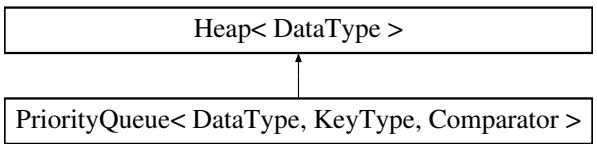
The documentation for this class was generated from the following file:

- Heap.h

4.4 PriorityQueue< DataType, KeyType, Comparator > Class - Template Reference

`#include <PriorityQueue.h>`

Inheritance diagram for PriorityQueue< DataType, KeyType, Comparator >:



Public Member Functions

- PriorityQueue (int maxNumber=defMaxQueueSize)
- void enqueue (const DataType &newDataItem)
- DataType dequeue ()

template<typename DataType, typename KeyType = int, typename Comparator = Less<KeyType>> class PriorityQueue< DataType, KeyType, Comparator >

4.4.1 Constructor & Destructor Documentation

4.4.1.1 template<typename DataType , typename KeyType , typename Comparator > PriorityQueue< DataType, KeyType, Comparator >::PriorityQueue (int *maxNumber* = defMaxQueueSize)

default constructor

Creates an empty priority queue. Allocates enough memory for a queue containing maxNumber data items.

Parameters

maxNumber	int of the max queue size

4.4.2 Member Function Documentation

4.4.2.1 template<typename DataType , typename KeyType , typename Comparator > DataType PriorityQueue< DataType, KeyType, Comparator >::dequeue ( )

dequeue

Removes the highest priority (front) data item from the priority queue and returns it.

Returns

DataType with data removed

Precondition

queue is not empty

Exceptions

logic_error	thrown if queue is empty

4.4.2.2 template<typename DataType , typename KeyType , typename Comparator > void PriorityQueue< DataType, KeyType, Comparator >::enqueue (const DataType & *newDataItem*)

enqueue

Inserts newDataItem into the priority queue.

Parameters

newDataItem	(const DataType&) of new data to queue

Precondition

queue is not full

Exceptions

logic_error	thrown if queue is full

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

- PriorityQueue.h
- PriorityQueue.cpp

4.5 TaskData Struct Reference

Declaration for the task data struct.

Public Member Functions

- int getPriority () const

Public Attributes

- int priority

  Returns the priority. Needed by the heap.
- int arrived

  Task's priority.

4.5.1 Detailed Description

Declaration for the task data struct.

4.5.2 Member Function Documentation

4.5.2.1 int TaskData::getPriority ( ) const [inline]

4.5.3 Member Data Documentation

4.5.3.1 int TaskData::arrived

Task's priority.

4.5.3.2 int TaskData::priority

Returns the priority. Needed by the heap.

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

- ossim.cpp

4.6 TestData Class Reference

Public Member Functions

- void setPriority (int newPriority)
- int getPriority () const
- void setPriority (int newPriority)
- int getPriority () const

Private Attributes

- int priority

4.6.1 Member Function Documentation

4.6.1.1 int TestData::getPriority ( ) const [inline]

4.6.1.2 int TestData::getPriority ( ) const [inline]

4.6.1.3 void TestData::setPriority (int *newPriority*) [inline]

4.6.1.4 void TestData::setPriority (int *newPriority*) [inline]

4.6.2 Member Data Documentation

4.6.2.1 int TestData::priority [private]

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

- test11hs.cpp
- test11pq.cpp

4.7 TestDataItem< KeyType > Class Template Reference

Public Member Functions

- TestDataItem ()

- void setPriority (KeyType newPty)
- KeyType getPriority () const

Private Attributes

- KeyType priority

template<typename KeyType> class TestDataItem< KeyType >

4.7.1 Constructor & Destructor Documentation

4.7.1.1 template<typename KeyType > TestDataItem< KeyType >::TestDataItem ( ) [inline]

4.7.2 Member Function Documentation

4.7.2.1 template<typename KeyType > KeyType TestDataItem< KeyType >::getPriority ( ) const [inline]

4.7.2.2 template<typename KeyType > void TestDataItem< KeyType >::setPriority (KeyType *newPty*) [inline]

4.7.3 Member Data Documentation

4.7.3.1 template<typename KeyType > KeyType TestDataItem< KeyType >::priority [private]

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

- test11.cpp

Chapter 5

File Documentation

5.1 config.h File Reference

Defines

- #define LAB11_TEST1 1

5.1.1 Define Documentation

5.1.1.1 #define LAB11_TEST1 1

Heap class configuration file. Activate test #N by defining the corresponding LAB11_TESTN to have the value 1.

5.2 Heap.cpp File Reference

```
#include <stdexcept> #include <iostream> #include <cmath> ×
#include "Heap.h"
```

5.2.1 Detailed Description

Author

CatherinePollock

Date

11/5/14

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

5.3 Heap.h File Reference

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

Classes

- class Less< KeyType >
- class Heap< DataType, KeyType, Comparator >

5.4 heapsort.cs File Reference

Functions

- template<typename DataType >
  void moveDown (DataType dataItems[], int root, int size)
- template<typename DataType >
  void heapSort (DataType dataItems[], int size)

5.4.1 Function Documentation

5.4.1.1 template<typename DataType > void heapSort (DataType *dataItems[],* int *size*)

5.4.1.2 template<typename DataType > void moveDown (DataType *dataItems[],* int *root,* int *size*)

5.5 ossim.cpp File Reference

`#include <iostream> #include <cstdlib> #include "Priority-Queue.cpp"`

Classes

- struct TaskData

  Declaration for the task data struct.

Functions

- int main ()

5.5.1 Function Documentation

5.5.1.1 int main ( )

Priority queue of tasks

Task

Length of simulation (minutes)

Current minute

Number of priority levels

Number of new tasks arriving

Loop counter

Seed the random number generator

Dequeue the first task in the queue (if any).

Determine the number of new tasks and add them to the queue.

queue one value if 1 or 2 arrivals

enqueue a second value if 2 arrivals

enqueue nothing if 0 or 3 arrivals

5.6 PriorityQueue.cpp File Reference

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

5.6.1 Detailed Description

Author

CatherinePollock

Date

11/5/14

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

5.7 PriorityQueue.h File Reference

```
#include <stdexcept> #include <iostream> #include "Heap.-
cpp"
```

Classes

- class PriorityQueue< DataType, KeyType, Comparator >

Variables

- const int defMaxQueueSize = 10

5.7.1 Variable Documentation

5.7.1.1 const int defMaxQueueSize = 10

5.8 show11.cpp File Reference

5.9 test11.cpp File Reference

```
#include <iostream> #include <string> #include <cctype> ×
#include "Heap.cpp" #include "config.h"
```

Classes

- class TestDataItem< KeyType >
- class Greater< KeyType >

Functions

- void printHelp ()
- int main ()

5.9.1 Function Documentation

5.9.1.1 int main ( )

5.9.1.2 void printHelp ( )

5.10 test11hs.cpp File Reference

```
#include <iostream> #include "heapsort.cpp"
```

Classes

- class TestData

Functions

- int main ()

Variables

- const int MAX_NUM_DATA_ITEMS = 10

5.10.1 Function Documentation

5.10.1.1 int main ( )

5.10.2 Variable Documentation

5.10.2.1 const int MAX_NUM_DATA_ITEMS = 10

5.11 test11pq.cpp File Reference

```
#include <iostream> #include <cctype> #include "Priority-
Queue.cpp"
```

Classes

- class TestData

Functions

- void printHelp ()
- int main ()

5.11.1 Function Documentation

5.11.1.1 int main ( )

5.11.1.2 void printHelp ( )