Heap

1.0

Generated by Doxygen 1.8.8

Wed Nov 5 2014 01:21:19

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Greater< KeyType > . . . . . ??
Heap< DataType, KeyType, Comparator > . . . . . ??
Heap< DataType > . . . . . ??
    PriorityQueue< DataType, KeyType, Comparator > . . . . . ??
Less< KeyType > . . . . . ??
TaskData . . . . . ??
TestData . . . . . ??
TestDataItem< KeyType > . . . . . ??

Chapter 2

Class Index

2.1 Class List

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

Greater< KeyType > . . . . . ??
Heap< DataType, KeyType, Comparator > . . . . . ??
Less< KeyType > . . . . . ??
PriorityQueue< DataType, KeyType, Comparator > . . . . . ??
TaskData . . . . . ??
TestData . . . . . ??
TestDataItem< KeyType > . . . . . ??

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

config.h . . . . . ??
Heap.cpp . . . . . ??
Heap.h . . . . . ??
heapsort.cs . . . . . ??
ossim.cpp . . . . . ??
PriorityQueue.cpp . . . . . ??
PriorityQueue.h . . . . . ??
show11.cpp . . . . . ??
test11.cpp . . . . . ??
test11hs.cpp . . . . . ??
test11pq.cpp . . . . . ??

Chapter 4

Class Documentation

4.1 Greater< KeyType > Class Template Reference

Public Member Functions

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

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 ()
- int leftchild (int i)
- int rightchild (int i)
- int parentof (int i)
- bool isEmpty () const
- bool isFull () const
- void swap (int i, int n)
- void showStructure () const
- void writeLevels () const

Static Public Attributes

- static const int DEFAULT_MAX_HEAP_SIZE = 10

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 for the heap

allocates the memory for the heap of the appropriate size. saves the max size and initializes size to 0.

Parameters

None.	

Returns

Constructor.

Precondition

None.

Postcondition

Initialized.

allocate the new memory

save max size

set actual size right now

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

Copy constructor for the heap

Copies max size from the source and allocates memory for that size. Then copies sets the two heaps equal to eachother.

Parameters

A	heap to copy from.

Returns

Constructor.

Precondition

Heaps created.

Postcondition

Two equivalent heaps.

copy size

allocate the memory

copy

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

Destructor

Deallocates the memory or array.

Parameters

None.	

Returns

Destructor.

Precondition

Array initialized.

Postcondition

Memory released.

delete

4.2.2 Member Function Documentation

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

Clears the heap.

Makes size 0.

Parameters

none.	

Returns

void.

Precondition

Heap initialized.

Postcondition

Empty heap.

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

Inserts new item.

inserts newest item to end of the array then checks the parent to see if it is bigger. if it is, they get swapped and then that repeats until the parent is not smaller.

Parameters

item	to insert.

Returns

void.

Exceptions

if	it is full throw error

Precondition

Heap initialized.

Postcondition

New item inserted.

check if full

insert

work through its parents to see if it is a max heap set indecies

check if inserted if larger than parent

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

Is Empty.

If the size if 0 then it is empty.

Parameters

none.	

Returns

True if empty. False if not.

Precondition

Heap initialized.

Postcondition

Nothing different.

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

Is Full.

If size is equal to the max size of the heap, then the heap is full.

Parameters

none.	

Returns

True if full. False if not.

Precondition

Heap initialized.

Postcondition

Nothing different.

4.2.2.5 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::leftchild (int *i*) [inline]

the left child is known to be store at the equation in the function.

Parameters

The	index of which you want the left child.

Returns

The index of the left child of the param.

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

Overloaded assigment operator.

Checks if the heaps are the same. If they are returns it. Then sees if the sizes are the same. If they aren't, copies them then deletes current memory allocated then allocates new memory. Lastly copies items from the source array to the home array. Copies size and returns.

Parameters

Heap	to copy from.

Returns

Copied heap.

Precondition

Two heaps.

Postcondition

Heaps will be equal.

check equality

make same size array

copy items

copy size

return

4.2.2.7 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::parentof (int *i*) [inline]

the parent is known to be store at the equation in the function.

Parameters

The	index of which you want the parent.

Returns

The index of the parent of the child in the param.

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

Removes highest priority item.

saves highest item in a temp and inserts last item into the first item. then compares the item to it's children, gets the larger of the children then swaps if the parent is smaller. this repeats until the end of the heap is reached.

Parameters

none.	

Returns

void.

Exceptions

if	it is empty throw error

Precondition

Heap initialized.

Postcondition

front item removed.

if empty throw error

save top item

put last item to the front

compare the two children

if left is bigger swap

swaps

update index to where we moved

update size

return removed item

4.2.2.9 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::rightchild (int *i*) [inline]

the right child is known to be store at the equation in the function.

Parameters

The	index of which you want the right child.

Returns

The index of the right child of the param.

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

4.2.2.11 template<typename DataType , typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::swap (int *i,* int *n*)

Simple swap of the items at the given indicies.

Parameters

The	indicies you want swapped.

Returns

Void.

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

Prints heap by level.

Prints out 2^n number of items, n being the number of the row (starting at zero), then inserts endl. This will repeat in a loop, every loop will print out a row.

Parameters

none.	

Returns

void.

Precondition

Heap initialized.

Postcondition

Nothing changes.

check if empty

for row N, print out all items

end the line

go to next row

4.2.3 Member Data Documentation

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

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

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]

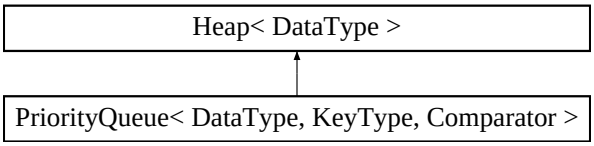
The documentation for this class was generated from the following file:

- 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 ()

Additional Inherited Members

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 for the priority queue

Just calls the heap constructor to initialize a heap.

Parameters

Size	of the heap.

Returns

Constructor.

Precondition

None.

Postcondition

Initialized.

4.4.2 Member Function Documentation

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

Dequeue

Just calls the heap remove to remove the highest priority item.

Parameters

None.	

Returns

The item of highest priority.

Precondition

None.

Postcondition

Initialized.

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

Enqueue

Using the heap insert, enqueues item into the priority queue.

Parameters

The	thing to add to the queue.

Returns

Void.

Precondition

Initialized tree.

Postcondition

New item will be in the queue.

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

- PriorityQueue.h
- PriorityQueue.cpp

4.5 TaskData Struct Reference

Public Member Functions

- int getPriority () const

Public Attributes

- int priority
- int arrived

4.5.1 Member Function Documentation

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

4.5.2 Member Data Documentation

4.5.2.1 int TaskData::arrived

4.5.2.2 int TaskData::priority

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

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]

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

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]

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

- test11.cpp

Chapter 5

File Documentation

5.1 config.h File Reference

Macros

- #define LAB11_TEST1 1

5.1.1 Macro Definition 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 "Heap.h"
#include "show11.cpp"
```

5.3 Heap.h File Reference

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

Classes

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

5.4 heapsort.cs File Reference

Functions

- void moveDown (DataType dataItems[], int root, int size)

- void heapSort (DataType dataItems[], int size)

5.4.1 Function Documentation

5.4.1.1 void heapSort (DataType *dataItems[],* int *size*)

5.4.1.2 void moveDown (DataType *dataItems[],* int *root,* int *size*)

5.5 ossim.cpp File Reference

```
#include <iostream>
#include <cstdlib>
#include "PriorityQueue.cpp"
```

Classes

- struct TaskData

Functions

- int main ()

5.5.1 Function Documentation

5.5.1.1 int main ( )

Seed the random number generator

clear

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

dequeue

show info

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

get # of new tasks

insert first new item set priority

record current time

save to queue

if two new items, insert second

5.6 PriorityQueue.cpp File Reference

```
#include "PriorityQueue.h"
```

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 "PriorityQueue.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 ( )