PA09_Lab11_Huffman_Henry

Generated by Doxygen 1.7.6.1

Tue Nov 4 2014 23:54:33

Contents

1 Class Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 Greater< KeyType > Class Template Reference 7

4.1.1 Member Function Documentation 7

4.1.1.1 operator() 7

4.2 Heap< DataType, KeyType, Comparator > Class Template Reference . 7

4.2.1 Constructor & Destructor Documentation 8

4.2.1.1 Heap 8

4.2.1.2 Heap 9

4.2.1.3 ~Heap 9

4.2.2 Member Function Documentation 10

4.2.2.1 clear 10

4.2.2.2 getLeft 10

4.2.2.3 getParent 11

4.2.2.4 getRight 11

4.2.2.5 heapDown 12

4.2.2.6 heapUp 12

4.2.2.7 insert 13

4.2.2.8 isEmpty . . . 13
4.2.2.9 isFull . . . 14
4.2.2.10 operator= . . . 14
4.2.2.11 remove . . . 15
4.2.2.12 showStructure . . . 15
4.2.2.13 showSubtree . . . 15
4.2.2.14 writeLevels . . . 15
4.2.3 Member Data Documentation . . . 16
4.2.3.1 comparator . . . 16
4.2.3.2 dataItems . . . 16
4.2.3.3 DEFAULT_MAX_HEAP_SIZE . . . 16
4.2.3.4 maxSize . . . 16
4.2.3.5 size . . . 16
4.3 Less< KeyType > Class Template Reference . . . 16
4.3.1 Member Function Documentation . . . 17
4.3.1.1 operator() . . . 17
4.4 PriorityQueue< DataType, KeyType, Comparator > Class Template Reference . . . 17
4.4.1 Constructor & Destructor Documentation . . . 17
4.4.1.1 PriorityQueue . . . 17
4.4.2 Member Function Documentation . . . 18
4.4.2.1 dequeue . . . 18
4.4.2.2 enqueue . . . 18
4.5 TaskData Struct Reference . . . 19
4.5.1 Member Function Documentation . . . 19
4.5.1.1 getPriority . . . 19
4.5.1.2 getPriority . . . 19
4.5.2 Member Data Documentation . . . 19
4.5.2.1 arrived . . . 19
4.5.2.2 priority . . . 20
4.6 TestData Class Reference . . . 20
4.6.1 Member Function Documentation . . . 20
4.6.1.1 getPriority . . . 20
4.6.1.2 getPriority . . . 20

4.6.1.3 setPriority 20
4.6.1.4 setPriority 20
4.6.2 Member Data Documentation 20
4.6.2.1 priority 20
4.7 TestDataItem< KeyType > Class Template Reference 20
4.7.1 Constructor & Destructor Documentation 21
4.7.1.1 TestDataItem 21
4.7.2 Member Function Documentation 21
4.7.2.1 getPriority 21
4.7.2.2 setPriority 21
4.7.3 Member Data Documentation 21
4.7.3.1 priority 21

5 File Documentation **23**

5.1 config.h File Reference 23
5.1.1 Define Documentation 23
5.1.1.1 LAB11_TEST1 23
5.2 Heap.cpp File Reference 23
5.2.1 Detailed Description 23
5.3 Heap.h File Reference 24
5.4 heapsort.cs File Reference 24
5.4.1 Function Documentation 24
5.4.1.1 heapSort 24
5.4.1.2 moveDown 24
5.5 ossim.cpp File Reference 24
5.5.1 Detailed Description 25
5.5.2 Function Documentation 25
5.5.2.1 main 25
5.6 ossim.cs File Reference 25
5.6.1 Function Documentation 26
5.6.1.1 main 26
5.7 PriorityQueue.cpp File Reference 26
5.7.1 Detailed Description 26
5.8 PriorityQueue.h File Reference 26

5.8.1 Variable Documentation 27
5.8.1.1 defMaxQueueSize 27
5.9 show11.cpp File Reference 27
5.10 test11.cpp File Reference 27
5.10.1 Function Documentation 27
5.10.1.1 main 27
5.10.1.2 printHelp 27
5.11 test11hs.cpp File Reference 27
5.11.1 Function Documentation 28
5.11.1.1 main 28
5.11.2 Variable Documentation 28
5.11.2.1 MAX_NUM_DATA_ITEMS 28
5.12 test11pq.cpp File Reference 28
5.12.1 Function Documentation 28
5.12.1.1 main 28
5.12.1.2 printHelp 28

Chapter 1

Class Index

1.1 Class Hierarchy

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

Greater< KeyType > 7
Heap< DataType, KeyType, Comparator > 7
Heap< DataType > 7
    PriorityQueue< DataType, KeyType, Comparator > 17
Less< KeyType > 16
Less< int > 16
TaskData 19
TestData 20
TestDataItem< KeyType > 20

Chapter 2

Class Index

2.1 Class List

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

Greater< KeyType > 7
Heap< DataType, KeyType, Comparator > 7
Less< KeyType > 16
PriorityQueue< DataType, KeyType, Comparator > 17
TaskData 19
TestData 20
TestDataItem< KeyType > 20

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

config.h 23
Heap.cpp
This program contains the basic function for manipulating data in the heap 23
Heap.h 24
heapsort.cs 24
ossim.cpp
This program uses a priority queue to simulate a line 24
ossim.cs 25
PriorityQueue.cpp
The priority queue uses the predefined heap functions to perform basic priority queue operations 26
PriorityQueue.h 26
show11.cpp 27
test11.cpp 27
test11hs.cpp 27
test11pq.cpp 28

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
- int getLeft (int index) const
- int getRight (int index) const
- int getParent (int index) const
- void heapUp (int me)
- void heapDown (int me)

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)

Heap Constructor

This constructor creates an array of new dataItems, sets the maxSize, and current size.

Parameters

maxNumber	- an integer that sets the maxSize of the number of dataItems in the array

Returns

none

Precondition

there will not be an initialized heap

Postcondition

there will be an initialized heap with the maxSize, and size set

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

Heap copy constructor

This constructor creates an array of new dataItems with identical values of another specified heap.

Parameters

other	- another heap that is to be copied

Returns

none

Precondition

there will only be one initialized heap

Postcondition

there will be two initialized heaps with identical values

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

Heap deconstructor

The deconstructor deallocates the array of data and sets the pointer to null

Parameters

none	

Returns

none

Precondition

there will be an initialized heap with an array set

Postcondition

there will not be any memory allocated to the 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 function

This function deallocates the memory, reallocates the memory, and sets the size of the array to zero

Parameters

none	

Returns

none

Precondition

there will an array that may or may not be empty

Postcondition

there will be an that is empty

4.2.2.2 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::getLeft (int *index*) const [private]

getLeft function

This function returns the index of the left child of the current index

Parameters

index	- the current location in the array

Returns

none

Precondition

the left child index will not be returned

Postcondition

the left child index will be returned

4.2.2.3 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::getParent (int *index*) const [private]

getParent function

This function returns the index of the parent of the current index

Parameters

index	- the current location in the array

Returns

none

Precondition

the parent of current index will not be returned

Postcondition

the parent of current index will be returned

4.2.2.4 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::getRight (int *index*) const [private]

getRight function

This function returns the index of the right child of the current index

Parameters

index	- the current location in the array

Returns

none

Precondition

the right child index will not be returned

Postcondition

the right child index will be returned

4.2.2.5 template<typename DataType , typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::heapDown (int *me*) [private]

heapDown function

This function places data in the correct order after the highest priority.

Parameters

me	- index of current dataItem

Returns

none

Precondition

the data may or may not be in the correct locations

Postcondition

the data will be in the correct locations

4.2.2.6 template<typename DataType , typename KeyType , typename Comparator > void Heap< DataType, KeyType, Comparator >::heapUp (int *me*) [private]

heapUp function

This helper function aides in placing the values of the heap in the correct location after new data is inserted

Parameters

me	- current index

Returns

none

Precondition

the new data may not be placed in the correct location

Postcondition

the new data item and all other values will be correctly placed into the tree

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

insert function

This function checks to see if any more data can be placed into the current array. If not, an error is thrown, otherwise, the data is inserted into the array

Parameters

newDataItem	- the dataItem that is to be placed into the current heap

Returns

none

Precondition

the newDataItem will not be placed into the heap

Postcondition

the newDataItem will be placed into the heap, if not full

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

isEmpty function

This function checks to see if the size is set to zero

Parameters

none	

Returns

bool stating whether or not the size is set to zero

Precondition

the heap may or may not be empty

Postcondition

the function will report true or false in accordance to its size

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

isFull function

This function checks to see if the current size of the array matches the maxsize

Parameters

none	

Returns

bool - states whether or not the array is full

Precondition

the array may or may not be full

Postcondition

the function will report true or false in accordance to its size and maxsize

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

operator=

This function returns a heap. If the heap is assigned to itself, it simply returns itself. Otherwise, the current heap will reallocate memory to match the other heap, and copy the values of the other heap. Then it will return itself.

Parameters

other	- another heap that is to be copied

Returns

Heap& - the current tree

Precondition

there may or may not be two initialized trees with different values

Postcondition

there may or may not be two initialized trees with identical values

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

remove function

This function removes the dataItem of highest priority, unless the array is empty.

Parameters

none	

Returns

DataType - the dataItem that was removed

Precondition

dataItems may or may not be in the array

Postcondition

the dataItem of highest priority is removed, if not empty

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

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

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

writeLevels function

This function writes the levels of the heap, starting with the highest priority, and ending with the lowest priority.

Parameters

noen	

Returns

none

Precondition

nothing will be output

Postcondition

each level of the heap will be output in the correct order

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]`

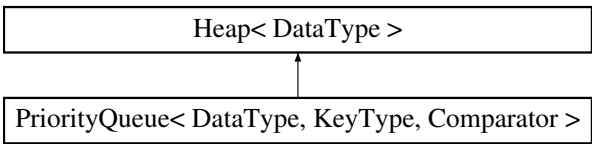
The documentation for this class was generated from the following file:

- 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)

PriorityQueue Constructor

This constructor inherently calls the heap consructor

Parameters

maxNumber	- an integer that sets the maxSize of the number of dataItems in the heap array

Returns

none

Precondition

there will not be an initialized heap or priority queue

Postcondition

there will be an initialized heap with the maxSize, and size set (and priority queue)

4.4.2 Member Function Documentation

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

dequeue function

This function calls the remove function of the heap

Parameters

none	

Returns

DataType - the dataItem that was removed from the priority queue

Precondition

the dataItem of highest priority will be in the priority queue

Postcondition

the dataItem of highest priority will not be in the priority queue

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

enqueue function

This function calls the insert function of the heap.

Parameters

newDataItem	- the dataItem that is to be placed into the priority queue

Returns

none

Precondition

the newDataItem will not be in the current priority queue

Postcondition

the dataItem will be placed into the priority queue if the heap array is not full

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
- int getPriority () const

Public Attributes

- int priority
- int arrived

4.5.1 Member Function Documentation

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

4.5.1.2 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 files:

- ossim.cpp
- ossim.cs

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

  configuration is set to one for testing of write levels function

5.1.1 Define Documentation

5.1.1.1 #define LAB11_TEST1 1

configuration is set to one for testing of write levels function

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

5.2 Heap.cpp File Reference

This program contains the basic function for manipulating data in the heap.

```
#include "Heap.h"
```

5.2.1 Detailed Description

This program contains the basic function for manipulating data in the heap.

Author

Henry Huffman

Version

1.1

More specifically, this program has the following basic member functions: heap constructor, copy constructor, overloaded = operator, and heap deconstructor. This program also contains the following functions: insert, remove, clear, isEmpty, isFull, showStructure, and writeLevels. There are also several helper functions which are also used to help in manipulating data. These functions being: showSubtree, getLeft, getRight, getParent, heap up, and heap down.

Date

Friday, October 17th, 2014

5.3 Heap.h File Reference

`#include <stdexcept> #include <iostream> #include <cstdlib> ×`

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

This program uses a priority queue to simulate a line.

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

Classes

- struct TaskData

Functions

- int main ()

5.5.1 Detailed Description

This program uses a priority queue to simulate a line.

Author

Henry Huffman

Version

1.1

Moreover, this program will generate random numbers for a specified amount of times. Then it will dequeue one data item per minute and enqueue randomly generated values. The value of the dequeued dataItem will be output.

Date

Tuesday, October 28th, 2014

5.5.2 Function Documentation

5.5.2.1 int main ( )

5.6 ossim.cs File Reference

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

Classes

- struct TaskData

Functions

- int main ()

5.6.1 Function Documentation

5.6.1.1 int main ( )

5.7 PriorityQueue.cpp File Reference

The priority queue uses the predefined heap functions to perform basic priority queue operations.

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

5.7.1 Detailed Description

The priority queue uses the predefined heap functions to perform basic priority queue operations.

Author

Henry Huffman

Version

1.1

More specifically, this program contains the following functions: priorirty queue constructor, enqueue, and dequeue.

Date

Friday, October 17th, 2014

5.8 PriorityQueue.h File Reference

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

Classes

- class PriorityQueue< DataType, KeyType, Comparator >

Variables

- const int defMaxQueueSize = 10

5.8.1 Variable Documentation

5.8.1.1 const int defMaxQueueSize = 10

5.9 show11.cpp File Reference

5.10 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 ()

5.10.1 Function Documentation

5.10.1.1 int main ( )

5.10.1.2 void printHelp ( )

5.11 test11hs.cpp File Reference

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

Classes

- class TestData

Functions

- int main ()

Variables

- const int MAX_NUM_DATA_ITEMS = 10

5.11.1 Function Documentation

5.11.1.1 int main ( )

5.11.2 Variable Documentation

5.11.2.1 const int MAX_NUM_DATA_ITEMS = 10

5.12 test11pq.cpp File Reference

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

Classes

- class TestData

Functions

- void printHelp ()
- int main ()

5.12.1 Function Documentation

5.12.1.1 int main ( )

5.12.1.2 void printHelp ( )