Lab 11 - Heap Implementation

Generated by Doxygen 1.7.6.1

Wed Nov 13 2013 17:55:18

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
- int parent (int cIndex) const
- int lchild (int pIndex) const
- int rchild (int pIndex) 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

Default constructor which creats an empty heap. Allocates memory for heap of given size.

Algorithm

1. Set maxSize equal to maxNumber.
2. Set size to 0 to designate empty heap.
3. Allocate array of size maxSize and assign dataItems to point to this array.

Precondition

1. Function must be called with valid integer for maxNumber.

Postcondition

1. All data members will be updated.
2. dataItems will now have memory allocated to it.

Parameters

maxNumber	Integer variable designating size of heap. This integer has a default value of Default_Max_Heap_Size = 10.

Returns

None

Exceptions

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

Copy Constructor

Copy constructor initializes a heap which will be set to a deep copy of the given heap from the parameter.

Algorithm

1. Set all data members equal to equivalent data members of other heap.
2. Allocate memory for dataItems of array of DataType with size maxSize.
3. Assign current heap to other heap.

Precondition

1. Function must be called with valid heap as parameter.

Postcondition

1. All data members of current heap will be updated.
2. dataItems will have memory allocated to it.
3. Contents of dataItems will be equivalent to dataitems of other heap.

Parameters

other	Heap object to be copied to current heap.

Returns

None

Exceptions

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

Destructor

Deallocates the memory used to store the heap. Resets other data members to indicate an empty heap.

Algorithm

1. If dataItems has memory allocated to it, deallocate this memory.
2. Set maxSize and actual size to zero.

Precondition

1. Heap must be initialized and should have memory allocated to it.

Postcondition

1. Heap will no longer have memory allocated to it.
2. Data members will be updated to indicate empty heap.

Parameters

None	

Returns

None

Exceptions

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 from the heap.

Algorithm

1. Set number of actual data items to zero.

Precondition

1. Heap must be declared and have memory allocated to it.

Postcondition

1. Number of actual items will be set to 0.

Parameters

None	

Returns

None

Exceptions

None	

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

Insert

Description

Algorithm

1.

2.

Precondition

1.
2.

Postcondition

1.
2.

Parameters

None	

Returns

None

Exceptions

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

IsEmpty

Function determines whether heap is empty. Returns true if heap is empty.

Algorithm

1. Determine whether actual size is equal to zero.
2. Returns true if it is.

Precondition

1. Heap must be initialized.

Postcondition

1. Heap will be unchanged.

Parameters

None	

Returns

Boolean value. Returns true is heap is empty. Otherwise, returns false.

Exceptions

None	

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

IsFull

Function determines whether heap is full. Returns true if heap is full.

Algorithm

1. Determine whether actual size is equal to maxSize.
2. Returns true if it is.

Precondition

1. Heap must be initialized.

Postcondition

1. Heap will be unchanged.

Parameters

None	

Returns

Boolean value. Returns true is heap is full. Otherwise, returns false.

Exceptions

None	

4.2.2.5 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::lchild (int *pIndex*) const `[private]`

lchild

Function calculates and returns the index of the left child of the given index, if index is valid. This assumes that any error checking for out of bounds is handled outside of the function.

Algorithm

1. If index is valid (greater than or equal to zero), calculate left child index. This index is twice the parent index plus one.
2. Otherwise, set the left child index equal to negative one to designate an out of bounds index.
3. Return left child index.

Precondition

1. Function must be called with valid integer for parameter. This parameter does not have to be within bounds of the array, but should be.

Postcondition

1. The heap will be unchanged.
2. The calculated value will be returned to the calling function.

Parameters

pIndex	Integer value indicating the index of the parent item which is used to calculate the left child index.

Returns

Integer value of the index of the right child, if it exists. Otherwise, negative one.

Exceptions

None	

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 assignment operator

Overloaded assignment operator which sets the current heap to be a deep copy of the other heap. Returns current heap object.

Algorithm

1. Check if current object has the same address of other heap.
2. If not, check if maxSize of current heap is different from other maxSize.
3. If it is, then set maxSize equal to maxSize of other heap.
4. Deallocate memory allocated to dataItems.
5. Reallocate memory of new size.
6. Set actual size equal to actual size of other heap.
7. Iterate through other heap and copy contents to current heap.
8. Return current heap.

Precondition

1. Function must be called with valid heap object as parameter.
2. Other heap object must be previously initialized and have memory allocated to it.
3. Current heap object must be previously initialized and have memory allocated to it.

Postcondition

1. Current heap will be updated to be a deep copy of other heap.
2. Other heap will be unchanged.

Parameters

other	Heap object to be copied to current heap.

Returns

Heap object of current heap.

Exceptions

4.2.2.7 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::parent (int *cIndex*) const [private]

Parent

Function calculates and returns the index of the parent of the given index, if index is valid. This assumes that any error checking for out of bounds is handled outside of the function.

Algorithm

1. If index is valid (greater than or equal to zero and within array), calculate parent index. This index is the child index minus one, all divided by two.
2. Otherwise, set the parent index equal to negative one to designate an out of bounds index.
3. Return parent index.

Precondition

1. Function must be called with valid integer for parameter. This parameter does not have to be within bounds of the array, but should be.

Postcondition

1. The heap will be unchanged.
2. The calculated value will be returned to the calling function.

Parameters

pIndex	Integer value indicating the index of the child item which is used to calculate the parent index.

Returns

Integer value of the index of the parent, if it exists. Otherwise, negative one.

Exceptions

None	

4.2.2.8 template<typename DataType , typename KeyType , typename Comparator > int Heap< DataType, KeyType, Comparator >::rchild (int *pIndex*) const [private]

rchild

Function calculates and returns the index of the right child of the given index, if index is valid. This assumes that any error checking for out of bounds is handled outside of the function.

Algorithm

1. If index is valid (greater than or equal to zero), calculate right child index. This index is twice the parent index plus two.
2. Otherwise, set the right child index equal to negative one to designate an out of bounds index.
3. Return right child index.

Precondition

1. Function must be called with valid integer for parameter. This parameter does not have to be within bounds of the array, but should be.

Postcondition

1. The heap will be unchanged.
2. The calculated value will be returned to the calling function.

Parameters

pIndex	Integer value indicating the index of the parent item which is used to calculate the right child index.

Returns

Integer value of the index of the right child, if it exists. Otherwise, negative one.

Exceptions

None	

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

Remove

Function removes the item with the largest priority from the heap. The heap is then rearranged so that the next largest item is at the top of the heap, and the other elements in the heap maintain the proper relationships.

Algorithm

1. Check if heap is empty.
2.

Precondition

1.
2.

Postcondition

1.
2.

Parameters

None	

Returns

None

Exceptions

logic_error	Exception is thrown if trying to remove a data item from an empty list

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

Show Structure

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.

Algorithm

1. Initialize variables.
2. Output array form.
3. Output tree form.

Precondition

1. Heap must be initialized.
2. showSubtree function must be declared and properly functioning.
3. getPriority function must be declared and properly functioning.

Postcondition

1. Heap will be unchanged.

Parameters

None	

Returns

None

Exceptions

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

Show Subtree Helper Function

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.

Algorithm

1. Check if index is less than size.
2. Output right subtree.
3. Tab over to level.
4. Output dataItem's priority.
5. Output connector character.
6. output left subtree.

Precondition

1. Heap must be initialized.
2. getPriority function must be declared and properly functioning.

Postcondition

1. Heap will be unchanged.

Parameters

index	Integer of index of item to be displayed.
level	Integer of level of item to be displayed

Returns

None

Exceptions

None	

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

writeLevels

Function displays the heap separated by level

Algorithm

1. Initialize variables for the first element in the heap.
2. Loop through the heap.
3. If the current index is equal to the index of the left child of the previous left child, then output an endline.
4. Output the priority of the element in the heap.

Precondition

1. Heap must be declared.
2. Function lchild must be properly functioning.

Postcondition

1. Heap will be unchanged.

Parameters

None	

Returns

None

Exceptions

None	

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]`

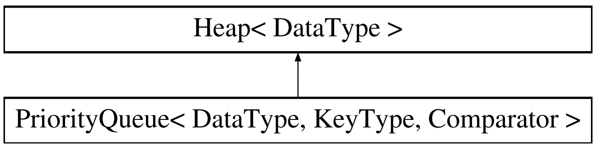
The documentation for this class was generated from the following file:

- 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**)

4.4.2 Member Function Documentation

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

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

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

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_T-ESTN to have the value 1.

5.2 Heap.cpp File Reference

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

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

Functions

- int main ()

5.5.1 Function Documentation

5.5.1.1 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

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

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 ( )