BSTree

7

Generated by Doxygen 1.8.6

Tue Nov 7 2017 11:55:17

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 AccountRecord Struct Reference

Public Attributes

- int acctID
- char firstName [nameLength]
- char lastName [nameLength]
- double balance

3.1.1 Member Data Documentation

3.1.1.1 int AccountRecord::acctID

3.1.1.2 double AccountRecord::balance

3.1.1.3 char AccountRecord::firstName[nameLength]

3.1.1.4 char AccountRecord::lastName[nameLength]

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

- database.cpp

3.2 BSTree< DataType, KeyType > Class Template Reference

`#include <BSTree.h>`

Classes

- class BSTreeNode

Public Member Functions

- BSTree ()

  Default Binary Tree Constructor.
- BSTree (const BSTree< DataType, KeyType > &other)

Copy Constructor.

- BSTree & operator= (const BSTree< DataType, KeyType > &other)

  Overloaded Assignment Operator.

- ~BSTree ()

  Destructor.

- void insert (const DataType &newDataItem)

  Inserts new data item.

- bool retrieve (const KeyType &searchKey, DataType &searchDataItem) const

  Retrieves data item.

- bool remove (const KeyType &deleteKey)

  Removes data item.

- void writeKeys () const

  Output keys.

- void clear ()

  Clears the tree.

- bool isEmpty () const

  Checks if the binary tree is empty.

- void showStructure () const
- int getHeight () const

  Gets the height of the binary tree.

- int getCount () const

  Gets the number of nodes in tree.

Protected Member Functions

- void showHelper (BSTreeNode *p, int level) const
- void insertHelper (BSTreeNode *&ptr, DataType data)
- bool retrieveHelper (BSTreeNode *ptr, const KeyType &searchKey, DataType &searchDataItem) const

  Recursive Helper function for public retrieve function.

- void clearHelper (BSTreeNode *&ptr)
- bool removeHelper (BSTreeNode *&ptr, const KeyType &deleteKey)
- void writeHelper (BSTreeNode *ptr) const

  Recursive helper function to output keys.

- void copyHelper (const BSTreeNode *sourcePtr, BSTreeNode *&newPtr)
- int heightHelper (BSTreeNode *ptr) const

  Recursively helps the getHeight function get the height of the binary tree.

- int countHelper (BSTreeNode *ptr, int &count) const

  Recursively helps the getCount function get the number of nodes in tree.

Protected Attributes

- BSTreeNode * root

  Pointer to the root node.

3.2.1 Constructor & Destructor Documentation

3.2.1.1 template<typename DataType , typename KeyType > BSTree< DataType, KeyType >::BSTree ( )

Default Binary Tree Constructor.

Precondition

None

Postcondition

Creates an empty binary search tree

3.2.1.2 template<typename DataType , typename KeyType > BSTree< DataType, KeyType >::BSTree (const BSTree< DataType, KeyType > & *other*)

Copy Constructor.

Parameters

other	Reference to binary tree to be copied

Precondition

None

Postcondition

Inititalizes the binary search tree to be equivalent to the other BSTree object parameter

3.2.1.3 template<typename DataType , typename KeyType > BSTree< DataType, KeyType >::~BSTree ( )

Destructor.

Parameters

None	

Precondition

None

Postcondition

Deallocates the memory used to store the binary search tree

3.2.2 Member Function Documentation

3.2.2.1 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::clear ( )

Clears the tree.

Postcondition

Removes all the data items in the binary search tree

See Also

{ references }

Note

{ text . . . ex. An algorithm }

3.2.2.2 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::clearHelper (BSTreeNode *& *ptr*) [protected]

3.2.2.3 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::copyHelper (const BSTreeNode * *sourcePtr,* BSTreeNode *& *newPtr*) [protected]

3.2.2.4 template<typename DataType , typename KeyType > int BSTree< DataType, KeyType >::countHelper (BSTreeNode * *ptr,* int & *count*) const [protected]

Recursively helps the getCount function get the number of nodes in tree.

Parameters

ptr	points to the current node
count	keeps track of the number of nodes during the traversal of the tree

Postcondition

Returns the count of the nubmer of data items in the binary search tree

See Also

BSTree<DataType,KeyType>::getCount() const

Note

Uses preOrder traversal to get the count of the tree

Returns

An int representing the amount of data items in the binary search tree

3.2.2.5 template<typename DataType , typename KeyType > int BSTree< DataType, KeyType >::getCount ( ) const

Gets the number of nodes in tree.

Postcondition

Returns the count of the nubmer of data items in the binary search tree

See Also

BSTree<DataType,KeyType>::countHelper(BSTreeNode* ptr, int& count) const

Note

Uses preOrder traversal to get the count of the tree

Returns

An int representing the amount of data items in the binary search tree

3.2.2.6 template<typename DataType , typename KeyType > int BSTree< DataType, KeyType >::getHeight ( ) const

Gets the height of the binary tree.

Postcondition

Returns the geight of the binary search tree

See Also

BSTree<DataType, KeyType>::heightHelper(BSTreeNode *& ptr, int& height) const

Note

Height is defined as the number of nodes on the longest path from the root node to any leaf node.

Returns

an int representing the height of the tree

3.2.2.7 template<typename DataType , typename KeyType > int BSTree< DataType, KeyType >::heightHelper (BSTreeNode * *ptr*) const [protected]

Recursively helps the getHeight function get the height of the binary tree.

Parameters

ptr	points to the nodes in the tree
height	keeps count of the height of the tree

Postcondition

Returns the height of the binary search tree

See Also

BSTree<DataType,KeyType>::getHeight() const

Note

Height is defined as the number of nodes on the longest path from the root node to any leaf node.

Returns

an int representing the height of the tree

3.2.2.8 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::insert (const DataType & *newDataItem*)

Inserts new data item.

Parameters

newDataItem	Reference to DataType to be added to binary tree

Precondition

None

Postcondition

Inserts newDataItem into the binary search tree. If a data item with the same key as newDataItem already exists in the tree, then updates that data item with newDataItem

See Also

insertHelper Function

3.2.2.9 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::insertHelper (BSTreeNode *& *ptr,* DataType *data*) [protected]

3.2.2.10 template<typename DataType , typename KeyType > bool BSTree< DataType, KeyType >::isEmpty ( ) const

Checks if the binary tree is empty.

Returns

Returns true if tree is empty, otherwise, returns false

3.2.2.11 template<typename DataType , typename KeyType > BSTree< DataType, KeyType > & BSTree< DataType, KeyType >::operator= (const BSTree< DataType, KeyType > & *other*)

Overloaded Assignment Operator.

Parameters

other	Reference to binary tree to be copied

Precondition

None

Postcondition

Sets the binary search tree to be equivalent to the other BSTree object parameter

Returns

Reference to the calling object

3.2.2.12 template<typename DataType , typename KeyType > bool BSTree< DataType, KeyType >::remove (const KeyType & *deleteKey*)

Removes data item.

Parameters

deleteKey	Reference to the index value that needs to be deleted from the binary tree

Precondition

None

Postcondition

Deletes the data item with key deleteKey from the binary search tree. If this data item is found, then deletes it from the tree and returns true. Otherwise, returns false

See Also

{ references }

Note

{ text . . . ex. An algorithm }

Returns

Returns true if data item is found and deleted. Otherwise, returns false.

3.2.2.13 template<typename DataType , typename KeyType > bool BSTree< DataType, KeyType >::removeHelper (BSTreeNode *& *ptr,* const KeyType & *deleteKey*) [protected]

3.2.2.14 template<typename DataType , typename KeyType > bool BSTree< DataType, KeyType >::retrieve (const KeyType & *searchKey,* DataType & *searchDataItem*) const

Retrieves data item.

Parameters

searchKey	The key of the data item that is being searched for in the binary tree
searchDataItem	If a data item is found that matches the searchKey, then the data item is copied to searchDataItem, otherwise, it is left undefined

Precondition

None

Postcondition

Searches the binary search tree for the data item with key searchKey. If this data item is found, then copies the data item to searchDataItem and returns true. Otherwise, returns false with searchDataItem undefined.

See Also

bool BSTree<DataType,KeyType>::retrieveHelper(BSTreeNode* ptr, const KeyType& searchKey, DataType& searchDataItem)

Returns

Returns true if data item is found, otherwise returns false.

3.2.2.15 template<typename DataType , typename KeyType > bool BSTree< DataType, KeyType >::retrieveHelper (BSTreeNode * *ptr,* const KeyType & *searchKey,* DataType & *searchDataItem*) const [protected]

Recursive Helper function for public retrieve function.

Parameters

ptr	A pointer to the current node
searchKey	The key of the data item that is being searched for in the binary tree
searchDataItem	If a data item is found that matches the searchKey, then the data item is copied to searchDataItem, otherwise, it is left undefined

Precondition

None

Postcondition

Searches the binary search tree for the data item with key searchKey. If this data item is found, then copies the data item to searchDataItem and returns true. Otherwise, returns false with searchDataItem undefined.

See Also

bool BSTree<DataType,KeyType>::retrieve(const KeyType &searchKey, DataType &searchDataItem) const

Returns

Returns true if data item is found, otherwise returns false.

3.2.2.16 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::showHelper (BSTreeNode * *p,* int *level*) const [protected]

3.2.2.17 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::showStructure ( ) const

3.2.2.18 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::writeHelper (BSTreeNode * *ptr*) const [protected]

Recursive helper function to output keys.

Precondition

None

Postcondition

Outputs the keys of the data items in the binary search tree. The keys are output in ascending order on one line, separated by spaces.

See Also

void BSTree<DataType,KeyType>::writeKeys() const

Note

In-Order Traversal

3.2.2.19 template<typename DataType , typename KeyType > void BSTree< DataType, KeyType >::writeKeys ( ) const

Output keys.

Precondition

None

Postcondition

Outputs the keys of the data items in the binary search tree. The keys are output in ascending order on one line, separated by spaces.

See Also

void BSTree::writeHelper(BSTreeNode*& ptr)

Note

{ text ... ex. An algorithm }

3.2.3 Member Data Documentation

3.2.3.1 template<typename DataType, class KeyType> BSTreeNode* BSTree< DataType, KeyType >::root [protected]

Pointer to the root node.

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

- BSTree.h
- BSTree.cpp
- show9.cpp

3.3 BSTree< DataType, KeyType >::BSTreeNode Class Reference

`#include <BSTree.h>`

Public Member Functions

- BSTreeNode (const DataType &nodeDataItem, BSTreeNode *leftPtr, BSTreeNode *rightPtr)

  Tree Node Constructor.

Public Attributes

- DataType dataItem

  Binary search tree data item.
- BSTreeNode * left

  Pointer to the left child.
- BSTreeNode * right

  Pointer to the right child.

3.3.1 Constructor & Destructor Documentation

3.3.1.1 template<typename DataType, class KeyType> BSTree< DataType, KeyType >::BSTreeNode::BSTreeNode (const DataType & *nodeDataItem,* BSTreeNode * *leftPtr,* BSTreeNode * *rightPtr*)

Tree Node Constructor.

Parameters

<*parameter-name*>	{ parameter description }

Precondition

{ description of the precondition }

Postcondition

{ description of the postcondition }

Exceptions

<*exception-object*>	{ exception description }

See Also

{ references }

Note

{ text . . . ex. An algorithm }

Returns

{ description of the return value }

3.3.2 Member Data Documentation

3.3.2.1 template<typename DataType, class KeyType> DataType BSTree< DataType, KeyType >::BSTreeNode::dataItem

Binary search tree data item.

3.3.2.2 template<typename DataType, class KeyType> BSTreeNode* BSTree< DataType, KeyType >::BSTreeNode::left

Pointer to the left child.

3.3.2.3 template<typename DataType, class KeyType> BSTreeNode* BSTree< DataType, KeyType >::BSTreeNode::right

Pointer to the right child.

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

- BSTree.h
- BSTree.cpp

3.4 IndexEntry Struct Reference

Public Member Functions

- int getKey () const

Public Attributes

- int acctID
- long recNum

3.4.1 Member Function Documentation

3.4.1.1 int IndexEntry::getKey ( ) const [inline]

3.4.2 Member Data Documentation

3.4.2.1 int IndexEntry::acctID

3.4.2.2 long IndexEntry::recNum

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

- database.cpp

3.5 TestData Class Reference

Public Member Functions

- void setKey (int newKey)
- int getKey () const
- TestData & operator= (const TestData &orig)
- bool operator> (const TestData &right) const
- bool operator< (const TestData &right) const
- bool operator== (const TestData &right) const
- bool operator== (const int right) const
- bool operator> (const int right) const
- bool operator< (const int right) const

Friends

- ostream & operator<< (ostream &out, const TestData &dataOut)

3.5.1 Member Function Documentation

3.5.1.1 int TestData::getKey ( ) const [inline]

3.5.1.2 bool TestData::operator< (const TestData & *right*) const [inline]

3.5.1.3 bool TestData::operator< (const int *right*) const [inline]

3.5.1.4 TestData& TestData::operator= (const TestData & *orig*) [inline]

3.5.1.5 bool TestData::operator== (const TestData & *right*) const [inline]

3.5.1.6 bool TestData::operator== (const int *right*) const [inline]

3.5.1.7 bool TestData::operator> (const TestData & *right*) const [inline]

3.5.1.8 bool TestData::operator> (const int *right*) const [inline]

3.5.1.9 void TestData::setKey (int *newKey*) [inline]

3.5.2 Friends And Related Function Documentation

3.5.2.1 ostream& operator<< (ostream & *out,* const TestData & *dataOut*) [friend]

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

- test9.cpp

Chapter 4

File Documentation

4.1 BSTree.cpp File Reference

```
#include <iostream>
#include <algorithm>
#include "BSTree.h"
#include "show9.cpp"
```

4.2 BSTree.h File Reference

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

Classes

- class BSTree< DataType, KeyType >
- class BSTree< DataType, KeyType >::BSTreeNode

4.2.1 Detailed Description

Author

Leah Kramer

Date

01/01/1970

4.3 database.cpp File Reference

```
#include <iostream>
#include <fstream>
#include "BSTree.cpp"
```

Classes

- struct AccountRecord
- struct IndexEntry

Functions

- int main ()

Variables

- const int nameLength = 11
- const long bytesPerRecord = 37

4.3.1 Function Documentation

4.3.1.1 int main ( )

4.3.2 Variable Documentation

4.3.2.1 const long bytesPerRecord = 37

4.3.2.2 const int nameLength = 11

4.4 show9.cpp File Reference

```
#include "BSTree.h"
```

4.5 test9.cpp File Reference

```
#include <iostream>
#include "BSTree.cpp"
#include "config.h"
```

Classes

- class TestData

Functions

- void print_help ()
- int main ()

4.5.1 Function Documentation

4.5.1.1 int main ( )

4.5.1.2 void print_help ( )

Index