Project 9 - Binary Search Tree ADT

AUTHOR

Version 1.00

10/20/2014

Table of Contents

Project 9 (Binary Search Tree ADT)

This program will implement a Binary Search Tree using a linked tree structure.

Author:

Saharrath Kleips

The specifications of this project match those of the book C++ Data Structures - A Laboratory Course (3rd Edition) Project 9. Data items within the data structure form a binary tree. Data items are of generic type DataType and each data item has a key of generic type KeyType. For each data item D in the tree, all the data items in D's left subtree have keys that are less than D's key and all the data items in D's right subtree have keys that are greater than D's key. /n Note: Binary Search Tree == BSTree, Binary Search Tree Node == BSTreeNode

Todo List

Member BSTree< DataType, KeyType >::writeLessThan (const KeyType &searchKey) const

Function implementation.

Member BSTree< DataType, KeyType >::writeLessThanHelper (const KeyType &searchKey, BSTreeNode *p) const

Function implementation.

Class Index

Class List

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

File Index

File List

Here is a list of all documented files with brief descriptions:

Class Documentation

AccountRecord Struct Reference

Public Attributes

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

Detailed Description

Definition at line 29 of file database.cpp.

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

1 database.cpp

BSTree< DataType, KeyType > Class Template Reference

Classes

class BSTreeNode

Public Member Functions

BSTree (const BSTree< DataType, KeyType > &other)

The copy constructor that initializes this BSTree to be equivalent to the other BSTree object parameter.

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

The overloaded assignment operator that sets this BSTree to be equivalent to the other BSTree object parameter and returrns a reference to this object.

~BSTree ()

The destructor that deallocates the memory used to store this BSTree.

void insert (const DataType &newDataItem)

Inserts newDataItem into this BSTree.

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

Searches this BSTree for the data item with key searchKey.

bool remove (const KeyType &deleteKey)

Deletes the data item with key deleteKey from this BSTree.

void writeKeys () const

Outputs the keys of the data items in this BSTree.

void clear ()

Removes all data items in this BSTree.

bool isEmpty () const

Returns true if this BSTree is empty.

void **showStructure** () const

int getHeight () const

Returns the height of this BSTree.

int getCount () const

Returns the count of the number of data items in this BSTree.

void writeLessThan (const KeyType &searchKey) const

Outputs all keys in this BSTree that are less than searchKey.

Protected Member Functions

void copyHelper (BSTreeNode *&p, BSTreeNode *other)

Recursive helper function.

void insertHelper (const DataType &newDataItem, BSTreeNode *&p)

Recursive helper function.

bool retrieveHelper (const KeyType &searchKey, DataType &searchDataItem, BSTreeNode *p) const

Recursive helper function.

bool removeHelper (const KeyType &deleteKey, BSTreeNode *&p)

Recursive helper function.

void writeKeysHelper (BSTreeNode *p) const

Recursive helper function.

void clearHelper (BSTreeNode *&p)
Recursive helper function.

int getHeightHelper (BSTreeNode *p, int currentLevel) const
Recursive helper function.

int getCountHelper (BSTreeNode *p) const
Recursive helper function.

void writeLessThanHelper (const KeyType &searchKey, BSTreeNode *p) const
Recursive helper function.

void **showHelper** (BSTreeNode *p, int level) const

Protected Attributes

BSTreeNode * **root**

Detailed Description

template<typename DataType, class KeyType>class BSTree< DataType, KeyType >

Definition at line 20 of file BSTree.h.

Constructor & Destructor Documentation

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

The copy constructor that initializes this BSTree to be equivalent to the other BSTree object parameter.

Precondition:
other is a valid BSTree.

Postcondition:
This BSTree will be a deep copy of the other BSTree.

Parameters:

other	is the BSTree that this BSTree will be made equivalent to.

BSTree<DataType,KeyType>::operator=(const BSTree<DataType,KeyType>&)
Definition at line 47 of file BSTree.cpp.

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

The destructor that deallocates the memory used to store this BSTree.

Postcondition:
This BSTree will be an empty, deallocated, BSTree.

See also:

BSTree<DataType,KeyType>::clear()

Definition at line 96 of file BSTree.cpp.

Member Function Documentation

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

Removes all data items in this BSTree.

Postcondition:

This BSTree will be an empty, deallocated, BSTree.

See also:

BSTree<DataType,KeyType>::clearHelper(BSTreeNode*& p)

Definition at line 305 of file BSTree.cpp.

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

Recursive helper function.

Parameters:

p	is the current node to check against.

See also:

BSTree<DataType,KeyType>::clear()

Definition at line 315 of file BSTree.cpp.

template<typename DataType , class KeyType > void BSTree< DataType, KeyType >::copyHelper (BSTreeNode *& *p*, BSTreeNode * *other*) [protected]

Recursive helper function.

Parameters:

p	is the node to copy into (destination node).
other	is the node to copy from (source node).

See also:

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

Definition at line 78 of file BSTree.cpp.

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

Returns the count of the number of data items in this BSTree.

Returns:

An integer representation of how many data items are in this BSTree.

See also:

BSTree<DataType,KeyType>::getCountHelper(BSTreeNode*)

Definition at line 387 of file BSTree.cpp.

template<typename DataType , class KeyType > int BSTree< DataType, KeyType >::getCountHelper (BSTreeNode * *p*) const [protected]

Recursive helper function.

Parameters:

See also:

p	is the current node to check against.

BSTree<DataType,KeyType>::getCount()

Definition at line 397 of file BSTree.cpp.

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

Returns the height of this BSTree.

Returns:

An integer representation of the height of this BSTree. see BSTree<DataType,KeyType>::getHeightHelper(BSTreeNode*, int)

Definition at line 351 of file BSTree.cpp.

template<typename DataType , class KeyType > int BSTree< DataType, KeyType >::getHeightHelper (BSTreeNode * *p*, int *currentLevel*) const [protected]

Recursive helper function.

Parameters:

See also:

p	is the current node to check against.
currentLevel	is the level of the current node.

BSTree<DataType,KeyType>::getHeight()

Definition at line 362 of file BSTree.cpp.

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

Inserts newDataItem into this BSTree.

If a data item with the same key as newDataItem already exists in this tree, then updates that data item with newDataItem.

Postcondition:

newDataItem will be inserted with respect to left and right BSTreeNodes.

Parameters:

See also:

newDataItem	is the data item to be inserted into this BSTree.

template<typename DataType , class KeyType > void BSTree< DataType, KeyType >::insertHelper (const DataType & *newDataItem*, BSTreeNode *& *p*) [protected]

Recursive helper function.

Parameters:

Definition at line 108 of file BSTree.cpp.

newDataItem	is the data item to insert into the BSTree.
p	is the current node to check against.

BSTree<DataType,KeyType>::insert(const DataType&)

Definition at line 119 of file BSTree.cpp.

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

Returns true if this BSTree is empty.

Otherwise, returns false.

Returns:

True if this BSTree is empty. False if this BSTree is not empty.

Definition at line 339 of file BSTree.cpp.

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

The overloaded assignment operator that sets this BSTree to be equivalent to the other BSTree object parameter and returrns a reference to this object.

Precondition:

other is a valid BSTree.

Postcondition:

This BSTree will be a deep copy of the other BSTree

Parameters:

See also:

other	is this BSTree that this BSTree will be made equivalent to.

The reference to this BSTree.

See also:

BSTree<DataType,KeyType>::copyHelper(BSTreeNode*&, BSTreeNode*)

Definition at line 62 of file BSTree.cpp.

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

Deletes the data item with key deleteKey from this BSTree.

If the data item is found, then deletes it from the tree and returns true. Otherwise, returns false.

Postcondition:

This BSTree will no longer contain the data item with key deleteKey if found.

Parameters:

Returns:

deleteKey	is the key to search which data item to delete in this BSTree.

True if the data item is found. False if the data item is not found.

See also:

BSTree<DataType,KeyType>::removeHelper(const KeyType&, BSTreeNode*&)

Definition at line 201 of file BSTree.cpp.

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

Recursive helper function.

Parameters:

Returns:

deleteKey	is the key to compare all the nodes within BSTree to.
p	is the current node to check against.

BSTree<DataType,KeyType>::remove(const KeyType&)

Definition at line 212 of file BSTree.cpp.

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

Searches this BSTree 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 equal to null.

Postcondition:

searchDataItem will be the copied data item if the data item is found.

Parameters:

See also:

searchKey	is the key to search this BSTree for.
searchDataItem	is the data item that will contain the search key's data item if found.

True if the data item is found. False if the data item is not found.

See also:

BSTree<DataType,KeyType>::retrieveHelper(const KeyType&, DataType&, BSTreeNode*)

Definition at line 153 of file BSTree.cpp.

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

Recursive helper function.

Parameters:

Returns:

searchKey	is the key to compare all nodes within the BSTree to.
searchDataItem	will contain the search key's data item if found.
p	is the current node to check against.

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

Definition at line 166 of file BSTree.cpp.

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

Outputs the keys of the data items in this BSTree.

The keys are output in ascending order on one line, separated by spaces.

Postcondition:

The keys of each data item are outputted to the console.

See also:

BSTree<DataType,KeyType>::writeKeysHelper(BSTreeNode*)

Definition at line 277 of file BSTree.cpp.

template<typename DataType , class KeyType > void BSTree< DataType, KeyType >::writeKeysHelper (BSTreeNode * *p*) const [protected]

Recursive helper function.

Parameters:

See also:

p	is the current node to check against.

BSTree<DataType,KeyType>::writeKeys()

Definition at line 288 of file BSTree.cpp.

template<typename DataType , class KeyType > void BSTree< DataType, KeyType >::writeLessThan (const KeyType & *searchKey*) const

Outputs all keys in this BSTree that are less than searchKey.

The keys are output in ascending order on one line, separated by spaces. searchKey does not need to be a key in this BSTree.

Postcondition:

The keys less than searchKey are outputted to the console.

Parameters:

searchKey	is the key to compare if this BSTree's keys are less than to.

See also:

BSTree<DataType,KeyType>::writeLessThanHelper(const keyType&, BSTreeNode*)

Todo:

Function implementation.

Definition at line 417 of file BSTree.cpp.

template<typename DataType , class KeyType > void BSTree< DataType, KeyType >::writeLessThanHelper (const KeyType & *searchKey*, BSTreeNode * *p*) const [protected]

Recursive helper function.

Parameters:

searchKey	is the key to compare if this BSTree's keys are less than to.
p	is the current node to check against.

See also:

BSTree<DataType,KeyType>::writeLessThan(const KeyType&)

Todo:

Function implementation.

Definition at line 429 of file BSTree.cpp.

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

2 BSTree.h
3 BSTree.cpp
4 show9.cpp

BSTree< DataType, KeyType >::BSTreeNode Class Reference

Public Member Functions

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

The parameterized constructor that sets the BSTreeNode's data item to the value nodeDataItem, BSTreeNode's previous pointer to the value leftPtr, and BSTreeNode's next pointer to the value rightPtr.

Public Attributes

DataType **dataItem**
BSTreeNode * **left**
BSTreeNode * **right**

Detailed Description

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

Definition at line 55 of file BSTree.h.

Constructor & Destructor Documentation

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

The parameterized constructor that sets the BSTreeNode's data item to the value nodeDataItem, BSTreeNode's previous pointer to the value leftPtr, and BSTreeNode's next pointer to the value rightPtr.

Postcondition:

This BSTreeNode will be a valid initialized BSTreeNode.

Parameters:

See also:

nodeDataItem	is the data to be stored within the node.
leftPtr	is the pointer to the previous BSTreeNode that this BSTreeNode is linked to.
rightPtr	is the pointer to the next BSTreeNode that this BSTreeNode is linked to.

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

5 BSTree.h
6 BSTree.cpp

IndexEntry Struct Reference

Public Member Functions

int **getKey** () const

Public Attributes

int **acctID**
long **recNum**

Detailed Description

Definition at line 42 of file database.cpp.

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

7 database.cpp

TestData Class Reference

Public Member Functions

void **setKey** (int newKey)
int **getKey** () const

Private Attributes

int **keyField**

Detailed Description

Definition at line 21 of file test9.cpp.

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

8 test9.cpp

File Documentation

BSTree.cpp File Reference

```
#include "BSTree.h"
#include "show9.cpp"
```

Detailed Description

Definition in file BSTree.cpp.

Index

INDE