Binary Search Tree

Tim Kwist
Version 1.0
10/20/14

Table of Contents

Table of contents

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 files with brief descriptions:

Class Documentation

AccountRecord Struct Reference

Public Attributes

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

Member Data Documentation

int AccountRecord::acctID

double AccountRecord::balance

char AccountRecord::firstName[nameLength]

char AccountRecord::lastName[nameLength]

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

- **database.cpp**

BSTree< DataType, KeyType > Class Template Reference

`#include <BSTree.h>`

Classes

- class **BSTreeNode**

Public Member Functions

- **BSTree** ()
- **BSTree** (const **BSTree**< DataType, KeyType > &other)
- **BSTree** & **operator=** (const **BSTree**< DataType, KeyType > &other)
- **~BSTree** ()
- void **insert** (const DataType &newDataItem)
- bool **retrieve** (const KeyType &searchKey, DataType &searchDataItem) const
- bool **remove** (const KeyType &deleteKey)
- void **writeKeys** () const
- void **clear** ()
- bool **isEmpty** () const
- void **showStructure** () const
- int **getHeight** () const
- int **getCount** () const

Protected Member Functions

- void **showHelper** (**BSTreeNode** *p, int level) const
- void **copyHelper** (**BSTreeNode** *&current, **BSTreeNode** *other)
- void **insertHelper** (**BSTreeNode** *&p, const DataType &newDataItem)
- bool **retrieveHelper** (**BSTreeNode** *p, const KeyType &searchKey, DataType &searchDataItem) const
- bool **removeHelper** (**BSTreeNode** *&p, const KeyType &deleteKey)
- void **writeKeysHelper** (const **BSTreeNode** *p) const
- void **clearHelper** (**BSTreeNode** *&p)
- int **getHeightHelper** (const **BSTreeNode** *p) const
- int **getCountHelper** (const **BSTreeNode** *p) const

Protected Attributes

- **BSTreeNode** * **root**

Constructor & Destructor Documentation

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

Default constructor for **BSTree**

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Creates an empty binary search tree

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Copy constructor for **BSTree**

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Initializes the binary search tree to be equivalent to the other **BSTree** object parameter

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

Destructor for **BSTree**

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Deallocates (frees) the memory used to store the binary search tree.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

Member Function Documentation

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

Removes all the data items in the binary search tree.

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

This binary search tree will have nothing in it and all memory will be deallocated.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Recursively assist the clear method by deleting nodes one by one. Go as far as left as possible, go as far right as possible, delete the child nodes, then go back to the previous stack call and delete the child nodes which now have no other children

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

This binary search tree will have nothing in it and all memory will be deallocated.

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

Helper function for copy constructor and assignment operator. Copies the contents, recursively, from one **BSTree** to this one

Parameters:

current	Current node of this **BSTree**
other	Current node of other **BSTree**

Returns:

None

Precondition:

None

Postcondition:

Copies other **BSTree** to current **BSTree**

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

Return the count of the number of data items in the binary search tree.

Parameters:

None	

Returns:

Number of data items in the binary search tree

Precondition:

None

Postcondition:

The contents of this **BSTree** will be unchanged.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Recursive helper function for getCount Check every node on the left side, check every node on the right side. Add one for each that isn't null, return total. Return the count of the number of data items in the binary search tree.

Parameters:

None	

Returns:

Number of data items in the binary search tree

Precondition:

None

Postcondition:

The contents of this **BSTree** will be unchanged.

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

Calculate and return the height of the **BSTree**

Parameters:

None	

Returns:

The height of the binary search tree

Precondition:

None

Postcondition:

The contents of this **BSTree** will be unchanged.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Recursive helper function for getHeight If left not NULL, go down left side; if right not NULL, go down right side. Once at the bottom, return 1, then as we go back up compare whether the left or right side has a higher value to determine what to return Calculate and return the height of the **BSTree**

Parameters:

None	

Returns:

The height of the binary search tree

Precondition:

None

Postcondition:

The contents of this **BSTree** will be unchanged.

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

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.

Parameters:

newDataItem	item to be inserted into binary search tree

Returns:

None

Precondition:

None

Postcondition:

Another item will be added to this binary search tree if the passed in data is new.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Recursive helper for insert method. If null, create new node Else continue down tree until we find a null node If data greater than current val, go down right If data less than current val, go down left 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.

Parameters:

p	current node being evaluated for whether or not to insert into
newDataItem	item to be inserted into binary search tree

Returns:

None

Precondition:

None

Postcondition:

Another item will be added to this binary search tree if the passed in data is new.

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

Return if tree is empty or not.

Parameters:

None	

Returns:

True if the binary search tree is empty. Otherwise, returns false.

Precondition:

None

Postcondition:

The contents of this binary search tree will not be changed.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Overloaded assignment operator for **BSTree**

Parameters:

other	**BSTree** object to be set equal to

Returns:

BSTree A reference to this **BSTree** object

Precondition:

None

Postcondition:

Sets the binary search tree to be equivalent to the other **BSTree** object parameter and returns a reference to this object

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

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

Parameters:

deleteKey	key to be deleted

Returns:

True if item is found and deleted; false otherwise.

Precondition:

None

Postcondition:

The deleteKey will be deleted from the tree if it exists in the tree. Otherwise the tree will be unchanged.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Recursive helper function for remove If no children, delete the node If one child, set child to current position then delete node If two children, find the predecessor to replace node, then delete node Deletes the data item with the key deleteKey from the binary search tree. If this data item is found, then deletes it from the tree and returns true. Otherwise return false.

Parameters:

p	current node being looked for deletion
deleteKey	key to be deleted

Returns:

True if item is found and deleted; false otherwise.

Precondition:

None

Postcondition:

The deleteKey will be deleted from the tree if it exists in the tree. Otherwise the tree will be unchanged.

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

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 return true. Otherwise returns false with searchDataItem

Parameters:

searchKey	key to be searched for
searchDataItem	data to be updated if key found

Returns:

True if data item found; false otherwise

Precondition:

None

Postcondition:

The contents of this tree will not be changed

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Recursive helper function for retrieve. If current value greater than search key, go down left If current value less than search key, go down right If null, return false 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 return true. Otherwise returns false with searchDataItem

Parameters:

p	current node being checked for if it is the searchKey
searchKey	key to be searched for
searchDataItem	data to be updated if key found

Returns:

True if data item found; false otherwise

Precondition:

None

Postcondition:

The contents of this tree will not be changed

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

Recursive helper for showStructure. Outputs the subtree whose root node is pointed to by p.

Parameters:

p	**BSTreeNode** currently being outputted
level	the level of this node within the tree

Returns:

None

Precondition:

None

Postcondition:

The contents of thi **BSTree** will be unchanged.

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

Outputs the keys in a binary search tree. The tree is output rotated counterclockwie 90 degrees from its conventional orientation using a "reverse" inorder traversal. This operation is intended for testing and debugging purposes only.

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

The contents of this **BSTree** will be unchanged.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

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.

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

The contents of this **BSTree** will be unchanged.

Parameters:

source	Expression tree that this expression tree will become a copy of

Returns:

None

Precondition:

None

Postcondition:

Initializes the expression tree to be equivalent to the other ExprTree object parameter.

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

Recursive helper function for writeKeys 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.

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

The contents of this **BSTree** will be unchanged.

Member Data Documentation

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

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

- **BSTree.h**
- **BSTree.cpp**
- **BSTree.cs**
- **show9.cpp**

BSTree< DataType, KeyType >::BSTreeNode Class Reference

`#include <BSTree.h>`

Public Member Functions

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

Public Attributes

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

Constructor & Destructor Documentation

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

Default constructor for **BSTreeNode**

Parameters:

None	

Returns:

None

Precondition:

None

Postcondition:

Creates an empty binary search tree

Member Data Documentation

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

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

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

---

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

- **BSTree.h**
- **BSTree.cpp**

IndexEntry Struct Reference

Public Member Functions

- int **getKey** () const

Public Attributes

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

Member Function Documentation

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

Member Data Documentation

int IndexEntry::acctID

long IndexEntry::recNum

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

- **database.cpp**

TestData Class Reference

Public Member Functions

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

Member Function Documentation

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

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

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

- **test9.cpp**

File Documentation

BSTree.cpp File Reference

This program will implement a Binary Search Tree.
`#include "BSTree.h"`

Detailed Description

This program will implement a Binary Search Tree.

Author:
Tim Kwist

Version:
1.0

The specifications of this program are defined by C++ Data Structures: A Laboratory Course (3rd edition) by Brandle, JGeisler, Roberge, Whittington, lab 9.

Date:
Wednesday, October 8, 2014

BSTree.cs File Reference

BSTree.h File Reference

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

Classes

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

config.h File Reference

Macros

- #define **LAB9_TEST1** 1
- #define **LAB9_TEST2** 1
- #define **LAB9_TEST3** 0

Macro Definition Documentation

#define LAB9_TEST1 1

BSTree class (Lab 9) configuration file. Activate test 'N' by defining the corresponding LAB9_TESTN to have the value 1. Deactive test 'N' by setting the value to 0.

#define LAB9_TEST2 1

#define LAB9_TEST3 0

database.cpp File Reference

This program will implement Exercise 1 for **BSTree**.

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

Classes

- struct **AccountRecord**
- struct **IndexEntry**

Functions

- int **main** ()

Variables

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

Detailed Description

This program will implement Exercise 1 for **BSTree**.

Author:

Tim Kwist

Version:

1.0

The specifications of this program are defined by C++ Data Structures: A Laboratory Course (3rd edition) by Brandle, JGeisler, Roberge, Whittington, lab 9 exercise 1.

Date:

Wednesday, October 8, 2014

Function Documentation

int main ()

Get the account ids and store them into a binary search tree with appropriate record number, then output the account ids from least to greatest. Next user will input an account id; if it exists in the list, find it and output all of the account details of the account id.

Variable Documentation

const long bytesPerRecord = 37

const int nameLength = 11

show9.cpp File Reference

test9.cpp File Reference

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

Classes

- class **TestData**

Functions

- void **print_help** ()
- int **main** ()

Function Documentation

int main ()

void print_help ()

Index

INDEX