

PA08 - Hash

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	Account Struct Reference	5
3.1.1	Member Function Documentation	5
3.1.1.1	getKey() const	5
3.1.1.2	hash(const string &str)	5
3.1.2	Member Data Documentation	5
3.1.2.1	password	5
3.1.2.2	username	5
3.2	BSTree< DataType, KeyType > Class Template Reference	6
3.2.1	Constructor & Destructor Documentation	7
3.2.1.1	BSTree()	7
3.2.1.2	BSTree(const BSTree< DataType, KeyType > &other)	7
3.2.1.3	~BSTree()	7
3.2.2	Member Function Documentation	7
3.2.2.1	clear()	7
3.2.2.2	clearHelper(BSTreeNode *&ptr)	8
3.2.2.3	copyHelper(const BSTreeNode *sourcePtr, BSTreeNode *&newPtr)	8
3.2.2.4	countHelper(BSTreeNode *ptr, int &count) const	8

3.2.2.5	<code>getCount() const</code>	8
3.2.2.6	<code>getHeight() const</code>	9
3.2.2.7	<code>heightHelper(BSTreeNode *ptr) const</code>	9
3.2.2.8	<code>insert(const DataType &newDataItem)</code>	9
3.2.2.9	<code>insertHelper(BSTreeNode *&ptr, DataType data)</code>	10
3.2.2.10	<code>isEmpty() const</code>	10
3.2.2.11	<code>operator=(const BSTree< DataType, KeyType > &other)</code>	10
3.2.2.12	<code>remove(const KeyType &deleteKey)</code>	10
3.2.2.13	<code>removeHelper(BSTreeNode *&ptr, const KeyType &deleteKey)</code>	11
3.2.2.14	<code>retrieve(const KeyType &searchKey, DataType &searchDataItem) const</code>	11
3.2.2.15	<code>retrieveHelper(BSTreeNode *ptr, const KeyType &searchKey, DataType &searchDataItem) const</code>	11
3.2.2.16	<code>showHelper(BSTreeNode *p, int level) const</code>	12
3.2.2.17	<code>showStructure() const</code>	12
3.2.2.18	<code>writeHelper(BSTreeNode *ptr) const</code>	12
3.2.2.19	<code>writeKeys() const</code>	13
3.2.3	Member Data Documentation	13
3.2.3.1	<code>root</code>	13
3.3	<code>BSTree< DataType, KeyType >::BSTreeNode</code> Class Reference	13
3.3.1	Constructor & Destructor Documentation	13
3.3.1.1	<code>BSTreeNode(const DataType &nodeDataItem, BSTreeNode *leftPtr, BSTreeNode *rightPtr)</code>	13
3.3.2	Member Data Documentation	14
3.3.2.1	<code>dataItem</code>	14
3.3.2.2	<code>left</code>	14
3.3.2.3	<code>right</code>	14
3.4	<code>HashTable< DataType, KeyType ></code> Class Template Reference	14
3.4.1	Constructor & Destructor Documentation	15
3.4.1.1	<code>HashTable(int initTableSize)</code>	15
3.4.1.2	<code>HashTable(const HashTable &other)</code>	15
3.4.1.3	<code>~HashTable()</code>	15

3.4.2	Member Function Documentation	16
3.4.2.1	clear()	16
3.4.2.2	insert(const DataType &newDataItem)	16
3.4.2.3	isEmpty() const	16
3.4.2.4	operator=(const HashTable &other)	16
3.4.2.5	remove(const KeyType &deleteKey)	17
3.4.2.6	retrieve(const KeyType &searchKey, DataType &returnItem) const	17
3.4.2.7	showStructure() const	18
3.4.2.8	standardDeviation() const	18
3.5	TestData Class Reference	19
3.5.1	Constructor & Destructor Documentation	19
3.5.1.1	TestData()	19
3.5.1.2	TestData()	19
3.5.2	Member Function Documentation	19
3.5.2.1	getKey() const	19
3.5.2.2	getKey() const	19
3.5.2.3	getValue() const	19
3.5.2.4	getValue() const	19
3.5.2.5	hash(const string &str)	19
3.5.2.6	hash(const string &str)	19
3.5.2.7	setKey(const string &newKey)	19
3.5.2.8	setKey(const string &newKey)	19
3.5.2.9	setValue(const string &newValue)	19

4 File Documentation	21
4.1 BSTree.cpp File Reference	21
4.2 BSTree.h File Reference	21
4.2.1 Detailed Description	21
4.3 HashTable.cpp File Reference	21
4.4 HashTable.h File Reference	22
4.4.1 Detailed Description	22
4.5 login.cpp File Reference	22
4.5.1 Function Documentation	22
4.5.1.1 main()	22
4.6 loginCL.cpp File Reference	22
4.6.1 Function Documentation	23
4.6.1.1 main()	23
4.6.1.2 promptUser(HashTable< TestData, string > &table)	23
4.6.1.3 readln(HashTable< TestData, string > &table)	23
4.7 test10.cpp File Reference	23
4.7.1 Function Documentation	23
4.7.1.1 main(int argc, char **argv)	23
4.7.1.2 print_help()	23
Index	25

Chapter 1

Class Index

1.1 Class List

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

Account	5
BSTree< DataType, KeyType >	6
BSTree< DataType, KeyType >::BSTreeNode	13
HashTable< DataType, KeyType >	14
TestData	19

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

BSTree.cpp	21
BSTree.h	21
HashTable.cpp	21
HashTable.h	22
login.cpp	22
loginCL.cpp	22
test10.cpp	23

Chapter 3

Class Documentation

3.1 Account Struct Reference

Public Member Functions

- string [getKey](#) () const

Static Public Member Functions

- static unsigned int [hash](#) (const string &str)

Public Attributes

- string [username](#)
- string [password](#)

3.1.1 Member Function Documentation

3.1.1.1 string Account::getKey () const [inline]

3.1.1.2 unsigned int Account::hash (const string & *str*) [static]

3.1.2 Member Data Documentation

3.1.2.1 string Account::password

3.1.2.2 string Account::username

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

- [login.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.

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

<i>other</i>	Reference to binary tree to be copied
--------------	---------------------------------------

Postcondition

Initializes 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.

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

`void BSTree<DataType,KeyType>::clearHelper\(BSTreeNode *&ptr\)`

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

<i>ptr</i>	points to the current node
<i>count</i>	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 height 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

<i>ptr</i>	points to the nodes in the tree
<i>height</i>	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 & newDataType)`

Inserts new data item.

Parameters

<i>newDataItem</i>	Reference to DataType to be added to binary tree
--------------------	--------------------------------------------------

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

<i>other</i>	Reference to binary tree to be copied
--------------	---------------------------------------

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

<i>deleteKey</i>	Reference to the index value that needs to be deleted from the binary tree
------------------	----------------------------------------------------------------------------

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

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

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

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

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

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

3.2.2.17 `template<typename DataType, class 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.

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.

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)`

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](#)

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

Tree Node Constructor.

Parameters

<i>nodeDataItem</i>	Address of data item that node will hold
<i>leftPtr</i>	Pointer to left node
<i>rightPtr</i>	Pointer to right node

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 HashTable< DataType, KeyType > Class Template Reference

```
#include <HashTable.h>
```

Public Member Functions

- [HashTable](#) (int initTableSize)
Constructor.
- [HashTable](#) (const [HashTable](#) &other)
Copy Constructor.
- [HashTable](#) & [operator=](#) (const [HashTable](#) &other)
Overloaded Assignment Operator.
- [~HashTable](#) ()
Destructor.
- void [insert](#) (const DataType &newDataItem)
Insert.
- bool [remove](#) (const KeyType &deleteKey)
Remove.
- bool [retrieve](#) (const KeyType &searchKey, DataType &returnItem) const
Retrieve.
- void [clear](#) ()
Clear.
- bool [isEmpty](#) () const
Empty Check.
- void [showStructure](#) () const
Show Structure.
- double [standardDeviation](#) () const
<brief description>="">

3.4.1 Constructor & Destructor Documentation

3.4.1.1 `template<typename DataType , typename KeyType > HashTable< DataType, KeyType >::HashTable (int initTableSize)`

Constructor.

Parameters

<i>initTableSize</i>	{parameter description}
----------------------	-------------------------

Postcondition

Creates the empty hash table

3.4.1.2 `template<typename DataType , typename KeyType > HashTable< DataType, KeyType >::HashTable (const HashTable< DataType, KeyType > & other)`

Copy Constructor.

Parameters

<i>other</i>	Address to the table to be copied
--------------	-----------------------------------

Postcondition

Initializes the hash table to be equivalent to the [HashTable](#) object parameter *other*

See also

{references}

3.4.1.3 `template<typename DataType , typename KeyType > HashTable< DataType, KeyType >::~~HashTable ()`

Destructor.

Postcondition

Deallocates (frees) the memory used to store a hash table

See also

void [HashTable](#) <DataType, KeyType> :: [clear\(\)](#)

3.4.2 Member Function Documentation

3.4.2.1 `template<typename DataType , typename KeyType > void HashTable< DataType, KeyType >::clear ()`

Clear.

Postcondition

Removes all data items in the hash table

3.4.2.2 `template<typename DataType , typename KeyType > void HashTable< DataType, KeyType >::insert (const DataType & newDataItem)`

Insert.

Parameters

<i>data</i>	{parameter description}
-------------	-------------------------

Postcondition

Inserts *newDataItem* into the appropriate BST. If a data item with the same key as *newDataItem* already exists in the BST, then updates that data item with *newDataItem*. Otherwise, it inserts it in the binary search tree

3.4.2.3 `template<typename DataType , typename KeyType > bool HashTable< DataType, KeyType >::isEmpty () const`

Empty Check.

Returns

Returns true if the hash table is empty. Otherwise returns false

3.4.2.4 `template<typename DataType , typename KeyType > HashTable< DataType, KeyType > & HashTable< DataType, KeyType >::operator= (const HashTable< DataType, KeyType > & other)`

Overloaded Assignment Operator.

Parameters

<i>other</i>	Address to the table to be copied
--------------	-----------------------------------

Postcondition

Sets the hash table to be equivalent to the other [HashTable](#) object parameter

Returns

Reference to this object

See also

{references}

3.4.2.5 `template<typename DataType , typename KeyType > bool HashTable< DataType, KeyType >::remove (const KeyType & deleteKey)`

Remove.

Parameters

<i>deleteKey</i>	key for object to be deleted
------------------	------------------------------

Postcondition

Searches the hash table for the data item with the key deleteKey. If the data item is found, then removes the data item.

Returns

True if the data item is found and removed. Otherwise returns false

3.4.2.6 `template<typename DataType , typename KeyType > bool HashTable< DataType, KeyType >::retrieve (const KeyType & searchKey, DataType & returnItem) const`

Retrieve.

Parameters

<i>key</i>	The key for the item to be searched for
<i>returnItem</i>	DataType object that the data returned from the search returns

Postcondition

Searches the hash table for the data item with key searchKey. If the data item is found, then copies the data item to returnItem.

Note

gets hash index from key parameter then calls [BSTree](#) class retrieve method.

Returns

Returns true if data item is found. Otherwise, returns false with returnItem undefined.

3.4.2.7 `template<typename DataType , typename KeyType > void HashTable< DataType, KeyType >::showStructure ()
const`

Show Structure.

Postcondition

Outputs the data items in the hash table. If the has table is empty, outputs "Empty hash table". For testing/debugging purposes.

3.4.2.8 `template<typename DataType , typename KeyType > double HashTable< DataType, KeyType >::standardDeviation () const`

<brief description>="">

Parameters

<parameter-name>	{parameter description}
------------------	-------------------------

Precondition

{description of the precondition}

Postcondition

{description of the postcondition}

Exceptions

<exception-object>	{exception description}
--------------------	-------------------------

Returns

{description of the return value}

See also

{references}

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

- [HashTable.h](#)
- [HashTable.cpp](#)

3.5 TestData Class Reference

Public Member Functions

- [TestData](#) ()
- void [setKey](#) (const string &newKey)
- string [getKey](#) () const
- void [setValue](#) (const string &newValue)
- string [getValue](#) () const
- [TestData](#) ()
- void [setKey](#) (const string &newKey)
- string [getKey](#) () const
- int [getValue](#) () const

Static Public Member Functions

- static unsigned int [hash](#) (const string &str)
- static unsigned int [hash](#) (const string &str)

3.5.1 Constructor & Destructor Documentation

3.5.1.1 [TestData::TestData](#) ()

3.5.1.2 [TestData::TestData](#) ()

3.5.2 Member Function Documentation

3.5.2.1 string [TestData::getKey](#) () const

3.5.2.2 string [TestData::getKey](#) () const

3.5.2.3 int [TestData::getValue](#) () const

3.5.2.4 int [TestData::getValue](#) () const

3.5.2.5 static unsigned int [TestData::hash](#) (const string & *str*) [static]

3.5.2.6 unsigned int [TestData::hash](#) (const string & *str*) [static]

3.5.2.7 void [TestData::setKey](#) (const string & *newKey*)

3.5.2.8 void [TestData::setKey](#) (const string & *newKey*)

3.5.2.9 void [TestData::setValue](#) (const string & *newValue*)

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

- [loginCL.cpp](#)
- [test10.cpp](#)

Chapter 4

File Documentation

4.1 BSTree.cpp File Reference

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

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 HashTable.cpp File Reference

```
#include <stdexcept>
#include <iostream>
#include "HashTable.h"
```

4.4 HashTable.h File Reference

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

Classes

- class [HashTable](#)< [DataType](#), [KeyType](#) >

4.4.1 Detailed Description

Author

Leah Kramer

Date

10/08/2017

4.5 login.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include "HashTable.cpp"
```

Classes

- struct [Account](#)

Functions

- int [main](#) ()

4.5.1 Function Documentation

4.5.1.1 int main ()

4.6 loginCL.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include "HashTable.cpp"
```

Classes

- class [TestData](#)

Functions

- void [readIn](#) ([HashTable](#)< [TestData](#), string > &table)
- void [promptUser](#) ([HashTable](#)< [TestData](#), string > &table)
- int [main](#) ()

4.6.1 Function Documentation

4.6.1.1 int main ()

4.6.1.2 void promptUser ([HashTable](#)< [TestData](#), string > & *table*)

4.6.1.3 void readIn ([HashTable](#)< [TestData](#), string > & *table*)

4.7 test10.cpp File Reference

```
#include <iostream>
#include <string>
#include "HashTable.cpp"
```

Classes

- class [TestData](#)

Functions

- void [print_help](#) ()
- int [main](#) (int argc, char **argv)

4.7.1 Function Documentation

4.7.1.1 int main (int *argc*, char ** *argv*)

4.7.1.2 void print_help ()

Index

- ~BSTree
 - BSTree, 7
- ~HashTable
 - HashTable, 15
- Account, 5
 - getKey, 5
 - hash, 5
 - password, 5
 - username, 5
- BSTree
 - ~BSTree, 7
 - BSTree, 7
 - clear, 7
 - clearHelper, 7
 - copyHelper, 8
 - countHelper, 8
 - getCount, 8
 - getHeight, 8
 - heightHelper, 9
 - insert, 9
 - insertHelper, 10
 - isEmpty, 10
 - operator=, 10
 - remove, 10
 - removeHelper, 11
 - retrieve, 11
 - retrieveHelper, 11
 - root, 13
 - showHelper, 12
 - showStructure, 12
 - writeHelper, 12
 - writeKeys, 12
- BSTree< DataType, KeyType >, 6
- BSTree< DataType, KeyType >::BSTreeNode, 13
- BSTree.cpp, 21
- BSTree.h, 21
- BSTree::BSTreeNode
 - BSTreeNode, 13
 - datalItem, 14
 - left, 14
 - right, 14
- BSTreeNode
 - BSTree::BSTreeNode, 13
- clear
 - BSTree, 7
 - HashTable, 16
- clearHelper
 - BSTree, 7
- copyHelper
 - BSTree, 8
- countHelper
 - BSTree, 8
- datalItem
 - BSTree::BSTreeNode, 14
- getCount
 - BSTree, 8
- getHeight
 - BSTree, 8
- getKey
 - Account, 5
 - TestData, 19
- getValue
 - TestData, 19
- hash
 - Account, 5
 - TestData, 19
- HashTable
 - ~HashTable, 15
 - clear, 16
 - HashTable, 15
 - insert, 16
 - isEmpty, 16
 - operator=, 16
 - remove, 17
 - retrieve, 17
 - showStructure, 17
 - standardDeviation, 18
- HashTable< DataType, KeyType >, 14
- HashTable.cpp, 21
- HashTable.h, 22
- heightHelper
 - BSTree, 9
- insert
 - BSTree, 9
 - HashTable, 16
- insertHelper
 - BSTree, 10
- isEmpty
 - BSTree, 10
 - HashTable, 16
- left
 - BSTree::BSTreeNode, 14
- login.cpp, 22

- main, [22](#)
- loginCL.cpp, [22](#)
 - main, [23](#)
 - promptUser, [23](#)
 - readIn, [23](#)
- main
 - login.cpp, [22](#)
 - loginCL.cpp, [23](#)
 - test10.cpp, [23](#)
- operator=
 - BSTree, [10](#)
 - HashTable, [16](#)
- password
 - Account, [5](#)
- print_help
 - test10.cpp, [23](#)
- promptUser
 - loginCL.cpp, [23](#)
- readIn
 - loginCL.cpp, [23](#)
- remove
 - BSTree, [10](#)
 - HashTable, [17](#)
- removeHelper
 - BSTree, [11](#)
- retrieve
 - BSTree, [11](#)
 - HashTable, [17](#)
- retrieveHelper
 - BSTree, [11](#)
- right
 - BSTree::BSTreeNode, [14](#)
- root
 - BSTree, [13](#)
- setKey
 - TestData, [19](#)
- setValue
 - TestData, [19](#)
- showHelper
 - BSTree, [12](#)
- showStructure
 - BSTree, [12](#)
 - HashTable, [17](#)
- standardDeviation
 - HashTable, [18](#)
- test10.cpp, [23](#)
 - main, [23](#)
 - print_help, [23](#)
- TestData, [19](#)
 - getKey, [19](#)
 - getValue, [19](#)
 - hash, [19](#)
 - setKey, [19](#)
 - setValue, [19](#)
- TestData, [19](#)
- username
 - Account, [5](#)
- writeHelper
 - BSTree, [12](#)
- writeKeys
 - BSTree, [12](#)