

## CS 250 Project 4: Hash Table Dictionary

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Project 4: Hash Table Dictionary</b>	<b>1</b>
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	HashTable< TK, TD > Class Template Reference . . . . .	9
5.1.1	Detailed Description . . . . .	10
5.1.2	Constructor & Destructor Documentation . . . . .	10
5.1.2.1	HashTable() . . . . .	10
5.1.3	Member Function Documentation . . . . .	11
5.1.3.1	Clear() . . . . .	11
5.1.3.2	CollisionUsedElement(int index) . . . . .	11
5.1.3.3	CollisionWrongKey(int index, int key) . . . . .	11
5.1.3.4	Contains(const TK &key) . . . . .	11
5.1.3.5	GetCollisionMethod() . . . . .	11
5.1.3.6	GetIndex(int key, bool unused) . . . . .	11
5.1.3.7	GetItem(const TK &key) . . . . .	12
5.1.3.8	HashFunction(int key) . . . . .	13

5.1.3.9	HashFunction2(int key)	13
5.1.3.10	Insert(const TK &key, const TD &data)	13
5.1.3.11	IsEmpty()	14
5.1.3.12	LinearProbe(int key)	14
5.1.3.13	Print(const string &filename)	14
5.1.3.14	QuadraticProbe(int key, int &addValue)	15
5.1.3.15	Remove(const TK &key)	15
5.1.3.16	SetCollisionMethod(CollisionMethod cm)	15
5.1.3.17	Size()	15
5.1.4	Friends And Related Function Documentation	16
5.1.4.1	Tester	16
5.1.5	Member Data Documentation	16
5.1.5.1	m_collisionMethod	16
5.1.5.2	m_data	16
5.1.5.3	m_size	16
5.2	Node< TK, TD > Struct Template Reference	16
5.2.1	Detailed Description	17
5.2.2	Constructor & Destructor Documentation	17
5.2.2.1	Node()	17
5.2.3	Member Data Documentation	17
5.2.3.1	data	17
5.2.3.2	hasData	17
5.2.3.3	key	17
5.3	Student Struct Reference	17
5.3.1	Friends And Related Function Documentation	18
5.3.1.1	operator<<	18
5.3.2	Member Data Documentation	18
5.3.2.1	gpa	18
5.3.2.2	name	18
5.3.2.3	studentId	18

5.4	StudentManager Class Reference	18
5.4.1	Member Function Documentation	19
5.4.1.1	ReadFile(const string &filename)	19
5.4.1.2	Run(CollisionMethod method)	19
5.4.1.3	SaveFile()	19
5.4.2	Member Data Documentation	19
5.4.2.1	m_students	19
5.5	Tester Class Reference	19
5.5.1	Detailed Description	20
5.5.2	Constructor & Destructor Documentation	20
5.5.2.1	Tester()	20
5.5.3	Member Function Documentation	20
5.5.3.1	Test_Contains()	20
5.5.3.2	Test_DoubleHash()	20
5.5.3.3	Test_GetItem()	20
5.5.3.4	Test_GetItemIndex()	20
5.5.3.5	Test_GetUnusedIndex()	20
5.5.3.6	Test_HashFunction()	21
5.5.3.7	Test_HashFunction2()	21
5.5.3.8	Test_Insert()	21
5.5.3.9	Test_IsEmpty()	21
5.5.3.10	Test_LinearProbe()	21
5.5.3.11	Test_QuadraticProbe()	21
5.5.3.12	Test_Remove()	21
5.5.3.13	Test_Size()	21
<b>6</b>	<b>File Documentation</b>	<b>23</b>
6.1	HashTable.hpp File Reference	23
6.2	main.cpp File Reference	23
6.2.1	Function Documentation	23
6.2.1.1	main()	23
6.3	main.md File Reference	23
6.4	Node.hpp File Reference	23
6.5	StudentManager.hpp File Reference	24
6.5.1	Function Documentation	24
6.5.1.1	operator<<(ostream &out, const Student &stu)	24
6.6	students.txt File Reference	24
6.7	Tester.hpp File Reference	24
6.7.1	Macro Definition Documentation	24
6.7.1.1	_TESTER_HPP	24
	<b>Index</b>	<b>25</b>



# Chapter 1

## Project 4: Hash Table Dictionary

### Viewing the documentation

Click on the **Classes** tab to view files, classes, and their documentation.

### About

For this project, you will read the instructions from the Doxygen file (or the comments in the code) to complete the program.

Use the **unit tests** provided to test your Binary Search Tree as you are implementing features.

### Turn in

When turning in your work, please put your name on the project folder and zip your project folder. This helps out because Canvas isn't great at file management.

### Files

For this project, you will only be working with the [HashTable.hpp](#) file.





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

HashTable< TK, TD > . . . . .	9
HashTable< int, Student > . . . . .	9
Node< TK, TD > . . . . .	16
Node< int, Student > . . . . .	16
Student . . . . .	17
StudentManager . . . . .	18
TesterBase	
Tester . . . . .	19



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">HashTable&lt; TK, TD &gt;</a>	
Key-value array using a hash table . . . . .	9
<a href="#">Node&lt; TK, TD &gt;</a>	
The nodes in the array . . . . .	16
<a href="#">Student</a> . . . . .	17
<a href="#">StudentManager</a> . . . . .	18
<a href="#">Tester</a>	
Unit tests for the Hash Table . . . . .	19



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">HashTable.hpp</a>	23
<a href="#">main.cpp</a>	23
<a href="#">Node.hpp</a>	23
<a href="#">StudentManager.hpp</a>	24
<a href="#">Tester.hpp</a>	24



## Chapter 5

# Class Documentation

### 5.1 HashTable< TK, TD > Class Template Reference

key-value array using a hash table

```
#include <HashTable.hpp>
```

#### Public Member Functions

- [HashTable](#) ()  
*Initialize the hash table.*
- void [SetCollisionMethod](#) ([CollisionMethod](#) cm)  
*Set the collision method.*
- [CollisionMethod](#) [GetCollisionMethod](#) ()  
*Returns the collision method.*
- bool [IsEmpty](#) ()  
*Returns true if the table is empty, or false if it is not.*
- int [Size](#) ()  
*Returns the amount of items stored in the table.*
- void [Clear](#) ()  
*Clears out the table.*
- void [Insert](#) (const TK &key, const TD &data)  
*Generates an index for the new item using the key and stores it in the array.*
- void [Remove](#) (const TK &key)  
*Soft-removes the item at the given key.*
- TD & [GetItem](#) (const TK &key)  
*Returns the data for the item, given some key.*
- bool [Contains](#) (const TK &key)  
*Tries to find an item given some key, and returns true if found, false if not.*
- void [Print](#) (const string &filename)  
*Writes out all the set data from the table to a file.*

## Private Member Functions

- bool [CollisionWrongKey](#) (int index, int key)
- bool [CollisionUsedElement](#) (int index)
- int [GetIndex](#) (int key, bool unused)  
*Get the index of an existing item, given some key.*
- int [HashFunction](#) (int key)  
*Primary hash function.*
- int [LinearProbe](#) (int key)  
*Steps the index forward by one.*
- int [QuadraticProbe](#) (int key, int &addValue)  
*Steps the index forward by some quadratically-expanding value.*
- int [HashFunction2](#) (int key)  
*Secondary hash function.*

## Private Attributes

- [Node](#)< TK, TD > [m\\_data](#) [[TABLE\\_SIZE](#)]  
*The internal array.*
- [CollisionMethod](#) [m\\_collisionMethod](#)  
*Which collision method is being used.*
- int [m\\_size](#)  
*The amount of items being stored in the array.*

## Friends

- class [Tester](#)

### 5.1.1 Detailed Description

```
template<typename TK, typename TD>
class HashTable< TK, TD >
```

key-value array using a hash table

### 5.1.2 Constructor & Destructor Documentation

5.1.2.1 `template<typename TK , typename TD > HashTable< TK, TD >::HashTable ( )`

Initialize the hash table.



### 5.1.3 Member Function Documentation

#### 5.1.3.1 `template<typename TK , typename TD > void HashTable< TK, TD >::Clear ( )`

Clears out the table.

Goes through all the items in the array and lazy-deletes it; that is - sets the "hasData" flag to false for everything, and sets the size to 0.

#### Returns

<void>

#### 5.1.3.2 `template<typename TK , typename TD > bool HashTable< TK, TD >::CollisionUsedElement ( int index )` [private]

#### 5.1.3.3 `template<typename TK , typename TD > bool HashTable< TK, TD >::CollisionWrongKey ( int index, int key )` [private]

#### 5.1.3.4 `template<typename TK, typename TD > bool HashTable< TK, TD >::Contains ( const TK & key )`

Tries to find an item given some key, and returns true if found, false if not.

Try to find the index of the item given the key. If the item doesn't exist, return false. If it does exist, return true.

#### Parameters

<i>key</i>	const TK&, the key of the item to search for
------------	--

#### Returns

<bool> Whether the item is found or not

#### 5.1.3.5 `template<typename TK , typename TD > CollisionMethod HashTable< TK, TD >::GetCollisionMethod ( )`

Returns the collision method.

#### 5.1.3.6 `template<typename TK , typename TD > int HashTable< TK, TD >::GetIndex ( int key, bool unused )` [private]

Get the index of an existing item, given some key.

- Generate an index with the HashFunction.
- You might want to create some temporary variables to help out with calculations, such as the amount of collisions encountered.

- While there's a collision (There's an item there, hasData is true, and the key doesn't match), figure out which collision method is being used.
  - For LINEAR:
    - \* Use the LinearProbe function on the index, and use % TABLE\_SIZE, to get the new index.
  - For QUADRATIC:
    - \* Use the QuadraticProbe function on the index, plus how many collisions have occurred, and use % TABLE\_SIZE, to get the new index.
    - \* (First collision, pass in the index and 1. Second collision, pass in the index and 2. etc...)
  - For DOUBLE:
    - \* Use HashFunction2 on the key. This amount, times the number of collisions, will be added onto the original hashed value to get a new index.
    - \*  $\text{index} = (\text{HashFunction}(\text{key}) + \text{collisions} * \text{HashFunction2}(\text{key})) \% \text{TABLE\_SIZE};$
- GET UNUSED INDEX:
  - Collisions occur for this function if the INDEX contains data
  - $(\text{m\_data}[\text{index}].\text{hasData})$
- GET ITEM INDEX:
  - Collisions occur for this function if the INDEX contains data, but it's not the key we're looking for ( $\text{table}[\text{index}].\text{key} \neq \text{searchkey}$ )
  - $(\text{m\_data}[\text{index}].\text{hasData} \ \&\& \ \text{m\_data}[\text{index}].\text{key} \neq \text{key})$
  - If we ever encounter a hasData = false, that means the item with that key is NOT in the table, so return -1 or throw an exception.

#### Parameters

<i>key</i>	const TK&, the key for the new item
------------	-------------------------------------

#### Returns

<int> The generated index

#### 5.1.3.7 `template<typename TK, typename TD > TD & HashTable< TK, TD >::GetItem ( const TK & key )`

Returns the data for the item, given some key.

Use the GetItemIndex to find the index of the item (given the key). If the item isn't found, throw an exception. Otherwise, if it is found, return the item's data.

#### Parameters

<i>key</i>	const TK&, the key of the item to retrieve
------------	--

#### Returns

<TD&> The data of the item found

5.1.3.8 `template<typename TK , typename TD > int HashTable< TK, TD >::HashFunction ( int key ) [private]`

Primary hash function.

Generate an index by using a hash function with the key.

Hash function to use:

- $\text{key mod TABLE\_SIZE}$

#### Parameters

<i>int</i>	key, the key of the item to generate an index for
------------	---

#### Returns

<int> Generated index

5.1.3.9 `template<typename TK , typename TD > int HashTable< TK, TD >::HashFunction2 ( int key ) [private]`

Secondary hash function.

Generate an index by using a hash function with the key.

Hash function to use:

- $7 - (\text{key mod } 7)$

#### Parameters

<i>int</i>	key, the key of the item to generate an index for
------------	---

#### Returns

<int> Generated index

5.1.3.10 `template<typename TK, typename TD> void HashTable< TK, TD >::Insert ( const TK & key, const TD & data )`

Generates an index for the new item using the key and stores it in the array.

First, check if the table is full. Only allow new items to be added if the table is NOT FULL.

Next, use the GetUnusedIndex function to get an index, passing in the key as the input.

Once an index has been retrieved, set up the [Node](#) at that index:

- Set the key
- Set the data
- Set hasData to true

Lastly, increment the size.

#### Parameters

<i>key</i>	const TK&, the key for the new item
<i>data</i>	const TD&, the data for the new item

#### Returns

<void>

5.1.3.11 `template<typename TK , typename TD > bool HashTable< TK, TD >::isEmpty ( )`

Returns true if the table is empty, or false if it is not.

If the table is empty (check m\_size), then return true. Otherwise, return false.

#### Returns

<bool> Whether the table is empty

5.1.3.12 `template<typename TK , typename TD > int HashTable< TK, TD >::LinearProbe ( int index ) [private]`

Steps the index forward by one.

Return the index immediately after the index passed in.

#### Parameters

<i>int</i>	index, the index generated
------------	----------------------------

#### Returns

<int> Generated index

5.1.3.13 `template<typename TK , typename TD > void HashTable< TK, TD >::Print ( const string & filename )`

Writes out all the set data from the table to a file.

**5.1.3.14** `template<typename TK, typename TD> int HashTable< TK, TD >::QuadraticProbe ( int key, int & addValue )`  
`[private]`

Steps the index forward by some quadratically-expanding value.

Return the value of the index + the value of

#### Parameters

<i>int</i>	index, the index plus the # of collisions squared.
------------	--

#### Returns

<int> Generated index

**5.1.3.15** `template<typename TK, typename TD > void HashTable< TK, TD >::Remove ( const TK & key )`

Soft-removes the item at the given key.

Use the GetItemIndex to get the index of the existing item in the array, using the key passed in. If the index is found, then set the item's hasData variable to false and decrement the size.

#### Parameters

<i>key</i>	const TK&, the key of the item to remove
------------	--

#### Returns

<void>

**5.1.3.16** `template<typename TK , typename TD > void HashTable< TK, TD >::SetCollisionMethod ( CollisionMethod cm )`

Set the collision method.

**5.1.3.17** `template<typename TK , typename TD > int HashTable< TK, TD >::Size ( )`

Returns the amount of items stored in the table.

#### Returns

<int> The amount of items stored in the array

### 5.1.4 Friends And Related Function Documentation

5.1.4.1 `template<typename TK, typename TD> friend class Tester` `[friend]`

### 5.1.5 Member Data Documentation

5.1.5.1 `template<typename TK, typename TD> CollisionMethod HashTable< TK, TD >::m_collisionMethod`  
`[private]`

Which collision method is being used.

5.1.5.2 `template<typename TK, typename TD> Node<TK,TD> HashTable< TK, TD >::m_data[TABLE_SIZE]`  
`[private]`

The internal array.

5.1.5.3 `template<typename TK, typename TD> int HashTable< TK, TD >::m_size` `[private]`

The amount of items being stored in the array.

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

- [HashTable.hpp](#)

## 5.2 Node< TK, TD > Struct Template Reference

The nodes in the array.

```
#include <Node.hpp>
```

### Public Member Functions

- [Node](#) ()  
*Initializes the [Node](#) by setting `hasData` to false.*

### Public Attributes

- TK [key](#)  
*The [Node](#)'s identifier key.*
- TD [data](#)  
*The data stored in the [Node](#).*
- bool [hasData](#)  
*A flag for whether the [Node](#) is in use.*

### 5.2.1 Detailed Description

```
template<typename TK, typename TD>
struct Node< TK, TD >
```

The nodes in the array.

### 5.2.2 Constructor & Destructor Documentation

5.2.2.1 `template<typename TK, typename TD> Node< TK, TD >::Node ( ) [inline]`

Initializes the [Node](#) by setting `hasData` to false.

### 5.2.3 Member Data Documentation

5.2.3.1 `template<typename TK, typename TD> TD Node< TK, TD >::data`

The data stored in the [Node](#).

5.2.3.2 `template<typename TK, typename TD> bool Node< TK, TD >::hasData`

A flag for whether the [Node](#) is in use.

5.2.3.3 `template<typename TK, typename TD> TK Node< TK, TD >::key`

The [Node](#)'s identifier key.

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

- [Node.hpp](#)

## 5.3 Student Struct Reference

```
#include <StudentManager.hpp>
```

### Public Attributes

- string [name](#)
- float [gpa](#)
- int [studentId](#)

## Friends

- ostream & [operator<<](#) (ostream &out, const [Student](#) &stu)

### 5.3.1 Friends And Related Function Documentation

5.3.1.1 ostream& [operator<<](#) ( ostream & *out*, const [Student](#) & *stu* ) [*friend*]

### 5.3.2 Member Data Documentation

5.3.2.1 float [Student::gpa](#)

5.3.2.2 string [Student::name](#)

5.3.2.3 int [Student::studentId](#)

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

- [StudentManager.hpp](#)

## 5.4 StudentManager Class Reference

```
#include <StudentManager.hpp>
```

Collaboration diagram for StudentManager:

### Public Member Functions

- void [Run](#) ([CollisionMethod](#) method)

### Private Member Functions

- void [ReadFile](#) (const string &filename)
- void [SaveFile](#) ()

### Private Attributes

- [HashTable](#)< int, [Student](#) > [m\\_students](#)



### 5.4.1 Member Function Documentation

5.4.1.1 void StudentManager::ReadFile ( const string & *filename* ) [private]

5.4.1.2 void StudentManager::Run ( CollisionMethod *method* )

5.4.1.3 void StudentManager::SaveFile ( ) [private]

### 5.4.2 Member Data Documentation

5.4.2.1 HashTable<int, Student> StudentManager::m\_students [private]

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

- [StudentManager.hpp](#)

## 5.5 Tester Class Reference

Unit tests for the Hash Table.

```
#include <Tester.hpp>
```

Inheritance diagram for Tester:

Collaboration diagram for Tester:

### Public Member Functions

- [Tester](#) ()

### Private Member Functions

- int [Test\\_HashFunction](#) ()  
*Tests the output index of the HashFunction given some input key.*
- int [Test\\_HashFunction2](#) ()  
*Tests the output index of the HashFunction2 given some input key.*
- int [Test\\_LinearProbe](#) ()  
*Tests the output of the LinearProbe given some input.*
- int [Test\\_QuadraticProbe](#) ()  
*Tests the output of the QuadraticProbe given some input.*
- int [Test\\_DoubleHash](#) ()  
*Tests the output of the DoubleHash given some input.*
- int [Test\\_Insert](#) ()  
*Tests the Insert function using the various probing methods; Prerequisites: GetUnusedindex, HashFunction, LinearProbe, QuadraticProbe, HashFunction2.*
- int [Test\\_Remove](#) ()  
*Tests the removal of an item; Prerequisites: Insert, SetCollisionMethod.*

- int [Test\\_GetItem](#) ()  
*Tests getting an item given some key using various probing methods; Prerequisites: SetCollisionMethod, Insert.*
- int [Test\\_GetItemIndex](#) ()  
*Gets an item's index given some key; Prerequisite: SetCollisionMethod.*
- int [Test\\_GetUnusedIndex](#) ()  
*Gets an index of an unused space in the array given some key; Prerequisite: SetCollisionMethod.*
- int [Test\\_Contains](#) ()  
*Checks whether Contains finds or doesn't find an item; Prerequisite function: Insert()*
- int [Test\\_IsEmpty](#) ()  
*Checks whether table is empty; Prerequisite function: Insert()*
- int [Test\\_Size](#) ()  
*Checks whether size is correct after inserts; Prerequisite function: Insert()*

### 5.5.1 Detailed Description

Unit tests for the Hash Table.

### 5.5.2 Constructor & Destructor Documentation

5.5.2.1 `Tester::Tester ( ) [inline]`

### 5.5.3 Member Function Documentation

5.5.3.1 `int Tester::Test_Contains ( ) [private]`

Checks whether Contains finds or doesn't find an item; Prerequisite function: Insert()

5.5.3.2 `int Tester::Test_DoubleHash ( ) [private]`

Tests the output of the DoubleHash given some input.

5.5.3.3 `int Tester::Test_GetItem ( ) [private]`

Tests getting an item given some key using various probing methods; Prerequisites: SetCollisionMethod, Insert.

5.5.3.4 `int Tester::Test_GetItemIndex ( ) [private]`

Gets an item's index given some key; Prerequisite: SetCollisionMethod.

5.5.3.5 `int Tester::Test_GetUnusedIndex ( ) [private]`

Gets an index of an unused space in the array given some key; Prerequisite: SetCollisionMethod.

5.5.3.6 `int Tester::Test_HashFunction ( ) [private]`

Tests the output index of the HashFunction given some input key.

5.5.3.7 `int Tester::Test_HashFunction2 ( ) [private]`

Tests the output index of the HashFunction2 given some input key.

5.5.3.8 `int Tester::Test_Insert ( ) [private]`

Tests the Insert function using the various probing methods; Prerequisites: GetUnusedindex, HashFunction, LinearProbe, QuadraticProbe, HashFunction2.

5.5.3.9 `int Tester::Test_IsEmpty ( ) [private]`

Checks whether table is empty; Prerequisite function: Insert()

5.5.3.10 `int Tester::Test_LinearProbe ( ) [private]`

Tests the output of the LinearProbe given some input.

5.5.3.11 `int Tester::Test_QuadraticProbe ( ) [private]`

Tests the output of the QuadraticProbe given some input.

5.5.3.12 `int Tester::Test_Remove ( ) [private]`

Tests the removal of an item; Prerequisites: Insert, SetCollisionMethod.

5.5.3.13 `int Tester::Test_Size ( ) [private]`

Checks whether size is correct after inserts; Prerequisite function: Insert()

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

- [Tester.hpp](#)



## Chapter 6

# File Documentation

### 6.1 HashTable.hpp File Reference

```
#include <string>
#include <iomanip>
#include <iostream>
#include <stdexcept>
#include "cuTEST/StringUtil.hpp"
#include "Node.hpp"
```

Include dependency graph for HashTable.hpp:

### 6.2 main.cpp File Reference

```
#include <iostream>
#include <string>
#include <fstream>
#include "StudentManager.hpp"
#include "Tester.hpp"
#include "cuTEST/Menu.hpp"
```

Include dependency graph for main.cpp:

#### Functions

- int `main` ()

#### 6.2.1 Function Documentation

##### 6.2.1.1 int main ( )

### 6.3 main.md File Reference

### 6.4 Node.hpp File Reference

This graph shows which files directly or indirectly include this file:

## Classes

- struct [Node< TK, TD >](#)  
*The nodes in the array.*

## 6.5 StudentManager.hpp File Reference

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

Include dependency graph for StudentManager.hpp: This graph shows which files directly or indirectly include this file:

## Classes

- struct [Student](#)
- class [StudentManager](#)

## Functions

- ostream & [operator<<](#) (ostream &out, const [Student](#) &stu)

### 6.5.1 Function Documentation

6.5.1.1 ostream& operator<< ( ostream & *out*, const [Student](#) & *stu* )

## 6.6 students.txt File Reference

## 6.7 Tester.hpp File Reference

```
#include <iostream>
#include <string>
#include "cuTEST/TesterBase.hpp"
#include "cuTEST/Menu.hpp"
#include "cuTEST/StringUtil.hpp"
#include "HashTable.hpp"
```

Include dependency graph for Tester.hpp: This graph shows which files directly or indirectly include this file:

## Classes

- class [Tester](#)  
*Unit tests for the Hash Table.*

## Macros

- [#define \\_TESTER\\_HPP](#)

### 6.7.1 Macro Definition Documentation

6.7.1.1 [#define \\_TESTER\\_HPP](#)

# Index

`_TESTER_HPP`

Tester.hpp, 24

Clear

HashTable, 11

CollisionUsedElement

HashTable, 11

CollisionWrongKey

HashTable, 11

Contains

HashTable, 11

data

Node, 17

GetCollisionMethod

HashTable, 11

GetIndex

HashTable, 11

GetItem

HashTable, 12

gpa

Student, 18

hasData

Node, 17

HashFunction

HashTable, 12

HashFunction2

HashTable, 13

HashTable

Clear, 11

CollisionUsedElement, 11

CollisionWrongKey, 11

Contains, 11

GetCollisionMethod, 11

GetIndex, 11

GetItem, 12

HashFunction, 12

HashFunction2, 13

HashTable, 10

Insert, 13

IsEmpty, 14

LinearProbe, 14

m\_collisionMethod, 16

m\_data, 16

m\_size, 16

Print, 14

QuadraticProbe, 14

Remove, 15

SetCollisionMethod, 15

Size, 15

Tester, 16

HashTable< TK, TD >, 9

HashTable.hpp, 23

Insert

HashTable, 13

IsEmpty

HashTable, 14

key

Node, 17

LinearProbe

HashTable, 14

m\_collisionMethod

HashTable, 16

m\_data

HashTable, 16

m\_size

HashTable, 16

m\_students

StudentManager, 19

main

main.cpp, 23

main.cpp, 23

main, 23

main.md, 23

name

Student, 18

Node

data, 17

hasData, 17

key, 17

Node, 17

Node< TK, TD >, 16

Node.hpp, 23

operator<<

Student, 18

StudentManager.hpp, 24

Print

HashTable, 14

QuadraticProbe

HashTable, 14

ReadFile

- StudentManager, [19](#)
- Remove
  - HashTable, [15](#)
- Run
  - StudentManager, [19](#)
- SaveFile
  - StudentManager, [19](#)
- SetCollisionMethod
  - HashTable, [15](#)
- Size
  - HashTable, [15](#)
- Student, [17](#)
  - gpa, [18](#)
  - name, [18](#)
  - operator<<, [18](#)
  - studentId, [18](#)
- studentId
  - Student, [18](#)
- StudentManager, [18](#)
  - m\_students, [19](#)
  - ReadFile, [19](#)
  - Run, [19](#)
  - SaveFile, [19](#)
- StudentManager.hpp, [24](#)
  - operator<<, [24](#)
- students.txt, [24](#)
- Test\_Contains
  - Tester, [20](#)
- Test\_DoubleHash
  - Tester, [20](#)
- Test\_GetItem
  - Tester, [20](#)
- Test\_GetItemIndex
  - Tester, [20](#)
- Test\_GetUnusedIndex
  - Tester, [20](#)
- Test\_HashFunction
  - Tester, [20](#)
- Test\_HashFunction2
  - Tester, [21](#)
- Test\_Insert
  - Tester, [21](#)
- Test\_IsEmpty
  - Tester, [21](#)
- Test\_LinearProbe
  - Tester, [21](#)
- Test\_QuadraticProbe
  - Tester, [21](#)
- Test\_Remove
  - Tester, [21](#)
- Test\_Size
  - Tester, [21](#)
- Tester, [19](#)
  - HashTable, [16](#)
  - Test\_Contains, [20](#)
  - Test\_DoubleHash, [20](#)
  - Test\_GetItem, [20](#)
  - Test\_GetItemIndex, [20](#)
  - Test\_GetUnusedIndex, [20](#)
  - Test\_HashFunction, [20](#)
  - Test\_HashFunction2, [21](#)
  - Test\_Insert, [21](#)
  - Test\_IsEmpty, [21](#)
  - Test\_LinearProbe, [21](#)
  - Test\_QuadraticProbe, [21](#)
  - Test\_Remove, [21](#)
  - Test\_Size, [21](#)
  - Tester, [20](#)
  - Tester.hpp, [24](#)
  - \_TESTER\_HPP, [24](#)