CS 250 Project 4: Hash Table Dictionary

Generated by Doxygen 1.8.11

Contents

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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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

key	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.
    - index = (HashFunction(key) + collisions ∗ HashFunction2(key)) % TABLE_SIZE;
- GET UNUSED INDEX:
  - Collisions occur for this function if the INDEX contains data
  - (m_data[index].hasData)
- GET ITEM INDEX:
  - Collisions occur for this function if the INDEX contains data, but it's not the key we're looking for (table[index].key != searchkey)
  - (m_data[index].hasData && m_data[index].key != key)
  - If we ever encouter a hasData = false, that means the item with that key is NOT in the table, so return -1 or throw an exception.

Parameters

key	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

key	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:

- key mod TABLE_SIZE

Parameters

int	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 - (key mod 7)

Parameters

int	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

key	const TK&, the key for the new item
data	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

int	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

int	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

key	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, Linear↩ Probe, 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