HashTable

0.1

Generated by Doxygen 1.8.6

Thu Mar 27 2014 23:22:20

Contents

Chapter 1

glossygloss

Glossygloss est un petit programme écrit en C++ permettant de stocker dans un dictionnaire un mot associé à une valeur.

Usefull : `http://fr.wikibooks.org/wiki/Programmation_C%2B%2B`

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 Alveole Class Reference

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

Public Member Functions

- Alveole (const Alveole< K, V > &other)
- Alveole (K key, V value) _key(key)
- c_value (value)
- _next (END)
- Alveole (K key, V value, Alveole< K, V > *next) _key(key)
- _value (value)
- _next (next)
- bool isQueue ()
- K getKey ()
- V getValue ()
- void setValue (V n_value)

5.1.1 Detailed Description

Alveole class embodies a hashtable's alveole. An alveole store a pair <k,v>. Alveoles are simply-linked elements.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Alveole::Alveole (const Alveole< K, V > & *other*) `[inline]`

Copy constructor

Parameters

`in`	*other*	the alveole to copy

5.1.2.2 Alveole::Alveole (K *key,* V *value*)

Pair constructor

Parameters

`in`	*key*	key of the pair
`in`	*value*	value of the pair

5.1.2.3 Alveole::Alveole (K *key*, V *value*, Alveole< K, V > * *next*)

Complex constructor

Parameters

`in`	*key*	key of the pair
`in`	*value*	value of the pair
`in`	*next*	adresse to the next alveole

5.1.3 Member Function Documentation

5.1.3.1 Alveole::_next (END) `[inline]`

5.1.3.2 Alveole::_next (next) `[inline]`

5.1.3.3 Alveole::_value (value)

5.1.3.4 Alveole::c_value (value)

5.1.3.5 K Alveole::getKey ( ) `[inline]`

Get the key of an alveole

Parameters

`out`	*key*	stored into the alveole

5.1.3.6 V Alveole::getValue ( ) `[inline]`

Get the value stored into an alveole

Parameters

`out`	*value*	of the alveole

5.1.3.7 bool Alveole::isQueue ( ) `[inline]`

Does alveole have next ?

Parameters

`out`	*true*	if elements coming next, else false

5.1.3.8 void Alveole::setValue (V *n_value*) `[inline]`

Set the value stored into an alveole

Parameters

in	*n_value*	The new value of the pair

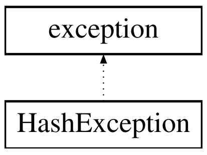
The documentation for this class was generated from the following file:

- HashTable.hpp

5.2 HashException Class Reference

```
#include <HashException.hpp>
```

Inheritance diagram for HashException:



Public Member Functions

- HashException (char ∗cause) _cause(cause)
- virtual ∼BagException () throw ()
- virtual const char ∗ what () const throw ()

5.2.1 Detailed Description

Exception class to manage hashtable errors

5.2.2 Constructor & Destructor Documentation

5.2.2.1 HashException::HashException (char ∗ *cause*) `[inline]`

constructor called then HashExceptions are threw

Parameters

in	*cause*	description of exception origin

5.2.2.2 virtual HashException::∼BagException ( ) throw) `[inline], [virtual]`

destructor currently, do anything special

5.2.3 Member Function Documentation

5.2.3.1 virtual const char∗ HashException::what ( ) const throw) `[inline], [virtual]`

virtual fonction from superclass, usefull to get the exception description

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

- HashException.hpp

5.3 Knot Class Reference

class for knots of a tree

```
#include <tree.hpp>
```

Public Member Functions

- Knot (const Knot< T > &other) _tag(other._tag)
- _children (other._children) Knot< T >(T data)
- ∼Knot ()
- Knot< T > & operator= (Knot< T > &other)
- operator== (const Knot< T > &lhs, const Knot< T > &rhs)
- bool operator!= (const Knot< T > &lhs, const Knot< T > &rhs)
- bool isLeaf ()
- int height ()
- void append (< T > n_data)
- void remove (< T > data)
- string toString ()

  get a representation of the knot

5.3.1 Detailed Description

class for knots of a tree

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Knot::Knot (const Knot< T > & *other*)

Copy constructor

Parameters

in	*other*	Knot to copy

5.3.2.2 Knot::∼Knot ( ) [inline]

Destructor for knot

5.3.3 Member Function Documentation

5.3.3.1 Knot::_children (other. *_children*) [inline]

Simple constructor

Parameters

in	*data*	to store into the knot

5.3.3.2 void Knot::append (< T > *n_data*) [inline]

Hook up a new child to the knot

Parameters

in	*n_data*	new data to store as a child of the knot

5.3.3.3 int Knot::height ( ) [inline]

The height of the knot

Parameters

out	*height*	of the knot

5.3.3.4 bool Knot::isLeaf ( ) [inline]

Is the knot a leaf ?

Parameters

out	*true,if*	no child, else false

5.3.3.5 bool Knot::operator!= (const Knot< T > & *lhs*, const Knot< T > & *rhs*) [inline]

inequality operator

Parameters

in	*lhs*	first knot to compare
in	*rhs*	second knot to compare
out	*true*	if knots have not the same memory adress, else false

5.3.3.6 Knot<T>& Knot::operator= (Knot< T > & *other*) [inline]

assignment operator overload

Parameters

in	*other*	knot to assign
out	*assigned*	knot

5.3.3.7 Knot::operator== (const Knot< T > & *lhs*, const Knot< T > & *rhs*) [inline]

equality operator

Parameters

in	*lhs*	left hand side, first knot to compare
in	*rhs*	right hand side, second knot to compare
out	*true*	if knots have the same memory adress, else false

5.3.3.8 void Knot::remove (< T > *data*) [inline]

Remove a leaf from the knot

Parameters

in	*data*	data of the knot's tag to remove

5.3.3.9 string Knot::toString ( )

get a representation of the knot

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

- tree.hpp

5.4 Tree Class Reference

class for the tree, use Knot

```
#include <tree.hpp>
```

Public Member Functions

- Tree (const Tree< T > &)

    copy constructor
- Tree ()

    common constructor
- ~Tree ()

    destructor
- Tree< T > & operator= (Tree< T >)

    assignment operator
- bool operator== (const Tree< T > &, const Tree< T > &)

    equal operator
- bool operator!= (const Tree< T > &, const Tree< T > &)

    ne operator
- bool contains (T)

    Is the element in the tree ?
- int count (T)

    count among of appearances of a particular element
- int height ()

    the height of the tree
- void add (T)

    add an element in the tree
- void remove (T)

    remove an element from the tree
- T[] elements ()

    get the whole list of elements in the tree

5.4.1 Detailed Description

class for the tree, use Knot

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Tree::Tree (const Tree< T > &)

copy constructor

5.4.2.2 Tree::Tree ( )

common constructor

5.4.2.3 Tree::∼Tree ( )

destructor

5.4.3 Member Function Documentation

5.4.3.1 void Tree::add (T)

add an element in the tree

5.4.3.2 bool Tree::contains (T)

Is the element in the tree ?

5.4.3.3 int Tree::count (T)

count among of appearances of a particular element

5.4.3.4 T [ ] Tree::elements ( )

get the whole list of elements in the tree

5.4.3.5 int Tree::height ( )

the height of the tree

5.4.3.6 bool Tree::operator!= (const Tree< T > & , const Tree< T > &)

ne operator

5.4.3.7 Tree<T>& Tree::operator= (Tree< T >)

assignment operator

5.4.3.8 bool Tree::operator== (const Tree< T > & , const Tree< T > &)

equal operator

5.4.3.9 void Tree::remove (T)

remove an element from the tree

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

- tree.hpp

Chapter 6

File Documentation

6.1 HashException.hpp File Reference

```
#include <string>
```

Classes

- class HashException

6.1.1 Detailed Description

6.1.2 File description

Exception class for hash classes.

6.1.3 Copyright

6.1.4 File informations

$Date$ 2014/03/27 $Rev$ 0.1 $Author$ Benjamin Sientzoff $URL$ `http://www.github.com/blasterbug`

6.2 HashTable.hpp File Reference

```
#include <string>
#include "HashException.hpp"
```

Classes

- class Alveole

Macros

- #define END 0

6.2.1 Detailed Description

6.2.2 File description

data structure to store pair in a table a hashcode is compute with k to evaluate the suitable place to store the pair

!! WARNING: int hashCode(K key) must be implemented !!

6.2.3 Copyright

6.2.4 File informations

$Date$ 2014/03/27 Rev 0.1 $Author$ Benjamin Sientzoff URL http://www.github.com/blasterbug

6.2.5 Macro Definition Documentation

6.2.5.1 #define END 0

6.3 README.md File Reference

6.4 tree.hpp File Reference

```
#include <string>
#include <list>
```

Classes

- class Knot

  class for knots of a tree

- class Tree

  class for the tree, use Knot

Index