

# Glossygloss

0.2

Generated by Doxygen 1.8.6

Thu Apr 3 2014 19:35:47



# Contents

<b>1</b>	<b>Main Page</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	Alveole< K, V > Class Template Reference . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Constructor & Destructor Documentation . . . . .	9
5.1.2.1	Alveole . . . . .	9
5.1.2.2	Alveole . . . . .	10
5.1.2.3	Alveole . . . . .	10
5.1.2.4	Alveole . . . . .	10
5.1.2.5	~Alveole . . . . .	10
5.1.3	Member Function Documentation . . . . .	10
5.1.3.1	getKey . . . . .	10
5.1.3.2	getNext . . . . .	10
5.1.3.3	getValue . . . . .	10
5.1.3.4	setNext . . . . .	11
5.1.3.5	setValue . . . . .	11
5.1.3.6	toString . . . . .	11
5.1.4	Member Data Documentation . . . . .	11
5.1.4.1	_key . . . . .	11
5.1.4.2	_next . . . . .	11
5.1.4.3	_value . . . . .	11
5.2	Dictionnaire< V > Class Template Reference . . . . .	11
5.2.1	Constructor & Destructor Documentation . . . . .	12

5.2.1.1	Dictionnaire	12
5.2.1.2	~Dictionnaire	12
5.2.2	Member Function Documentation	12
5.2.2.1	ajouterMot	12
5.2.2.2	associerMot	12
5.2.2.3	contientMot	12
5.2.2.4	supprimerMot	13
5.2.2.5	valeurAssociee	13
5.2.3	Member Data Documentation	13
5.2.3.1	dico	13
5.3	HashException Class Reference	13
5.3.1	Detailed Description	14
5.3.2	Constructor & Destructor Documentation	14
5.3.2.1	HashException	14
5.3.2.2	~HashException	14
5.3.3	Member Function Documentation	14
5.3.3.1	what	14
5.3.4	Member Data Documentation	14
5.3.4.1	_cause	14
5.4	Hashtable< K, V > Class Template Reference	14
5.4.1	Detailed Description	15
5.4.2	Constructor & Destructor Documentation	15
5.4.2.1	Hashtable	15
5.4.2.2	~Hashtable	15
5.4.3	Member Function Documentation	15
5.4.3.1	contains	15
5.4.3.2	get	15
5.4.3.3	isEmpty	15
5.4.3.4	put	16
5.4.3.5	remove	16
5.4.3.6	toString	16
5.4.4	Member Data Documentation	16
5.4.4.1	_table	16
5.5	Node< T > Class Template Reference	16
5.5.1	Detailed Description	17
5.5.2	Constructor & Destructor Documentation	17
5.5.2.1	Node	17
5.5.2.2	Node	17
5.5.2.3	~Node	17
5.5.3	Member Function Documentation	17

5.5.3.1	<a href="#">append</a>	17
5.5.3.2	<a href="#">contains</a>	18
5.5.3.3	<a href="#">getTag</a>	18
5.5.3.4	<a href="#">height</a>	18
5.5.3.5	<a href="#">isLeaf</a>	18
5.5.3.6	<a href="#">operator!=</a>	18
5.5.3.7	<a href="#">operator=</a>	18
5.5.3.8	<a href="#">operator==</a>	19
5.5.3.9	<a href="#">remove</a>	19
5.5.3.10	<a href="#">toString</a>	19
5.5.4	<a href="#">Member Data Documentation</a>	19
5.5.4.1	<a href="#">_children</a>	19
5.5.4.2	<a href="#">_tag</a>	19
5.6	<a href="#">Tree&lt; T &gt; Class Template Reference</a>	19
5.6.1	<a href="#">Detailed Description</a>	20
5.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	20
5.6.2.1	<a href="#">Tree</a>	20
5.6.2.2	<a href="#">Tree</a>	20
5.6.2.3	<a href="#">Tree</a>	20
5.6.2.4	<a href="#">~Tree</a>	20
5.6.3	<a href="#">Member Function Documentation</a>	20
5.6.3.1	<a href="#">contains</a>	20
5.6.3.2	<a href="#">height</a>	21
5.6.3.3	<a href="#">put</a>	21
5.6.3.4	<a href="#">remove</a>	21
5.6.4	<a href="#">Member Data Documentation</a>	21
5.6.4.1	<a href="#">_root</a>	21
5.7	<a href="#">TreeException Class Reference</a>	21
5.7.1	<a href="#">Detailed Description</a>	22
5.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	22
5.7.2.1	<a href="#">TreeException</a>	22
5.7.2.2	<a href="#">~TreeException</a>	22
5.7.3	<a href="#">Member Function Documentation</a>	22
5.7.3.1	<a href="#">what</a>	22
5.7.4	<a href="#">Member Data Documentation</a>	22
5.7.4.1	<a href="#">_cause</a>	22
<b>6</b>	<b><a href="#">File Documentation</a></b>	<b>23</b>
6.1	<a href="#">README.md File Reference</a>	23
6.2	<a href="#">src/application.cpp File Reference</a>	23

6.2.1	Function Documentation	23
6.2.1.1	computehash< string >	23
6.2.1.2	main	23
6.3	src/dictionnaire/dictionnaire.hpp File Reference	23
6.3.1	Detailed Description	23
6.3.2	File description	23
6.3.3	Copyright	24
6.3.4	File informations	24
6.3.5	File description	24
6.3.6	Copyright	24
6.3.7	File informations	24
6.4	src/hashtable.hpp File Reference	24
6.4.1	Detailed Description	25
6.4.2	File description	25
6.4.3	Copyright	25
6.4.4	File informations	25
6.4.5	Macro Definition Documentation	25
6.4.5.1	ARRAYSIZE	25
6.4.5.2	END	25
6.4.6	Function Documentation	26
6.4.6.1	computehash	26
6.5	src/test_tree.cpp File Reference	27
6.5.1	Detailed Description	27
6.5.2	File description	27
6.5.3	Copyright	27
6.5.4	File informations	27
6.5.5	Macro Definition Documentation	27
6.5.5.1	K	27
6.5.6	Function Documentation	27
6.5.6.1	main	28
6.6	src/tree.hpp File Reference	28
6.6.1	Detailed Description	28
6.6.2	File description	28
6.6.3	Copyright	28
6.6.4	File informations	28
<b>Index</b>		<b>29</b>

# Chapter 1

## Main Page

Glossygloss is set of classes to use several data structures as [Tree](#) and [Hashtable](#). More might come soon.

### Documentation

All documented things are [here](#).

A PDF file *refman.pdf* is also available for offline doc.

You can generate the doc using [doxygen](#) and the config file *doxygen\_config*

Usefull links :

- [C++ programming on wikibooks](#)
- what else ?

### Copyright

This source code is protected by the French intellectual property law.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 of the License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Alveole< K, V > . . . . .	9
Alveole< string, V > . . . . .	9
Dictionnaire< V > . . . . .	11
exception	
HashException . . . . .	13
TreeException . . . . .	21
Hashtable< K, V > . . . . .	14
Hashtable< string, V > . . . . .	14
Node< T > . . . . .	16
Tree< T > . . . . .	19



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">Alveole&lt; K, V &gt;</a>	
Class to define <a href="#">Hashtable</a> alveoles . . . . .	9
<a href="#">Dictionnaire&lt; V &gt;</a> . . . . .	11
<a href="#">HashException</a>	
Exception class to manage <a href="#">Hashtable</a> errors . . . . .	13
<a href="#">Hashtable&lt; K, V &gt;</a>	
Maps a key to a value . . . . .	14
<a href="#">Node&lt; T &gt;</a>	
Defines tree nodes . . . . .	16
<a href="#">Tree&lt; T &gt;</a>	
<a href="#">Tree</a> is a recursive structure using nodes . . . . .	19
<a href="#">TreeException</a>	
Exception class for trees . . . . .	21



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">application.cpp</a> . . . . .	23
src/ <a href="#">hashtable.hpp</a> . . . . .	24
src/ <a href="#">test_tree.cpp</a> . . . . .	27
src/ <a href="#">tree.hpp</a> . . . . .	28
src/dictionnaire/ <a href="#">dictionnaire.hpp</a> . . . . .	23



## Chapter 5

# Class Documentation

### 5.1 Alveole< K, V > Class Template Reference

Class to define [Hashtable](#) alveoles.

```
#include <hashtable.hpp>
```

#### Public Member Functions

- [Alveole](#) (const [Alveole](#)< [K](#), [V](#) > &other)
- [Alveole](#) ([K](#) key, [V](#) value)
- [Alveole](#) ()
- [Alveole](#) ([K](#) key, [V](#) value, [Alveole](#)< [K](#), [V](#) > \*next)
- [~Alveole](#) ()
- [K](#) getKey ()
- [V](#) getValue ()
- [Alveole](#)< [K](#), [V](#) > \* getNext ()
- void setValue ([V](#) n\_value)
- void setNext ([Alveole](#)< [K](#), [V](#) > \*n\_next)
- string toString ()

#### Private Attributes

- [K](#) \_key
- [V](#) \_value
- [Alveole](#)< [K](#), [V](#) > \* \_next

#### 5.1.1 Detailed Description

```
template<typename K, typename V>class Alveole< K, V >
```

Class to define [Hashtable](#) alveoles.

[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 `template<typename K, typename V> Alveole< K, V >::Alveole ( const Alveole< K, V > & other )`

next aveole Copy constructor

## Parameters

in	<i>other</i>	the alveole to copy
----	--------------	---------------------

5.1.2.2 `template<typename K, typename V> Alveole< K, V >::Alveole ( K key, V value )`

## Pair constructor

## Parameters

in	<i>key</i>	key of the pair
in	<i>value</i>	value of the pair

5.1.2.3 `template<typename K, typename V> Alveole< K, V >::Alveole ( )`

Empty constructor create an 'empty' alveole

5.1.2.4 `template<typename K, typename V> Alveole< K, V >::Alveole ( K key, V value, Alveole< K, V > * next )`

## Complex constructor

## Parameters

in	<i>key</i>	key of the pair
in	<i>value</i>	value of the pair
in	<i>next</i>	adresse to the next alveole

5.1.2.5 `template<typename K, typename V> Alveole< K, V >::~~Alveole ( )`

Destructor for [Alveole](#)

## 5.1.3 Member Function Documentation

5.1.3.1 `template<typename K, typename V> K Alveole< K, V >::getKey ( )`

Get the key of an alveole

## Parameters

out	<i>key</i>	key stored into the alveole
-----	------------	-----------------------------

5.1.3.2 `template<typename K, typename V> Alveole<K,V>* Alveole< K, V >::getNext ( )`

Which alveole coming next ?

## Parameters

out	<i>ptr</i>	memory adress of the next alveole
-----	------------	-----------------------------------

5.1.3.3 `template<typename K, typename V> V Alveole< K, V >::getValue ( )`

Get the value stored into an alveole



## Parameters

out	<i>value</i>	value of the alveole
-----	--------------	----------------------

5.1.3.4 `template<typename K, typename V> void Alveole< K, V >::setNext ( Alveole< K, V > * n_next )`

Set the next address of the next alveole

## Parameters

in	<i>n_next</i>	adress of the new next alveole
----	---------------	--------------------------------

5.1.3.5 `template<typename K, typename V> void Alveole< K, V >::setValue ( V n_value )`

Set the value stored into an alveole

## Parameters

in	<i>n_value</i>	The new value of the pair
----	----------------	---------------------------

5.1.3.6 `template<typename K, typename V> string Alveole< K, V >::toString ( )`

Return a string description of the pair stored into the alveole

## Parameters

out	<i>desc</i>	a string representation of the alveole
-----	-------------	--

## 5.1.4 Member Data Documentation

5.1.4.1 `template<typename K, typename V> K Alveole< K, V >::_key [private]`

5.1.4.2 `template<typename K, typename V> Alveole<K,V>* Alveole< K, V >::_next [private]`

value of the pair

5.1.4.3 `template<typename K, typename V> V Alveole< K, V >::_value [private]`

key of the pair

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

- [src/hashtable.hpp](#)

## 5.2 Dictionnaire&lt; V &gt; Class Template Reference

```
#include <dictionnaire.hpp>
```

## Public Member Functions

- [Dictionnaire](#) ()
- [~Dictionnaire](#) ()
- bool [contientMot](#) (string mot)

- void [ajouterMot](#) (string mot, V v)
- void [associerMot](#) (string mot, V v)
- void [supprimerMot](#) (string mot)
- V [valeurAssociee](#) (string mot)

### Private Attributes

- [Hashtable](#)< string, V > [dico](#)

## 5.2.1 Constructor & Destructor Documentation

### 5.2.1.1 `template<typename V > Dictionnaire< V >::Dictionnaire ( )`

Constructeur de la classe [Dictionnaire](#)

### 5.2.1.2 `template<typename V > Dictionnaire< V >::~~Dictionnaire ( )`

Destructeur de la classe [Dictionnaire](#)

## 5.2.2 Member Function Documentation

### 5.2.2.1 `template<typename V > void Dictionnaire< V >::ajouterMot ( string mot, V v )`

Fonction qui ajoute un mot non présent dans le dictionnaire

Parameters

in	<i>mot, le</i>	mot à ajouter
in	<i>v, la</i>	valeur associée

### 5.2.2.2 `template<typename V > void Dictionnaire< V >::associerMot ( string mot, V v )`

Fonction qui modifie la valeur d'un mot présent dans le dictionnaire

Parameters

in	<i>mot, le</i>	mot à modifier
in	<i>v, la</i>	valeur à modifier

Exceptions

<i>lève</i>	une exception si le mot n'est pas présent
-------------	---

### 5.2.2.3 `template<typename V > bool Dictionnaire< V >::contientMot ( string mot )`

Fonction qui renvoie vrai le mot est présent dans le [Dictionnaire](#)

Parameters

in	<i>mot, le</i>	mot à tester
----	----------------	--------------

out	<i>bool,vrai</i>	si présent, faux sinon.
-----	------------------	-------------------------

#### 5.2.2.4 `template<typename V > void Dictionnaire< V >::supprimerMot ( string mot )`

Fonction qui supprime un mot présent dans le dictionnaire

Parameters

in	<i>mot,le</i>	mot à supprimer
----	---------------	-----------------

Exceptions

<i>lève</i>	une exception si le mot n'est pas présent
-------------	---

#### 5.2.2.5 `template<typename V > V Dictionnaire< V >::valeurAssociee ( string mot )`

Fonction qui récupère la valeur associée au mot

Parameters

in	<i>mot,le</i>	mot dont on souhaite savoir la valeur associée
out	<i>valeur,la</i>	valeur associée

Exceptions

<i>lève</i>	une exception si le mot n'est pas présent dans le dictionnaire
-------------	--

### 5.2.3 Member Data Documentation

#### 5.2.3.1 `template<typename V > Hashtable<string,V> Dictionnaire< V >::dico [private]`

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

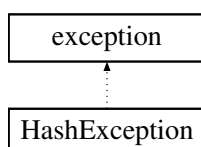
- [src/dictionnaire/dictionnaire.hpp](#)

## 5.3 HashException Class Reference

Exception class to manage [Hashtable](#) errors.

```
#include <hashtable.hpp>
```

Inheritance diagram for HashException:



### Public Member Functions

- [HashException](#) (const char \*cause)
- virtual [~HashException](#) () throw ()
- virtual const char \* [what](#) () const throw ()

## Private Attributes

- `const char * \_cause`

### 5.3.1 Detailed Description

Exception class to manage [Hashtable](#) errors.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 `HashException::HashException ( const char * cause )`

store exception description constructor called then HashExceptions are threw

Parameters

<code>in</code>	<code><i>cause</i></code>	description of exception origin
-----------------	---------------------------	---------------------------------

#### 5.3.2.2 `virtual HashException::~HashException ( ) throw )` `[virtual]`

destructor currently, do anything special

### 5.3.3 Member Function Documentation

#### 5.3.3.1 `virtual const char* HashException::what ( ) const throw )` `[virtual]`

virtual fonction from superclass, usefull to get the exception description

### 5.3.4 Member Data Documentation

#### 5.3.4.1 `const char* HashException::_cause` `[private]`

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

- `src/hashtable.hpp`

## 5.4 `Hashtable< K, V >` Class Template Reference

Maps a key to a value.

```
#include <hashtable.hpp>
```

## Public Member Functions

- [Hashtable](#) ()
- [~Hashtable](#) ()
- `bool` [contains](#) (const `K` &key)
- `V` [get](#) (const `K` &key)
- `bool` [isEmpty](#) ()
- `void` [put](#) (`K` key, `V` value)
- `void` [remove](#) (const `K` &key)

*FIXME : remove last element of a list lead to a seg. fault.*

- string [toString](#) ()

## Private Attributes

- [Alveole](#)< K, V > \*\* [\\_table](#)

### 5.4.1 Detailed Description

```
template<typename K, typename V>class Hashtable< K, V >
```

Maps a key to a value.

### 5.4.2 Constructor & Destructor Documentation

5.4.2.1 `template<typename K, typename V> Hashtable< K, V >::Hashtable ( )`

array of alveoles Simple constructor

5.4.2.2 `template<typename K, typename V> Hashtable< K, V >::~~Hashtable ( )`

Destructor

### 5.4.3 Member Function Documentation

5.4.3.1 `template<typename K, typename V> bool Hashtable< K, V >::contains ( const K & key )`

Do table contains key ?

Parameters

in	<i>key</i>	key to find
out	<i>bool</i>	True if the key is here, else false

5.4.3.2 `template<typename K, typename V> V Hashtable< K, V >::get ( const K & key )`

Return the value mapped to the specified key

Parameters

in	<i>key</i>	a key in the hashtable
out	<i>value</i>	value associated with the key

Exceptions

<a href="#">HashException</a>	threw if key is not in the hashtable
-------------------------------	--------------------------------------

5.4.3.3 `template<typename K, typename V> bool Hashtable< K, V >::isEmpty ( )`

Tests if this hashtable maps no keys to values.

## Parameters

out	<i>bool</i>	true if no elements in the hashtable, else false;
-----	-------------	---

5.4.3.4 `template<typename K, typename V> void Hashtable< K, V >::put ( K key, V value )`

Map the specified key to the specified value in this hashtable. or update the mapped value to the key

## Parameters

in	<i>key</i>	key of the pair
in	<i>value</i>	value of the pair

5.4.3.5 `template<typename K, typename V> void Hashtable< K, V >::remove ( const K & key )`

FIXME : remove last element of a list lead to a seg. fault.

Remove the key (and its corresponding value) from this hashtable.

## Parameters

in	<i>key</i>	Key of the pair to delete
----	------------	---------------------------

## Exceptions

<a href="#"><i>HashException</i></a>	threw if table does not contain key
--------------------------------------	-------------------------------------

5.4.3.6 `template<typename K, typename V> string Hashtable< K, V >::toString ( )`

Return a description of the hashtable, enclosed in braces as well as {key, value}.

## Parameters

out	<i>desc</i>	a string representation of this hashtable.
-----	-------------	--

## 5.4.4 Member Data Documentation

5.4.4.1 `template<typename K, typename V> Alveole<K,V>** Hashtable< K, V >::_table [private]`

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

- [src/hashtable.hpp](#)

5.5 `Node< T >` Class Template Reference

Defines tree nodes.

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

## Public Member Functions

- [Node](#) (const [Node](#)< T > &other)
- [Node](#) (T data)
- [~Node](#) ()

- `Node< T > & operator= (Node< T > &other)`
- `bool operator== (const Node< T > &rhs)`
- `bool operator!= (const Node< T > &rhs)`
- `bool isLeaf ()`
- `int height ()`
- `void append (T n_data)`
- `void remove (T data)`
- `T getTag ()`
- `bool contains (T element)`
- `string toString ()`

### Private Attributes

- `T _tag`  
*letter stored into `Node`, the tag*
- `forward_list< Node< T > > _children`  
*children of the `Node`*

### 5.5.1 Detailed Description

`template<typename T = char> class Node< T >`

Defines tree nodes.

Class for nodes of a tree. A `Node` store a tag and can have several children

### 5.5.2 Constructor & Destructor Documentation

5.5.2.1 `template<typename T = char> Node< T >::Node ( const Node< T > & other )`

Copy constructor

Parameters

<code>in</code>	<code>other</code>	<code>Node</code> to copy
-----------------	--------------------	---------------------------

5.5.2.2 `template<typename T = char> Node< T >::Node ( T data )`

Simple constructor

Parameters

<code>in</code>	<code>data</code>	to store into the <code>Node</code>
-----------------	-------------------	-------------------------------------

5.5.2.3 `template<typename T = char> Node< T >::~~Node ( )`

Destructor for `Node`

### 5.5.3 Member Function Documentation

5.5.3.1 `template<typename T = char> void Node< T >::append ( T n_data )`

Hook up a new child to the node

## Parameters

in	<i>n_data</i>	new data to store as a child of the node
----	---------------	--

5.5.3.2 `template<typename T = char> bool Node< T >::contains ( T element )`

Do the tag is element or one of his children ?

## Parameters

in	<i>element</i>	Element to look for
out	<i>bool</i>	True if node or one of his child has the right tag, else false.

5.5.3.3 `template<typename T = char> T Node< T >::getTag ( )`

What is the tag of the [Node](#) ?

## Parameters

out	<i>tag</i>	The tag of the node
-----	------------	---------------------

5.5.3.4 `template<typename T = char> int Node< T >::height ( )`

The height of the node

## Parameters

out	<i>hgt</i>	height of the node
-----	------------	--------------------

5.5.3.5 `template<typename T = char> bool Node< T >::isLeaf ( )`

Is the node a leaf ?

## Parameters

out	<i>bool</i>	true, if no child, else false
-----	-------------	-------------------------------

5.5.3.6 `template<typename T = char> bool Node< T >::operator!= ( const Node< T > & rhs )`

inequality operator

## Parameters

in	<i>lhs</i>	first node to compare
in	<i>rhs</i>	second node to compare
out	<i>bool</i>	true if nodes have not the same memory adress, else false

5.5.3.7 `template<typename T = char> Node<T> & Node< T >::operator= ( Node< T > & other )`

assignment operator overload



## Parameters

in	<i>other</i>	node to assign
out	<i>note</i>	assigned node

5.5.3.8 `template<typename T = char> bool Node< T >::operator==( const Node< T > & rhs )`

equality operator

## Parameters

in	<i>lhs</i>	left hand side, first node to compare
in	<i>rhs</i>	right hand side, second node to compare
out	<i>bool</i>	true if nodes have the same memory adress, else false

5.5.3.9 `template<typename T = char> void Node< T >::remove ( T data )`

Remove a leaf from the node

## Parameters

in	<i>data</i>	data of the node's tag to remove
----	-------------	----------------------------------

## Exceptions

<a href="#"><i>TreeException</i></a>	Threw if data is not removed
--------------------------------------	------------------------------

5.5.3.10 `template<typename T = char> string Node< T >::toString ( )`

Get a string representation of the node and his child

## Parameters

out	<i>desc</i>	Description of the node (and his child)
-----	-------------	---

## 5.5.4 Member Data Documentation

5.5.4.1 `template<typename T = char> forward_list<Node<T> > Node< T >::_children [private]`

children of the [Node](#)

5.5.4.2 `template<typename T = char> T Node< T >::_tag [private]`

letter stored into [Node](#), the tag

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

- [src/tree.hpp](#)

## 5.6 Tree&lt; T &gt; Class Template Reference

[Tree](#) is a recursive structure using nodes.

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

## Public Member Functions

- [Tree](#) ()
- [Tree](#) (const [Tree](#)< T > &other)
- [Tree](#) (T element)
- [~Tree](#) ()
- bool [contains](#) (T element)
- int [height](#) ()
- void [put](#) (T element)
- void [remove](#) (T element)

## Private Attributes

- [Node](#)< T > [\\_root](#)

### 5.6.1 Detailed Description

`template<typename T = string> class Tree< T >`

[Tree](#) is a recursive structure using nodes.

A root value and subtrees of children, represented as a set of linked nodes.

### 5.6.2 Constructor & Destructor Documentation

5.6.2.1 `template<typename T = string> Tree< T >::Tree ( )`

First node of the tree Default constructor

5.6.2.2 `template<typename T = string> Tree< T >::Tree ( const Tree< T > & other )`

Copy constructor

5.6.2.3 `template<typename T = string> Tree< T >::Tree ( T element )`

Common constructor, create an tree

Parameters

<code>in</code>	<code>element</code>	Root of the tree
-----------------	----------------------	------------------

5.6.2.4 `template<typename T = string> Tree< T >::~~Tree ( )`

Destructor, destroy the whole tree

### 5.6.3 Member Function Documentation

5.6.3.1 `template<typename T = string> bool Tree< T >::contains ( T element )`

Is the element in the tree ?

## Parameters

in	<i>element</i>	Search the element in the <a href="#">Tree</a>
out	<i>bool</i>	True if element is here, else false.

5.6.3.2 `template<typename T = string> int Tree< T >::height ( )`

The height of the tree

## Parameters

out	<i>hgt</i>	Height of the tree
-----	------------	--------------------

5.6.3.3 `template<typename T = string> void Tree< T >::put ( T element )`

Put an element in the tree

## Parameters

in	<i>element</i>	New element to put into the tree
----	----------------	----------------------------------

5.6.3.4 `template<typename T = string> void Tree< T >::remove ( T element )`

Remove an element from the tree

## Parameters

in	<i>data</i>	Element to remove
----	-------------	-------------------

## 5.6.4 Member Data Documentation

5.6.4.1 `template<typename T = string> Node<T> Tree< T >::_root [private]`

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

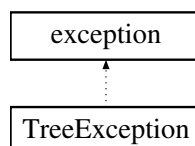
- [src/tree.hpp](#)

## 5.7 TreeException Class Reference

exception class for trees

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

Inheritance diagram for TreeException:



## Public Member Functions

- [TreeException](#) (char \*cause)

- virtual [~TreeException](#) () throw ()
- virtual const char \* [what](#) () const throw ()

## Private Attributes

- char \* [\\_cause](#)

### 5.7.1 Detailed Description

exception class for trees

Usefull to manage errors and the unforeseen

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 `TreeException::TreeException ( char * cause )`

store exception description constructor called then TreeExceptions are threw

Parameters

<code>in</code>	<code><i>cause</i></code>	description of exception origin
-----------------	---------------------------	---------------------------------

#### 5.7.2.2 `virtual TreeException::~~TreeException ( ) throw )` `[virtual]`

destructor currently, do anything special

### 5.7.3 Member Function Documentation

#### 5.7.3.1 `virtual const char* TreeException::what ( ) const throw )` `[virtual]`

virtual fonction from superclass, usefull to get the exception description

### 5.7.4 Member Data Documentation

#### 5.7.4.1 `char* TreeException::_cause` `[private]`

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

- [src/tree.hpp](#)

## Chapter 6

# File Documentation

### 6.1 README.md File Reference

### 6.2 src/application.cpp File Reference

```
#include <functional>
#include <iostream>
#include <fstream>
#include "dictionnaire/dictionnaire.hpp"
```

#### Functions

- `template<> unsigned computehash< string > (string element)`
- `int main ()`

#### 6.2.1 Function Documentation

6.2.1.1 `template<> unsigned computehash< string > ( string element )`

6.2.1.2 `int main ( )`

### 6.3 src/dictionnaire/dictionnaire.hpp File Reference

```
#include "../hashtable.hpp"
```

#### Classes

- `class Dictionnaire< V >`

#### 6.3.1 Detailed Description

#### 6.3.2 File description

Classe Test utilisant les deux implémentations du dictionnaire

### 6.3.3 Copyright

This source code is protected by the French intellectual property law.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 of the License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

### 6.3.4 File informations

\$Date\$ 2014/03/27 \$Rev\$ 0.2 \$Author\$ Benjamin Sientzoff & François Hallereau \$URL\$ <http://www.-github.com/blasterbug>

### 6.3.5 File description

[Dictionnaire](#) utilisant une hashtable

### 6.3.6 Copyright

This source code is protected by the French intellectual property law.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 of the License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

### 6.3.7 File informations

\$Date\$ 2014/03/27 \$Rev\$ 0.2 \$Author\$ Benjamin Sientzoff & François Hallereau \$URL\$ <http://www.-github.com/blasterbug>

## 6.4 src/hashtable.hpp File Reference

```
#include <string>
#include <cassert>
```

### Classes

- class [HashException](#)  
*Exception class to manage [Hashtable](#) errors.*
- class [Alveole](#)< K, V >  
*Class to define [Hashtable](#) alveoles.*

- class `Hashtable< K, V >`  
*Maps a key to a value.*

## Macros

- `#define END 0`  
*macro to define end of alveole chains*
- `#define ARRAYSIZE 10`  
*macro to define size of hash arrays*

## Functions

- `template<typename K >`  
`unsigned computehash (K element)`

### 6.4.1 Detailed Description

### 6.4.2 File description

data structure to store pairs 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 !!

### 6.4.3 Copyright

This source code is protected by the French intellectual property law.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 of the License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

### 6.4.4 File informations

\$Date\$ 2014/03/27 \$Rev\$ 0.2 \$Author\$ Benjamin Sientzoff \$URL\$ <http://www.github.com/blasterbug>

### 6.4.5 Macro Definition Documentation

#### 6.4.5.1 `#define ARRAYSIZE 10`

macro to define size of hash arrays

#### 6.4.5.2 `#define END 0`

macro to define end of alveole chains

## 6.4.6 Function Documentation

### 6.4.6.1 `template<typename K > unsigned computehash ( K element )`

Fonction you must define



**Parameters**

in	<i>element</i>	element to compute hashcode from
out	<i>hashcode</i>	the hashcode of element, an unsigned integer

template<> unsigned [computehash](#)<[string](#)>([string element](#)) { your implementation of hashcode function }

**6.5 src/test\_tree.cpp File Reference**

```
#include <functional>
#include <iostream>
#include <cstdlib>
#include <fstream>
#include "tree.hpp"
```

**Macros**

- `#define` [K](#) string

**Functions**

- int [main](#) (int argc, const char \*\*argv)

**6.5.1 Detailed Description****6.5.2 File description**

File to test tree.cpp classes

**6.5.3 Copyright**

This source code is protected by the French intellectual property law.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 of the License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

**6.5.4 File informations**

\$Date\$ 2014/04/03 \$Rev\$ 0.1 \$Author\$ Benjamin Sientzoff \$URL\$ <http://www.github.com/blasterbug>

**6.5.5 Macro Definition Documentation****6.5.5.1 #define K string****6.5.6 Function Documentation**

6.5.6.1 `int main ( int argc, const char ** argv )`

## 6.6 `src/tree.hpp` File Reference

```
#include <cassert>
#include <string>
#include <forward_list>
```

### Classes

- class `TreeException`  
*exception class for trees*
- class `Node< T >`  
*Defines tree nodes.*
- class `Tree< T >`  
*Tree is a recursive structure using nodes.*

### 6.6.1 Detailed Description

### 6.6.2 File description

Implémentation d'un arbre pour stocker des mots. Chaque noeud stocke une lettre.

### 6.6.3 Copyright

This source code is protected by the French intellectual property law.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 of the License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

### 6.6.4 File informations

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

# Index

- ~Alveole
  - Alveole, 10
- ~Dictionnaire
  - Dictionnaire, 12
- ~HashException
  - HashException, 14
- ~Hashtable
  - Hashtable, 15
- ~Node
  - Node, 17
- ~Tree
  - Tree, 20
- ~TreeException
  - TreeException, 22
- \_cause
  - HashException, 14
  - TreeException, 22
- \_children
  - Node, 19
- \_key
  - Alveole, 11
- \_next
  - Alveole, 11
- \_root
  - Tree, 21
- \_table
  - Hashtable, 16
- \_tag
  - Node, 19
- \_value
  - Alveole, 11
- ARRAYSIZE
  - hashtable.hpp, 25
- ajouterMot
  - Dictionnaire, 12
- Alveole
  - ~Alveole, 10
  - \_key, 11
  - \_next, 11
  - \_value, 11
  - Alveole, 9, 10
  - getKey, 10
  - getNext, 10
  - getValue, 10
  - setNext, 11
  - setValue, 11
  - toString, 11
- Alveole< K, V >, 9
- append
  - Node, 17
- application.cpp
  - computehash< string >, 23
  - main, 23
- associerMot
  - Dictionnaire, 12
- computehash
  - hashtable.hpp, 26
- computehash< string >
  - application.cpp, 23
- contains
  - Hashtable, 15
  - Node, 18
  - Tree, 20
- contientMot
  - Dictionnaire, 12
- dico
  - Dictionnaire, 13
- Dictionnaire
  - ~Dictionnaire, 12
  - ajouterMot, 12
  - associerMot, 12
  - contientMot, 12
  - dico, 13
  - Dictionnaire, 12
  - supprimerMot, 13
  - valeurAssociee, 13
- Dictionnaire< V >, 11
- END
  - hashtable.hpp, 25
- get
  - Hashtable, 15
- getKey
  - Alveole, 10
- getNext
  - Alveole, 10
- getTag
  - Node, 18
- getValue
  - Alveole, 10
- HashException, 13
  - ~HashException, 14
  - \_cause, 14
  - HashException, 14
  - HashException, 14
  - what, 14

Hashtable  
   ~Hashtable, 15  
   \_table, 16  
   contains, 15  
   get, 15  
   Hashtable, 15  
   isEmpty, 15  
   put, 16  
   remove, 16  
   toString, 16  
 Hashtable< K, V >, 14  
 hashtable.hpp  
   ARRAYSIZE, 25  
   computehash, 26  
   END, 25  
 height  
   Node, 18  
   Tree, 21  
  
 isEmpty  
   Hashtable, 15  
 isLeaf  
   Node, 18  
  
 K  
   test\_tree.cpp, 27  
  
 main  
   application.cpp, 23  
   test\_tree.cpp, 27  
  
 Node  
   ~Node, 17  
   \_children, 19  
   \_tag, 19  
   append, 17  
   contains, 18  
   getTag, 18  
   height, 18  
   isLeaf, 18  
   Node, 17  
   operator=, 18  
   operator==, 19  
   remove, 19  
   toString, 19  
 Node< T >, 16  
  
 operator=  
   Node, 18  
 operator==  
   Node, 19  
  
 put  
   Hashtable, 16  
   Tree, 21  
  
 README.md, 23  
 remove  
   Hashtable, 16  
   Node, 19  
  
   Tree, 21  
  
 setNext  
   Alveole, 11  
 setValue  
   Alveole, 11  
 src/application.cpp, 23  
 src/dictionnaire/dictionnaire.hpp, 23  
 src/hashtable.hpp, 24  
 src/test\_tree.cpp, 27  
 src/tree.hpp, 28  
 suppresserMot  
   Dictionnaire, 13  
  
 test\_tree.cpp  
   K, 27  
   main, 27  
 toString  
   Alveole, 11  
   Hashtable, 16  
   Node, 19  
 Tree  
   ~Tree, 20  
   \_root, 21  
   contains, 20  
   height, 21  
   put, 21  
   remove, 21  
   Tree, 20  
 Tree< T >, 19  
 TreeException, 21  
   ~TreeException, 22  
   \_cause, 22  
   TreeException, 22  
   TreeException, 22  
   what, 22  
  
 valeurAssociee  
   Dictionnaire, 13  
  
 what  
   HashException, 14  
   TreeException, 22