cuckoo

Generated by Doxygen 1.6.3

Mon May 30 18:53:26 2011

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 cuckoo< Key, Value, Hash, Equal >::const_iterator Class Reference

```
#include <cuckoo.hpp>
```

Public Member Functions

- const_iterator ()
- void operator= (const const_iterator &it)
- const_iterator (const const_iterator &it)
- const_iterator & operator++ ()
- const_iterator operator++ (int)
- const Data & operator* () const
- const Data * operator-> () const
- bool operator== (const const_iterator &it)
- bool operator!= (const const_iterator &it)

Private Member Functions

- const_iterator (const size_t p, const cuckoo *h)

Private Attributes

- size_t pos
- const cuckoo * hash

Friends

- class cuckoo

3.1.1 Detailed Description

template<class Key, class Value, class Hash, class Equal> class cuckoo< Key, Value, Hash, Equal >::const_iterator

Used with const cuckoo objects.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::const_iterator::const_iterator (const size_t *p*, const cuckoo * *h*) `[inline, private]`

3.1.2.2 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::const_iterator::const_iterator () `[inline]`

Default constructor.

3.1.2.3 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::const_iterator::const_iterator (const const_iterator & *it*) `[inline]`

3.1.3 Member Function Documentation

3.1.3.1 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::const_iterator::operator= (const const_iterator & *it*) `[inline]`

3.1.3.2 template<class Key, class Value, class Hash, class Equal> const_iterator& cuckoo< Key, Value, Hash, Equal >::const_iterator::operator++ () `[inline]`

3.1.3.3 template<class Key, class Value, class Hash, class Equal> const_iterator cuckoo< Key, Value, Hash, Equal >::const_iterator::operator++ (int) `[inline]`

3.1.3.4 template<class Key, class Value, class Hash, class Equal> const Data& cuckoo< Key, Value, Hash, Equal >::const_iterator::operator∗ () const `[inline]`

3.1.3.5 template<class Key, class Value, class Hash, class Equal> const Data∗ cuckoo< Key, Value, Hash, Equal >::const_iterator::operator-> () const `[inline]`

3.1.3.6 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::const_iterator::operator== (const const_iterator & *it*) `[inline]`

3.1.3.7 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::const_iterator::operator!= (const const_iterator & *it*) `[inline]`

3.1.4 Friends And Related Function Documentation

3.1.4.1 template<class Key, class Value, class Hash, class Equal> friend class cuckoo `[friend]`

3.1.5 Member Data Documentation

3.1.5.1 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::const_iterator::pos `[private]`

3.1.5.2 template<class Key, class Value, class Hash, class Equal> const cuckoo∗ cuckoo< Key, Value, Hash, Equal >::const_iterator::hash `[private]`

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

- cuckoo.hpp

3.2 cuckoo< Key, Value, Hash, Equal > Class Template Reference

```
#include <cuckoo.hpp>
```

Classes

- class const_iterator
- class iterator

Public Member Functions

- cuckoo (size_t d=D, size_t init_length=INIT_LENGTH, size_t max_loop=MAX_LOOP, double step=STEP, const Hash &hasher=Hash(), const Equal &equal=Equal())
- ~cuckoo ()
- cuckoo< Key, Value, Hash, Equal > & operator= (const cuckoo< Key, Value, Hash, Equal > &Cuckoo)
- cuckoo (const cuckoo< Key, Value, Hash, Equal > &Cuckoo)
- template<class InputIterator >
  cuckoo (InputIterator first, InputIterator last, const Hash &hasher=Hash(), const Equal &equal=Equal())
- void set_up (size_t d=D, size_t init_length=INIT_LENGTH, size_t max_loop=MAX_LOOP, double step=STEP)
- bool operator== (const cuckoo< Key, Value, Hash, Equal > &Cuckoo)
- bool operator!= (const cuckoo< Key, Value, Hash, Equal > &Cuckoo)
- void swap (cuckoo< Key, Value, Hash, Equal > &Cuckoo)
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- Value & operator[ ] (const Key &k)
- void erase (iterator it)
- void erase (iterator first, iterator last)
- size_t erase (const Key &k)
- iterator find (const Key &k)
- const_iterator find (const Key &k) const
- size_t count (const Key &k) const
- pair< iterator, iterator > equal_range (const Key &k)
- pair< const_iterator, const_iterator > equal_range (const Key &k) const
- pair< iterator, bool > insert (const Data &k)
- template<class InputIterator >
  void insert (InputIterator first, InputIterator last)
- void clear ()
- bool empty () const
- size_t size () const
- size_t length () const

Private Types

- typedef pair< Key, Value > Data

Private Member Functions

- bool get_exists (size_t pos) const
- void set_exists (size_t pos)
- void unset_exists (size_t pos)
- void init ()
- void copy (const cuckoo< Key, Value, Hash, Equal > &Cuckoo)
- void clear_all ()
- Data & data_from (size_t pos) const
- bool is_here (const Key &k, size_t pos) const
- size_t hash (const Key &k, size_t hash_num) const
- void update_exists (size_t len_temp_)
- void update_data (size_t len_temp_)
- void rehash ()
- size_t add_new (Data p)
- iterator remove (iterator &it)

Private Attributes

- size_t d_
- size_t init_length_
- size_t max_loop_
- double step_
- Hash hasher_
- Equal key_equal_
- Data ** data_
- char * exists_
- size_t len_
- size_t len_part_
- size_t size_
- bool is_rehashed_

Friends

- class iterator
- class const_iterator

template<class Key, class Value, class Hash, class Equal> class cuckoo< Key, Value, Hash, Equal >

3.2.1 Member Typedef Documentation

3.2.1.1 template<class Key, class Value, class Hash, class Equal> typedef pair<Key, Value> cuckoo< Key, Value, Hash, Equal >::Data `[private]`

3.2.2 Constructor & Destructor Documentation

3.2.2.1 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::cuckoo (size_t *d* = `D`, size_t *init_length* = `INIT_LENGTH`, size_t *max_loop* = `MAX_LOOP`, double *step* = `STEP`, const Hash & *hasher* = `Hash()`, const Equal & *equal* = `Equal()`) `[inline, explicit]`

Default constructor.

Parameters

d The number of hash functions (thus arrays also) that will be used in the program (can be >= 2).

init_length The initial length of the whole structure. When you know the approximate number of records to be used, it is a good idea to take this value in 1.05-1.1 times more and small value of step.

max_loop The maximum number of kick cycles during insertion before rehash.

step The ratio of increasing the size of hash during rehash. The less it is the less memory will be used but the more time is needed.

hasher The hash function object (template parameter by default).

equal The equal predicator object (template parameter by default).

3.2.2.2 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::~cuckoo () `[inline]`

Destructor.

3.2.2.3 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::cuckoo (const cuckoo< Key, Value, Hash, Equal > & *Cuckoo*) `[inline]`

Copy constructor.

Parameters

Cuckoo The source of information

3.2.2.4 template<class Key, class Value, class Hash, class Equal> template<class InputIterator > cuckoo< Key, Value, Hash, Equal >::cuckoo (InputIterator *first*, InputIterator *last*, const Hash & *hasher* = `Hash()`, const Equal & *equal* = `Equal()`) `[inline]`

Constructor from range [fisrt, last).

Parameters

first The begin of range iterator.
last The end of range iterator.

3.2.3 Member Function Documentation

3.2.3.1 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::get_exists (size_t *pos*) const `[inline, private]`

Check whether there is a Data element at pos.

Parameters

pos Position in hash arrays.

Returns

true if some element presents on this position, false otherwise.

3.2.3.2 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::set_exists (size_t *pos*) `[inline, private]`

Set flag of existence of Data element at pos.

Parameters

pos Position in hash arrays.

3.2.3.3 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::unset_exists (size_t *pos*) `[inline, private]`

Unset flag of existence Data element at pos.

Parameters

pos Position in hash arrays.

3.2.3.4 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::init () `[inline, private]`

Initialize all the variables of cuckoo.

3.2.3.5 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::copy (const cuckoo< Key, Value, Hash, Equal > & *Cuckoo*) `[inline, private]`

Copy information from Cuckoo to clean object.

Parameters

Object for information to be copied from.

3.2.3.6 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::clear_all () `[inline, private]`

Clear all the data from cuckoo.

3.2.3.7 template<class Key, class Value, class Hash, class Equal> Data& cuckoo< Key, Value, Hash, Equal >::data_from (size_t *pos*) const `[inline, private]`

Get Data element from pos.

Parameters

pos Position in hash arrays.

Returns

Reference to Data element.

3.2.3.8 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::is_here (const Key & *k*, size_t *pos*) const `[inline, private]`

Check whether element at pos has key k.

Parameters

k Key value.

pos Position in hash arrays.

Returns

true if element at pos is equal to k.

3.2.3.9 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::hash (const Key & *k*, size_t *hash_num*) const `[inline, private]`

Return hash function result for key k.

Parameters

k Key value.

hash_num The number of hash function from Hash family.

Returns

Hash value.

3.2.3.10 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::update_exists (size_t *len_temp_*) `[inline, private]`

Increase size of exists_ up to len_temp_.

Parameters

len_temp_ New size of exists_.

3.2.3.11 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::update_data (size_t *len_temp_*) `[inline, private]`

Increase size of data_ up to len_temp_.

Parameters

*len_temp_* New size of data_.

3.2.3.12 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::rehash () `[inline, private]`

Rehash all the cuckoo (i.e. change size and replace Data elements).

3.2.3.13 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::add_new (Data *p*) `[inline, private]`

Add new Data element.

Parameters

p New element.

Returns

0 to finish a recursion if it was needed.

3.2.3.14 template<class Key, class Value, class Hash, class Equal> iterator cuckoo< Key, Value, Hash, Equal >::remove (iterator & *it*) `[inline, private]`

Remove element from cuckoo.

Parameters

it Iterator to element to be removed.

Returns

Iterator to next element after removed.

3.2.3.15 template<class Key, class Value, class Hash, class Equal> cuckoo<Key, Value, Hash, Equal>& cuckoo< Key, Value, Hash, Equal >::operator= (const cuckoo< Key, Value, Hash, Equal > & *Cuckoo*) `[inline]`

Copy information from cuckoo of the same type.

Parameters

Cuckoo The source of data

3.2.3.16 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::set_up (size_t *d* = `D`, size_t *init_length* = `INIT_LENGTH`, size_t *max_loop* = `MAX_LOOP`, double *step* = `STEP`) `[inline]`

Update parameter of cuckoo, deleting all the data from it.

Warning

For test only!!!

3.2.3.17 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::operator== (const cuckoo< Key, Value, Hash, Equal > & *Cuckoo*) `[inline]`

Operator==

Parameters

Cuckoo Object to be compared with.

Returns

true if objects are equal (they are references to the same object).

3.2.3.18 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::operator!= (const cuckoo< Key, Value, Hash, Equal > & *Cuckoo*) `[inline]`

See also

operator==

3.2.3.19 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::swap (cuckoo< Key, Value, Hash, Equal > & *Cuckoo*) `[inline]`

Swap data with Cuckoo.

Parameters

Cuckoo Cuckoo of the same type

3.2.3.20 template<class Key, class Value, class Hash, class Equal> iterator cuckoo< Key, Value, Hash, Equal >::begin () `[inline]`

Get iterator to begin of cuckoo.

Returns

Iterator to the first element of cuckoo.

3.2.3.21 template<class Key, class Value, class Hash, class Equal> const_iterator cuckoo< Key, Value, Hash, Equal >::begin () const `[inline]`

Get const_iterator to begin of cuckoo (for const objects).

See also

begin()

3.2.3.22 template<class Key, class Value, class Hash, class Equal> iterator cuckoo< Key, Value, Hash, Equal >::end () `[inline]`

Get iterator to end of cuckoo.

Returns

Iterator to the position AFTER last element of cuckoo.

3.2.3.23 template<class Key, class Value, class Hash, class Equal> const_iterator cuckoo< Key, Value, Hash, Equal >::end () const `[inline]`

Get const_iterator to begin of cuckoo (for const objects).

See also

end()

3.2.3.24 template<class Key, class Value, class Hash, class Equal> Value& cuckoo< Key, Value, Hash, Equal >::operator[ ] (const Key & *k*) `[inline]`

Get value by key or created pair key-value.

Parameters

k Key value.

Returns

Reference to Value, associated with k.

3.2.3.25 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::erase (iterator *it*) `[inline]`

Erase data at iterator.

Parameters

it Iterator to Data element to be removed.

3.2.3.26 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::erase (iterator *first*, iterator *last*) `[inline]`

Erase range of Data elements.

Parameters

first The begin of range iterator.

last The end of range iterator.

3.2.3.27 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::erase (const Key & *k*) `[inline]`

Erase element by key.

Parameters

k Key value.

Returns

1 if element was erased and 0 if it didn't exist.

3.2.3.28 template<class Key, class Value, class Hash, class Equal> iterator cuckoo< Key, Value, Hash, Equal >::find (const Key & *k*) `[inline]`

Find element by key.

Parameters

k Key value

Returns

Iterator to element or to end of cuckoo, if element doesn't exist.

3.2.3.29 template<class Key, class Value, class Hash, class Equal> const_iterator cuckoo< Key, Value, Hash, Equal >::find (const Key & *k*) const `[inline]`

Find element by key (for const objects).

Parameters

k Key value

Returns

Const_iterator to element or to end of cuckoo, if element doesn't exist.

3.2.3.30 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::count (const Key & *k*) const `[inline]`

Count number of elements with this key.

Parameters

k Key value.

Returns

1 if element exists and 0 otherwise.

3.2.3.31 template<class Key, class Value, class Hash, class Equal> pair<iterator, iterator> cuckoo< Key, Value, Hash, Equal >::equal_range (const Key & *k*) `[inline]`

Find range of elements with key.

Parameters

k Key value.

Returns

Pair that determines the range [fisrt, last) or pair with both iterators pointing to the end of cuckoo.

3.2.3.32 template<class Key, class Value, class Hash, class Equal> pair<const_iterator, const_iterator> cuckoo< Key, Value, Hash, Equal >::equal_range (const Key & *k*) const `[inline]`

Find range of elements with key (for const objects).

Parameters

k Key value

Returns

Pair that determines the range [fisrt, last) or pair with both iterator pointing to the end of cuckoo.

3.2.3.33 template<class Key, class Value, class Hash, class Equal> pair<iterator, bool> cuckoo< Key, Value, Hash, Equal >::insert (const Data & *k*) `[inline]`

Insert Data element to cuckoo.

Parameters

k The new Data element.

Returns

Pair with iterator to existing element and bool value, which is true if element was inserted or false if it existed before.

3.2.3.34 template<class Key, class Value, class Hash, class Equal> template<class InputIterator > void cuckoo< Key, Value, Hash, Equal >::insert (InputIterator *first*, InputIterator *last*) `[inline]`

Insert range of Data elements.

Parameters

first The begin of range iterator.

last The end of range iterator.

3.2.3.35 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::clear () `[inline]`

Clear all data from cuckoo.

3.2.3.36 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::empty () const `[inline]`

Check whether cuckoo is empty.

Returns

true of cuckoo is empty and false otherwise.

3.2.3.37 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::size () const `[inline]`

Show size of cuckoo.

Returns

Size of cuckoo.

3.2.3.38 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::length () const `[inline]`

Show length of cuckoo (actual number of elements).

Returns

Length of cuckoo.

3.2.4 Friends And Related Function Documentation

3.2.4.1 template<class Key, class Value, class Hash, class Equal> friend class iterator `[friend]`

3.2.4.2 template<class Key, class Value, class Hash, class Equal> friend class const_iterator `[friend]`

3.2.5 Member Data Documentation

3.2.5.1 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::d_ `[private]`

The number of hash functions (thus arrays also) that will be used in the program (can be >= 2).

3.2.5.2 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::init_length_ `[private]`

The initial length of the whole structure. When you know the approximate number of records to be used, it is a good idea to take this value in 1.05-1.1 times more and small value of step.

3.2.5.3 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::max_loop_ `[private]`

The maximum number of kick cycles during insertion before rehash.

3.2.5.4 template<class Key, class Value, class Hash, class Equal> double cuckoo< Key, Value, Hash, Equal >::step_ `[private]`

The ratio of increasing the size of hash during rehash. The less it is the less memory will be used but the more time is needed.

3.2.5.5 template<class Key, class Value, class Hash, class Equal> Hash cuckoo< Key, Value, Hash, Equal >::hasher_ `[private]`

The hash function object (template parameter by default).

3.2.5.6 template<class Key, class Value, class Hash, class Equal> Equal cuckoo< Key, Value, Hash, Equal >::key_equal_ `[private]`

The equal predicator object (template parameter by default).

3.2.5.7 template<class Key, class Value, class Hash, class Equal> Data** cuckoo< Key, Value, Hash, Equal >::data_ `[private]`

The array of vectors, each of which is hash array.

3.2.5.8 template<class Key, class Value, class Hash, class Equal> char∗ cuckoo< Key, Value, Hash, Equal >::exists_ `[private]`

The array of flags indicating existence of the element in hash.

3.2.5.9 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::len_ `[private]`

The total length of all the hash arrays.

3.2.5.10 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::len_part_ `[private]`

The length of every hash array.

3.2.5.11 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::size_ `[private]`

The actual number of elements in cuckoo hash.

3.2.5.12 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::is_rehashed_ `[private]`

The flag that anounces that rehash was made recently.

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

- cuckoo.hpp

3.3 cuckoo< Key, Value, Hash, Equal >::iterator Class Reference

```
#include <cuckoo.hpp>
```

Public Member Functions

- iterator ()
- operator const_iterator ()
- void operator= (const iterator &it)
- iterator (const iterator &it)
- iterator & operator++ ()
- iterator operator++ (int)
- Data & operator∗ ()
- Data ∗ operator-> ()
- bool operator== (const iterator &it)
- bool operator!= (const iterator &it)

Private Member Functions

- iterator (size_t p, cuckoo ∗h)

Private Attributes

- size_t pos
- cuckoo ∗ hash

Friends

- class cuckoo

3.3.1 Detailed Description

template<class Key, class Value, class Hash, class Equal> class cuckoo< Key, Value, Hash, Equal >::iterator

Used with non-const cuckoo objects.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::iterator::iterator (size_t *p*, cuckoo ∗ *h*) `[inline, private]`

3.3.2.2 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::iterator::iterator () `[inline]`

Default constructor.

3.3.2.3 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::iterator::iterator (const iterator & *it*) `[inline]`

3.3.3 Member Function Documentation

3.3.3.1 template<class Key, class Value, class Hash, class Equal> cuckoo< Key, Value, Hash, Equal >::iterator::operator const_iterator () `[inline]`

Convertion to const_iterator.

3.3.3.2 template<class Key, class Value, class Hash, class Equal> void cuckoo< Key, Value, Hash, Equal >::iterator::operator= (const iterator & *it*) `[inline]`

3.3.3.3 template<class Key, class Value, class Hash, class Equal> iterator& cuckoo< Key, Value, Hash, Equal >::iterator::operator++ () `[inline]`

3.3.3.4 template<class Key, class Value, class Hash, class Equal> iterator cuckoo< Key, Value, Hash, Equal >::iterator::operator++ (int) `[inline]`

3.3.3.5 template<class Key, class Value, class Hash, class Equal> Data& cuckoo< Key, Value, Hash, Equal >::iterator::operator* () `[inline]`

3.3.3.6 template<class Key, class Value, class Hash, class Equal> Data* cuckoo< Key, Value, Hash, Equal >::iterator::operator-> () `[inline]`

3.3.3.7 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::iterator::operator== (const iterator & *it*) `[inline]`

3.3.3.8 template<class Key, class Value, class Hash, class Equal> bool cuckoo< Key, Value, Hash, Equal >::iterator::operator!= (const iterator & *it*) `[inline]`

3.3.4 Friends And Related Function Documentation

3.3.4.1 template<class Key, class Value, class Hash, class Equal> friend class cuckoo `[friend]`

3.3.5 Member Data Documentation

3.3.5.1 template<class Key, class Value, class Hash, class Equal> size_t cuckoo< Key, Value, Hash, Equal >::iterator::pos `[private]`

3.3.5.2 template<class Key, class Value, class Hash, class Equal> cuckoo* cuckoo< Key, Value, Hash, Equal >::iterator::hash `[private]`

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

- cuckoo.hpp

Chapter 4

File Documentation

4.1 cuckoo.hpp File Reference

```
#include <cstring>
```

Classes

- class cuckoo< Key, Value, Hash, Equal >
- class cuckoo< Key, Value, Hash, Equal >::const_iterator
- class cuckoo< Key, Value, Hash, Equal >::iterator

4.1.1 Detailed Description

Author

Mariya Fomkina

Version

1.0

4.1.2 LICENSE

4.1.3 DESCRIPTION

Cuckoo hashing implementation with the interface close to standard std::map and std::unordered_map interface.

See `http://en.wikipedia.org/wiki/Cuckoo_hashing`

Chapter 5

Example Documentation

5.1 cuckoo

Parameters

Key key type in hash.

Value value type in hash.

Hash function that gets data of type Key and some number (which is the number of appropriate hash function) and returns size_t (e.g. Hash(int, size_t)).

Equal predicator that compares two Key values. <int, int, Hasher, std::key_equal_to<int> > Cuckoo;

Index