# Symbol Table

Generated by Doxygen 1.8.16

# Chapter 1

# Class Index

## 1.1 Class List

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

# Chapter 2

# Class Documentation

## 2.1 Set< KeyType, Hash, Equals >::Iterator Class Reference

Const iterator for a Set. May become invalid/give errors after inserting data into the set; to check if an insertion would render the iterators invalid, use willInvalidateIteratorsOnInsert()

`#include <Set.hpp>`

### Public Types

- typedef std::input_iterator_tag **iterator_category**
- typedef KeyType **value_type**
- typedef std::ptrdiff_t **difference_type**

### Public Member Functions

- **Iterator** (const KeyType *elem, const State *state, const KeyType *end)
- Iterator & **operator++** ()
- Iterator **operator++** (int)
- bool **operator==** (Iterator other) const
- bool **operator!=** (Iterator other) const
- const reference **operator*** () const
- const pointer **operator->** () const

### Public Attributes

- const typedef KeyType * **pointer**
- const typedef KeyType & **reference**

### Friends

- class **Set**

### 2.1.1 Detailed Description

**template<class KeyType, class Hash = std::hash<KeyType>, class Equals = std::equal_to<KeyType>>**
**class Set< KeyType, Hash, Equals >::Iterator**

Const iterator for a Set. May become invalid/give errors after inserting data into the set; to check if an insertion would render the iterators invalid, use willInvalidateIteratorsOnInsert()

Definition at line 63 of file Set.hpp.

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

- Set.hpp

## 2.2 SymbolTable::Position Class Reference

Symbol table const iterator. Will still be valid after inserts in the parent SymbolTable

`#include <SymbolTable.h>`

### Public Types

- typedef std::input_iterator_tag **iterator_category**
- typedef std::ptrdiff_t **difference_type**

### Public Member Functions

- **Position** (Position &&other)
- **Position** (const Position &other)
- **Position** (SymbolTable *parent, Set< std::string >::Iterator iterator)
- SymbolTable & **symbolTable** () const
- Position & **operator++** ()
- Position **operator++** (int)
- bool **operator==** (Position other) const
- bool **operator!=** (Position other) const
- reference **operator*** () const
- pointer **operator->** () const

### Public Attributes

- const typedef std::string **value_type**
- const typedef std::string * **pointer**
- const typedef std::string & **reference**

### Friends

- class **SymbolTable**

### 2.2.1 Detailed Description

Symbol table const iterator. Will still be valid after inserts in the parent SymbolTable

Definition at line 15 of file SymbolTable.h.

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

- SymbolTable.h
- SymbolTable.cpp

## 2.3 Set< KeyType, Hash, Equals > Class Template Reference

Hashtable with coalesced chaining

`#include <Set.hpp>`

### Classes

- class Iterator

  *Const iterator for a Set. May become invalid/give errors after inserting data into the set; to check if an insertion would render the iterators invalid, use willInvalidateIteratorsOnInsert()*

### Public Types

- typedef Iterator **const_iterator_type**

### Public Member Functions

- Set ()

  *Initializes an empty Set with initial capacity 1*
- Set (size_t initial_size)

  *Initializes an empty Set with a custom initial capacity*
- std::pair< Iterator, bool > insert (const KeyType &item)

  *Inserts a copy of an item into the set*
- std::pair< Iterator, bool > insert (KeyType &&item)

  *Inserts an item into the set*
- Iterator find (const KeyType &item) const

  *Searches for an item in the set*
- void erase (Iterator iterator)

  *Removes the item at an iterator*
- size_t size ()

  ***Returns***

  *The number of items in the set*
- size_t capacity ()
- bool willInvalidateIteratorsOnInsert () const

  ***Returns***

  *true if the next insert would trigger a resize, therefore invalidating the iterators*
- Iterator **begin** () const
- Iterator **end** () const

### 2.3.1 Detailed Description

**template<class KeyType, class Hash = std::hash<KeyType>, class Equals = std::equal_to<KeyType>>**
**class Set< KeyType, Hash, Equals >**

Hashtable with coalesced chaining

**Template Parameters**

| | |
|---|---|
| *KeyType* | The type of the items held in this set |
| *Hash* | Hashing functor for an item of type KeyType |
| *Equals* | Equality functor for an item of type KeyType |

Definition at line 16 of file Set.hpp.

### 2.3.2 Constructor & Destructor Documentation

#### 2.3.2.1 Set() [1/2]

```
template<class KeyType, class Hash = std::hash<KeyType>, class Equals = std::equal_to<Key↩
Type>>
Set< KeyType, Hash, Equals >::Set ( )  [inline]
```

Initializes an empty Set with initial capacity 1

Definition at line 47 of file Set.hpp.

#### 2.3.2.2 Set() [2/2]

```
template<class KeyType , class Hash , class Equals >
Set< KeyType, Hash, Equals >::Set (
            size_t initial_size )  [inline]
```

Initializes an empty Set with a custom initial capacity

**Parameters**

| | |
|---|---|
| *initial_size* | The initial capacity |

Definition at line 138 of file Set.hpp.

## 2.3.3 Member Function Documentation

### 2.3.3.1 capacity()

```
template<class KeyType, class Hash = std::hash<KeyType>, class Equals = std::equal_to<Key↩
Type>>
size_t Set< KeyType, Hash, Equals >::capacity ( )  [inline]
```

**Returns**

The capacity of the set

The number of insertions (without subtracting deletions) is compared with the capacity when resizing

Definition at line 122 of file Set.hpp.

### 2.3.3.2 erase()

```
template<class KeyType , class Hash , class Equals >
void Set< KeyType, Hash, Equals >::erase (
            Iterator iterator )  [inline]
```

Removes the item at an iterator

**Parameters**

| | |
|---|---|
| *iterator* | The position of the item to remove |

Definition at line 271 of file Set.hpp.

### 2.3.3.3 find()

```
template<class KeyType, class Hash , class Equals >
Set< KeyType, Hash, Equals >::Iterator Set< KeyType, Hash, Equals >::find (
            const KeyType & item ) const  [inline]
```

Searches for an item in the set

**Parameters**

| | |
|---|---|
| *item* | The item to search for |

**Returns**

An iterator to the similar item in the set or end()

Definition at line 257 of file Set.hpp.

#### 2.3.3.4 insert() [1/2]

```
template<class KeyType, class Hash , class Equals >
std::pair< typename Set< KeyType, Hash, Equals >::Iterator, bool > Set< KeyType, Hash, Equals
>::insert (
            const KeyType & item )
```

Inserts a copy of an item into the set

**Parameters**

| | |
|---|---|
| *item* | The item to insert |

**Returns**

An iterator to an item equal to the parameter and whether the insertion happened (true) or a similar item was already in the set (false)

Definition at line 251 of file Set.hpp.

#### 2.3.3.5 insert() [2/2]

```
template<class KeyType, class Hash , class Equals >
std::pair< typename Set< KeyType, Hash, Equals >::Iterator, bool > Set< KeyType, Hash, Equals
>::insert (
            KeyType && item )
```

Inserts an item into the set

**Parameters**

| | |
|---|---|
| *item* | The item to insert |

**Returns**

An iterator to an item equal to the parameter and whether the insertion happened (true) or a similar item was already in the set (false)

Definition at line 245 of file Set.hpp.

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

- Set.hpp

## 2.4 SymbolTable Class Reference

Symbol table data structure. Features iterators that will sometimes get updated on insert, so that they are never invalid

`#include <SymbolTable.h>`

### Classes

- class Position

  *Symbol table const iterator. Will still be valid after inserts in the parent SymbolTable*

### Public Member Functions

- std::pair< Position, bool > insert (const std::string &symbol)

  *Inserts a symbol into the symbol table*
- Position find (const std::string &symbol)

  *Searches for an entry in the SymbolTable*
- Position **begin** ()
- Position **end** ()

### 2.4.1 Detailed Description

Symbol table data structure. Features iterators that will sometimes get updated on insert, so that they are never invalid

Definition at line 9 of file SymbolTable.h.

### 2.4.2 Member Function Documentation

#### 2.4.2.1 find()

```
SymbolTable::Position SymbolTable::find (
            const std::string & symbol )
```

Searches for an entry in the SymbolTable

**Parameters**

| | |
|---|---|
| *symbol* | The symbol to look for |

**Returns**

Position to the symbol or end() if it was not found

Definition at line 38 of file SymbolTable.cpp.

#### 2.4.2.2 insert()

```
std::pair< typename SymbolTable::Position, bool > SymbolTable::insert (
            const std::string & symbol )
```

Inserts a symbol into the symbol table

**Parameters**

| | |
|---|---|
| *symbol* | The symbol to insert |

**Returns**

A Position of where the symbol is and whether or not the insertion happened; if a similar symbol already exists in the table, the Position points to that

Definition at line 6 of file SymbolTable.cpp.

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

- SymbolTable.h
- SymbolTable.cpp

# Index