

Generic Containers

0.1

Generated by Doxygen 1.7.1

Thu Dec 2 2010 00:50:09

Contents

1	Class Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	GenericContainer< T >::ContainerNode Class Reference	7
4.1.1	Detailed Description	7
4.1.2	Member Data Documentation	7
4.1.2.1	element	7
4.1.2.2	next	7
4.1.2.3	previous	8
4.2	GenericContainer< T > Class Template Reference	8
4.2.1	Detailed Description	9
4.2.2	Constructor & Destructor Documentation	9
4.2.2.1	GenericContainer	9
4.2.2.2	~GenericContainer	9
4.2.3	Member Function Documentation	10
4.2.3.1	exists	10
4.2.3.2	get_element	10
4.2.3.3	insert	10
4.2.3.4	is_empty	10
4.2.3.5	remove	11
4.2.3.6	remove_element	11
4.2.3.7	size	11

4.2.4	Member Data Documentation	11
4.2.4.1	head	11
4.2.4.2	mutex	11
4.2.4.3	n_elements	12
4.2.4.4	tail	12
4.3	GenericQueue< T > Class Template Reference	12
4.3.1	Detailed Description	13
4.3.2	Member Function Documentation	13
4.3.2.1	exists	13
4.3.2.2	front	13
4.3.2.3	get_element	14
4.3.2.4	insert	14
4.3.2.5	is_empty	14
4.3.2.6	pop	14
4.3.2.7	push	15
4.3.2.8	remove	15
4.3.2.9	remove_element	15
4.3.2.10	size	15
4.4	GenericStack< T > Class Template Reference	15
4.4.1	Detailed Description	17
4.4.2	Member Function Documentation	17
4.4.2.1	exists	17
4.4.2.2	get_element	17
4.4.2.3	insert	17
4.4.2.4	is_empty	18
4.4.2.5	pop	18
4.4.2.6	push	18
4.4.2.7	remove	18
4.4.2.8	remove_element	18
4.4.2.9	size	19
4.4.2.10	top	19
4.5	ThreadSafe Class Reference	19
4.5.1	Detailed Description	20
4.5.2	Constructor & Destructor Documentation	20
4.5.2.1	ThreadSafe	20
4.5.3	Member Function Documentation	20

4.5.3.1	lock_readwrite	20
4.5.3.2	lock_write	20
4.5.3.3	unlock_readwrite	20
4.5.3.4	unlock_write	21
4.5.4	Member Data Documentation	21
4.5.4.1	logic_mutex	21
4.5.4.2	read_counter	21
4.5.4.3	write_mutex	21
5	File Documentation	23
5.1	/home/ferreira/dev-github/random/last.fm/src/GenericContainer.h File Reference	23
5.2	/home/ferreira/dev-github/random/last.fm/src/GenericQueue.h File Reference	23
5.3	/home/ferreira/dev-github/random/last.fm/src/GenericStack.h File Reference	24
5.4	/home/ferreira/dev-github/random/last.fm/src/ThreadSafe.cpp File Reference	24
5.5	/home/ferreira/dev-github/random/last.fm/src/ThreadSafe.h File Reference	24
5.6	/home/ferreira/dev-github/random/last.fm/src/ThreadUnsafe.cpp File Reference	24
5.7	/home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericContainer.cpp File Reference	24
5.7.1	Function Documentation	25
5.7.1.1	insert_remove_string_test	25
5.7.1.2	lots_of_inserts_test	25
5.7.1.3	main	25
5.7.1.4	multiple_operations_int_test	25
5.7.1.5	out_of_bounds_test	25
5.8	/home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericQueue.cpp File Reference	26
5.8.1	Function Documentation	26
5.8.1.1	main	26
5.8.1.2	order_test	26
5.9	/home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericStack.cpp File Reference	26
5.9.1	Function Documentation	27
5.9.1.1	main	27
5.9.1.2	order_test	27
5.10	/home/ferreira/dev-github/random/last.fm/unit-tests/unittestMultiThreaded.cpp File Reference	27
5.10.1	Define Documentation	28
5.10.1.1	N_INSERTS	28
5.10.1.2	N_THREADS	28

5.10.2	Function Documentation	28
5.10.2.1	insert_access_remove_run	28
5.10.2.2	insert_remove_run	28
5.10.2.3	main	28
5.10.2.4	test_insert_access_remove	28
5.10.2.5	test_insert_remove	28

Chapter 1

Class Index

1.1 Class Hierarchy

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

GenericContainer< T >::ContainerNode	7
GenericContainer< T >	8
GenericQueue< T >	12
GenericStack< T >	15
ThreadSafe	19

Chapter 2

Class Index

2.1 Class List

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

GenericContainer< T >::ContainerNode (Represents the node of a doubly-linked list)	7
GenericContainer< T > (A container for generic objects)	8
GenericQueue< T > (A queue for generic objects)	12
GenericStack< T > (A stack for generic objects)	15
ThreadSafe (Provides thread safeness to containers)	19

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/ferreira/dev-github/random/last.fm/src/ GenericContainer.h	23
/home/ferreira/dev-github/random/last.fm/src/ GenericQueue.h	23
/home/ferreira/dev-github/random/last.fm/src/ GenericStack.h	24
/home/ferreira/dev-github/random/last.fm/src/ ThreadSafe.cpp	24
/home/ferreira/dev-github/random/last.fm/src/ ThreadSafe.h	24
/home/ferreira/dev-github/random/last.fm/src/ ThreadUnsafe.cpp	24
/home/ferreira/dev-github/random/last.fm/unit-tests/ unittestGenericContainer.cpp	24
/home/ferreira/dev-github/random/last.fm/unit-tests/ unittestGenericQueue.cpp	26
/home/ferreira/dev-github/random/last.fm/unit-tests/ unittestGenericStack.cpp	26
/home/ferreira/dev-github/random/last.fm/unit-tests/ unittestMultiThreaded.cpp	27

Chapter 4

Class Documentation

4.1 GenericContainer< T >::ContainerNode Class Reference

Represents the node of a doubly-linked list.

Public Attributes

- [T element](#)
The copy of the element inserted in the container.
- [ContainerNode * next](#)
A link to the next element in the linked list.
- [ContainerNode * previous](#)
A link to the previous element in the linked list.

4.1.1 Detailed Description

template<class T> class GenericContainer< T >::ContainerNode

Represents the node of a doubly-linked list. This is a private class that can only be accessed by the container and that has the elements and links between the nodes in the linked list

4.1.2 Member Data Documentation

4.1.2.1 **template<class T> T GenericContainer< T >::ContainerNode::element**

The copy of the element inserted in the container.

This is the only copy of the element in the container.

4.1.2.2 **template<class T> ContainerNode* GenericContainer< T >::ContainerNode::next**

A link to the next element in the linked list.

This is NULL if it's the tail of the list

4.1.2.3 `template<class T> ContainerNode* GenericContainer< T >::ContainerNode::previous`

A link to the previous element in the linked list.

This is NULL if it's the tail of the list

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

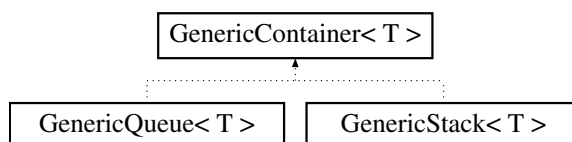
- `/home/ferreira/dev-github/random/last.fm/src/`[GenericContainer.h](#)

4.2 `GenericContainer< T >` Class Template Reference

A container for generic objects.

```
#include <GenericContainer.h>
```

Inheritance diagram for `GenericContainer< T >`:



Classes

- class [ContainerNode](#)
Represents the node of a doubly-linked list.

Public Member Functions

- [GenericContainer](#) ()
Create a [GenericContainer](#).
- [~GenericContainer](#) ()
Destroy a [GenericContainer](#).
- `bool is_empty () const`
Check if the Container is empty.
- `int size () const`
Get number of elements in the container.
- `bool exists (const T &element)`
Check if an element exists in the container.
- `void insert (const T &element, int position) throw (std::out_of_range)`

Insert an element at a specified position of the container.

- void [remove](#) (int position) throw (std::out_of_range)
Remove the element at a specified position of the container.
- bool [remove_element](#) (const T &element)
Remove the first occurrence of an element.
- T [get_element](#) (int position) throw (std::out_of_range)
Retrieve the element at a certain position of the container.

Private Attributes

- int [n_elements](#)
The number of elements in the container.
- [ContainerNode](#) * [head](#)
A pointer to the first element of the linked list.
- [ContainerNode](#) * [tail](#)
A pointer to the last element of the linked list.
- [ThreadSafe](#) mutex
A object to support safe race conditions.

4.2.1 Detailed Description

template<class T> class GenericContainer< T >

A container for generic objects. This class implements a container for generic objects using doubly-linked lists.

A copy of the elements inserted is kept in memory.

If this is linked with [ThreadSafe.cpp](#), it becomes thread safe and supports concurrent reading of the elements of the container. When writing, it makes sure that no other thread is reading or writing concurrently

4.2.2 Constructor & Destructor Documentation

4.2.2.1 **template<class T> GenericContainer< T >::GenericContainer ()**

Create a [GenericContainer](#).

Initialization of internal variables

4.2.2.2 **template<class T> GenericContainer< T >::~~GenericContainer ()**

Destroy a [GenericContainer](#).

Frees objects allocated in the container

4.2.3 Member Function Documentation

4.2.3.1 `template<class T> bool GenericContainer< T >::exists (const T & element)`

Check if an element exists in the container.

This method takes O(n) time for a random position

Returns

True if the element exists in the container, False otherwise

4.2.3.2 `template<class T> T GenericContainer< T >::get_element (int position) throw (std::out_of_range)`

Retrieve the element at a certain position of the container.

When asked to remove an element out of bounds gives an assert error.

Parameters

position The position of the element to return

Returns

The element at the specified position

4.2.3.3 `template<class T> void GenericContainer< T >::insert (const T & element, int position = 0) throw (std::out_of_range)`

Insert an element at a specified position of the container.

Insert an element in the container at a specified position. A Copy of the element is performed

Positions start counting from 0. The element is inserted at the first position if none other is specified.

When asked to insert an element out of bounds gives an assert error.

This method takes O(n) time for a random position This method takes O(1) time for inserting at left of the linked list

Parameters

element The element to insert in the container

position The position where to insert the element

4.2.3.4 `template<class T> bool GenericContainer< T >::is_empty () const`

Check if the Container is empty.

This method takes O(1) time

Returns

True if the container is empty, False otherwise

4.2.3.5 template<class T > void GenericContainer< T >::remove (int *position* = 0) throw (std::out_of_range)

Remove the element at a specified position of the container.

Positions start counting from 0.

When asked to remove an element out of bounds gives an assert error.

This method takes O(n) time for a random position This method takes O(1) time for removing at left of the linked list

Parameters

position The position where the element to remove is

4.2.3.6 template<class T > bool GenericContainer< T >::remove_element (const T & *element*)

Remove the first occurrence of an element.

Parameters

element The element to remove

Returns

True if removed an element, False otherwise

4.2.3.7 template<class T > int GenericContainer< T >::size () const

Get number of elements in the container.

This method takes O(1) time

Returns

Number of elements in the container

4.2.4 Member Data Documentation**4.2.4.1 template<class T > ContainerNode* GenericContainer< T >::head [private]**

A pointer to the first element of the linked list.

This first element is a dummy element and it is always created. This is done to avoid special cases when inserting/removing from the endpoints of the list

4.2.4.2 template<class T > ThreadSafe GenericContainer< T >::mutex [private]

A object to support safe race conditions.

If the code is linked with threadunsafe, there are no actions being performed. If threading is enabled, it allows concurrent reading of the list and locks reading and writing (inserting/removing) when writing

4.2.4.3 `template<class T> int GenericContainer< T >::n_elements` `[private]`

The number of elements in the container.

4.2.4.4 `template<class T> ContainerNode* GenericContainer< T >::tail` `[private]`

A pointer to the last element of the linked list.

This last element is a dummy element and it is always created. This is done to avoid special cases when inserting/removing from the endpoints of the list

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

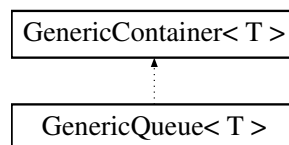
- `/home/ferreira/dev-github/random/last.fm/src/`[GenericContainer.h](#)

4.3 `GenericQueue< T >` Class Template Reference

A queue for generic objects.

```
#include <GenericQueue.h>
```

Inheritance diagram for `GenericQueue< T >`:



Public Member Functions

- `T front` (void) throw (std::out_of_range)
Get element at the front of the queue.
- `void pop` (void) throw (std::out_of_range)
Remove element at the front of the queue.
- `void push` (const T &element)
Push an element to the back of the queue.

Private Member Functions

- `bool is_empty` () const
Check if the Container is empty.
- `int size` () const
Get number of elements in the container.
- `bool exists` (const T &element)

Check if an element exists in the container.

- void [insert](#) (const T &element, int position) throw (std::out_of_range)

Insert an element at a specified position of the container.

- void [remove](#) (int position) throw (std::out_of_range)

Remove the element at a specified position of the container.

- bool [remove_element](#) (const T &element)

Remove the first occurrence of an element.

- T [get_element](#) (int position) throw (std::out_of_range)

Retrieve the element at a certain position of the container.

4.3.1 Detailed Description

template<class T> class GenericQueue< T >

A queue for generic objects. This class implements a queue for generic objects using the doubly-linked list in Generic Container.

If this is linked with [ThreadSafe.cpp](#), it becomes thread safe and supports concurrent reading of the elements of the container. When writing, it makes sure that no other thread is reading or writing concurrently

4.3.2 Member Function Documentation

4.3.2.1 **template<class T > bool GenericContainer< T >::exists (const T & element)** [**inherited**]

Check if an element exists in the container.

This method takes O(n) time for a random position

Returns

True if the element exists in the container, False otherwise

4.3.2.2 **template<class T > T GenericQueue< T >::front (void) throw (std::out_of_range)**

Get element at the front of the queue.

This method reuses the [get_element](#) method implemented in [GenericContainer](#) This method takes O(1) time

Returns

Returns a copy of the element at the front of the queue

4.3.2.3 `template<class T> T GenericContainer< T >::get_element (int position) throw (std::out_of_range) [inherited]`

Retrieve the element at a certain position of the container.

When asked to remove an element out of bounds gives an assert error.

Parameters

position The position of the element to return

Returns

The element at the specified position

4.3.2.4 `template<class T> void GenericContainer< T >::insert (const T & element, int position = 0) throw (std::out_of_range) [inherited]`

Insert an element at a specified position of the container.

Insert an element in the container at a specified position. A Copy of the element is performed

Positions start counting from 0. The element is inserted at the first position if none other is specified.

When asked to insert an element out of bounds gives an assert error.

This method takes O(n) time for a random position This method takes O(1) time for inserting at left of the linked list

Parameters

element The element to insert in the container

position The position where to insert the element

4.3.2.5 `template<class T> bool GenericContainer< T >::is_empty () const [inherited]`

Check if the Container is empty.

This method takes O(1) time

Returns

True if the container is empty, False otherwise

4.3.2.6 `template<class T> void GenericQueue< T >::pop (void) throw (std::out_of_range)`

Remove element at the front of the queue.

This method reuses the remove method implemented in [GenericContainer](#)

This method takes O(1) time

4.3.2.7 template<class T > void GenericQueue< T >::push (const T & *element*)

Push an element to the back of the queue.

This method implement a insert from the back of the linked list method in order to be more efficient.

This method takes O(1) time

4.3.2.8 template<class T > void GenericContainer< T >::remove (int *position* = 0) throw (std::out_of_range) [inherited]

Remove the element at a specified position of the container.

Positions start counting from 0.

When asked to remove an element out of bounds gives an assert error.

This method takes O(n) time for a random position This method takes O(1) time for removing at left of the linked list

Parameters

position The position where the element to remove is

4.3.2.9 template<class T > bool GenericContainer< T >::remove_element (const T & *element*) [inherited]

Remove the first occurrence of an element.

Parameters

element The element to remove

Returns

True if removed an element, False otherwise

4.3.2.10 template<class T > int GenericContainer< T >::size () const [inherited]

Get number of elements in the container.

This method takes O(1) time

Returns

Number of elements in the container

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

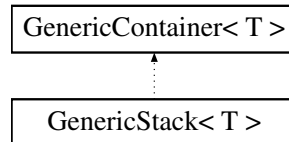
- /home/ferreira/dev-github/random/last.fm/src/[GenericQueue.h](#)

4.4 GenericStack< T > Class Template Reference

A stack for generic objects.

```
#include <GenericStack.h>
```

Inheritance diagram for GenericStack< T >:



Public Member Functions

- T [top](#) (void) throw (std::out_of_range)
Get element at the top of the stack.
- void [pop](#) (void) throw (std::out_of_range)
Remove element at the top of the stack.
- void [push](#) (T &element)
Push an element to the top of the stack.

Private Member Functions

- bool [is_empty](#) () const
Check if the Container is empty.
- int [size](#) () const
Get number of elements in the container.
- bool [exists](#) (const T &element)
Check if an element exists in the container.
- void [insert](#) (const T &element, int position) throw (std::out_of_range)
Insert an element at a specified position of the container.
- void [remove](#) (int position) throw (std::out_of_range)
Remove the element at a specified position of the container.
- bool [remove_element](#) (const T &element)
Remove the first occurrence of an element.
- T [get_element](#) (int position) throw (std::out_of_range)
Retrieve the element at a certain position of the container.

4.4.1 Detailed Description

template<class T> class GenericStack< T >

A stack for generic objects. This class implements a stack for generic objects using the doubly-linked list in Generic Container.

If this is linked with [ThreadSafe.cpp](#), it becomes thread safe and supports concurrent reading of the elements of the container. When writing, it makes sure that no other thread is reading or writing concurrently

4.4.2 Member Function Documentation

**4.4.2.1 template<class T > bool GenericContainer< T >::exists (const T & *element*)
[*inherited*]**

Check if an element exists in the container.

This method takes O(n) time for a random position

Returns

True if the element exists in the container, False otherwise

**4.4.2.2 template<class T > T GenericContainer< T >::get_element (int *position*) throw
(std::out_of_range) [*inherited*]**

Retrieve the element at a certain position of the container.

When asked to remove an element out of bounds gives an assert error.

Parameters

position The position of the element to return

Returns

The element at the specified position

**4.4.2.3 template<class T > void GenericContainer< T >::insert (const T & *element*, int
position = 0) throw (std::out_of_range) [*inherited*]**

Insert an element at a specified position of the container.

Insert an element in the container at a specified position. A Copy of the element is performed

Positions start counting from 0. The element is inserted at the first position if none other is specified.

When asked to insert an element out of bounds gives an assert error.

This method takes O(n) time for a random position This method takes O(1) time for inserting at left of the linked list

Parameters

element The element to insert in the container

position The position where to insert the element

4.4.2.4 `template<class T> bool GenericContainer< T >::is_empty () const [inherited]`

Check if the Container is empty.

This method takes O(1) time

Returns

True if the container is empty, False otherwise

4.4.2.5 `template<class T> void GenericStack< T >::pop (void) throw (std::out_of_range)`

Remove element at the top of the stack.

This method reuses the `get_element` method implemented in [GenericContainer](#) This method takes O(1) time

4.4.2.6 `template<class T> void GenericStack< T >::push (T & element)`

Push an element to the top of the stack.

This method reuses the `get_element` method implemented in [GenericContainer](#)

This method takes O(1) time

4.4.2.7 `template<class T> void GenericContainer< T >::remove (int position = 0) throw (std::out_of_range) [inherited]`

Remove the element at a specified position of the container.

Positions start counting from 0.

When asked to remove an element out of bounds gives an assert error.

This method takes O(n) time for a random position This method takes O(1) time for removing at left of the linked list

Parameters

position The position where the element to remove is

4.4.2.8 `template<class T> bool GenericContainer< T >::remove_element (const T & element) [inherited]`

Remove the first occurrence of an element.

Parameters

element The element to remove

Returns

True if removed an element, False otherwise

4.4.2.9 `template<class T> int GenericContainer< T >::size () const [inherited]`

Get number of elements in the container.

This method takes O(1) time

Returns

Number of elements in the container

4.4.2.10 `template<class T> T GenericStack< T >::top (void) throw (std::out_of_range)`

Get element at the top of the stack.

This method reuses the `get_element` method implemented in [GenericContainer](#) This method takes O(1) time

Returns

Returns a copy of the element at the front of the queue

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

- </home/ferreira/dev-github/random/last.fm/src/GenericStack.h>

4.5 ThreadSafe Class Reference

Provides thread safeness to containers.

```
#include <ThreadSafe.h>
```

Public Member Functions

- [ThreadSafe \(\)](#)
Constuctor of the thread safe class.
- void [lock_write \(\)](#)
Locks the container for writing.
- void [unlock_write \(\)](#)
Unlocks the container for writing.
- void [lock_readwrite \(\)](#)
Locks the container for Reading and Writing.
- void [unlock_readwrite \(\)](#)
Unlocks the container for Reading and Writing.

Private Attributes

- pthread_mutex_t [write_mutex](#)
Mutex for writing (insert/remove).
- pthread_mutex_t [logic_mutex](#)
Mutex for internal synch logic.
- int [read_counter](#)
Number of threads reading the container.

4.5.1 Detailed Description

Provides thread safeness to containers. This class implements methods that provide thread safeness to containers.

It allows that many threads are used at the same time. If a method that modifies the container is used, i.e. insert/remove, it guarantees that no other thread is reading or writing. Read operations, i.e. get_element, can be used concurrently.

If this is linked with [ThreadSafe.cpp](#), it enables thread safeness and supports concurrent reading of the elements of the container.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ThreadSafe::ThreadSafe ()

Constructor of the thread safe class.

Initializes the class that provides safe race conditions.

4.5.3 Member Function Documentation

4.5.3.1 void ThreadSafe::lock_readwrite ()

Locks the container for Reading and Writing.

Waits until there are no threads reading the container. Locks the container for writing if there are no threads reading at the moment.

4.5.3.2 void ThreadSafe::lock_write ()

Locks the container for writing.

Waits until there are no threads writing in the container. Locks the container for writing without doing so for reading. Keeps count of how many threads are reading the container at the moment.

4.5.3.3 void ThreadSafe::unlock_readwrite ()

Unlocks the container for Reading and Writing.

Unlocks the container for reading and writing.

4.5.3.4 void ThreadSafe::unlock_write ()

Unlocks the container for writing.

Unlocks the container for writing if there are no threads reading at the moment Keeps count of how many threads are reading the container at the moment

4.5.4 Member Data Documentation

4.5.4.1 pthread_mutex_t ThreadSafe::logic_mutex [private]

Mutex for internal synch logic.

This mutex makes sure that operations in the synchorinzation methods are atomic

4.5.4.2 int ThreadSafe::read_counter [private]

Number of threads reading the container.

4.5.4.3 pthread_mutex_t ThreadSafe::write_mutex [private]

Mutex for writing (insert/remove).

If locked a thread has to wait to write

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

- [/home/ferreira/dev-github/random/last.fm/src/ThreadSafe.h](#)
- [/home/ferreira/dev-github/random/last.fm/src/ThreadSafe.cpp](#)
- [/home/ferreira/dev-github/random/last.fm/src/ThreadUnsafe.cpp](#)

Chapter 5

File Documentation

5.1 /home/ferreira/dev-github/random/last.fm/src/GenericContainer.h File Reference

```
#include <utility>
#include <assert.h>
#include <stdexcept>
#include "ThreadSafe.h"
#include <iostream>
```

Classes

- class [GenericContainer< T >](#)
A container for generic objects.
- class [GenericContainer< T >::ContainerNode](#)
Represents the node of a doubly-linked list.

5.2 /home/ferreira/dev-github/random/last.fm/src/GenericQueue.h File Reference

```
#include "GenericContainer.h"
```

Classes

- class [GenericQueue< T >](#)
A queue for generic objects.

5.3 /home/ferreira/dev-github/random/last.fm/src/GenericStack.h File Reference

```
#include "GenericContainer.h"
```

Classes

- class [GenericStack< T >](#)
A stack for generic objects.

5.4 /home/ferreira/dev-github/random/last.fm/src/ThreadSafe.cpp File Reference

```
#include "ThreadSafe.h"  
#include <stdlib.h>  
#include <iostream>
```

5.5 /home/ferreira/dev-github/random/last.fm/src/ThreadSafe.h File Reference

```
#include <pthread.h>
```

Classes

- class [ThreadSafe](#)
Provides thread safeness to containers.

5.6 /home/ferreira/dev-github/random/last.fm/src/ThreadUnsafe.cpp File Reference

```
#include "ThreadSafe.h"
```

5.7 /home/ferreira/dev-github/random/last.fm/unit- tests/unittestGenericContainer.cpp File Reference

```
#include <assert.h>  
#include <stdexcept>  
#include "../src/GenericContainer.h"
```

```
#include <iostream>
#include <string>
```

Functions

- void [insert_remove_string_test](#) (void)
test insertion and removal of elements
- void [out_of_bounds_test](#) (void)
Test exceptions thrown for access of out of bounds elements.
- void [multiple_operations_int_test](#) (void)
Test mixed operations.
- void [lots_of_inserts_test](#) (void)
Test the insertion of elements.
- int [main](#) (void)
Runs the unit tests for [GenericContainer](#).

5.7.1 Function Documentation

5.7.1.1 void [insert_remove_string_test](#) (void)

test insertion and removal of elements

5.7.1.2 void [lots_of_inserts_test](#) (void)

Test the insertion of elements.

5.7.1.3 int [main](#) (void)

Runs the unit tests for [GenericContainer](#).

5.7.1.4 void [multiple_operations_int_test](#) (void)

Test mixed operations.

5.7.1.5 void [out_of_bounds_test](#) (void)

Test exceptions thrown for access of out of bounds elements.

5.8 /home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericQueue.cpp File Reference

```
#include <assert.h>
#include <stdexcept>
#include "../src/GenericQueue.h"
#include <iostream>
#include <string>
```

Functions

- void [order_test](#) (void)
Test the order elements come out of the queue.
- int [main](#) (void)
Runs the unit tests for [GenericQueue](#).

5.8.1 Function Documentation

5.8.1.1 int main (void)

Runs the unit tests for [GenericQueue](#).

As the queue heavily reuses code from [GenericContainer](#) the only thing left to test is the order of the elements

5.8.1.2 void order_test (void)

Test the order elements come out of the queue.

Make sure of FIFO

5.9 /home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericStack.cpp File Reference

```
#include <assert.h>
#include <stdexcept>
#include "../src/GenericStack.h"
#include <iostream>
#include <string>
```

Functions

- void [order_test](#) (void)

Test the order elements come out of the queue.

- `int main (void)`

Runs the unit tests for [GenericStack](#).

5.9.1 Function Documentation

5.9.1.1 `int main (void)`

Runs the unit tests for [GenericStack](#).

As the stack heavily reuses code from [GenericContainer](#) the only thing left to test is the order of the elements

5.9.1.2 `void order_test (void)`

Test the order elements come out of the queue.

Make sure of LIFO

5.10 /home/ferreira/dev-github/random/last.fm/unit-tests/unittestMultiThreaded.cpp File Reference

```
#include <assert.h>
#include <stdexcept>
#include "../src/GenericContainer.h"
#include <iostream>
#include <pthread.h>
```

Defines

- `#define N_THREADS 50`
- `#define N_INSERTS 1000`

Functions

- `void * insert_remove_run (void *data)`
The thread that insert, accesses and removes elements.
- `void test_insert_remove ()`
Test concurrent insertion, access and removal of elements.
- `void * insert_access_remove_run (void *data)`
The thread that insert and removes elements.
- `void test_insert_access_remove ()`

Test concurrent insertion and removal of elements.

- `int main (void)`

Runs the unit tests for [GenericContainer](#).

5.10.1 Define Documentation

5.10.1.1 `#define N_INSERTS 1000`

5.10.1.2 `#define N_THREADS 50`

5.10.2 Function Documentation

5.10.2.1 `void* insert_access_remove_run (void * data)`

The thread that insert and removes elements.

5.10.2.2 `void* insert_remove_run (void * data)`

The thread that insert, accesses and removes elements.

5.10.2.3 `int main (void)`

Runs the unit tests for [GenericContainer](#).

As all the multi-threading logic is in the [GenericContainer](#) there is no need to test the queue and the stack

5.10.2.4 `void test_insert_access_remove ()`

Test concurrent insertion and removal of elements.

Start several threads that insert many elements and then remove the same number of elements inserted

5.10.2.5 `void test_insert_remove ()`

Test concurrent insertion, access and removal of elements.

Start several threads that insert many elements then access the same number of elements and then remove the same number of elements inserted

Index

- ~GenericContainer
 - GenericContainer, 9
- /home/ferreira/dev-
 - github/random/last.fm/src/GenericContainer.h, 23
- /home/ferreira/dev-
 - github/random/last.fm/src/GenericQueue.h, 23
- /home/ferreira/dev-
 - github/random/last.fm/src/GenericStack.h, 24
- /home/ferreira/dev-
 - github/random/last.fm/src/ThreadSafe.cpp, 24
- /home/ferreira/dev-
 - github/random/last.fm/src/ThreadSafe.h, 24
- /home/ferreira/dev-
 - github/random/last.fm/src/ThreadUnsafe.cpp, 24
- /home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericContainer.cpp, 24
- /home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericQueue.cpp, 26
- /home/ferreira/dev-github/random/last.fm/unit-tests/unittestGenericStack.cpp, 26
- /home/ferreira/dev-github/random/last.fm/unit-tests/unittestMultiThreaded.cpp, 27
- element
 - GenericContainer::ContainerNode, 7
- exists
 - GenericContainer, 10
 - GenericQueue, 13
 - GenericStack, 17
- front
 - GenericQueue, 13
- GenericContainer, 8
 - ~GenericContainer, 9
 - exists, 10
 - GenericContainer, 9
 - get_element, 10
 - head, 11
 - insert, 10
 - is_empty, 10
 - mutex, 11
 - n_elements, 11
 - remove, 10
 - remove_element, 11
 - size, 11
 - tail, 12
- GenericContainer::ContainerNode, 7
 - element, 7
 - next, 7
 - previous, 8
- GenericQueue, 12
 - exists, 13
 - front, 13
 - get_element, 13
 - insert, 14
 - is_empty, 14
 - pop, 14
 - push, 14
 - remove, 15
 - remove_element, 15
 - size, 15
- GenericStack, 15
 - exists, 17
 - get_element, 17
 - insert, 17
 - is_empty, 17
 - pop, 18
 - push, 18
 - remove, 18
 - remove_element, 18
 - size, 18
 - top, 19
- get_element
 - GenericContainer, 10
 - GenericQueue, 13
 - GenericStack, 17
- head
 - GenericContainer, 11
- insert
 - GenericContainer, 10
 - GenericQueue, 14

- GenericStack, 17
- insert_access_remove_run
 - unittestMultiThreaded.cpp, 28
- insert_remove_run
 - unittestMultiThreaded.cpp, 28
- insert_remove_string_test
 - unittestGenericContainer.cpp, 25
- is_empty
 - GenericContainer, 10
 - GenericQueue, 14
 - GenericStack, 17
- lock_readwrite
 - ThreadSafe, 20
- lock_write
 - ThreadSafe, 20
- logic_mutex
 - ThreadSafe, 21
- lots_of_inserts_test
 - unittestGenericContainer.cpp, 25
- main
 - unittestGenericContainer.cpp, 25
 - unittestGenericQueue.cpp, 26
 - unittestGenericStack.cpp, 27
 - unittestMultiThreaded.cpp, 28
- multiple_operations_int_test
 - unittestGenericContainer.cpp, 25
- mutex
 - GenericContainer, 11
- n_elements
 - GenericContainer, 11
- N_INSERTS
 - unittestMultiThreaded.cpp, 28
- N_THREADS
 - unittestMultiThreaded.cpp, 28
- next
 - GenericContainer::ContainerNode, 7
- order_test
 - unittestGenericQueue.cpp, 26
 - unittestGenericStack.cpp, 27
- out_of_bounds_test
 - unittestGenericContainer.cpp, 25
- pop
 - GenericQueue, 14
 - GenericStack, 18
- previous
 - GenericContainer::ContainerNode, 8
- push
 - GenericQueue, 14
 - GenericStack, 18
- read_counter
 - ThreadSafe, 21
- remove
 - GenericContainer, 10
 - GenericQueue, 15
 - GenericStack, 18
- remove_element
 - GenericContainer, 11
 - GenericQueue, 15
 - GenericStack, 18
- size
 - GenericContainer, 11
 - GenericQueue, 15
 - GenericStack, 18
- tail
 - GenericContainer, 12
- test_insert_access_remove
 - unittestMultiThreaded.cpp, 28
- test_insert_remove
 - unittestMultiThreaded.cpp, 28
- ThreadSafe, 19
 - lock_readwrite, 20
 - lock_write, 20
 - logic_mutex, 21
 - read_counter, 21
 - ThreadSafe, 20
 - unlock_readwrite, 20
 - unlock_write, 20
 - write_mutex, 21
- top
 - GenericStack, 19
- unittestGenericContainer.cpp
 - insert_remove_string_test, 25
 - lots_of_inserts_test, 25
 - main, 25
 - multiple_operations_int_test, 25
 - out_of_bounds_test, 25
- unittestGenericQueue.cpp
 - main, 26
 - order_test, 26
- unittestGenericStack.cpp
 - main, 27
 - order_test, 27
- unittestMultiThreaded.cpp
 - insert_access_remove_run, 28
 - insert_remove_run, 28
 - main, 28
 - N_INSERTS, 28
 - N_THREADS, 28
 - test_insert_access_remove, 28
 - test_insert_remove, 28

unlock_readwrite
 ThreadSafe, [20](#)
unlock_write
 ThreadSafe, [20](#)

write_mutex
 ThreadSafe, [21](#)