

The Container and Sequence Classes

pa11

Generated by Doxygen 1.8.17

1 Hierarchical 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 Container Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Member Typedef Documentation	8
4.1.2.1 size_type	8
4.1.2.2 value_type	8
4.1.3 Constructor & Destructor Documentation	8
4.1.3.1 Container()	8
4.1.4 Member Function Documentation	8
4.1.4.1 clear()	9
4.1.4.2 contains()	9
4.1.4.3 count()	9
4.1.4.4 empty()	10
4.1.4.5 erase()	11
4.1.4.6 insert()	11
4.1.4.7 size()	12
4.1.4.8 write()	13
4.1.5 Member Data Documentation	14
4.1.5.1 CAPACITY	14
4.1.5.2 data	14
4.1.5.3 used	14
4.2 Sequence Class Reference	15
4.2.1 Detailed Description	16
4.2.2 Constructor & Destructor Documentation	16
4.2.2.1 Sequence() [1/2]	16
4.2.2.2 Sequence() [2/2]	16
4.2.3 Member Function Documentation	16
4.2.3.1 at() [1/2]	17
4.2.3.2 at() [2/2]	17
4.2.3.3 back() [1/2]	18
4.2.3.4 back() [2/2]	18
4.2.3.5 contains()	18
4.2.3.6 erase()	19
4.2.3.7 find()	19

4.2.3.8 front() [1/2]	20
4.2.3.9 front() [2/2]	20
5 File Documentation	21
5.1 Container.cpp File Reference	21
5.1.1 Detailed Description	21
5.2 Container.h File Reference	22
5.2.1 Detailed Description	22
5.3 pa11-test.cpp File Reference	23
5.3.1 Macro Definition Documentation	24
5.3.1.1 CATCH_CONFIG_MAIN	24
5.3.2 Function Documentation	24
5.3.2.1 TEST_CASE() [1/8]	24
5.3.2.2 TEST_CASE() [2/8]	24
5.3.2.3 TEST_CASE() [3/8]	25
5.3.2.4 TEST_CASE() [4/8]	25
5.3.2.5 TEST_CASE() [5/8]	25
5.3.2.6 TEST_CASE() [6/8]	25
5.3.2.7 TEST_CASE() [7/8]	26
5.3.2.8 TEST_CASE() [8/8]	26
5.4 pa11.cpp File Reference	26
5.4.1 Macro Definition Documentation	27
5.4.1.1 CATCH_CONFIG_MAIN	27
5.4.2 Function Documentation	27
5.4.2.1 TEST_CASE() [1/8]	27
5.4.2.2 TEST_CASE() [2/8]	27
5.4.2.3 TEST_CASE() [3/8]	28
5.4.2.4 TEST_CASE() [4/8]	28
5.4.2.5 TEST_CASE() [5/8]	28
5.4.2.6 TEST_CASE() [6/8]	28
5.4.2.7 TEST_CASE() [7/8]	29
5.4.2.8 TEST_CASE() [8/8]	29
5.5 Sequence.cpp File Reference	29
5.5.1 Function Documentation	30
5.5.1.1 equal()	30
5.6 Sequence.h File Reference	30
5.6.1 Detailed Description	31
5.6.2 Function Documentation	31
5.6.2.1 equal()	32
Index	33

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Container	7
Sequence	15

Chapter 2

Class Index

2.1 Class List

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

Container	7
Sequence	15

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

- [Container.cpp](#)
This is the implementation file for the [Container](#) class 21
- [Container.h](#)
This is the interface file for the [Container](#) class 22
- [pa11-test.cpp](#) 23
- [pa11.cpp](#) 26
- [Sequence.cpp](#) 29
- [Sequence.h](#)
This is the interface for the [Sequence](#) class 30

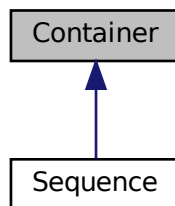
Chapter 4

Class Documentation

4.1 Container Class Reference

```
#include <Container.h>
```

Inheritance diagram for Container:



Public Types

- using `value_type` = int
- using `size_type` = std::size_t

Public Member Functions

- `Container ()`
Constructs the `Container`.
- `size_type size () const`
- `bool empty () const`
- `void insert (const value_type &value)`
- `void erase (const value_type &target)`
- `void clear ()`
After this call, `size()` returns zero.
- `size_type count (const value_type &target) const`
- `bool contains (const value_type &target) const`
- `void write (std::ostream &output=std::cout) const`

Static Public Attributes

- static const [size_type](#) `CAPACITY` = 20
Maximum storage capacity.

Protected Attributes

- [size_type](#) `used`
Number of items in [Container](#).
- [value_type](#) `data` [`CAPACITY`]
Array of items.

4.1.1 Detailed Description

A [Container](#) is a general-purpose container that stores a set of possibly non-unique values of type `int`. Internally, the items are not maintained in any particular order.

4.1.2 Member Typedef Documentation

4.1.2.1 `size_type`

```
using Container::size\_type = std::size_t
```

4.1.2.2 `value_type`

```
using Container::value\_type = int
```

4.1.3 Constructor & Destructor Documentation

4.1.3.1 `Container()`

```
Container::Container ( ) [inline]
```

Constructs the [Container](#).

4.1.4 Member Function Documentation

4.1.4.1 clear()

```
void Container::clear ( ) [inline]
```

After this call, [size\(\)](#) returns zero.

4.1.4.2 contains()

```
bool Container::contains (
    const value\_type & target ) const [inline]
```

Checks if there is an item with its value equal to the target.

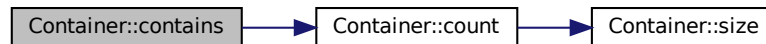
Parameters

<i>target</i>	Key value of the item to search for.
---------------	--------------------------------------

Returns

true if there is such an item, otherwise false.

Here is the call graph for this function:



4.1.4.3 count()

```
Container::size\_type Container::count (
    const value\_type & target ) const
```

Returns the number of items equal to the target.

Parameters

<i>target</i>	Key value of the item(s) to count.
---------------	------------------------------------

Returns

Number of items with value equal to the target.

Here is the call graph for this function:



Here is the caller graph for this function:

**4.1.4.4 empty()**

```
bool Container::empty ( ) const [inline]
```

Checks if the container has no items, i.e. whether `size() == 0`

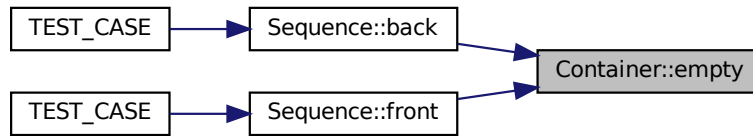
Returns

true if the container has no items, false otherwise.

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.5 erase()

```
void Container::erase (
    const value_type & target )
```

Removes all items from the container equal to the target value. Internally, the items are not maintained in any particular order.

Parameters

<i>target</i>	Key value of the items to remove.
---------------	-----------------------------------

Here is the call graph for this function:



4.1.4.6 insert()

```
void Container::insert (
    const value_type & value )
```

Inserts an item into the container. Internally, the items are not maintained in any particular order.

Precondition

`size() < CAPACITY`

Parameters

<i>value</i>	Element value to insert.
--------------	--------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.7 size()

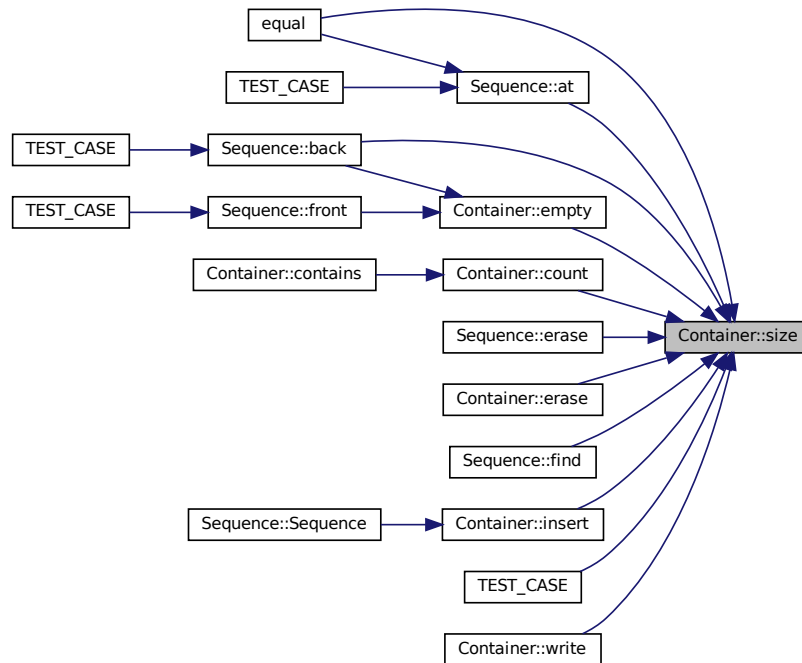
```
size_type Container::size ( ) const [inline]
```

Returns the number of items in the container.

Returns

The number of items in the container.

Here is the caller graph for this function:

**4.1.4.8 write()**

```
void Container::write (
    std::ostream & output = std::cout ) const
```

Writes all items to an output stream in the format: {42,73,0,-59,7}

Parameters

<i>output</i>	The output stream (defaults to <code>std::cout</code>).
---------------	----------------------------------------------------------

Here is the call graph for this function:



4.1.5 Member Data Documentation

4.1.5.1 CAPACITY

```
const size\_type Container::CAPACITY = 20 [static]
```

Maximum storage capacity.

4.1.5.2 data

```
value\_type Container::data[CAPACITY] [protected]
```

Array of items.

4.1.5.3 used

```
size\_type Container::used [protected]
```

Number of items in [Container](#).

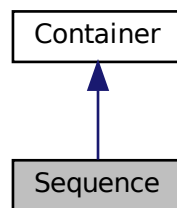
The documentation for this class was generated from the following files:

- [Container.h](#)
- [Container.cpp](#)

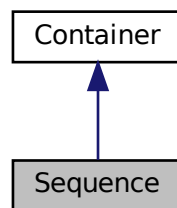
4.2 Sequence Class Reference

```
#include <Sequence.h>
```

Inheritance diagram for Sequence:



Collaboration diagram for Sequence:



Public Member Functions

- `Sequence ()=default`
Default ctor.
- `Sequence (const std::initializer_list< value_type > &ilist)`
Initializer list ctor.
- `void erase (size_type pos)`
Removes a single item from the container.
- `value_type & at (size_type pos)`
Access specified element.
- `const value_type & at (size_type pos) const`
- `value_type & front ()`
Access the first element.
- `const value_type & front () const`
- `value_type & back ()`

Access the last element.

- const `value_type` & `back` () const
- `size_type` `find` (const `value_type` &target, `size_type` pos=0) const

Finds the first element equal to the given target.

- bool `contains` (const `value_type` &) const =delete

Delete `contains()` function from `Sequence` interface.

Additional Inherited Members

4.2.1 Detailed Description

A `Sequence` is a `Container` that stores a set of values, maintaining insertion order.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 `Sequence()` [1/2]

```
Sequence::Sequence ( ) [default]
```

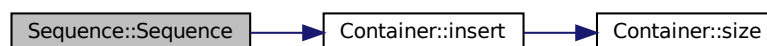
Default ctor.

4.2.2.2 `Sequence()` [2/2]

```
Sequence::Sequence (
    const std::initializer_list< value_type > & ilist )
```

Initializer list ctor.

Here is the call graph for this function:



4.2.3 Member Function Documentation

4.2.3.1 at() [1/2]

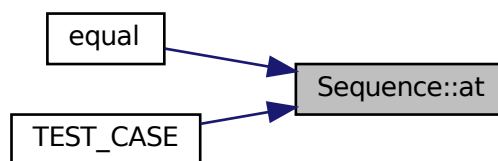
```
Sequence::value_type & Sequence::at (
    size_type pos )
```

Access specified element.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.2 at() [2/2]

```
const Sequence::value_type & Sequence::at (
    size_type pos ) const
```

Here is the call graph for this function:



4.2.3.3 back() [1/2]

```
const Sequence::value_type & Sequence::back ( )
```

Access the last element.

Here is the call graph for this function:



Here is the caller graph for this function:

**4.2.3.4 back() [2/2]**

```
const value_type& Sequence::back ( ) const
```

4.2.3.5 contains()

```
bool Sequence::contains (
    const value_type & ) const [delete]
```

Delete `contains()` function from `Sequence` interface.

4.2.3.6 erase()

```
void Sequence::erase (
    size_type pos )
```

Removes a single item from the container.

Here is the call graph for this function:



4.2.3.7 find()

```
Sequence::size_type Sequence::find (
    const value_type & target,
    size_type pos = 0 ) const
```

Finds the first element equal to the given target.

Here is the call graph for this function:

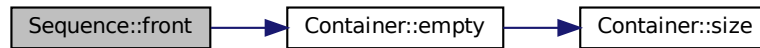


4.2.3.8 front() [1/2]

```
const Sequence::value_type & Sequence::front ( )
```

Access the first element.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.9 front() [2/2]

```
const value_type& Sequence::front ( ) const
```

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

- [Sequence.h](#)
- [Sequence.cpp](#)

Chapter 5

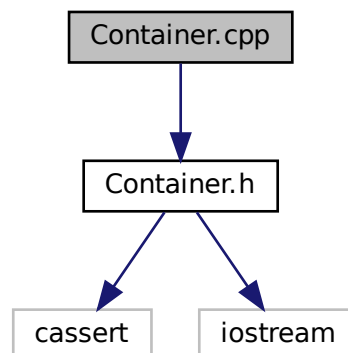
File Documentation

5.1 Container.cpp File Reference

This is the implementation file for the [Container](#) class.

```
#include "Container.h"
```

Include dependency graph for Container.cpp:



5.1.1 Detailed Description

This is the implementation file for the [Container](#) class.

Author

Kevin Mess kevin.mess@csn.edu

Date

05/26/2021

Note

I pledge my word of honor that I have complied with the CSN Academic Integrity Policy while completing this assignment.

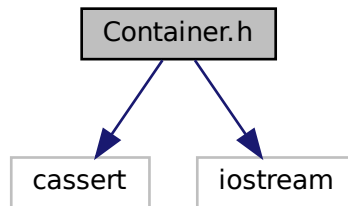
This is a sample solution and is but one way to solve the problem.

5.2 Container.h File Reference

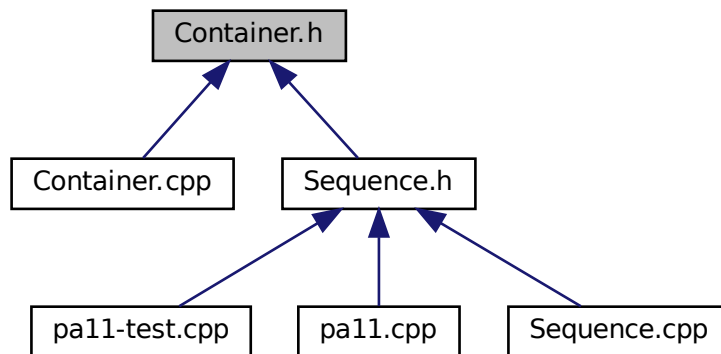
This is the interface file for the [Container](#) class.

```
#include <cassert>
#include <iostream>
```

Include dependency graph for Container.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Container](#)

5.2.1 Detailed Description

This is the interface file for the [Container](#) class.

Author

Kevin Mess kevin.mess@csn.edu

Date

05/26/2021

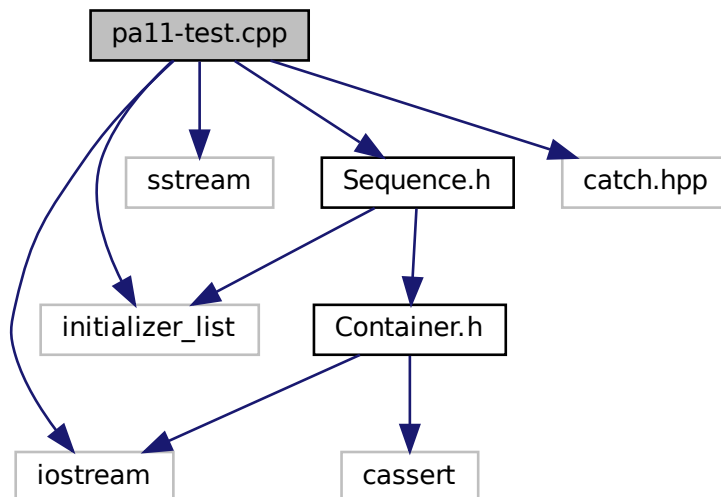
Note

I pledge my word of honor that I have complied with the CSN Academic Integrity Policy while completing this assignment.

This is a sample solution and is but one way to solve the problem.

5.3 pa11-test.cpp File Reference

```
#include <initializer_list>
#include <iostream>
#include <sstream>
#include "Sequence.h"
#include <catch.hpp>
Include dependency graph for pa11-test.cpp:
```

**Macros**

- `#define CATCH_CONFIG_MAIN`

Functions

- [TEST_CASE](#) ("Sequence::Sequence()")
- [TEST_CASE](#) ("Sequence::Sequence(std::initializer_list<>&")")
- [TEST_CASE](#) ("Sequence::erase()")
- [TEST_CASE](#) ("Sequence::at()")
- [TEST_CASE](#) ("Sequence::front()")
- [TEST_CASE](#) ("Sequence::back()")
- [TEST_CASE](#) ("Sequence::find()")
- [TEST_CASE](#) ("equal()")

5.3.1 Macro Definition Documentation

5.3.1.1 CATCH_CONFIG_MAIN

```
#define CATCH_CONFIG_MAIN
```

5.3.2 Function Documentation

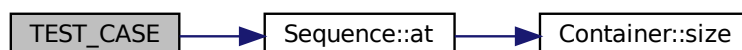
5.3.2.1 TEST_CASE() [1/8]

```
TEST_CASE (
    "equal()" )
```

5.3.2.2 TEST_CASE() [2/8]

```
TEST_CASE (
    "Sequence::at()" )
```

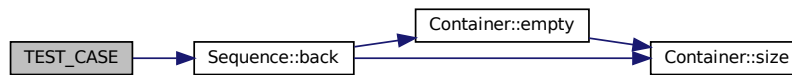
Here is the call graph for this function:



5.3.2.3 TEST_CASE() [3/8]

```
TEST_CASE (
    "Sequence::back()" )
```

Here is the call graph for this function:



5.3.2.4 TEST_CASE() [4/8]

```
TEST_CASE (
    "Sequence::erase()" )
```

5.3.2.5 TEST_CASE() [5/8]

```
TEST_CASE (
    "Sequence::find()" )
```

5.3.2.6 TEST_CASE() [6/8]

```
TEST_CASE (
    "Sequence::front()" )
```

Here is the call graph for this function:



5.3.2.7 TEST_CASE() [7/8]

```
TEST_CASE (
    "Sequence::Sequence()" )
```

Here is the call graph for this function:



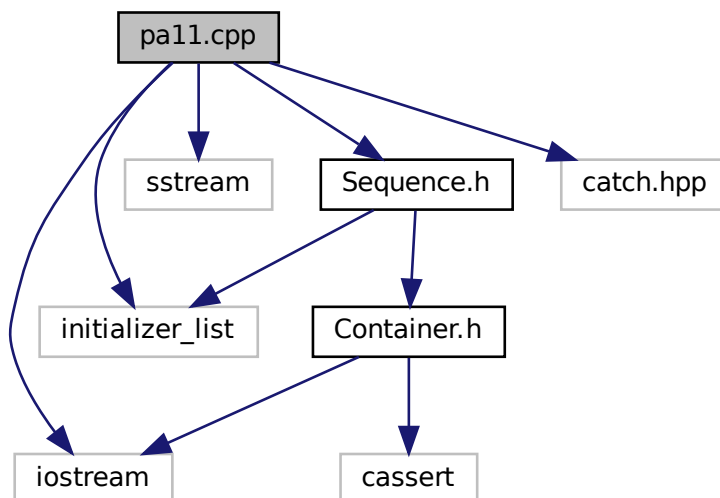
5.3.2.8 TEST_CASE() [8/8]

```
TEST_CASE ( )
```

5.4 pa11.cpp File Reference

```
#include <initializer_list>
#include <iostream>
#include <sstream>
#include "Sequence.h"
#include <catch.hpp>
```

Include dependency graph for pa11.cpp:



Macros

- `#define CATCH_CONFIG_MAIN`

Functions

- `TEST_CASE` ("Sequence::Sequence()")
- `TEST_CASE` ("Sequence::Sequence(std::initializer_list<>&")
- `TEST_CASE` ("Sequence::erase()")
- `TEST_CASE` ("Sequence::at()")
- `TEST_CASE` ("Sequence::front()")
- `TEST_CASE` ("Sequence::back()")
- `TEST_CASE` ("Sequence::find()")
- `TEST_CASE` ("equal()")

5.4.1 Macro Definition Documentation

5.4.1.1 CATCH_CONFIG_MAIN

```
#define CATCH_CONFIG_MAIN
```

5.4.2 Function Documentation

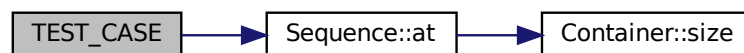
5.4.2.1 TEST_CASE() [1/8]

```
TEST_CASE (
    "equal()" )
```

5.4.2.2 TEST_CASE() [2/8]

```
TEST_CASE (
    "Sequence::at()" )
```

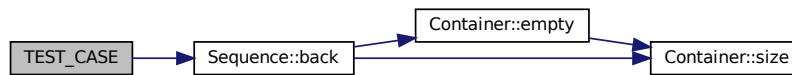
Here is the call graph for this function:



5.4.2.3 TEST_CASE() [3/8]

```
TEST_CASE (
    "Sequence::back()" )
```

Here is the call graph for this function:



5.4.2.4 TEST_CASE() [4/8]

```
TEST_CASE (
    "Sequence::erase()" )
```

5.4.2.5 TEST_CASE() [5/8]

```
TEST_CASE (
    "Sequence::find()" )
```

5.4.2.6 TEST_CASE() [6/8]

```
TEST_CASE (
    "Sequence::front()" )
```

Here is the call graph for this function:



5.4.2.7 TEST_CASE() [7/8]

```
TEST_CASE (
    "Sequence::Sequence()" )
```

Here is the call graph for this function:



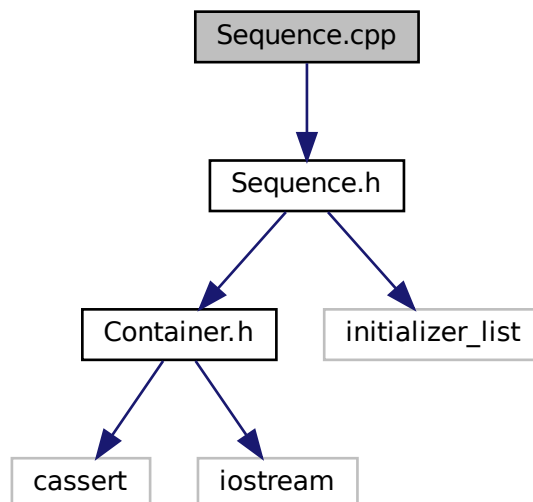
5.4.2.8 TEST_CASE() [8/8]

```
TEST_CASE ( )
```

5.5 Sequence.cpp File Reference

```
#include "Sequence.h"
```

Include dependency graph for Sequence.cpp:



Functions

- `bool equal (const Sequence &lhs, const Sequence &rhs)`
Returns true if the lhs parameter compares equal to the rhs parameter.

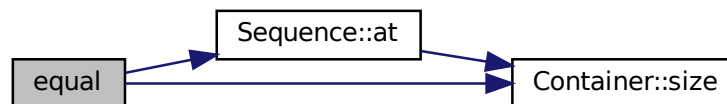
5.5.1 Function Documentation

5.5.1.1 equal()

```
bool equal (
    const Sequence & lhs,
    const Sequence & rhs )
```

Returns true if the lhs parameter compares equal to the rhs parameter.

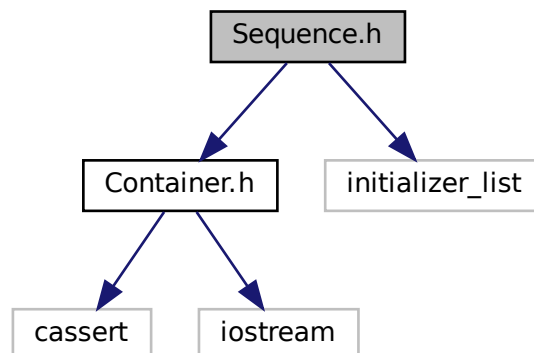
Here is the call graph for this function:



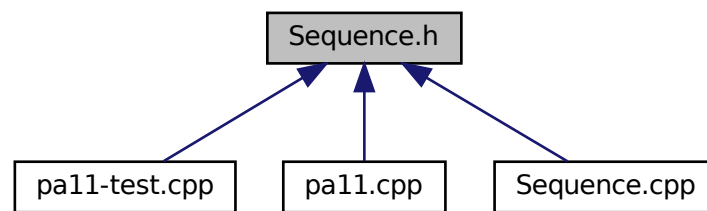
5.6 Sequence.h File Reference

This is the interface for the `Sequence` class.

```
#include "Container.h"
#include <initializer_list>
Include dependency graph for Sequence.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Sequence](#)

Functions

- bool [equal](#) (const [Sequence](#) &lhs, const [Sequence](#) &rhs)
Returns true if the lhs parameter compares equal to the rhs parameter.

5.6.1 Detailed Description

This is the interface for the [Sequence](#) class.

Author

Kevin Mess kevin.mess@csn.edu

Date

05/28/2021

Note

I pledge my word of honor that I have complied with the CSN Academic Integrity Policy while completing this assignment.

This is a sample solution and is but one way to solve the problem.

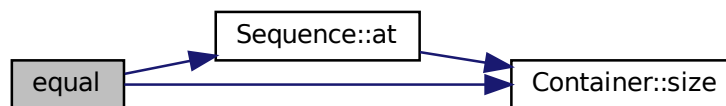
5.6.2 Function Documentation

5.6.2.1 equal()

```
bool equal (
    const Sequence & lhs,
    const Sequence & rhs )
```

Returns true if the lhs parameter compares equal to the rhs parameter.

Here is the call graph for this function:



Index

- at
 - Sequence, [16, 17](#)
- back
 - Sequence, [17, 18](#)
- CAPACITY
 - Container, [14](#)
- CATCH_CONFIG_MAIN
 - pa11-test.cpp, [24](#)
 - pa11.cpp, [27](#)
- clear
 - Container, [8](#)
- Container, [7](#)
 - CAPACITY, [14](#)
 - clear, [8](#)
 - Container, [8](#)
 - contains, [9](#)
 - count, [9](#)
 - data, [14](#)
 - empty, [10](#)
 - erase, [11](#)
 - insert, [11](#)
 - size, [12](#)
 - size_type, [8](#)
 - used, [14](#)
 - value_type, [8](#)
 - write, [13](#)
- Container.cpp, [21](#)
- Container.h, [22](#)
- contains
 - Container, [9](#)
 - Sequence, [18](#)
- count
 - Container, [9](#)
- data
 - Container, [14](#)
- empty
 - Container, [10](#)
- equal
 - Sequence.cpp, [30](#)
 - Sequence.h, [31](#)
- erase
 - Container, [11](#)
 - Sequence, [18](#)
- find
 - Sequence, [19](#)
- front
 - Sequence, [19, 20](#)
- insert
 - Container, [11](#)
- pa11-test.cpp, [23](#)
 - CATCH_CONFIG_MAIN, [24](#)
 - TEST_CASE, [24–26](#)
- pa11.cpp, [26](#)
 - CATCH_CONFIG_MAIN, [27](#)
 - TEST_CASE, [27–29](#)
- Sequence, [15](#)
 - at, [16, 17](#)
 - back, [17, 18](#)
 - contains, [18](#)
 - erase, [18](#)
 - find, [19](#)
 - front, [19, 20](#)
 - Sequence, [16](#)
- Sequence.cpp, [29](#)
 - equal, [30](#)
- Sequence.h, [30](#)
 - equal, [31](#)
- size
 - Container, [12](#)
- size_type
 - Container, [8](#)
- TEST_CASE
 - pa11-test.cpp, [24–26](#)
 - pa11.cpp, [27–29](#)
- used
 - Container, [14](#)
- value_type
 - Container, [8](#)
- write
 - Container, [13](#)