My Project

Generated by Doxygen 1.8.15

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 Sequence< Key, Info >::ConstIterator Class Reference 5

3.1.1 Detailed Description 5

3.2 Sequence< Key, Info >::Iterator Class Reference 6

3.2.1 Detailed Description 6

3.3 Sequence< Key, Info >::IterEnd Struct Reference 6

3.4 Sequence< Key, Info >::Node Class Reference 6

3.4.1 Detailed Description 7

3.5 OwningPtr< T > Class Template Reference 7

3.5.1 Detailed Description 8

3.6 Sequence< Key, Info > Struct Template Reference 8

3.6.1 Member Function Documentation 10

3.6.1.1 append() 10

3.6.1.2 first() [1/2] 10

3.6.1.3 first() [2/2] 10

3.6.1.4 get_elem_by() [1/2] 11

3.6.1.5 get_elem_by() [2/2] 11

3.6.1.6 get_iter_by() [1/2] 11

3.6.1.7 get_iter_by() [2/2] 11

3.6.1.8 insert() 12

3.6.1.9 insert_at() 12

3.6.1.10 last() [1/2] 12

3.6.1.11 last() [2/2] 13

3.6.1.12 popb() 13

3.6.1.13 popf() 13

3.6.1.14 remove_if() 13

4 File Documentation **15**

4.1 sequence.cc File Reference 15

Index **17**

Chapter 1

Class Index

1.1 Class List

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

Sequence< Key, Info >::ConstIterator
Type used to iterate const Sequence 5
Sequence< Key, Info >::Iterator
Type used to iterate non-const Sequence 6
Sequence< Key, Info >::IterEnd 6
Sequence< Key, Info >::Node
Used as a node of a Sequence 6
OwningPtr< T >
Std::unique_ptr wrapper allowing for copying owned value 7
Sequence< Key, Info > 8

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

sequence.cc 15

Chapter 3

Class Documentation

3.1 Sequence< Key, Info >::ConstIterator Class Reference

type used to iterate const Sequence

Public Types

- using **iterator_category** = std::output_iterator_tag
- using **value_type** = typename Node::Elem
- using **difference_type** = std::ptrdiff_t
- using **pointer** = value_type const ∗
- using **reference** = value_type const &

Public Member Functions

- auto **operator++** () -> ConstIterator &
- auto **operator** ∗ () const -> reference

Public Attributes

- std::reference_wrapper< OwningPtr< Node > const > **elem_**

3.1.1 Detailed Description

template<typename Key, typename Info>
class Sequence< Key, Info >::ConstIterator

type used to iterate const Sequence

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

- sequence.cc

3.2 Sequence< Key, Info >::Iterator Class Reference

type used to iterate non-const Sequence

Public Types

- using **iterator_category** = std::forward_iterator_tag
- using **value_type** = typename Node::Elem
- using **difference_type** = std::ptrdiff_t
- using **pointer** = value_type ∗
- using **reference** = value_type &

Public Member Functions

- auto **operator++** () -> Iterator &
- auto **operator** ∗ () const -> reference

Public Attributes

- std::reference_wrapper< OwningPtr< Node > > **elem_**

3.2.1 Detailed Description

template<typename Key, typename Info>
class Sequence< Key, Info >::Iterator

type used to iterate non-const Sequence

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

- sequence.cc

3.3 Sequence< Key, Info >::IterEnd Struct Reference

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

- sequence.cc

3.4 Sequence< Key, Info >::Node Class Reference

used as a node of a Sequence

Public Types

- using **Elem** = std::pair< Key, Info >

Public Member Functions

- template<typename Key_ , typename Info_ >
  Node (Key_ &&k, Info_ &&i)
- **Node** (Elem const &elem)
- **Node** (Elem &&elem)
- template<typename Key_ , typename Info_ , typename... Ts>
  Node (Key_ &&k, Info_ &&i, Ts &&... vs)
- template<typename... Ts>
  Node (Elem const &elem, Ts &&... vs)
- template<typename... Ts>
  Node (Elem &&elem, Ts &&... vs)
- **Node** (Node const &)=default
- **Node** (Node &&)=default
- auto **print** () const -> void
- auto **elem** () const -> Elem const &
- auto **elem** () -> Elem &
- auto **next** () const -> OwningPtr< Node > const &
- auto **next** () -> OwningPtr< Node > &

3.4.1 Detailed Description

template<typename Key, typename Info>
class Sequence< Key, Info >::Node

used as a node of a Sequence

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

- sequence.cc

3.5 OwningPtr< T > Class Template Reference

std::unique_ptr wrapper allowing for copying owned value

Public Types

- using **Inner** = std::unique_ptr< T >
- using **Pointer** = typename Inner::pointer

Public Member Functions

- constexpr **OwningPtr** (std::nullptr_t)
- **OwningPtr** (Pointer p)
- **OwningPtr** (OwningPtr &&)=default
- **OwningPtr** (Inner &&other)
- auto **operator=** (OwningPtr &&rhs) -> OwningPtr &
- **OwningPtr** (OwningPtr const &other)
- **OwningPtr** (Inner const &other)
- auto **operator=** (OwningPtr const &rhs) -> OwningPtr &

Public Attributes

- Inner **inner**

3.5.1 Detailed Description

template<typename T>
class OwningPtr< T >

std::unique_ptr wrapper allowing for copying owned value

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

- sequence.cc

3.6 Sequence< Key, Info > Struct Template Reference

Classes

- class ConstIterator

  type used to iterate const Sequence
- class Iterator

  type used to iterate non-const Sequence
- struct IterEnd
- class Node

  used as a node of a Sequence

Public Member Functions

- template<typename T , typename... Ts, typename = std::enable_if_t< sizeof...(Ts) != 0 || !std::is_same_v< std::remove_reference↵ _t<T>, Sequence > >>
  Sequence (T &&t, Ts &&... vs)
- **Sequence** (Sequence const &other)
- **Sequence** (Sequence &&other)
- auto **empty** () const -> bool
- auto **clear** () -> void
- auto **print** () const -> void
- auto first () const -> Node const &

  returns reference to first node, fires assertion if sequence is empty.
- auto first () -> Node &

  returns reference to first node, fires assertion if sequence is empty.
- auto last () const -> Node const &

  returns reference to last node, fires assertion if sequence is empty.
- auto last () -> Node &

  returns reference to last node, fires assertion if sequence is empty.
- template<typename... Ts>
  auto insert (Ts &&... vs) -> Sequence &

  inserts a sequence or a single node in front of the sequence.
- template<typename... Ts>
  auto append (Ts &&... vs) -> Sequence &

  appends a sequence or a single node at the end of the sequence.
- auto popf () -> typename Node::Elem

  removes and returns first element of the sequence, asserts on empty.
- auto popb () -> typename Node::Elem

  removes and returns last element of the sequence, assserts on empty.
- auto **begin** () -> Iterator
- auto **end** () -> IterEnd
- auto **begin** () const -> ConstIterator
- auto **end** () const -> IterEnd
- template<typename... Ts>
  auto insert_at (Iterator const &iter, Ts &&... vs) -> Sequence &

  insert node or nodes at position denoted by iterator.
- template<typename F >
  auto get_iter_by (F const &func) -> Iterator

  get iterator at elem for which func returned true.
- template<typename F >
  auto get_iter_by (F const &func) const -> ConstIterator

  get iterator at elem for which func returned true.
- template<typename F >
  auto get_elem_by (F const &func) const -> typename Node::Elem const &

  returns elem for which func returned true.
- template<typename F >
  auto get_elem_by (F const &func) -> typename Node::Elem &

  returns elem for which func returned true.
- template<typename F >
  auto remove_if (F const &func) -> Sequence &

  remove first elem fulfilling predicate func.

Friends

- auto **operator !=** (Iterator const &iter, IterEnd const &) -> bool
- auto **operator !=** (ConstIterator const &iter, IterEnd const &) -> bool

3.6.1 Member Function Documentation

3.6.1.1 append()

```
template<typename Key , typename Info >
template<typename... Ts>
auto Sequence< Key, Info >::append (
             Ts &&... vs ) -> Sequence&   [inline]
```

appends a sequence or a single node at the end of the sequence.

on empty sequence equivalent to insert

See also

insert

Returns

self

3.6.1.2 first() [1/2]

```
template<typename Key , typename Info >
auto Sequence< Key, Info >::first ( ) const -> Node const&   [inline]
```

returns reference to first node, fires assertion if sequence is empty.

Returns

reference to first node in sequence

3.6.1.3 first() [2/2]

```
template<typename Key , typename Info >
auto Sequence< Key, Info >::first ( ) -> Node&   [inline]
```

returns reference to first node, fires assertion if sequence is empty.

Returns

reference to first node in sequence

3.6.1.4 get_elem_by() [1/2]

```
template<typename Key , typename Info >
template<typename F >
auto Sequence< Key, Info >::get_elem_by (
             F const & func ) const -> typename Node::Elem const&   [inline]
```

returns elem for which func returned true.

if no such elem exists fires assertion

Returns

elem fulfilling predicate func

3.6.1.5 get_elem_by() [2/2]

```
template<typename Key , typename Info >
template<typename F >
auto Sequence< Key, Info >::get_elem_by (
             F const & func ) -> typename Node::Elem&   [inline]
```

returns elem for which func returned true.

if no such elem exists fires assertion

Returns

elem fulfilling predicate func

3.6.1.6 get_iter_by() [1/2]

```
template<typename Key , typename Info >
template<typename F >
auto Sequence< Key, Info >::get_iter_by (
             F const & func ) -> Iterator   [inline]
```

get iterator at elem for which func returned true.

if no such elem exists return iterator to one pass the end

3.6.1.7 get_iter_by() [2/2]

```
template<typename Key , typename Info >
template<typename F >
auto Sequence< Key, Info >::get_iter_by (
             F const & func ) const -> ConstIterator   [inline]
```

get iterator at elem for which func returned true.

if no such elem exists return iterator to one pass the end

3.6.1.8 insert()

```
template<typename Key , typename Info >
template<typename...  Ts>
auto Sequence< Key, Info >::insert (
             Ts &&...  vs ) -> Sequence&   [inline]
```

inserts a sequence or a single node in front of the sequence.

on empty sequence equivalent to append

See also

append

Returns

self

3.6.1.9 insert_at()

```
template<typename Key , typename Info >
template<typename...  Ts>
auto Sequence< Key, Info >::insert_at (
             Iterator const & iter,
             Ts &&...  vs ) -> Sequence&   [inline]
```

insert node or nodes at position denoted by iterator.

Returns

self

3.6.1.10 last() [1/2]

```
template<typename Key , typename Info >
auto Sequence< Key, Info >::last ( ) const -> Node const&   [inline]
```

returns reference to last node, fires assertion if sequence is empty.

Returns

reference to last node in sequence

3.6.1.11 last() [2/2]

```
template<typename Key , typename Info >
auto Sequence< Key, Info >::last ( ) -> Node&   [inline]
```

returns reference to last node, fires assertion if sequence is empty.

Returns

reference to last node in sequence

3.6.1.12 popb()

```
template<typename Key , typename Info >
auto Sequence< Key, Info >::popb ( ) -> typename Node::Elem   [inline]
```

removes and returns last element of the sequence, assserts on empty.

Returns

removed element

3.6.1.13 popf()

```
template<typename Key , typename Info >
auto Sequence< Key, Info >::popf ( ) -> typename Node::Elem   [inline]
```

removes and returns first element of the sequence, asserts on empty.

Returns

removed element

3.6.1.14 remove_if()

```
template<typename Key , typename Info >
template<typename F >
auto Sequence< Key, Info >::remove_if (
            F const & func ) -> Sequence&   [inline]
```

remove first elem fulfilling predicate func.

if no elem does do nothing

Returns

self

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

- sequence.cc

Chapter 4

File Documentation

4.1 sequence.cc File Reference

```
#include <functional>
#include <iostream>
#include <iterator>
#include <memory>
#include <type_traits>
#include <utility>
#include <cstdio>
#include <cassert>
```

Classes

- class OwningPtr< T >

  std::unique_ptr wrapper allowing for copying owned value
- struct Sequence< Key, Info >
- class Sequence< Key, Info >::Node

  used as a node of a Sequence
- class Sequence< Key, Info >::Iterator

  type used to iterate non-const Sequence
- struct Sequence< Key, Info >::IterEnd
- class Sequence< Key, Info >::ConstIterator

  type used to iterate const Sequence

Functions

- template<typename T , typename... Args>
  auto **make_owning** (Args &&... args) -> OwningPtr< T >
- template<typename Key , typename Info , typename... Ts>
  auto **make_seq** (Key &&k, Info &&i, Ts &&... vs) -> decltype(auto)
- template<typename Key , typename Info >
  auto **produce** (Sequence< Key, Info > const &a, uint start_a, uint len_a, Sequence< Key, Info > const &b, uint start_b, uint len_b, uint limit) -> Sequence< Key, Info >
- auto **main** () -> int

Index

append
    Sequence< Key, Info >, 10

first
    Sequence< Key, Info >, 10

get_elem_by
    Sequence< Key, Info >, 10, 11
get_iter_by
    Sequence< Key, Info >, 11

insert
    Sequence< Key, Info >, 11
insert_at
    Sequence< Key, Info >, 12

last
    Sequence< Key, Info >, 12

OwningPtr< T >, 7

popb
    Sequence< Key, Info >, 13
popf
    Sequence< Key, Info >, 13

remove_if
    Sequence< Key, Info >, 13

Sequence< Key, Info >, 8
    append, 10
    first, 10
    get_elem_by, 10, 11
    get_iter_by, 11
    insert, 11
    insert_at, 12
    last, 12
    popb, 13
    popf, 13
    remove_if, 13
Sequence< Key, Info >::ConstIterator, 5
Sequence< Key, Info >::Iterator, 6
Sequence< Key, Info >::IterEnd, 6
Sequence< Key, Info >::Node, 6
sequence.cc, 15