

My Project

Generated by Doxygen 1.10.0

1 Namespace Index	1
1.1 Namespace List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 std Namespace Reference	7
5 Class Documentation	9
5.1 std::Vector< T, Allocator > Class Template Reference	9
5.1.1 Detailed Description	11
5.1.2 Member Typedef Documentation	11
5.1.2.1 allocator_type	11
5.1.2.2 const_iterator	11
5.1.2.3 const_pointer	11
5.1.2.4 const_reference	11
5.1.2.5 const_reverse_iterator	11
5.1.2.6 difference_type	12
5.1.2.7 iterator	12
5.1.2.8 pointer	12
5.1.2.9 reference	12
5.1.2.10 reverse_iterator	12
5.1.2.11 size_type	12
5.1.2.12 value_type	12
5.1.3 Constructor & Destructor Documentation	13
5.1.3.1 Vector() [1/7]	13
5.1.3.2 Vector() [2/7]	13
5.1.3.3 Vector() [3/7]	13
5.1.3.4 Vector() [4/7]	13
5.1.3.5 ~Vector()	13
5.1.3.6 Vector() [5/7]	13
5.1.3.7 Vector() [6/7]	14
5.1.3.8 Vector() [7/7]	14
5.1.4 Member Function Documentation	14
5.1.4.1 assign() [1/3]	14
5.1.4.2 assign() [2/3]	14
5.1.4.3 assign() [3/3]	14
5.1.4.4 at() [1/2]	15
5.1.4.5 at() [2/2]	15
5.1.4.6 back() [1/2]	15

5.1.4.7 back() [2/2]	15
5.1.4.8 begin() [1/2]	15
5.1.4.9 begin() [2/2]	15
5.1.4.10 capacity()	16
5.1.4.11 cbegin()	16
5.1.4.12 cend()	16
5.1.4.13 clear()	16
5.1.4.14 crbegin()	16
5.1.4.15 crend()	16
5.1.4.16 data() [1/2]	17
5.1.4.17 data() [2/2]	17
5.1.4.18 emplace()	17
5.1.4.19 emplace_back()	17
5.1.4.20 empty()	17
5.1.4.21 end() [1/2]	17
5.1.4.22 end() [2/2]	18
5.1.4.23 erase() [1/2]	18
5.1.4.24 erase() [2/2]	18
5.1.4.25 front() [1/2]	18
5.1.4.26 front() [2/2]	18
5.1.4.27 get_allocator()	18
5.1.4.28 insert() [1/5]	19
5.1.4.29 insert() [2/5]	19
5.1.4.30 insert() [3/5]	19
5.1.4.31 insert() [4/5]	19
5.1.4.32 insert() [5/5]	19
5.1.4.33 max_size()	20
5.1.4.34 operator=() [1/3]	20
5.1.4.35 operator=() [2/3]	20
5.1.4.36 operator=() [3/3]	20
5.1.4.37 operator[]() [1/2]	20
5.1.4.38 operator[]() [2/2]	20
5.1.4.39 pop_back()	21
5.1.4.40 push_back() [1/2]	21
5.1.4.41 push_back() [2/2]	21
5.1.4.42 rbegin() [1/2]	21
5.1.4.43 rbegin() [2/2]	21
5.1.4.44 rend() [1/2]	21
5.1.4.45 rend() [2/2]	22
5.1.4.46 reserve()	22
5.1.4.47 resize() [1/2]	22
5.1.4.48 resize() [2/2]	22

5.1.4.49 shrink_to_fit()	22
5.1.4.50 size()	22
5.1.4.51 swap()	22
6 File Documentation	23
6.1 Testai.cpp File Reference	23
6.1.1 Macro Definition Documentation	23
6.1.1.1 CATCH_CONFIG_MAIN	23
6.2 Testai.cpp	23
6.3 Vector.h File Reference	24
6.4 Vector.h	24
Index	33

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

std	7
-------------------------------	-------------------

Chapter 2

Class Index

2.1 Class List

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

std::Vector< T, Allocator >	9
---	---

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Testai.cpp	23
Vector.h	24

Chapter 4

Namespace Documentation

4.1 std Namespace Reference

Classes

- class [Vector](#)

Chapter 5

Class Documentation

5.1 `std::Vector< T, Allocator >` Class Template Reference

```
#include <Vector.h>
```

Public Types

- `using value_type = T`
- `using allocator_type = Allocator`
- `using pointer = typename allocator_traits<Allocator>::pointer`
- `using const_pointer = typename allocator_traits<Allocator>::const_pointer`
- `using reference = value_type&`
- `using const_reference = const value_type&`
- `using size_type = size_t`
- `using difference_type = ptrdiff_t`
- `using iterator = pointer`
- `using const_iterator = const_pointer`
- `using reverse_iterator = std::reverse_iterator<iterator>`
- `using const_reverse_iterator = std::reverse_iterator<const_iterator>`

Public Member Functions

- `constexpr Vector () noexcept(noexcept(Allocator()))`
- `constexpr Vector (const Allocator &alloc) noexcept`
- `constexpr Vector (size_type n, const Allocator &alloc=Allocator())`
- `constexpr Vector (size_type n, const T &value, const Allocator &alloc=Allocator())`
- `~Vector ()`
- `constexpr Vector (const Vector &x)`
- `constexpr Vector (Vector &&x) noexcept`
- `constexpr Vector & operator= (const Vector &x)`
- `constexpr Vector & operator= (Vector &&x) noexcept(allocator_traits< Allocator >::propagate_on_↔ container_move_assignment::value||allocator_traits< Allocator >::is_always_equal::value)`
- `constexpr Vector (initializer_list< T > il, const Allocator &alloc=Allocator())`
- `constexpr Vector & operator= (initializer_list< T > il)`
- `constexpr iterator begin () noexcept`
- `constexpr const_iterator begin () const noexcept`

- `constexpr iterator end () noexcept`
- `constexpr const_iterator end () const noexcept`
- `constexpr reverse_iterator rbegin () noexcept`
- `constexpr const_reverse_iterator rbegin () const noexcept`
- `constexpr reverse_iterator rend () noexcept`
- `constexpr const_reverse_iterator rend () const noexcept`
- `constexpr const_iterator cbegin () const noexcept`
- `constexpr const_iterator cend () const noexcept`
- `constexpr const_reverse_iterator crbegin () const noexcept`
- `constexpr const_reverse_iterator crend () const noexcept`
- `constexpr bool empty () const noexcept`
- `constexpr size_type size () const noexcept`
- `constexpr size_type max_size () const noexcept`
- `constexpr size_type capacity () const noexcept`
- `constexpr void resize (size_type sz)`
- `constexpr void resize (size_type sz, const T &c)`
- `constexpr void reserve (size_type n)`
- `constexpr void shrink_to_fit ()`
- `constexpr reference operator[] (size_type n)`
- `constexpr const_reference operator[] (size_type n) const`
- `constexpr const_reference at (size_type n) const`
- `constexpr reference at (size_type n)`
- `constexpr reference front ()`
- `constexpr const_reference front () const`
- `constexpr reference back ()`
- `constexpr const_reference back () const`
- `constexpr pointer data () noexcept`
- `constexpr const_pointer data () const noexcept`
- `constexpr void assign (size_type n, const T &u)`
- `template<class InputIterator >`
`constexpr void assign (InputIterator first, InputIterator last)`
- `constexpr void assign (initializer_list< T > il)`
- `constexpr void push_back (const T &x)`
- `constexpr void push_back (T &&x)`
- `template<class... Args>`
`constexpr reference emplace_back (Args &&... args)`
- `constexpr void pop_back ()`
- `constexpr iterator insert (const_iterator position, const T &x)`
- `constexpr iterator insert (const_iterator position, T &&x)`
- `constexpr iterator insert (const_iterator position, size_type n, const T &x)`
- `template<class InputIterator >`
`constexpr iterator insert (const_iterator position, InputIterator first, InputIterator last)`
- `constexpr iterator insert (const_iterator position, initializer_list< T > il)`
- `template<class... Args>`
`constexpr iterator emplace (const_iterator position, Args &&... args)`
- `constexpr iterator erase (const_iterator position)`
- `constexpr iterator erase (const_iterator first, const_iterator last)`
- `constexpr void swap (Vector &x) noexcept(allocator_traits< Allocator >::propagate_on_container_swap←
::value||allocator_traits< Allocator >::is_always_equal::value)`
- `constexpr void clear () noexcept`
- `constexpr Allocator get_allocator () const noexcept`

5.1.1 Detailed Description

```
template<class T, class Allocator = allocator<T>>
class std::Vector< T, Allocator >
```

Definition at line 14 of file [Vector.h](#).

5.1.2 Member Typedef Documentation

5.1.2.1 allocator_type

```
template<class T , class Allocator = allocator<T>>
using std::Vector< T, Allocator >::allocator_type = Allocator
```

Definition at line 23 of file [Vector.h](#).

5.1.2.2 const_iterator

```
template<class T , class Allocator = allocator<T>>
using std::Vector< T, Allocator >::const_iterator = const_pointer
```

Definition at line 31 of file [Vector.h](#).

5.1.2.3 const_pointer

```
template<class T , class Allocator = allocator<T>>
using std::Vector< T, Allocator >::const_pointer = typename allocator_traits<Allocator>↵
::const_pointer
```

Definition at line 25 of file [Vector.h](#).

5.1.2.4 const_reference

```
template<class T , class Allocator = allocator<T>>
using std::Vector< T, Allocator >::const_reference = const value_type&
```

Definition at line 27 of file [Vector.h](#).

5.1.2.5 const_reverse_iterator

```
template<class T , class Allocator = allocator<T>>
using std::Vector< T, Allocator >::const_reverse_iterator = std::reverse_iterator<const_iterator>
```

Definition at line 33 of file [Vector.h](#).

5.1.2.6 difference_type

```
template<class T , class Allocator = allocator<T>>  
using std::Vector< T, Allocator >::difference_type = ptrdiff_t
```

Definition at line 29 of file [Vector.h](#).

5.1.2.7 iterator

```
template<class T , class Allocator = allocator<T>>  
using std::Vector< T, Allocator >::iterator = pointer
```

Definition at line 30 of file [Vector.h](#).

5.1.2.8 pointer

```
template<class T , class Allocator = allocator<T>>  
using std::Vector< T, Allocator >::pointer = typename allocator_traits<Allocator>::pointer
```

Definition at line 24 of file [Vector.h](#).

5.1.2.9 reference

```
template<class T , class Allocator = allocator<T>>  
using std::Vector< T, Allocator >::reference = value_type&
```

Definition at line 26 of file [Vector.h](#).

5.1.2.10 reverse_iterator

```
template<class T , class Allocator = allocator<T>>  
using std::Vector< T, Allocator >::reverse_iterator = std::reverse_iterator<iterator>
```

Definition at line 32 of file [Vector.h](#).

5.1.2.11 size_type

```
template<class T , class Allocator = allocator<T>>  
using std::Vector< T, Allocator >::size_type = size_t
```

Definition at line 28 of file [Vector.h](#).

5.1.2.12 value_type

```
template<class T , class Allocator = allocator<T>>  
using std::Vector< T, Allocator >::value_type = T
```

Definition at line 22 of file [Vector.h](#).

5.1.3 Constructor & Destructor Documentation

5.1.3.1 Vector() [1/7]

```
template<class T , class Allocator = allocator<T>>
constexpr std::Vector< T, Allocator >::Vector ( ) [inline], [constexpr], [noexcept]
```

Definition at line 36 of file [Vector.h](#).

5.1.3.2 Vector() [2/7]

```
template<class T , class Allocator = allocator<T>>
constexpr std::Vector< T, Allocator >::Vector (
    const Allocator & alloc ) [inline], [explicit], [constexpr], [noexcept]
```

Definition at line 37 of file [Vector.h](#).

5.1.3.3 Vector() [3/7]

```
template<class T , class Allocator = allocator<T>>
constexpr std::Vector< T, Allocator >::Vector (
    size_type n,
    const Allocator & alloc = Allocator() ) [inline], [explicit], [constexpr]
```

Definition at line 41 of file [Vector.h](#).

5.1.3.4 Vector() [4/7]

```
template<class T , class Allocator = allocator<T>>
constexpr std::Vector< T, Allocator >::Vector (
    size_type n,
    const T & value,
    const Allocator & alloc = Allocator() ) [inline], [constexpr]
```

Definition at line 45 of file [Vector.h](#).

5.1.3.5 ~Vector()

```
template<class T , class Allocator = allocator<T>>
std::Vector< T, Allocator >::~~Vector ( ) [inline]
```

Definition at line 50 of file [Vector.h](#).

5.1.3.6 Vector() [5/7]

```
template<class T , class Allocator = allocator<T>>
constexpr std::Vector< T, Allocator >::Vector (
    const Vector< T, Allocator > & x ) [inline], [constexpr]
```

Definition at line 55 of file [Vector.h](#).

5.1.3.7 Vector() [6/7]

```
template<class T , class Allocator = allocator<T>>
constexpr std::Vector< T, Allocator >::Vector (
    Vector< T, Allocator > && x ) [inline], [constexpr], [noexcept]
```

Definition at line 67 of file [Vector.h](#).

5.1.3.8 Vector() [7/7]

```
template<class T , class Allocator = allocator<T>>
constexpr std::Vector< T, Allocator >::Vector (
    initializer_list< T > il,
    const Allocator & alloc = Allocator() ) [inline], [constexpr]
```

Definition at line 132 of file [Vector.h](#).

5.1.4 Member Function Documentation**5.1.4.1 assign()** [1/3]

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::assign (
    initializer_list< T > il ) [inline], [constexpr]
```

Definition at line 370 of file [Vector.h](#).

5.1.4.2 assign() [2/3]

```
template<class T , class Allocator = allocator<T>>
template<class InputIterator >
constexpr void std::Vector< T, Allocator >::assign (
    InputIterator first,
    InputIterator last ) [inline], [constexpr]
```

Definition at line 364 of file [Vector.h](#).

5.1.4.3 assign() [3/3]

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::assign (
    size_type n,
    const T & u ) [inline], [constexpr]
```

Definition at line 352 of file [Vector.h](#).

5.1.4.4 `at()` [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr reference std::Vector< T, Allocator >::at (
    size_type n ) [inline], [constexpr]
```

Definition at line 318 of file [Vector.h](#).

5.1.4.5 `at()` [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_reference std::Vector< T, Allocator >::at (
    size_type n ) const [inline], [constexpr]
```

Definition at line 310 of file [Vector.h](#).

5.1.4.6 `back()` [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr reference std::Vector< T, Allocator >::back ( ) [inline], [constexpr]
```

Definition at line 334 of file [Vector.h](#).

5.1.4.7 `back()` [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_reference std::Vector< T, Allocator >::back ( ) const [inline], [constexpr]
```

Definition at line 338 of file [Vector.h](#).

5.1.4.8 `begin()` [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_iterator std::Vector< T, Allocator >::begin ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 155 of file [Vector.h](#).

5.1.4.9 `begin()` [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::begin ( ) [inline], [constexpr], [noexcept]
```

Definition at line 150 of file [Vector.h](#).

5.1.4.10 capacity()

```
template<class T , class Allocator = allocator<T>>
constexpr size_type std::Vector< T, Allocator >::capacity ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 223 of file [Vector.h](#).

5.1.4.11 cbegin()

```
template<class T , class Allocator = allocator<T>>
constexpr const_iterator std::Vector< T, Allocator >::cbegin ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 190 of file [Vector.h](#).

5.1.4.12 cend()

```
template<class T , class Allocator = allocator<T>>
constexpr const_iterator std::Vector< T, Allocator >::cend ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 195 of file [Vector.h](#).

5.1.4.13 clear()

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::clear ( ) [inline], [constexpr], [noexcept]
```

Definition at line 575 of file [Vector.h](#).

5.1.4.14 crbegin()

```
template<class T , class Allocator = allocator<T>>
constexpr const_reverse_iterator std::Vector< T, Allocator >::crbegin ( ) const [inline],
[constexpr], [noexcept]
```

Definition at line 200 of file [Vector.h](#).

5.1.4.15 crend()

```
template<class T , class Allocator = allocator<T>>
constexpr const_reverse_iterator std::Vector< T, Allocator >::crend ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 205 of file [Vector.h](#).

5.1.4.16 data() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_pointer std::Vector< T, Allocator >::data ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 346 of file [Vector.h](#).

5.1.4.17 data() [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr pointer std::Vector< T, Allocator >::data ( ) [inline], [constexpr], [noexcept]
```

Definition at line 342 of file [Vector.h](#).

5.1.4.18 emplace()

```
template<class T , class Allocator = allocator<T>>
template<class... Args>
constexpr iterator std::Vector< T, Allocator >::emplace (
    const_iterator position,
    Args &&... args ) [inline], [constexpr]
```

Definition at line 502 of file [Vector.h](#).

5.1.4.19 emplace_back()

```
template<class T , class Allocator = allocator<T>>
template<class... Args>
constexpr reference std::Vector< T, Allocator >::emplace_back (
    Args &&... args ) [inline], [constexpr]
```

Definition at line 387 of file [Vector.h](#).

5.1.4.20 empty()

```
template<class T , class Allocator = allocator<T>>
constexpr bool std::Vector< T, Allocator >::empty ( ) const [inline], [constexpr], [noexcept]
```

Definition at line 211 of file [Vector.h](#).

5.1.4.21 end() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_iterator std::Vector< T, Allocator >::end ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 165 of file [Vector.h](#).

5.1.4.22 end() [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::end ( ) [inline], [constexpr], [noexcept]
```

Definition at line 160 of file [Vector.h](#).

5.1.4.23 erase() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::erase (
    const_iterator first,
    const_iterator last ) [inline], [constexpr]
```

Definition at line 537 of file [Vector.h](#).

5.1.4.24 erase() [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::erase (
    const_iterator position ) [inline], [constexpr]
```

Definition at line 523 of file [Vector.h](#).

5.1.4.25 front() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr reference std::Vector< T, Allocator >::front ( ) [inline], [constexpr]
```

Definition at line 326 of file [Vector.h](#).

5.1.4.26 front() [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_reference std::Vector< T, Allocator >::front ( ) const [inline], [constexpr]
```

Definition at line 330 of file [Vector.h](#).

5.1.4.27 get_allocator()

```
template<class T , class Allocator = allocator<T>>
constexpr Allocator std::Vector< T, Allocator >::get_allocator ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 584 of file [Vector.h](#).

5.1.4.28 insert() [1/5]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::insert (
    const_iterator position,
    const T & x ) [inline], [constexpr]
```

Definition at line 404 of file [Vector.h](#).

5.1.4.29 insert() [2/5]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::insert (
    const_iterator position,
    initializer_list< T > il ) [inline], [constexpr]
```

Definition at line 496 of file [Vector.h](#).

5.1.4.30 insert() [3/5]

```
template<class T , class Allocator = allocator<T>>
template<class InputIterator >
constexpr iterator std::Vector< T, Allocator >::insert (
    const_iterator position,
    InputIterator first,
    InputIterator last ) [inline], [constexpr]
```

Definition at line 471 of file [Vector.h](#).

5.1.4.31 insert() [4/5]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::insert (
    const_iterator position,
    size_type n,
    const T & x ) [inline], [constexpr]
```

Definition at line 446 of file [Vector.h](#).

5.1.4.32 insert() [5/5]

```
template<class T , class Allocator = allocator<T>>
constexpr iterator std::Vector< T, Allocator >::insert (
    const_iterator position,
    T && x ) [inline], [constexpr]
```

Definition at line 425 of file [Vector.h](#).

5.1.4.33 max_size()

```
template<class T , class Allocator = allocator<T>>
constexpr size_type std::Vector< T, Allocator >::max_size ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 219 of file [Vector.h](#).

5.1.4.34 operator=() [1/3]

```
template<class T , class Allocator = allocator<T>>
constexpr Vector & std::Vector< T, Allocator >::operator= (
    const Vector< T, Allocator > & x ) [inline], [constexpr]
```

Definition at line 73 of file [Vector.h](#).

5.1.4.35 operator=() [2/3]

```
template<class T , class Allocator = allocator<T>>
constexpr Vector & std::Vector< T, Allocator >::operator= (
    initializer_list< T > il ) [inline], [constexpr]
```

Definition at line 140 of file [Vector.h](#).

5.1.4.36 operator=() [3/3]

```
template<class T , class Allocator = allocator<T>>
constexpr Vector & std::Vector< T, Allocator >::operator= (
    Vector< T, Allocator > && x ) [inline], [constexpr], [noexcept]
```

Definition at line 103 of file [Vector.h](#).

5.1.4.37 operator[]() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr reference std::Vector< T, Allocator >::operator[] (
    size_type n ) [inline], [constexpr]
```

Definition at line 302 of file [Vector.h](#).

5.1.4.38 operator[]() [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_reference std::Vector< T, Allocator >::operator[] (
    size_type n ) const [inline], [constexpr]
```

Definition at line 306 of file [Vector.h](#).

5.1.4.39 pop_back()

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::pop_back ( ) [inline], [constexpr]
```

Definition at line 398 of file [Vector.h](#).

5.1.4.40 push_back() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::push_back (
    const T & x ) [inline], [constexpr]
```

Definition at line 376 of file [Vector.h](#).

5.1.4.41 push_back() [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::push_back (
    T && x ) [inline], [constexpr]
```

Definition at line 381 of file [Vector.h](#).

5.1.4.42 rbegin() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_reverse_iterator std::Vector< T, Allocator >::rbegin ( ) const [inline],
[constexpr], [noexcept]
```

Definition at line 175 of file [Vector.h](#).

5.1.4.43 rbegin() [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr reverse_iterator std::Vector< T, Allocator >::rbegin ( ) [inline], [constexpr],
[noexcept]
```

Definition at line 170 of file [Vector.h](#).

5.1.4.44 rend() [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr const_reverse_iterator std::Vector< T, Allocator >::rend ( ) const [inline], [constexpr],
[noexcept]
```

Definition at line 185 of file [Vector.h](#).

5.1.4.45 `rend()` [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr reverse_iterator std::Vector< T, Allocator >::rend ( ) [inline], [constexpr], [noexcept]
```

Definition at line 180 of file [Vector.h](#).

5.1.4.46 `reserve()`

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::reserve (
    size_type n ) [inline], [constexpr]
```

Definition at line 274 of file [Vector.h](#).

5.1.4.47 `resize()` [1/2]

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::resize (
    size_type sz ) [inline], [constexpr]
```

Definition at line 228 of file [Vector.h](#).

5.1.4.48 `resize()` [2/2]

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::resize (
    size_type sz,
    const T & c ) [inline], [constexpr]
```

Definition at line 251 of file [Vector.h](#).

5.1.4.49 `shrink_to_fit()`

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::shrink_to_fit ( ) [inline], [constexpr]
```

Definition at line 293 of file [Vector.h](#).

5.1.4.50 `size()`

```
template<class T , class Allocator = allocator<T>>
constexpr size_type std::Vector< T, Allocator >::size ( ) const [inline], [constexpr], [noexcept]
```

Definition at line 215 of file [Vector.h](#).

5.1.4.51 `swap()`

```
template<class T , class Allocator = allocator<T>>
constexpr void std::Vector< T, Allocator >::swap (
    Vector< T, Allocator > & x ) [inline], [constexpr], [noexcept]
```

Definition at line 555 of file [Vector.h](#).

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

- [Vector.h](#)

Chapter 6

File Documentation

6.1 Testai.cpp File Reference

```
#include "catch2/catch.hpp"  
#include "Vector.h"
```

Macros

- #define [CATCH_CONFIG_MAIN](#)

6.1.1 Macro Definition Documentation

6.1.1.1 CATCH_CONFIG_MAIN

```
#define CATCH_CONFIG_MAIN
```

Definition at line 1 of file [Testai.cpp](#).

6.2 Testai.cpp

[Go to the documentation of this file.](#)

```
00001 #define CATCH_CONFIG_MAIN  
00002 #include "catch2/catch.hpp"  
00003 #include "Vector.h"  
00004  
00005  
00006  
00007
```

6.3 Vector.h File Reference

```
#include <memory>
#include <iterator>
#include <algorithm>
#include <initializer_list>
#include <stdexcept>
#include <utility>
```

Classes

- class [std::Vector< T, Allocator >](#)

Namespaces

- namespace [std](#)

6.4 Vector.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003
00004 #include <memory>
00005 #include <iterator>
00006 #include <algorithm>
00007 #include <initializer_list>
00008 #include <stdexcept>
00009 #include <utility>
00010
00011 namespace std
00012 {
00013     template<class T, class Allocator = allocator<T>
00014     class Vector
00015     {
00016     public:
00017         typename allocator_traits<Allocator>::pointer data__;
00018         size_t size__;
00019         size_t capacity__;
00020         Allocator alloc__;
00021
00022         // types
00023         using value_type = T;
00024         using allocator_type = Allocator;
00025         using pointer = typename allocator_traits<Allocator>::pointer;
00026         using const_pointer = typename allocator_traits<Allocator>::const_pointer;
00027         using reference = value_type&;
00028         using const_reference = const value_type&;
00029         using size_type = size_t;
00030         using difference_type = ptrdiff_t;
00031         using iterator = pointer;
00032         using const_iterator = const_pointer;
00033         using reverse_iterator = std::reverse_iterator<iterator>;
00034         using const_reverse_iterator = std::reverse_iterator<const_iterator>;
00035
00036         // construct/copy/destroy
00037         constexpr Vector() noexcept (noexcept (Allocator{})) : Vector(Allocator{}) { }
00038         constexpr explicit Vector(const Allocator& alloc) noexcept : data__(nullptr), size__(0),
00039         capacity__(0), alloc__(alloc)
00040         {
00041             reserve(1);
00042         };
00043         constexpr explicit Vector(size_type n, const Allocator& alloc = Allocator()) : data__(nullptr),
00044         size__(0), capacity__(0), alloc__(alloc)
00045         {
00046             resize(n);
00047         };
00048     };
00049 }
```

```

00045     constexpr Vector(size_type n, const T& value, const Allocator& alloc = Allocator()):
data__(nullptr), size__(0), capacity__(0), alloc__(alloc)
00046     {
00047         resize(n, value);
00048     };
00049
00050     ~Vector() // I. destructor
00051     {
00052         if(data__)
00053             allocator_traits<Allocator>::deallocate(alloc__, data__, capacity__);
00054     };
00055     constexpr Vector(const Vector& x) // II. copy constructor
00056     : data__(nullptr), size__(0),
00057       capacity__(0),
00058       alloc__(allocator_traits<Allocator>::select_on_container_copy_construction(x.alloc__))
00059     {
00060         reserve(x.size__);
00061         for (size_type i = 0; i < x.size__; ++i) {
00062             allocator_traits<Allocator>::construct(alloc__, &data__[i], x.data__[i]);
00063         }
00064         size__ = x.size__;
00065     }
00066
00067     constexpr Vector(Vector&& x) noexcept // III. move constructor
00068     : data__(std::exchange(x.data__, nullptr)),
00069       size__(std::exchange(x.size__, 0)),
00070       capacity__(std::exchange(x.capacity__, 0)),
00071       alloc__(std::move(x.alloc__)) {}
00072
00073     constexpr Vector& operator=(const Vector& x) { // IV. copy assignment
00074         if (this != &x) {
00075             if (allocator_traits<Allocator>::propagate_on_container_copy_assignment::value && alloc__
!= x.alloc__) {
00076                 if (data__) {
00077                     clear();
00078                     allocator_traits<Allocator>::deallocate(alloc__, data__, capacity__);
00079                 }
00080                 alloc__ = x.alloc__;
00081                 data__ = nullptr;
00082                 capacity__ = 0;
00083             }
00084             reserve(x.size__);
00085             if (x.size__ <= size__) {
00086                 std::copy(x.data__, x.data__ + x.size__, data__);
00087                 for (size_type i = x.size__; i < size__; ++i) {
00088                     allocator_traits<Allocator>::destroy(alloc__, &data__[i]);
00089                 }
00090             } else {
00091                 for (size_type i = 0; i < size__; ++i) {
00092                     allocator_traits<Allocator>::construct(alloc__, &data__[i], x.data__[i]);
00093                 }
00094                 for (size_type i = size__; i < x.size__; ++i) {
00095                     allocator_traits<Allocator>::construct(alloc__, &data__[i], x.data__[i]);
00096                 }
00097             }
00098             size__ = x.size__;
00099         }
00100         return *this;
00101     }
00102
00103     constexpr Vector& operator=(Vector&& x) // V. move assignment
00104     noexcept(
00105         allocator_traits<Allocator>::propagate_on_container_move_assignment::value ||
00106         allocator_traits<Allocator>::is_always_equal::value
00107     ){
00108         if (this != &x) {
00109             if (allocator_traits<Allocator>::propagate_on_container_move_assignment::value) {
00110                 if (data__) {
00111                     clear();
00112                     allocator_traits<Allocator>::deallocate(alloc__, data__, capacity__);
00113                 }
00114                 alloc__ = std::move(x.alloc__);
00115                 data__ = std::exchange(x.data__, nullptr);
00116                 size__ = std::exchange(x.size__, 0);
00117                 capacity__ = std::exchange(x.capacity__, 0);
00118             } else {
00119                 if (data__) {
00120                     clear();
00121                     allocator_traits<Allocator>::deallocate(alloc__, data__, capacity__);
00122                 }
00123
00124                 data__ = std::exchange(x.data__, nullptr);
00125                 size__ = std::exchange(x.size__, 0);
00126                 capacity__ = std::exchange(x.capacity__, 0);
00127             }
00128         }
00129         return *this;

```

```

00130     }
00131
00132     constexpr Vector(initializer_list<T> il, const Allocator& alloc = Allocator()) : data__(nullptr),
size__(0), capacity__(0), alloc__(alloc)
00133     {
00134         reserve(il.size());
00135         for (auto& elem : il) {
00136             allocator_traits<Allocator>::construct(alloc__, &data__[size__++], elem);
00137         }
00138     }
00139
00140     constexpr Vector& operator=(initializer_list<T> il) {
00141         clear();
00142         reserve(il.size());
00143         for (auto& elem : il) {
00144             allocator_traits<Allocator>::construct(alloc__, &data__[size__++], elem);
00145         }
00146         return *this;
00147     }
00148
00149     // iterators
00150     constexpr iterator begin() noexcept
00151     {
00152         return data__;
00153     }
00154
00155     constexpr const_iterator begin() const noexcept
00156     {
00157         return data__;
00158     }
00159
00160     constexpr iterator end() noexcept
00161     {
00162         return data__ + size__;
00163     }
00164
00165     constexpr const_iterator end() const noexcept
00166     {
00167         return data__ + size__;
00168     }
00169
00170     constexpr reverse_iterator rbegin() noexcept
00171     {
00172         return reverse_iterator(end());
00173     }
00174
00175     constexpr const_reverse_iterator rbegin() const noexcept
00176     {
00177         return const_reverse_iterator(end());
00178     }
00179
00180     constexpr reverse_iterator rend() noexcept
00181     {
00182         return reverse_iterator(begin());
00183     }
00184
00185     constexpr const_reverse_iterator rend() const noexcept
00186     {
00187         return const_reverse_iterator(begin());
00188     }
00189
00190     constexpr const_iterator cbegin() const noexcept
00191     {
00192         return data__;
00193     }
00194
00195     constexpr const_iterator cend() const noexcept
00196     {
00197         return data__ + size__;
00198     }
00199
00200     constexpr const_reverse_iterator crbegin() const noexcept
00201     {
00202         return const_reverse_iterator(end());
00203     }
00204
00205     constexpr const_reverse_iterator crend() const noexcept
00206     {
00207         return const_reverse_iterator(begin());
00208     }
00209
00210     // capacity
00211     [[nodiscard]] constexpr bool empty() const noexcept
00212     {
00213         return size__==0;
00214     }
00215     constexpr size_type size() const noexcept

```



```

00216     {
00217         return size__;
00218     }
00219     constexpr size_type max_size() const noexcept
00220     {
00221         return allocator_traits<Allocator>::max_size(alloc__);
00222     }
00223     constexpr size_type capacity() const noexcept
00224     {
00225         return capacity__;
00226     }
00227
00228     constexpr void      resize(size_type sz)
00229     {
00230         if (sz > capacity__)
00231         {
00232             reserve(sz);
00233         }
00234         if (sz > size__)
00235         {
00236             for (size_type i = size__; i < sz; ++i)
00237             {
00238                 allocator_traits<Allocator>::construct(alloc__, &data__[i]);
00239             }
00240         }
00241         else
00242         {
00243             for (size_type i = sz; i < size__; ++i)
00244             {
00245                 allocator_traits<Allocator>::destroy(alloc__, &data__[i]);
00246             }
00247         }
00248         size__ = sz;
00249     };
00250
00251     constexpr void      resize(size_type sz, const T& c)
00252     {
00253         if (sz > capacity__)
00254         {
00255             reserve(sz);
00256         }
00257         if (sz > size__)
00258         {
00259             for (size_type i = size__; i < sz; ++i)
00260             {
00261                 allocator_traits<Allocator>::construct(alloc__, &data__[i], c);
00262             }
00263         }
00264         else
00265         {
00266             for (size_type i = sz; i < size__; ++i)
00267             {
00268                 allocator_traits<Allocator>::destroy(alloc__, &data__[i]);
00269             }
00270         }
00271         size__ = sz;
00272     }
00273
00274     constexpr void      reserve(size_type n)
00275     {
00276         if (n > capacity__)
00277         {
00278             pointer new_data = allocator_traits<Allocator>::allocate(alloc__, n);
00279             for (size_type i = 0; i < size__; ++i)
00280             {
00281                 allocator_traits<Allocator>::construct(alloc__, &new_data[i], std::move(data__[i]));
00282                 allocator_traits<Allocator>::destroy(alloc__, &data__[i]);
00283             }
00284             if (data__)
00285             {
00286                 allocator_traits<Allocator>::deallocate(alloc__, data__, capacity__);
00287             }
00288             data__ = new_data;
00289             capacity__ = n;
00290         }
00291     }
00292
00293     constexpr void      shrink_to_fit()
00294     {
00295         if (size__ < capacity__)
00296         {
00297             resize(size__);
00298         }
00299     };
00300
00301     // element access
00302     constexpr reference  operator[](size_type n)

```

```

00303     {
00304         return data__[n];
00305     }
00306     constexpr const_reference operator[](size_type n) const
00307     {
00308         return data__[n];
00309     }
00310     constexpr const_reference at(size_type n) const
00311     {
00312         if (n >= size__)
00313         {
00314             throw std::out_of_range("Vector::at: index out of range");
00315         }
00316         return data__[n];
00317     }
00318     constexpr reference at(size_type n)
00319     {
00320         if (n >= size__)
00321         {
00322             throw std::out_of_range("Vector::at: index out of range");
00323         }
00324         return data__[n];
00325     }
00326     constexpr reference front()
00327     {
00328         return data__[0];
00329     }
00330     constexpr const_reference front() const
00331     {
00332         return data__[0];
00333     }
00334     constexpr reference back()
00335     {
00336         return data__[size__ - 1];
00337     }
00338     constexpr const_reference back() const
00339     {
00340         return data__[size__ - 1];
00341     }
00342     constexpr pointer data() noexcept
00343     {
00344         return data__;
00345     }
00346     constexpr const_pointer data() const noexcept
00347     {
00348         return data__;
00349     }
00350
00351     // modifiers
00352     constexpr void assign(size_type n, const T& u)
00353     {
00354         clear();
00355         reserve(n);
00356         for (size_type i = 0; i < n; ++i)
00357         {
00358             allocator_traits<Allocator>::construct(alloc__, &data__[i], u);
00359         }
00360         size__ = n;
00361     }
00362
00363     template<class InputIterator>
00364     constexpr void assign(InputIterator first, InputIterator last)
00365     {
00366         clear();
00367         insert(begin(), first, last);
00368     }
00369
00370     constexpr void assign(initializer_list<T> il)
00371     {
00372         clear();
00373         insert(begin(), il);
00374     }
00375
00376     constexpr void push_back(const T& x)
00377     {
00378         emplace_back(x);
00379     }
00380
00381     constexpr void push_back(T&& x)
00382     {
00383         emplace_back(std::move(x));
00384     }
00385
00386     template<class... Args>
00387     constexpr reference emplace_back(Args&&... args)
00388     {
00389         if (size__ == capacity__)

```

```

00390     {
00391         reserve(size__ > 0 ? 2 * size__ : 1);
00392     }
00393     allocator_traits<Allocator>::construct(alloc__, &data__[size__], std::forward<Args>(args)...);
00394     ++size__;
00395     return data__[size__ - 1];
00396 }
00397
00398 constexpr void pop_back()
00399 {
00400     allocator_traits<Allocator>::destroy(alloc__, &data__[size__ - 1]);
00401     --size__;
00402 }
00403
00404 constexpr iterator insert(const_iterator position, const T& x)
00405 {
00406     difference_type offset = position - cbegin();
00407     if (size__ == capacity__)
00408     {
00409         reserve(size__ > 0 ? 2 * size__ : 1);
00410     }
00411     iterator pos = begin() + offset;
00412     if (pos != end())
00413     {
00414         for (iterator it = end(); it != pos; --it)
00415         {
00416             allocator_traits<Allocator>::construct(alloc__, it, std::move(*(it - 1)));
00417             allocator_traits<Allocator>::destroy(alloc__, it - 1);
00418         }
00419     }
00420     allocator_traits<Allocator>::construct(alloc__, pos, x);
00421     ++size__;
00422     return pos;
00423 }
00424
00425 constexpr iterator insert(const_iterator position, T&& x)
00426 {
00427     difference_type offset = position - cbegin();
00428     if (size__ == capacity__)
00429     {
00430         reserve(size__ > 0 ? 2 * size__ : 1);
00431     }
00432     iterator pos = begin() + offset;
00433     if (pos != end())
00434     {
00435         for (iterator it = end(); it != pos; --it)
00436         {
00437             allocator_traits<Allocator>::construct(alloc__, it, std::move(*(it - 1)));
00438             allocator_traits<Allocator>::destroy(alloc__, it - 1);
00439         }
00440     }
00441     allocator_traits<Allocator>::construct(alloc__, pos, std::move(x));
00442     ++size__;
00443     return pos;
00444 }
00445
00446 constexpr iterator insert(const_iterator position, size_type n, const T& x)
00447 {
00448     difference_type offset = position - cbegin();
00449     if (size__ + n > capacity__)
00450     {
00451         reserve(size__ + n);
00452     }
00453     iterator pos = begin() + offset;
00454     if (pos != end())
00455     {
00456         for (iterator it = end() + n - 1; it != pos + n - 1; --it)
00457         {
00458             allocator_traits<Allocator>::construct(alloc__, it, std::move(*(it - n)));
00459             allocator_traits<Allocator>::destroy(alloc__, it - n);
00460         }
00461     }
00462     for (iterator it = pos; it != pos + n; ++it)
00463     {
00464         allocator_traits<Allocator>::construct(alloc__, it, x);
00465     }
00466     size__ += n;
00467     return pos;
00468 }
00469
00470 template<class InputIterator>
00471 constexpr iterator insert(const_iterator position, InputIterator first, InputIterator last)
00472 {
00473     difference_type offset = position - cbegin();
00474     size_type n = std::distance(first, last);
00475     if (size__ + n > capacity__)
00476     {

```

```

00477         reserve(size__ + n);
00478     }
00479     iterator pos = begin() + offset;
00480     if (pos != end())
00481     {
00482         for (iterator it = end() + n - 1; it != pos + n - 1; --it)
00483         {
00484             allocator_traits<Allocator>::construct(alloc__, it, std::move(*(it - n)));
00485             allocator_traits<Allocator>::destroy(alloc__, it - n);
00486         }
00487     }
00488     for (iterator it = pos; first != last; ++it, ++first)
00489     {
00490         allocator_traits<Allocator>::construct(alloc__, it, *first);
00491     }
00492     size__ += n;
00493     return pos;
00494 }
00495
00496 constexpr iterator insert(const_iterator position, initializer_list<T> il)
00497 {
00498     return insert(position, il.begin(), il.end());
00499 }
00500
00501 template<class... Args>
00502 constexpr iterator emplace(const_iterator position, Args&&... args)
00503 {
00504     difference_type offset = position - cbegin();
00505     if (size__ == capacity__)
00506     {
00507         reserve(size__ > 0 ? 2 * size__ : 1);
00508     }
00509     iterator pos = begin() + offset;
00510     if (pos != end())
00511     {
00512         for (iterator it = end(); it != pos; --it)
00513         {
00514             allocator_traits<Allocator>::construct(alloc__, it, std::move(*(it - 1)));
00515             allocator_traits<Allocator>::destroy(alloc__, it - 1);
00516         }
00517     }
00518     allocator_traits<Allocator>::construct(alloc__, pos, std::forward<Args>(args)...);
00519     ++size__;
00520     return pos;
00521 }
00522
00523 constexpr iterator erase(const_iterator position)
00524 {
00525     difference_type offset = position - cbegin();
00526     iterator pos = begin() + offset;
00527     allocator_traits<Allocator>::destroy(alloc__, pos);
00528     for (iterator it = pos; it != end() - 1; ++it)
00529     {
00530         allocator_traits<Allocator>::construct(alloc__, it, std::move(*(it + 1)));
00531         allocator_traits<Allocator>::destroy(alloc__, it + 1);
00532     }
00533     --size__;
00534     return pos;
00535 }
00536
00537 constexpr iterator erase(const_iterator first, const_iterator last)
00538 {
00539     difference_type offset = first - cbegin();
00540     iterator pos = begin() + offset;
00541     difference_type n = last - first;
00542     for (iterator it = pos; it != pos + n; ++it)
00543     {
00544         allocator_traits<Allocator>::destroy(alloc__, it);
00545     }
00546     for (iterator it = pos; it != end() - n; ++it)
00547     {
00548         allocator_traits<Allocator>::construct(alloc__, it, std::move(*(it + n)));
00549         allocator_traits<Allocator>::destroy(alloc__, it + n);
00550     }
00551     size__ -= n;
00552     return pos;
00553 }
00554
00555 constexpr void swap(Vector& x) noexcept(
00556     allocator_traits<Allocator>::propagate_on_container_swap::value ||
00557     allocator_traits<Allocator>::is_always_equal::value)
00558 {
00559     if (this != &x)
00560     {
00561         if (allocator_traits<Allocator>::propagate_on_container_swap::value || alloc__ ==
00562             x.alloc__)
00563         {

```

```
00563         std::swap(data__, x.data__);
00564         std::swap(size__, x.size__);
00565         std::swap(capacity__, x.capacity__);
00566         std::swap(alloc__, x.alloc__);
00567     }
00568     else
00569     {
00570         throw std::runtime_error("Allocator mismatch in Vector::swap");
00571     }
00572 }
00573 }
00574
00575 constexpr void clear() noexcept
00576 {
00577     for (size_type i = 0; i < size__; ++i)
00578     {
00579         allocator_traits<Allocator>::destroy(alloc__, &data__[i]);
00580     }
00581     size__ = 0;
00582 }
00583
00584 constexpr Allocator get_allocator() const noexcept
00585 {
00586     return alloc__;
00587 }
00588 };
00589
00590 }
00591 #endif
00592
00593
00594
```


Index

`~Vector`
 `std::Vector< T, Allocator >`, [13](#)

`allocator_type`
 `std::Vector< T, Allocator >`, [11](#)

`assign`
 `std::Vector< T, Allocator >`, [14](#)

`at`
 `std::Vector< T, Allocator >`, [14](#), [15](#)

`back`
 `std::Vector< T, Allocator >`, [15](#)

`begin`
 `std::Vector< T, Allocator >`, [15](#)

`capacity`
 `std::Vector< T, Allocator >`, [15](#)

`CATCH_CONFIG_MAIN`
 Testai.cpp, [23](#)

`cbegin`
 `std::Vector< T, Allocator >`, [16](#)

`cend`
 `std::Vector< T, Allocator >`, [16](#)

`clear`
 `std::Vector< T, Allocator >`, [16](#)

`const_iterator`
 `std::Vector< T, Allocator >`, [11](#)

`const_pointer`
 `std::Vector< T, Allocator >`, [11](#)

`const_reference`
 `std::Vector< T, Allocator >`, [11](#)

`const_reverse_iterator`
 `std::Vector< T, Allocator >`, [11](#)

`crbegin`
 `std::Vector< T, Allocator >`, [16](#)

`crend`
 `std::Vector< T, Allocator >`, [16](#)

`data`
 `std::Vector< T, Allocator >`, [16](#), [17](#)

`difference_type`
 `std::Vector< T, Allocator >`, [11](#)

`emplace`
 `std::Vector< T, Allocator >`, [17](#)

`emplace_back`
 `std::Vector< T, Allocator >`, [17](#)

`empty`
 `std::Vector< T, Allocator >`, [17](#)

`end`
 `std::Vector< T, Allocator >`, [17](#)

`erase`
 `std::Vector< T, Allocator >`, [18](#)

`front`
 `std::Vector< T, Allocator >`, [18](#)

`get_allocator`
 `std::Vector< T, Allocator >`, [18](#)

`insert`
 `std::Vector< T, Allocator >`, [18](#), [19](#)

`iterator`
 `std::Vector< T, Allocator >`, [12](#)

`max_size`
 `std::Vector< T, Allocator >`, [19](#)

`operator=`
 `std::Vector< T, Allocator >`, [20](#)

`operator[]`
 `std::Vector< T, Allocator >`, [20](#)

`pointer`
 `std::Vector< T, Allocator >`, [12](#)

`pop_back`
 `std::Vector< T, Allocator >`, [20](#)

`push_back`
 `std::Vector< T, Allocator >`, [21](#)

`rbegin`
 `std::Vector< T, Allocator >`, [21](#)

`reference`
 `std::Vector< T, Allocator >`, [12](#)

`rend`
 `std::Vector< T, Allocator >`, [21](#)

`reserve`
 `std::Vector< T, Allocator >`, [22](#)

`resize`
 `std::Vector< T, Allocator >`, [22](#)

`reverse_iterator`
 `std::Vector< T, Allocator >`, [12](#)

`shrink_to_fit`
 `std::Vector< T, Allocator >`, [22](#)

`size`
 `std::Vector< T, Allocator >`, [22](#)

`size_type`
 `std::Vector< T, Allocator >`, [12](#)

`std`, [7](#)
 `std::Vector< T, Allocator >`, [9](#)
 `~Vector`, [13](#)

- allocator_type, 11
- assign, 14
- at, 14, 15
- back, 15
- begin, 15
- capacity, 15
- cbegin, 16
- cend, 16
- clear, 16
- const_iterator, 11
- const_pointer, 11
- const_reference, 11
- const_reverse_iterator, 11
- crbegin, 16
- crend, 16
- data, 16, 17
- difference_type, 11
- emplace, 17
- emplace_back, 17
- empty, 17
- end, 17
- erase, 18
- front, 18
- get_allocator, 18
- insert, 18, 19
- iterator, 12
- max_size, 19
- operator=, 20
- operator[], 20
- pointer, 12
- pop_back, 20
- push_back, 21
- rbegin, 21
- reference, 12
- rend, 21
- reserve, 22
- resize, 22
- reverse_iterator, 12
- shrink_to_fit, 22
- size, 22
- size_type, 12
- swap, 22
- value_type, 12
- Vector, 13, 14

swap

- std::Vector< T, Allocator >, 22

Testai.cpp, 23

- CATCH_CONFIG_MAIN, 23

value_type

- std::Vector< T, Allocator >, 12

Vector

- std::Vector< T, Allocator >, 13, 14

Vector.h, 24