C++ Library - <vector>

Introduction

Vectors are sequence container that can change size. Container is a objects that hold data of same type. Sequence containers store elements strictly in linear sequence.
Vector stores elements in contiguous memory locations and enables direct access to any element using subscript operator []. Unlike array, vector can shrink or expand as needed at run time.
The storage of the vector is handled automatically.
To support shrink and expand functionality at runtime, vector container may allocate some extra storage to accommodate for possible growth thus container have actual capacity greater than the size. Therefore, compared to array, vector consumes more memory in exchange for the ability to manage storage and grow dynamically in an efficient way.
Zero sized vectors are also valid. In that case vector.begin() and vector.end() points to same location. But behavior of calling front() or back() is undefined.

Definition

Below is definition of std::vector from <vector> header file

```
template < class T, class Alloc = allocator<T> > class vector;
```

Parameters

- **T** – Type of the element contained. T may be substituted by any other data type including user-defined type.
- **Alloc** – Type of allocator object. By default, the allocator class template is used, which defines the simplest memory allocation model and is value-independent.

Member types

Following member types can be used as parameters or return type by member functions.

S.N.	Member types	Definition
1	value_type	T (First parameter of the template)
2	allocator_type	Alloc (Second parameter of the template)
3	reference	value_type&
4	const_reference	const value_type&
5	pointer	value_type*
6	const_pointer	const value_type*
7	iterator	a random access iterator to value_type
8	const_iterator	a random access iterator to const value_type
9	reverse_iterator	std::reverse_iterator <iterator>
10	const_reverse_iterator	std::reverse_iterator <const_iterator>
11	size_type	size_t
12	difference_type	ptrdiff_t

Functions from <vector>

Below is list of all methods from <vector> header.

Constructors

S.N.	Method & Description
1	**vector::vector** `default constructor` Constructs an empty container, with zero elements.
2	**vector::vector** `fill constructor` Constructs a container with `n` elements and assignd `val` to each element.
3	**vector::vector** `range constructor` Constructs a container with as many elements in range of `first` to `last`.
4	**vector::vector** `copy constructor` Constructs a container with copy of each elements present in existing container `x`.
5	**vector::vector** `move constructor` Constructs the container with the contents of other using `move` semantics.
6	**vector::vector** `initializer list constructor` Constructs a container from initializer list.

Destructor

S.N.	Method & Description
1	**vector::~vector** Destroys container by deallocating container memory.

Member functions

S.N.	Method & Description
1	**vector::assign** `fill version` Assign new values to the vector elements by replacing old ones.
2	**vector::assign** `range version` Assign new values to the vector elements by replacing old ones.
3	**vector::assign** `initializer list version` Assign new values to the vector elements by replacing old ones.
4	**vector::at** Returns reference to the element present at location `n` in the vector.
5	**vector::back** Returns a reference to the last element of the vector.
6	**vector::begin** Return a random access iterator pointing to the first element of the vector.
7	**vector::capacity** Returns the size of allocate storage, expressed in terms of elements.
8	**vector::cbegin** Returns a constant random access iterator which points to the beginning of the vector.
9	**vector::cend** Returns a constant random access iterator which points to the beginning of the vector.
10	**vector::clear** Destroys the vector by removing all elements from the vector and sets size of vector to zero.
11	**vector::crbegin** Returns a constant reverse iterator which points to the reverser beginning of the container.
12	**vector::crend** Returns a constant reverse iterator which points to the reverse end of the vector.
13	**vector::data** Returns a pointer to the first element of the vector container.
14	**vector::emplace** Extends container by inserting new element at `position`.
15	**vector::emplace_back** Inserts new element at the end of vector.
16	**vector::empty** Tests whether vector is empty or not.
17	**vector::end** Returns an iterator which points to `past-the-end element` in the vector container.
18	**vector::erase** `position version` Removes single element from the the vector.
19	**vector::erase** `range version` Removes single element from the the vector.
20	**vector::front** Returns a reference to the first element of the vector.
21	**vector::get_allocator** Returns an allocator associated with vector.
22	**vector::insert** `single element version` Extends iterator by inserting new element at `position`.
23	**vector::insert** `fill version` Extends vector by inserting new element in the container.
24	**vector::insert** `range version` Extends vector by inserting new element in the container.
25	**vector::insert** `move version` Extends vector by inserting new element in the container.
26	**vector::insert** `initializer list version` Extends vector by inserting new element in the container.

27	**vector::max_size** Returns the maximum number of elements can be held by vector.
28	**vector::operator=** `copy version` Assign new contents to the vector by replacing old ones and modifies size if necessary.
29	**vector::operator=** `move version` Assign new contents to the vector by replacing old ones and modifies size if necessary.
30	**vector::operator =** `initializer list version` Assign new contents to the vector by replacing old ones and modifies size if necessary.
31	**vector::operator[]** Returns a reference to the element present at location `n`.
32	**vector::pop_back** Removes last element from vector and reduces size of vector by one.
33	**vector::push_back** Inserts new element at the end of vector and increases size of vector by one.
34	**vector::rbegin** Returns a reverse iterator which points to the last element of the vector.
35	**vector::rend** Returns a reverse iterator which points to the reverse end of the vector.
36	**vector::reserve** Requests to reserve vector capacity be at least enough to contain `n` elements.
37	**vector::resize** Changes the size of vector.
38	**vector::shrink_to_fit** Requests the container to reduce it's capacity to fit its size.
39	**vector::size** Returns the number of elements present in the vector.
40	**vector::swap** Exchanges the content of vector with contents of vector `x`.

Non-member overloaded functions

S.N.	Method & Description
1	**operator ==** Tests whether two vectors are equal or not.
2	**operator !=** Tests whether two vectors are equal or not.
3	**operator <** Tests whether first vector is less than other or not.
4	**operator <=** Tests whether first vector is less than or equal to other or not.
5	**operator >** Tests whether first vector is greater than other or not.
6	**operator >=** Tests whether first vector is greater than or equal to other or not.
7	**swap** Exchanges the contents of two vector.