class template

std::vector

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

Vector

Vectors are sequence containers representing arrays that can change in size.

Just like arrays, vectors use contiguous storage locations for their elements, which means that their elements can also be accessed using offsets on regular pointers to its elements, and just as efficiently as in arrays. But unlike arrays, their size can change dynamically, with their storage being handled automatically by the container.

Internally, vectors use a dynamically allocated array to store their elements. This array may need to be reallocated in order to grow in size when new elements are inserted, which implies allocating a new array and moving all elements to it. This is a relatively expensive task in terms of processing time, and thus, vectors do not reallocate each time an element is added to the container.

Instead, vector containers may allocate some extra storage to accommodate for possible growth, and thus the container may have an actual capacity greater than the storage strictly needed to contain its elements (i.e., its size). Libraries can implement different strategies for growth to balance between memory usage and reallocations, but in any case, reallocations should only happen at logarithmically growing intervals of size so that the insertion of individual elements at the end of the vector can be provided with *amortized constant time* complexity (see push_back).

Therefore, compared to arrays, vectors consume more memory in exchange for the ability to manage storage and grow dynamically in an efficient way.

Compared to the other dynamic sequence containers (deques, lists and forward_lists), vectors are very efficient accessing its elements (just like arrays) and relatively efficient adding or removing elements from its end. For operations that involve inserting or removing elements at positions other than the end, they perform worse than the others, and have less consistent iterators and references than lists and forward_lists.

Container properties

Sequence
: Elements in sequence containers are ordered in a strict linear sequence. Individual elements are accessed by their position in this sequence. [zero-based indexing]

Dynamic array
: Allows direct access to any element in the sequence, even through pointer arithmetic, and provides relatively fast addition/removal of elements at the end of the sequence.

Allocator-aware
: The container uses an allocator object to dynamically handle its storage needs.

Member functions

(constructor)	
	Construct vector (public member function)
(destructor)	
	Vector destructor (public member function)
operator=	
	Assign content (public member function)

Iterators:	
begin	
	Return iterator to beginning (public member function)
end	
	Return iterator to end (public member function)
rbegin	
	Return reverse iterator to reverse beginning (public member function)
rend	
	Return reverse iterator to reverse end (public member function)
cbegin	
	Return const_iterator to beginning (public member function)
cend	
	Return const_iterator to end (public member function)
crbegin	
	Return const_reverse_iterator to reverse beginning (public member function)
crend	
	Return const_reverse_iterator to reverse end (public member function)

Capacity:	
size	
	Return size (public member function)
max_size	
	Return maximum size (public member function)
resize	
	Change size (public member function)
capacity	
	Return size of allocated storage capacity (public member function)
empty	
	Test whether vector is empty (public member function)
reserve	
	Request a change in capacity (public member function)
shrink_to_fit	
	Shrink to fit (public member function)

Element access:	
operator[]	
	Access element (public member function)
at	
	Access element (public member function) returns a reference
front	
	Access first element (public member function) returns a reference
back	
	Access last element (public member function) returns a reference
data	
	Access data (public member function)

Modifiers:
assign
Assign vector content (public member function)
push_back
Add element at the end (public member function)
pop_back
Delete last element (public member function)
insert
Insert elements (public member function)
erase
Erase elements (public member function)
swap
Swap content (public member function)
clear
Clear content (public member function)
emplace
Construct and insert element (public member function)
emplace_back
Construct and insert element at the end (public member function)
Allocator:
get_allocator
Get allocator (public member function)

Non-member functions overloads

relational operators
Relational operators for vector (function)
swap
Exchanges the contents of two vectors (function template)

Template specializations

vector<bool>
Vector of bool (class template specialization)