

some common functions in STL sequence Container. Give me in a table way

ChatGPT said:

ChatGPT

Here is a table summarizing the common functions available in **STL sequence containers** (like `vector`, `deque`, `list`, `array`, and `forward_list`):

Function	Description	Available Containers
<code>size()</code>	Returns the number of elements in the container.	All sequence containers
<code>empty()</code>	Checks if the container is empty (returns <code>true</code> if no elements).	All sequence containers
<code>clear()</code>	Removes all elements from the container.	All sequence containers
<code>front()</code>	Returns a reference to the first element of the container.	All sequence containers
<code>back()</code>	Returns a reference to the last element of the container.	All sequence containers
<code>push_back()</code>	Adds an element to the end of the container.	<code>vector</code> , <code>deque</code> , <code>list</code>
<code>push_front()</code>	Adds an element to the front of the container.	<code>deque</code> , <code>list</code> , <code>forward_list</code>
<code>pop_back()</code>	Removes the last element of the container.	<code>vector</code> , <code>deque</code> , <code>list</code>
<code>pop_front()</code>	Removes the first element of the container.	<code>deque</code> , <code>list</code> , <code>forward_list</code>
<code>insert()</code>	Inserts an element at a specific position in the container.	All sequence containers
<code>erase()</code>	Removes an element or a range of elements from the container.	All sequence containers
<code>emplace()</code>	Constructs and inserts an element at a specific position.	All sequence containers
<code>emplace_back()</code>	Constructs and inserts an element at the back of the container.	<code>vector</code> , <code>deque</code> , <code>list</code>
<code>emplace_front()</code>	Constructs and inserts an element at the front of the container.	<code>deque</code> , <code>list</code> , <code>forward_list</code>
<code>resize()</code>	Resizes the container to a specific size (may increase or decrease the size).	<code>vector</code> , <code>deque</code> , <code>list</code>
<code>swap()</code>	Swaps the contents of two containers of the same type.	All sequence containers

Function	Description	Available Containers
<code>operator[]</code>	Provides direct access to an element at a specific index (no bounds checking).	vector, deque, array
<code>at()</code>	Provides access to an element with bounds checking (throws <code>std::out_of_range</code> if index is invalid).	vector, deque, array
<code>begin()</code>	Returns an iterator to the first element in the container.	All sequence containers
<code>end()</code>	Returns an iterator to the position after the last element.	All sequence containers
<code>rbegin()</code>	Returns a reverse iterator to the last element.	vector, deque, list
<code>rend()</code>	Returns a reverse iterator to the position before the first element.	vector, deque, list
<code>cbegin()</code>	Returns a constant iterator to the first element.	All sequence containers
<code>cend()</code>	Returns a constant iterator to the position after the last element.	All sequence containers
<code>crbegin()</code>	Returns a constant reverse iterator to the last element.	vector, deque, list
<code>crend()</code>	Returns a constant reverse iterator to the position before the first element.	vector, deque, list

---

## Key Notes:

- `operator[]` and `at()` provide direct access to elements, but `at()` has bounds checking while `operator[]` does not.
- `resize()` and `swap()` are useful for modifying the size and exchanging contents of containers.
- `insert()`, `erase()`, `emplace()`, and `emplace_back()` are versatile functions that allow you to add, remove, and construct elements in-place at various positions within the container.
- `push_back()` and `pop_back()` are commonly used to manipulate the end of sequence containers.