C++ STL: Vectors - Overview and Detailed Explanation

Optimized Overview of Vectors in C++ STL

- Vectors: Dynamic arrays that can resize themselves automatically when elements are added or removed.
- Key Features:
  - Dynamic resizing.
  - Random access via index.
  - Efficient insertions/deletions at the end.
- Common Functions:
  - Accessing elements: at(), operator[].
  - Modifying elements: push_back(), pop_back(), insert(), erase().
  - Capacity management: size(), capacity(), reserve(), resize().
  - Iterators: begin(), end(), rbegin(), rend().

Detailed Explanation of Vectors in C++ STL

---

1. Definition and Characteristics

- Dynamic Array: A vector in C++ is a sequence container that encapsulates dynamic size arrays. Unlike static arrays, vectors can grow or shrink in size.

- Contiguous Memory: Elements are stored in contiguous memory, allowing direct access to elements via pointers or array-style indexing.

2. Declaration and Initialization

- Syntax:

```
std::vector<int> vec;                // Empty vector
std::vector<int> vec(10);            // Vector with 10 default-initialized elements
std::vector<int> vec(10, 5);         // Vector with 10 elements, each initialized to 5
std::vector<int> vec2(vec);          // Copy constructor
std::vector<int> vec = {1, 2, 3, 4};  // Initializer list
```

3. Accessing Elements

- operator[] and at():
  - operator[]: Provides direct access without bounds checking.
  - at(): Similar to operator[] but with bounds checking, throws an out_of_range exception if the index is invalid.

```
vec[0] = 10;
int val = vec.at(1);
```

4. Modifying Elements

- push_back(): Adds an element to the end of the vector.

```
vec.push_back(5);
```

- pop_back(): Removes the last element of the vector.

```
vec.pop_back();
```

- insert(): Inserts elements at a specified position.

```
vec.insert(vec.begin() + 2, 10);   // Insert 10 at position 2
```

- erase(): Removes elements from the vector.

```
vec.erase(vec.begin() + 1);        // Remove the element at position 1
vec.erase(vec.begin(), vec.begin() + 3);  // Remove a range of elements
```

5. Capacity Functions

- size(): Returns the number of elements currently in the vector.

  std::cout << vec.size();

- capacity(): Returns the size of the allocated storage, which is at least equal to the vector size.

  std::cout << vec.capacity();

- reserve(): Requests a change in capacity to at least the specified value.

  vec.reserve(100);

- resize(): Resizes the vector to contain a specified number of elements.

  vec.resize(5); // Shrink or expand the vector to size 5

  vec.resize(8, 2); // Resize to 8, filling new elements with 2

6. Iterators

- begin() and end(): Returns iterators to the beginning and end of the vector.

  for (auto it = vec.begin(); it != vec.end(); ++it) {

      std::cout << *it << " ";

  }

- rbegin() and rend(): Returns reverse iterators, allowing traversal from the end to the beginning.

  for (auto it = vec.rbegin(); it != vec.rend(); ++it) {

      std::cout << *it << " ";

  }

7. Additional Functions

- clear(): Removes all elements from the vector.

  vec.clear();

- empty(): Checks whether the vector is empty.

  if (vec.empty()) {

```
    std::cout << "Vector is empty";
}
```

- shrink_to_fit(): Reduces the capacity to fit the size.

```
vec.shrink_to_fit();
```

8. Complexity Considerations

- Accessing Elements: Constant time O(1).
- Insertion/Deletion at the End: Amortized constant time O(1).
- Insertion/Deletion at Arbitrary Positions: Linear time O(n).

List of All Functions in std::vector

---

1. Element Access

- operator[]: Access element at a specific index (no bounds checking).
- at(size_type pos): Access element with bounds checking.
- front(): Access the first element.
- back(): Access the last element.
- data(): Access the underlying array.

2. Iterators

- begin(): Returns an iterator to the first element.
- end(): Returns an iterator to the element following the last element.
- rbegin(): Returns a reverse iterator to the last element.
- rend(): Returns a reverse iterator to the element preceding the first element.
- cbegin(): Returns a constant iterator to the first element.
- cend(): Returns a constant iterator to the element following the last element.

- crbegin(): Returns a constant reverse iterator to the last element.
- crend(): Returns a constant reverse iterator to the element preceding the first element.

3. Capacity

- size(): Returns the number of elements in the vector.
- max_size(): Returns the maximum number of elements that the vector can hold.
- resize(size_type count): Resizes the container to contain count elements.
- resize(size_type count, const T& value): Resizes and fills new elements with value.
- capacity(): Returns the number of elements that can be held in currently allocated storage.
- empty(): Checks whether the container is empty.
- reserve(size_type new_cap): Requests the vector capacity be at least enough to contain new_cap elements.
- shrink_to_fit(): Reduces the capacity to fit the size.

4. Modifiers

- clear(): Clears the contents of the vector.
- insert(const_iterator pos, const T& value): Inserts an element at pos.
- insert(const_iterator pos, T&& value): Inserts an element (move).
- insert(const_iterator pos, size_type count, const T& value): Inserts count copies of value.
- insert(const_iterator pos, InputIt first, InputIt last): Inserts a range [first, last).
- insert(const_iterator pos, std::initializer_list<T> ilist): Inserts elements from an initializer list.
- emplace(const_iterator pos, Args&&... args): Constructs an element in-place at pos.
- erase(const_iterator pos): Erases the element at pos.
- erase(const_iterator first, const_iterator last): Erases the elements in the range [first, last).
- push_back(const T& value): Adds an element to the end of the vector.
- push_back(T&& value): Adds an element (move).

- emplace_back(Args&&... args): Constructs an element in-place at the end.
- pop_back(): Removes the last element.
- swap(vector& other): Swaps the contents with another vector.
- assign(size_type count, const T& value): Assigns new contents with count copies of value.
- assign(InputIt first, InputIt last): Assigns new contents from a range [first, last).
- assign(std::initializer_list<T> ilist): Assigns new contents from an initializer list.

5. Allocator

- get_allocator(): Returns the allocator associated with the vector.

6. Comparison Operators

- operator==: Checks if two vectors are equal.
- operator!=: Checks if two vectors are not equal.
- operator<: Checks if one vector is less than another.
- operator<=: Checks if one vector is less than or equal to another.
- operator>: Checks if one vector is greater than another.
- operator>=: Checks if one vector is greater than or equal to another.

7. Non-Member Functions

- swap(vector<T,Allocator>& lhs, vector<T,Allocator>& rhs): Swaps the contents of two vectors.
- std::erase(vector<T,Allocator>& c, const T& value): Removes all elements equal to value.
- std::erase_if(vector<T,Allocator>& c, Pred pred): Removes all elements satisfying the predicate pred.