📘 Understanding `std::vector` in C++

`std::vector` is a **sequence container** in the C++ Standard Template Library (STL) that provides **dynamic array** functionality. It manages memory automatically and allows random access to elements.

> The elements in the vector are stored **contiguously**, meaning that the elements are ordered in memory right next to each other.

🔹 1. Indexing

Vectors support **random access** via the subscript operator `[]` and the `.at()` method.

```
std::vector<int> v = {10, 20, 30};
int x = v[1];    // x = 20
int y = v.at(2);    // y = 30
```

- `v[i]` is fast but **not bounds-checked**.
- `v.at(i)` is **bounds-checked** and throws `std::out_of_range` if `i` is invalid.

🔹 2. Adding Items

`push_back()`

Adds an element to the **end** of the vector.

```
v.push_back(40);  // v = {10, 20, 30, 40}
```

`insert()`

Inserts an element at a **specific position**.

```
v.insert(v.begin() + 1, 15);  // v = {10, 15, 20, 30, 40}
```

🔹 3. Erasing Items

`pop_back()`

Removes the **last** element.

```
v.pop_back();  // v = {10, 15, 20, 30}
```

`erase()`

Removes an element or a range.

```
v.erase(v.begin() + 1);  // removes 15 → v = {10, 20, 30}
v.erase(v.begin(), v.begin() + 2);  // removes 10 and 20 → v = {30}
```

🔹 4. Iterating Over a Vector

for loops

```
for(int i=0;i<vec.size();i++)
{
   std::cout << vec[i] << "\n"; //or use vec.at(i)
}
```

Using Iterators

```
for (std::vector<int>::iterator it = v.begin(); it != v.end(); ++it) {
    std::cout << *it << " ";
}
```

Range-Based For Loop

```
for (int val : v) {
    std::cout << val << " ";
}
```

`auto`

```
for (auto it = v.begin(); it != v.end(); ++it) {
    std::cout << *it << " ";
}
```

🔹 5. Size

`size()`

Returns the **number of elements** in the vector.

```
std::cout << v.size();  // e.g., 3
```

🔹 6. Capacity

`capacity()`

Returns the **allocated storage** length (how big the internal array is). size() is ALWAYS <= capacity().

Think of it like the fuel tank in your car. capacity() is how big the fuel tank is. size() is how much gas is in the fuel tank.

```
std::cout << v.capacity();
```

`reserve(n)`

Pre-allocates memory for at least `n` elements to avoid frequent reallocations.

```
v.reserve(100);  // improves performance if you know the size in advance
```

`shrink_to_fit()`

Requests the container to reduce capacity to fit its size.

🔹 7. Copying

There are three ways to copy one vector to another.

Copy each element manually

```
std::vector<int> scores = { 1,2,3,4,5 };
std::vector<int> scores2;
scores2.reserve(scores.size()); // reserve the space needed
for (size_t i = 0; i < scores.size(); i++)
    scores2.push_back(scores[i]);
```

use the assignment `=` operator

```
std::vector<int> scores = { 1,2,3,4,5 };
std::vector<int> scores3 = scores;
```

passing a vector to the constructor of the new vector

```
std::vector<int> scores = { 1,2,3,4,5 };
std::vector<int> scores4(scores);
```

✅ Summary

Operation	Method	Notes
Indexing	`[]`, `.at()`	`.at()` is bounds-checked
Add	`push_back()`, `insert()`	Adds at end or specific position
Remove	`pop_back()`, `erase()`	Removes last or specific elements
Iterate	`iterator`, `range-for`	Multiple styles supported
Size	`size()`	Number of elements
Capacity	`capacity()`, `reserve()`	Memory management tools

🎯 Quiz!

Here's a short quiz on the topic: quiz

🔭 Markdown Viewer

How to view the markdown files in a browser...

- Markdown Viewer

🧠 Lecture Practices

Here are the lecture Practices...

- Day 1
- Day 2
- Day 3

🔍 Lecture Quizzes

Here are the lecture quizzes...

- Day 1
- Day 2
- Day 3

⚙️ Weekly Topics

Here are the topics for the week...

- Calling Methods
- Calling Methods 2
- Creating Methods
- Iterators
- Vectors
- References
- Const
- Erasing in a Loop
- Default Parameters