📘 C++ Iterators and `std::vector`

🔹 What is an Iterator?

An **iterator** in C++ is an object that acts like a pointer and allows you to **traverse** (i.e. move through) containers (like `std::vector`, `std::map`, etc.). Think of it as a **generalized pointer** that provides access to elements in a container without exposing the underlying structure.

🔹 Iterators and `std::vector`

`std::vector` is a **sequence container** that stores elements in a **contiguous memory block**. It supports **random access iterators**, which means you can move forward, backward, and jump to any position in constant time.

Basic Syntax:

```
std::vector<int>::iterator it;
```

🔹 Common Iterator Operations

Operation	Description
`begin()`	Returns iterator to the first element
`end()`	Returns iterator to one past the last element
`*it`	Dereferences iterator to access value
`it++` / `++it`	Moves to the next element
`it--` / `--it`	Moves to the previous element
`it + n`	Moves forward by `n` positions
`it - n`	Moves backward by `n` positions

🔹 Example: Iterating Over a Vector

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> nums = {10, 20, 30, 40, 50};

    // Using iterator
    for (std::vector<int>::iterator it = nums.begin(); it != nums.end(); ++it) {
        std::cout << *it << " ";
    }
    // Output: 10 20 30 40 50
}
```

🔹 Const Iterators

Use `const_iterator` when you want to **read** but not **modify** the elements.

```
std::vector<int>::const_iterator it;
```

🔹 Reverse Iterators

```
for (std::vector<int>::reverse_iterator rit = nums.rbegin(); rit != nums.rend(); ++rit) {
    std::cout << *rit << " ";
}
// Output: 50 40 30 20 10
```

🔹 Why Use Iterators?

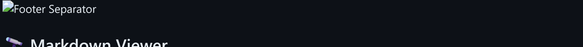
- **Abstraction**: You don't need to know how the container is implemented.
- **Generic Programming**: Algorithms like `std::sort`, `std::find`, etc., work with iterators.
- **Safety**: Iterators provide bounds-checked access when used with STL algorithms.

✅ Summary

- Iterators are like smart pointers for containers.
- `std::vector` supports **random access iterators**.
- Use `begin()` and `end()` to traverse a vector.
- Prefer `const_iterator` for read-only access.
- STL algorithms rely heavily on iterators.

🎯 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