Tutorial 71 - Vector in C++ STL

Introduction

- **Vectors** are **dynamic arrays** in C++ STL.
- Unlike **arrays**, vectors do **not require a predefined size**.
- To use vectors, include the `<vector>` header file.

📌 Syntax of Declaring a Vector

```
vector<data_type> vector_name;

```

✅ Example: Declaring a vector of integers

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> vec1;
    return 0;
}

```

📌 Advantages of Using Vectors

✔ **Dynamic in size** – No need to specify a fixed size as in arrays.
✔ **Provides built-in methods** for inserting, deleting, and accessing elements efficiently.
✔ **Can be easily copied and assigned to other vectors.**

📌 Vector Operations & Methods

1️⃣ `push_back()` - Adding Elements

- `push_back(value)` inserts an element **at the end** of the vector.
- **Example: Taking user input and adding elements dynamically**

```
#include <iostream>
#include <vector>
using namespace std;

void display(vector<int> &v) {
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
}

int main() {
    vector<int> vec1;
    int element, size;
    cout << "Enter the size of your vector: ";
    cin >> size;

    for (int i = 0; i < size; i++) {
        cout << "Enter an element to add to this vector: ";
        cin >> element;
        vec1.push_back(element);
    }

    display(vec1);
    return 0;
}

```

✅ **Output:**

```
Enter the size of your vector
3
Enter an element to add to this vector: 5
Enter an element to add to this vector: 3
Enter an element to add to this vector: 7
5 3 7

```

2️⃣ `pop_back()` - Removing the Last Element

- **Removes the last element** from the vector.

```
display(vec1);
vec1.pop_back(); // Removes the last element
display(vec1);

```

✅ **Output:**

```
5 3 7
5 3

```

3️⃣ `insert(iterator, value)` - Inserting at a Specific Position

- **Inserts an element at the position pointed by the iterator.**
- **Syntax:**

```
vector<int>::iterator iter = vec1.begin();
vec1.insert(iter, 566); // Inserts 566 at the beginning

```

✅ **Example:**

```
display(vec1);
vector<int>::iterator iter = vec1.begin();
vec1.insert(iter, 566);
display(vec1);

```

✅ **Output:**

```
5 3 7
566 5 3 7

```

4️⃣ `v.at(i)` - Accessing Elements

- Works **similar to** `v[i]`, but provides **bounds checking**.

```
cout << vec1.at(1); // Safer way to access elements

```

📌 Different Ways to Declare a Vector

```
vector<int> vec1;       // Empty vector
vector<char> vec2(4);   // Vector of size 4 (default initialized)
vector<char> vec3(vec2);// Copy vector vec2
vector<int> vec4(6,3);  // Vector of size 6, all elements initialized to 3

```

📌 Summary of Vector Methods

Method	Description
`push_back(x)`	Adds an element `x` at the end
`pop_back()`	Removes the last element
`insert(iter, x)`	Inserts `x` at position `iter`
`size()`	Returns the number of elements
`at(i)`	Accesses the element at index `i`
`begin()`	Returns an iterator to the first element
`end()`	Returns an iterator to the last element
`clear()`	Removes all elements
`empty()`	Checks if the vector is empty

Key Takeaways

✔ **Vectors are dynamic arrays in C++ STL**.
✔ **They provide built-in functions for efficient manipulation**.
✔ **Common operations include** `push_back()`**,** `pop_back()`**,** `insert()`**,** `size()`.
✔ **Use** `v.at(i)` **instead of** `v[i]` **for safer access**.
✔ **Multiple ways to declare and initialize vectors**.

🚀 **Next Topic:** Lists in C++ STL! Stay tuned! 🔥

Short Notes

📌 What is a Vector in C++?

- A **dynamic array** that can **grow and shrink** automatically.
- Requires `#include <vector>`.

📌 Vector Declaration

```
vector<int> vec1;       // Empty vector
vector<int> vec2(5, 0); // Vector of size 5, initialized with 0

```

📌 Common Vector Methods

Method	Function
`push_back(x)`	Adds `x` to the end
`pop_back()`	Removes the last element
`insert(iter, x)`	Inserts `x` at position `iter`
`size()`	Returns the number of elements
`at(i)`	Accesses element at index `i` safely

🚀 **Next Topic:** Lists in C++ STL! Keep Learning! 🔥