◆ STL (Standard Template Library) in C++

STL (Standard Template Library) in C++ is a powerful collection of template-based classes and functions that provide commonly used data structures (like vector, list, stack, map) and algorithms (like sort, search, count). It helps developers write efficient, reusable, and faster code with less effort.

Containers – Store data

- `vector` – Dynamic array
- `list` – Doubly linked list
- `deque` – Double-ended queue
- `stack` – LIFO (usually built over `deque`)
- `queue` – FIFO
- `priority_queue` – Max/Min heap
- `set` / `unordered_set` – Store unique elements
- `map` / `unordered_map` – Key-value pairs (like dictionaries)

□ STL Full Example in C++

```
#include <iostream>
#include <vector>
#include <list>
#include <deque>
#include <stack>
#include <queue>
#include <set>
#include <map>
#include <unordered_set>
#include <unordered_map>
#include <algorithm>
#include <numeric> // for accumulate

using namespace std;

int main() {
    cout << "=== VECTOR ===" << endl;
    vector<int> vec = {5, 1, 4, 2, 3};
    sort(vec.begin(), vec.end());
    for (int val : vec) cout << val << " ";
    cout << "\nSum: " << accumulate(vec.begin(), vec.end(), 0) << "\n\n";

    cout << "=== LIST ===" << endl;
    list<string> fruits = {"Mango", "Apple", "Banana"};
    fruits.push_front("Orange");
    fruits.sort();
    for (const auto &f : fruits) cout << f << " ";
    cout << "\n\n";
```

```
cout << "=== DEQUE ===" << endl;
deque<int> dq;
dq.push_back(10);
dq.push_front(20);
dq.push_back(30);
for (int x : dq) cout << x << " ";
cout << "\n\n";

cout << "=== STACK ===" << endl;
stack<int> st;
st.push(100);
st.push(200);
st.push(300);
while (!st.empty()) {
    cout << st.top() << " ";
    st.pop();
}
cout << "\n\n";

cout << "=== QUEUE ===" << endl;
queue<string> q;
q.push("C++");
q.push("Java");
q.push("Python");
while (!q.empty()) {
    cout << q.front() << " ";
    q.pop();
}
cout << "\n\n";

cout << "=== PRIORITY QUEUE ===" << endl;
priority_queue<int> pq; // Max heap
pq.push(40);
pq.push(10);
pq.push(30);
while (!pq.empty()) {
    cout << pq.top() << " ";
    pq.pop();
}
cout << "\n\n";

cout << "=== SET ===" << endl;
set<int> s = {5, 3, 5, 1, 2};
for (int x : s) cout << x << " ";
cout << "\n\n";

cout << "=== UNORDERED SET ===" << endl;
unordered_set<int> us = {7, 2, 7, 1, 9};
for (int x : us) cout << x << " ";
cout << "\n\n";
```

```
    cout << "=== MAP ===" << endl;
    map<string, int> ages;
    ages["Alice"] = 25;
    ages["Bob"] = 30;
    for (auto pair : ages)
        cout << pair.first << ": " << pair.second << endl;
    cout << "\n";

    cout << "=== UNORDERED MAP ===" << endl;
    unordered_map<string, int> scores = {
        {"Math", 90}, {"Physics", 85}, {"Chemistry", 88}};
    for (auto pair : scores)
        cout << pair.first << ": " << pair.second << endl;
    cout << "\n";

    cout << "=== ALGORITHMS (find, count, binary_search) ===" << endl;
    vector<int> data = {10, 20, 30, 40, 50};
    if (find(data.begin(), data.end(), 30) != data.end())
        cout << "30 Found\n";
    cout << "Count of 20: " << count(data.begin(), data.end(), 20) << "\n";
    sort(data.begin(), data.end()); // binary_search needs sorted container
    cout << "Binary search for 40: "
         << binary_search(data.begin(),data.end(), 40) << "\n\n";

    cout << "=== ITERATORS ===" << endl;
    vector<int>::iterator it;
    for (it = vec.begin(); it != vec.end(); ++it)
        cout << *it << " ";
    cout << "\n";

    return 0;
}
```

Output Preview (Shortened):

```
=== VECTOR ===
1 2 3 4 5
Sum: 15

=== LIST ===
Apple Banana Mango Orange

=== DEQUE ===
20 10 30

=== STACK ===
300 200 100

=== QUEUE ===
C++ Java Python
```

```
=== PRIORITY QUEUE ===
40 30 10

=== SET ===
1 2 3 5

=== UNORDERED SET ===
9 2 1 7

=== MAP ===
Alice: 25
Bob: 30

=== UNORDERED MAP ===
Chemistry: 88
Physics: 85
Math: 90

=== ALGORITHMS (find, count, binary_search) ===
30 Found
Count of 20: 1
Binary search for 40: 1

=== ITERATORS ===
1 2 3 4 5
```