

# C++ STL Containers Summary

## 1. Pairs

- Used to store two related values together.
- Syntax: `std::pair<int, string> p = {1, "Ali"};`
- Access elements using `p.first` and `p.second`.
- Commonly used in maps and other containers.

## 2. Sets

- Stores unique elements in sorted order.
- No duplicate values allowed.
- Syntax: `std::set<int> s;`
- Operations: `insert()`, `erase()`, `count()`, `find()`.
- Time complexity for insert/search:  $O(\log n)$ .

## 3. Multisets

- Allows duplicate elements, keeps sorted order.
- Syntax: `std::multiset<int> ms;`
- Operations: `insert()`, `erase()`, `count()`.

## 4. Unordered Sets

- Stores unique elements with no specific order.
- Faster average time complexity:  $O(1)$ .
- Syntax: `std::unordered_set<int> us;`
- No sorting guarantees.

## 5. Maps

- Stores key-value pairs with unique keys.
- Automatically sorted by key.
- Syntax: `std::map<string, int> m;`

## C++ STL Containers Summary

- Access value: `m["Ali"] = 20;`
- Operations: `insert()`, `erase()`, `count()`, `find()`.
- Time complexity for operations:  $O(\log n)$ .

### 6. Multimaps

- Allows duplicate keys in key-value pairs.
- Syntax: `std::multimap<string, int> mm;`

### 7. Unordered Maps

- Stores key-value pairs with unique keys.
- No specific order; uses hashing.
- Faster average time complexity:  $O(1)$ .
- Syntax: `std::unordered_map<string, int> um;`