C++ Map-like Data Structures: Types, Pros/Cons, and Performance

std::map

Keywords: Balanced Binary Search Tree (Red-Black Tree), ordered, unique keys

Pros:

- Keys kept in sorted order
- Efficient for ordered traversal
- Deterministic performance (O(log n))

Cons:

- Slower than hash maps for random lookups
- Higher memory usage than flat structures

Time Complexity:

Operation	Complexity
Lookup	O(log n)
Insertion	O(log n)
Deletion	O(log n)
Iteration	O(n)

std::multimap

Keywords: Balanced Binary Search Tree, ordered, allows duplicate keys

Pros:

- Keys stored in sorted order
- Supports multiple values per key
- Good for grouped data

Cons:

- Same drawbacks as std::map (slower lookups than hash maps)
- Duplicates may add overhead

Time Complexity:

Operation	Complexity
Lookup	O(log n)
Insertion	O(log n)
Deletion	O(log n)
Iteration	O(n)

std::unordered_map

Keywords: Hash table, unique keys, not ordered

Pros:

- Average constant-time lookup (O(1))
- Faster than std::map for direct key access
- Good for large datasets

Cons:

- No ordering of keys
- Worst-case O(n) if hash collisions occur
- More memory overhead due to buckets

Time Complexity:

Operation	Complexity
Lookup	Average O(1), Worst O(n)
Insertion	Average O(1), Worst O(n)
Deletion	Average O(1), Worst O(n)
Iteration	O(n)

std::unordered_multimap

Keywords: Hash table, allows duplicate keys, not ordered

Pros:

- Allows multiple values per key
- Fast average lookups (O(1))
- Good when duplicates are needed

Cons:

- No key ordering

- More memory overhead
- Worst-case O(n) with many collisions

Time Complexity:

Operation	Complexity
Lookup	Average O(1), Worst O(n)
Insertion	Average O(1), Worst O(n)
Deletion	Average O(1), Worst O(n)
Iteration	O(n)

boost::unordered_flat_map

Keywords: Hybrid: Hash table + flat contiguous storage

Pros:

- Fast lookups (average O(1))
- Better cache locality than std::unordered_map
- Lower memory overhead

Cons:

- Insertions may be slower due to relocations
- Requires Boost library

Time Complexity:

Operation	Complexity
Lookup	Average O(1)
Insertion	Average O(1)
Deletion	Average O(1)
Iteration	O(n), faster than std::unordered_map

boost::flat_map

Keywords: Sorted vector-like container, ordered keys, unique

Pros:

- Very memory efficient
- Fast iteration due to contiguous storage
- Good for small to medium datasets

Cons:

- Insertions/deletions are O(n) (need to shift elements)
- Not suitable for very large or frequently updated datasets

Time Complexity:

Operation	Complexity
Lookup	O(log n)
Insertion	O(n)
Deletion	O(n)
Iteration	O(n), but very fast due to cache locality