C++ Maps Cheat Sheet

1. Declaring a Map

```
#include <map>
std::map<int, int> m1;                  // Empty map with int keys and values
std::map<std::string, int> m2;          // Map with string keys and int values
std::map<int, int> m3 = {{1, 2}, {3, 4}}; // Map initialized with key-value pairs
```

2. Basic Functions

```
m.size();          // Returns the number of elements
m.empty();         // Returns true if the map is empty
m.clear();         // Removes all elements from the map
m.insert({key, value}); // Inserts a key-value pair
m.erase(key);      // Removes element with specified key
m.find(key);       // Returns iterator to element with specified key (or m.end())
```

3. Accessing Elements

```
m[key];            // Access or insert element with 'key'
m.at(key);         // Access element with 'key' (throws an exception if key doesn't exist)
```

4. Iterators

```
m.begin();         // Iterator to the beginning
m.end();           // Iterator to the end (one past the last element)
m.rbegin();        // Reverse iterator to the beginning (last element)
m.rend();          // Reverse iterator to the end (before first element)
```

5. Modifying Elements

```
m.insert({key, value});    // Inserts a key-value pair
m.erase(it);               // Removes element at iterator position 'it'
m.erase(startIt, endIt);   // Removes elements in the range [startIt, endIt)
m.swap(m2);                // Swaps elements with another map m2
std::swap(m1, m2);         // Alternative to swap two maps
```

6. Capacity Functions

```
m.max_size();      // Returns the maximum number of elements
```

7. Searching and Counting

```
m.find(key);                          // Finds an element by key, returns iterator
bool exists = (m.find(key) != m.end()); // Check if key exists
m.count(key);                         // Returns 1 if key exists, 0 otherwise
```

8. Common Operations

```
for (const auto &pair : m) {              // Loop through map with range-based for loop
    std::cout << pair.first << ": " << pair.second << std::endl;
}
auto it = m.find(key);                    // Find element by key
if (it != m.end()) { std::cout << it->second; } // Access value if found
```

9. Looping through a Map

```
// Using a range-based for loop
for (const auto &pair : m) {
    std::cout << pair.first << ": " << pair.second << std::endl;
}
// Using iterators
for (auto it = m.begin(); it != m.end(); ++it) {
    std::cout << it->first << ": " << it->second << std::endl;
}
```