C++ Vectors Cheat Sheet

1. Declaring a Vector

```
#include <vector>
std::vector<int> v1;                 // Empty vector of integers
std::vector<int> v2(5);              // Vector of size 5 with default values (0 for int)
std::vector<int> v3(5, 10);          // Vector of size 5, all elements initialized to 10
std::vector<int> v4 = {1, 2, 3, 4};  // Vector initialized with elements
```

2. Basic Functions

```
v.size();       // Returns the number of elements
v.empty();      // Returns true if the vector is empty
v.clear();      // Removes all elements from the vector
v.push_back(x); // Adds an element 'x' at the end
v.pop_back();   // Removes the last element
```

3. Accessing Elements

```
v[i];           // Access the element at index i (no bounds check)
v.at(i);        // Access the element at index i (with bounds check)
v.front();      // Returns the first element
v.back();       // Returns the last element
v.data();       // Returns a pointer to the underlying array
```

4. Iterators

```
v.begin();      // Iterator to the beginning
```

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

5. Modifying Elements

```
v.insert(v.begin() + i, x);        // Inserts 'x' at position 'i'
v.insert(v.begin() + i, n, x);     // Inserts 'n' copies of 'x' at position 'i'
v.erase(v.begin() + i);            // Removes the element at position 'i'
v.erase(v.begin() + i, v.begin() + j); // Removes elements in the range [i, j)
v.resize(n);                       // Resizes the vector to contain 'n' elements
v.resize(n, x);                    // Resizes and fills new elements with 'x'
v.swap(v2);                        // Swaps elements with another vector v2
std::swap(v1, v2);                 // Alternative to swap two vectors
```

6. Capacity Functions

```
v.capacity();    // Returns the current capacity
v.reserve(n);    // Increases the capacity to at least 'n'
v.shrink_to_fit(); // Reduces capacity to fit size (non-binding request)
```

7. Sorting and Searching

```
#include <algorithm>
std::sort(v.begin(), v.end());             // Sort in ascending order
std::sort(v.rbegin(), v.rend());           // Sort in descending order
auto it = std::find(v.begin(), v.end(), x); // Find an element (returns iterator)
bool exists = (std::find(v.begin(), v.end(), x) != v.end()); // Check existence
std::binary_search(v.begin(), v.end(), x); // Binary search (true if found)
```

8. Common Operations

```
std::vector<int> v = {1, 2, 3, 4, 5};

int sum = std::accumulate(v.begin(), v.end(), 0);   // Sum of all elements

int min = *std::min_element(v.begin(), v.end());    // Minimum element

int max = *std::max_element(v.begin(), v.end());    // Maximum element

int count = std::count(v.begin(), v.end(), x);      // Count occurrences
```

9. Looping through a Vector

```
// Using a for loop with index

for (int i = 0; i < v.size(); i++) { std::cout << v[i] << " "; }

// Using a range-based for loop

for (int x : v) { std::cout << x << " "; }

// Using iterators

for (auto it = v.begin(); it != v.end(); ++it) { std::cout << *it << " "; }
```