#include <iostream>

#include <vector>

#include <algorithm>

int main() {

// 1. Construction

std::vector<int> vec1; // Default constructor

std::vector<int> vec2(10, 5); // Fill constructor (10 elements with value 5)

std::vector<int> vec3{1, 2, 3, 4, 5}; // Initializer list constructor

std::vector<int> vec4(vec3.begin(), vec3.end()); // Range constructor

std::vector<int> vec5(vec3); // Copy constructor

std::vector<int> vec6(std::move(vec5)); // Move constructor

// 2. Assignment

vec1 = vec2; // Copy assignment

vec1 = std::move(vec2); // Move assignment

vec1 = {10, 20, 30}; // Initializer list assignment

// 3. Element Access

std::cout << "Element at index 1: " << vec1[1] << std::endl; // Operator[]

std::cout << "Element at index 2: " << vec1.at(2) << std::endl; // at()

std::cout << "First element: " << vec1.front() << std::endl; // front()

std::cout << "Last element: " << vec1.back() << std::endl; // back()

int\* data = vec1.data(); // data()

std::cout << "Element via data pointer: " << data[0] << std::endl;

// 4. Iterators

std::cout << "Elements in vec1: ";

for (auto it = vec1.begin(); it != vec1.end(); ++it) { // begin() and end()

std::cout << \*it << " ";

}

std::cout << std::endl;

std::cout << "Elements in reverse: ";

for (auto it = vec1.rbegin(); it != vec1.rend(); ++it) { // rbegin() and rend()

std::cout << \*it << " ";

}

std::cout << std::endl;

// 5. Capacity

std::cout << "Size: " << vec1.size() << std::endl; // size()

std::cout << "Capacity: " << vec1.capacity() << std::endl; // capacity()

std::cout << "Is empty: " << vec1.empty() << std::endl; // empty()

vec1.resize(5); // resize()

std::cout << "Resized vec1 size: " << vec1.size() << std::endl;

vec1.reserve(20); // reserve()

std::cout << "Reserved capacity: " << vec1.capacity() << std::endl;

// 6. Modifiers

vec1.assign(7, 100); // assign()

vec1.push\_back(200); // push\_back()

vec1.pop\_back(); // pop\_back()

vec1.insert(vec1.begin() + 1, 300); // insert()

vec1.erase(vec1.begin() + 2); // erase()

vec1.emplace(vec1.begin(), 400); // emplace()

vec1.emplace\_back(500); // emplace\_back()

vec1.swap(vec3); // swap()

vec1.clear(); // clear()

// 7. Non-member Functions

std::cout << "Is vec1 == vec3? " << (vec1 == vec3) << std::endl; // operator==

std::swap(vec1, vec3); // swap()

std::cout << "Elements after swap: ";

for (const auto& elem : vec1) {

std::cout << elem << " ";

}

std::cout << std::endl;

// 8. Algorithms

std::sort(vec1.begin(), vec1.end()); // sort()

std::cout << "Sorted elements: ";

for (const auto& elem : vec1) {

std::cout << elem << " ";

}

std::cout << std::endl;

return 0;

}

Ouput:

