***Kumail Raza***

Code 2:

#include <iostream>

#include <vector>

#include <algorithm>

#include <chrono>

// Bubble Sort implementation

void bubbleSort(std::vector<int>& arr) {

int n = arr.size();

for (int i = 0; i < n - 1; ++i) {

for (int j = 0; j < n - i - 1; ++j) {

if (arr[j] > arr[j + 1]) {

std::swap(arr[j], arr[j + 1]);

}

}

}

}

int main() {

// Initialize a vector of 100,000 integers in descending order

std::vector<int> data(100000);

for (int i = 0; i < 100000; ++i) {

data[i] = 100000 - i;

}

// Measure time taken by Bubble Sort

auto startBubbleSort = std::chrono::high\_resolution\_clock::now();

bubbleSort(data);

auto stopBubbleSort = std::chrono::high\_resolution\_clock::now();

auto durationBubbleSort = std::chrono::duration\_cast<std::chrono::milliseconds>(stopBubbleSort - startBubbleSort);

// Reset the vector to descending order

std::reverse(data.begin(), data.end());

// Measure time taken by std::sort

auto startStdSort = std::chrono::high\_resolution\_clock::now();

std::sort(data.begin(), data.end());

auto stopStdSort = std::chrono::high\_resolution\_clock::now();

auto durationStdSort = std::chrono::duration\_cast<std::chrono::milliseconds>(stopStdSort - startStdSort);

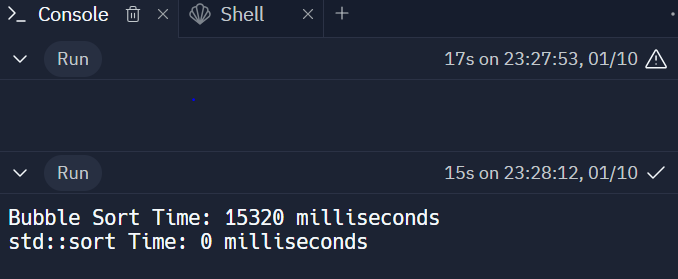
// Output the results

std::cout << "Bubble Sort Time: " << durationBubbleSort.count() << " milliseconds\n";

std::cout << "std::sort Time: " << durationStdSort.count() << " milliseconds\n";

return 0;

}



Code 1:

#include <iostream>

#include <vector>

#include <algorithm> // For std::remove\_if

struct Product {

int id;

std::string name;

double price;

int quantity;

};

class Inventory {

private:

std::vector<Product> products;

public:

// Function to add a new product to the inventory

void addProduct(const Product& newProduct) {

products.push\_back(newProduct);

}

// Function to remove a product based on its ID

void removeProductById(int productId) {

products.erase(std::remove\_if(products.begin(), products.end(),

[productId](const Product& product) { return product.id == productId; }),

products.end());

}

// Function to display the current inventory

void displayInventory() const {

std::cout << "Inventory:\n";

for (const auto& product : products) {

std::cout << "ID: " << product.id << ", Name: " << product.name

<< ", Price: " << product.price << ", Quantity: " << product.quantity << "\n";

}

}

};

int main() {

Inventory inventory;

// Adding products to the inventory

inventory.addProduct({1, "Product A", 20.0, 50});

inventory.addProduct({2, "Product B", 30.0, 30});

inventory.addProduct({3, "Product C", 15.0, 40});

// Displaying the initial inventory

inventory.displayInventory();

// Removing a product by ID

inventory.removeProductById(2);

// Displaying the updated inventory after removal

inventory.displayInventory();

return 0;

}

