**Блочная сортировка массива длиной 25 элементов.**

**Код программы**

#include <iostream>

#include <vector>

#include <ctime> // Для генерации случайных чисел

using namespace std;

// Функция для блочной сортировки

void bucketSort(vector<int>& arr, int n) {

    // Находим максимальное значение в массиве

    int maxVal = arr[0];

    for (int i = 1; i < n; ++i) {

        if (arr[i] > maxVal) {

            maxVal = arr[i];

        }

    }

    // Создаем "карманы" (buckets) для каждого диапазона значений

    vector<vector<int>> buckets(maxVal + 1);

    // Распределяем элементы по "карманам"

    for (int i = 0; i < n; ++i) {

        int bucketIndex = (arr[i] \* n) / (maxVal + 1);

        buckets[bucketIndex].push\_back(arr[i]);

    }

    // Сортируем каждый "карман" отдельно (например, с помощью вставки)

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

        for (int j = 1; j < buckets[i].size(); ++j) {

            int key = buckets[i][j];

            int k = j - 1;

            while (k >= 0 && buckets[i][k] > key) {

                buckets[i][k + 1] = buckets[i][k];

                --k;

            }

            buckets[i][k + 1] = key;

        }

    }

    // Собираем отсортированный массив из "карманов"

    int outputIndex = 0;

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

        for (int j = 0; j < buckets[i].size(); ++j) {

            arr[outputIndex] = buckets[i][j];

            ++outputIndex;

        }

    }

    // Выводим отсортированный массив

    cout << "Отсортированный массив:\n";

    for (int i = 0; i < n; ++i) {

        cout << arr[i] << " ";

    }

}

int main() {

    int n = 25; // Размер массива

    vector<int> arr(n);

    for (int i = 0; i < n; ++i) {

        cout<<"Введите элемент массива: ";

        int new\_element;

        cin>>new\_element;

        arr[i] = new\_element;

    }

    // Выводим исходный массив

    cout << "Исходный массив:\n";

    for (int i = 0; i < n; ++i) {

        cout << arr[i] << " ";

    }

    cout << endl;

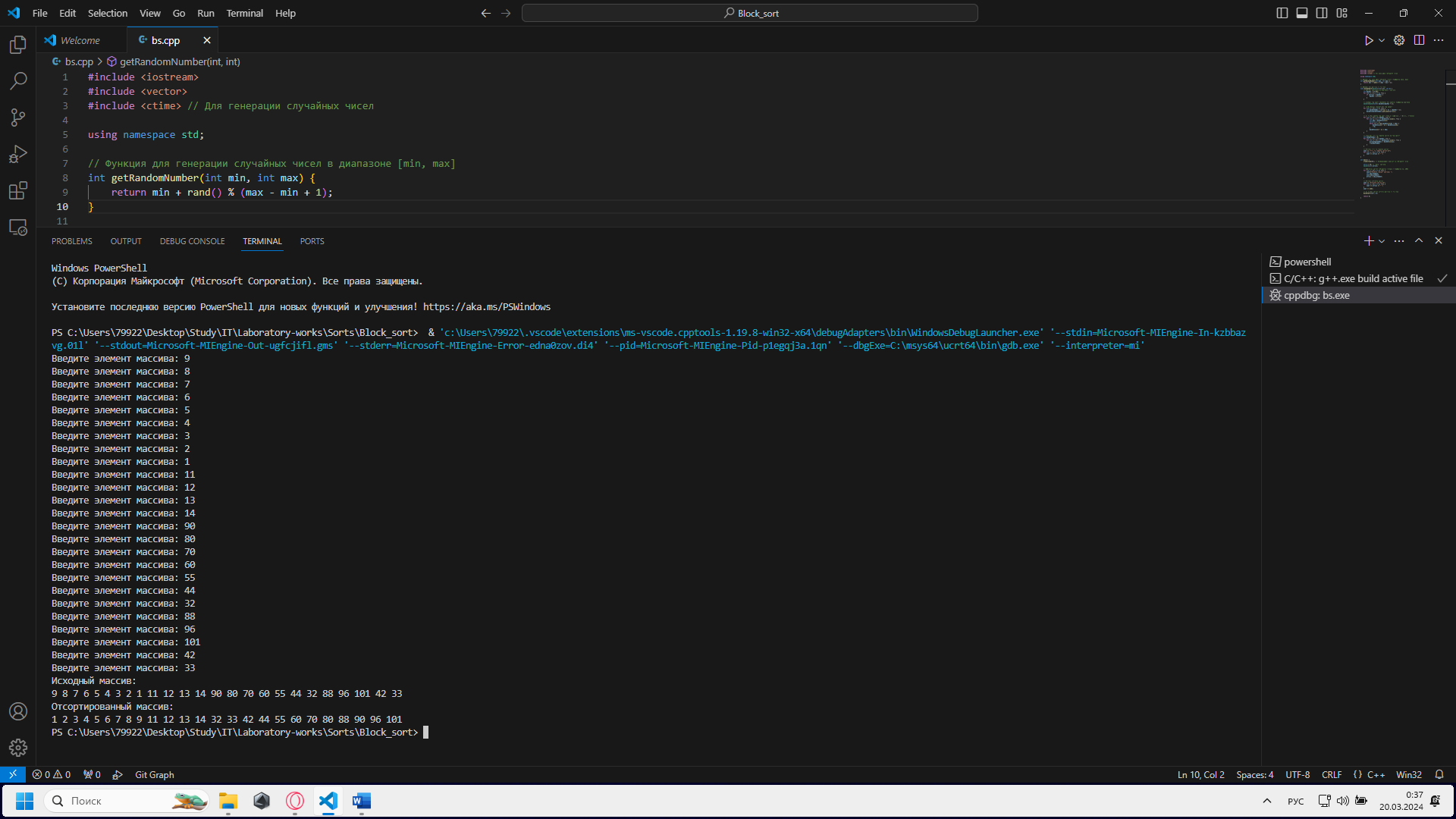
    // Сортируем массив методом блочной сортировки

    bucketSort(arr, n);

    return 0;

}

**Результаты работы**



**Сортировка подсчетом**

#include <iostream>

#include <vector>

using namespace std;

// Функция сортировки подсчетом

void countSort(vector<int>& arr, int n) {

    int maxVal = arr[0];

    for (int i = 1; i < n; ++i) {

        if (arr[i] > maxVal) {

            maxVal = arr[i];

        }

    }

    vector<int> output(n);

    // Создаем массив `count` размера `maxVal + 1`, инициализируем его нулями

    vector<int> count(maxVal + 1, 0);

    // Используя значение каждого элемента входного массива в качестве индекса,

    // сохраняем счетчик каждого целого числа в `count[]`

    for (int i = 0; i < n; ++i) {

        ++count[arr[i]];

    }

    // Копируем в выходной массив, сохраняя порядок входов с одинаковыми ключами

    int outputIndex = 0;

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

        while (count[i] > 0) {

            output[outputIndex] = i;

            --count[i];

            ++outputIndex;

        }

    }

    // Выводим отсортированный массив

    cout << "Отсортированный массив:\n";

    for (int i = 0; i < n; ++i) {

        cout << output[i] << " ";

    }

}

int main() {

    int n = 25; // Размер массива

    vector<int> arr(n);

    for (int i = 0; i < n; ++i) {

        int new\_number;

        cout<<"Введите элемент массива: ";

        cin>>new\_number;

        arr[i] = new\_number;

    }

    // Выводим исходный массив

    cout << "Исходный массив:\n";

    for (int i = 0; i < n; ++i) {

        cout << arr[i] << " ";

    }

    cout << endl;

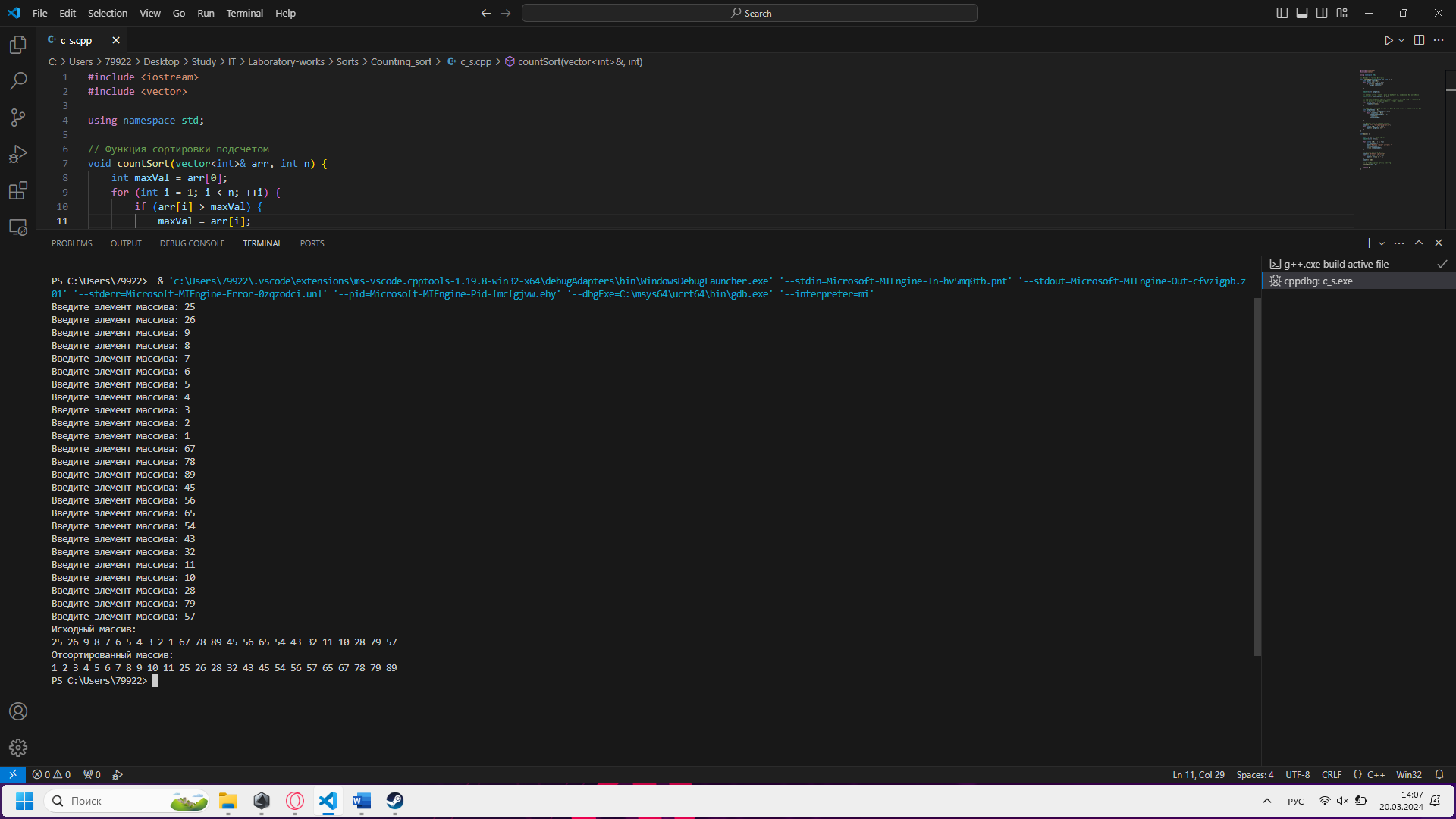
    // Сортируем массив методом подсчета

    countSort(arr, n);

    return 0;

}

**Результаты работы**



**Сортировка слиянием (Merge sort)**

#include <iostream>

using namespace std;

//функция, сливающая массивы

void Merge(int\* A, int first, int last) {

    int middle, start, final, j;

    int \*mas = new int[100];

    middle = (first + last) / 2; //вычисление среднего элемента

    start = first; //начало левой части

    final = middle + 1; //начало правой части

    for (j = first; j <= last; j++) //выполнять от начала до конца

    {

        if ((start <= middle) && ((final > last) || (A[start] < A[final]))) {

            mas[j] = A[start];

            start++;

        }

        else {

            mas[j] = A[final];

            final++;

        }

    }

    //возвращение результата в список

    for (j = first; j <= last; j++) {

        A[j] = mas[j];

    }

    delete[] mas;

}

//рекурсивная процедура сортировки

void MergeSort(int\* A, int first, int last) {

    if (first < last) {

        MergeSort(A, first, (first + last) / 2); //сортировка левой части

        MergeSort(A, (first + last) / 2 + 1, last); //сортировка правой части

        Merge(A, first, last); //слияние двух частей

    }

}

int main() {

    setlocale(LC\_ALL, "RU");

    int n;

    cout << "Введите количество элементов: ";

    cin >> n;

    // Генерация случайных чисел и заполнение массива

    int\* A = new int[n];

    for (int i = 0; i < n; i++) {

        int new\_element;

        cout<<"Введите элемент массива: ";

        cin>>new\_element;

        A[i] = new\_element;

    }

    cout << endl;

    MergeSort(A, 0, n - 1);

    cout << "Сортированный массив: ";

    for (int i = 0; i < n; i++) {

        cout << A[i] << " ";

    }

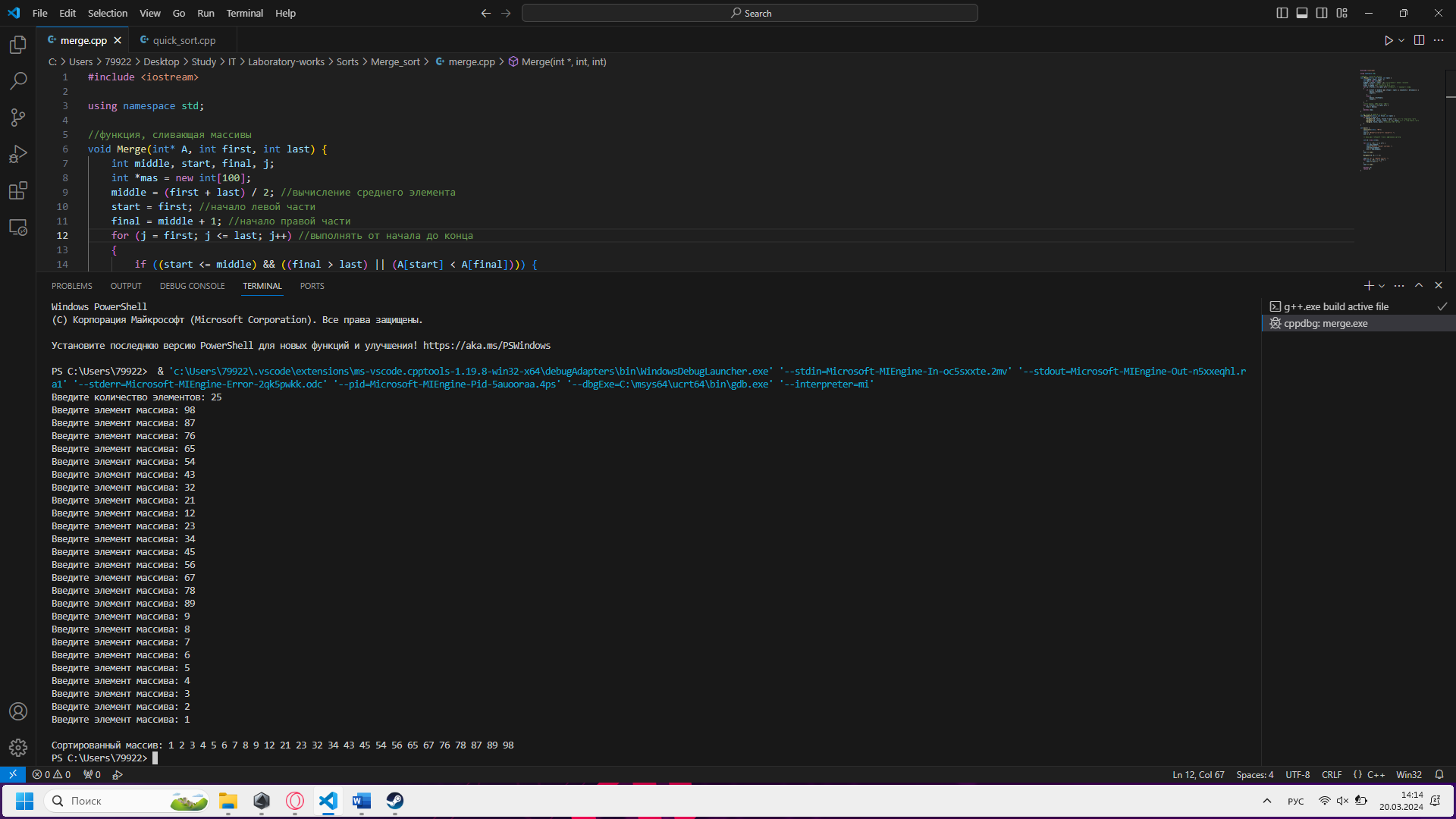
    cout << endl;

    delete[] A;

    return 0;

}

**Результаты работы**



**Quick sort**

#include <iostream>

#include <vector>

using namespace std;

// Функция для быстрой сортировки

void quicksort(vector<int>& items, int left, int right) {

    int i = left, j = right, pivot = items[(left + right) / 2], tmp;

    do {

        while ((items[i] < pivot) && (i < right)) i++;

        while ((pivot < items[j]) && (j > left)) j--;

        if (i <= j) {

            tmp = items[i];

            items[i] = items[j];

            items[j] = tmp;

            i++;

            j--;

        }

    } while (i <= j);

    if (left < j) quicksort(items, left, j);

    if (i < right) quicksort(items, i, right);

}

int main() {

    int n = 25; // Размер массива

    vector<int> arr(n);

    for (int i = 0; i < n; ++i) {

        int new\_element;

        cout<<"Введите элемент массива: ";

        cin>>new\_element;

        arr[i] = new\_element;

    }

    // Выводим исходный массив

    cout << "Исходный массив:\n";

    for (int i = 0; i < n; ++i) {

        cout << arr[i] << " ";

    }

    cout << endl;

    // Сортируем массив методом быстрой сортировки

    quicksort(arr, 0, n - 1);

    // Выводим отсортированный массив

    cout << "Отсортированный массив:\n";

    for (int i = 0; i < n; ++i) {

        cout << arr[i] << " ";

    }

    return 0;

}

**Результат работы**

