МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
федеральное государственное автономное образовательное учреждение высшего образования
«Санкт–Петербургский государственный университет
аэрокосмического приборостроения»

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
РУКОВОДИТЕЛЬ _____

преподаватель		
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине МДК. 01.02

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	С326		Э.С. Тигранян
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Лабораторная работа № 6

Тестирование производительности

Цель работы:
Научиться определять время работы программы на примере алгоритмов сортировки массивов.

Вариант 11.
Сравнить время работы следующих алгоритмов:
1) Сортировка методом выбора с обменом
2) Сортировка «слиянием»
Зафиксировать время для размерностей массива 100, 1000, 10000, 100000.

Описание алгоритмов сортировки:
Сортировка методом выбора с обменом
Алгоритм сортировки выбором заключается в следующем:

1. Нахождение минимального элемент в неотсортированной части массива.
2. Измена местами с первым элементом неотсортированной части.
3. Уменьшение размера неотсортированной части на один элемент.
4. Повторения шагов 1-3, пока неотсортированная часть не станет пустой.

Сортировка «слиянием»
Алгоритм сортировки слиянием:

1. Разделение массива на две примерно равные части.
2. Рекурсивная сортировка каждой части.
3. Слияние двух отсортированных частей в один массив.

Код программы:

```
#include <iostream>
#include <vector>
#include <chrono>
#include <random>
#include <algorithm>
#include <iomanip>

using namespace std;

// Сортировка методом выбора с обменом
void selectionSort(vector<double>& arr) {
    int n = arr.size();
    for (int i = 0; i < n - 1; i++) {
        int min_idx = i;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        swap(arr[i], arr[min_idx]);
```

```
    }
}

// Сортировка методом слияния
void merge(double* l, double* m, double* r, double* temp) {
    double* cl = l, * cr = m;
    int cur = 0;
    while (cl < m && cr < r) {
        if (*cl < *cr) temp[cur++] = *cl, cl++;
        else temp[cur++] = *cr, cr++;
    }
    while (cl < m) temp[cur++] = *cl, cl++;
    while (cr < r) temp[cur++] = *cr, cr++;
    cur = 0;
    for (double* i = l; i < r; i++)
        *i = temp[cur++];
}

void _mergesort(double* l, double* r, double* temp) {
    if (r - l <= 1) return;
    double* m = l + (r - l) / 2;
    _mergesort(l, m, temp);
    _mergesort(m, r, temp);
    merge(l, m, r, temp);
}

void mergeSort(vector<double>& arr) {
    vector<double> temp(arr.size());
    _mergesort(arr.data(), arr.data() + arr.size(), temp.data());
}

// Функция случайного заполнения вектора
void fillArray(vector<double>& arr) {
    srand(static_cast<unsigned int>(time(nullptr)));
    for (auto& num : arr) {
        // Генерация числа от 1.0 до 10000.0
        num = 1.0 + (rand() / (RAND_MAX / (10000.0 - 1.0)));
    }
}

// Вывод фрагмента массива
void printArray(const vector<double>& arr, int m = 0, int n = -1) {
    if (n == -1) n = arr.size();
    cout << "[";
    for (int i = m; i < n; i++) {
        cout << fixed << setprecision(2) << arr[i];
        if (i < n - 1) cout << ", ";
    }
    cout << "]\n";
}

// Измерение времени выполнения функции
template<typename Func>
double measureTime(Func func) {
    auto start = chrono::high_resolution_clock::now();
    func();
    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration = end - start;
    return duration.count();
}

int main() {
    setlocale(LC_ALL, "ru");
    vector<int> sizes = {10, 100, 1000, 10000, 100000 };
```

```
    for (int size : sizes) {
        vector<double> original(size);
        fillArray(original);

        cout << "\nРазмер массива = " << size << "\n";

        if (size <= 20) {
            cout << "Оригинальный массив: ";
            printArray(original);
        }

        // Сортировка выбором
        vector<double> selectionArr = original;
        double selectionTime = measureTime([&]() {
            selectionSort(selectionArr);
            });

        cout << "Скорость методом выбора: " << fixed << setprecision(6)
            << selectionTime << " сек.\n";
        if (size <= 20) {
            cout << "Отсортированный массив: ";
            printArray(selectionArr);
        }

        // Сортировка слиянием
        vector<double> mergeArr = original;
        double mergeTime = measureTime([&]() {
            mergeSort(mergeArr);
            });

        cout << "Скорость методом слияния: " << fixed << setprecision(6)
            << mergeTime << " сек.\n";
        if (size <= 20) {
            cout << "Отсортированный массив: ";
            printArray(mergeArr);
        }
    }

    return 0;
}
```

График зависимости времени сортировки от количества входных данных:

Метод сортировки	Скорость для 100 эл	Скорость для 1000 эл	Скорость для 10000 эл	Скорость для 100000 эл
Выборка	0,000075	0,004687	0,481062	29,108374
Слияние	0,000017	0,000129	0,002051	0,013250

Результаты:
Сортировка выбором имеет квадратичную сложность $O(n^2)$, что приводит к резкому увеличению времени при росте размера массива.
Сортировка слиянием имеет сложность $O(n \log n)$, поэтому время растет гораздо медленнее.

Выводы
Для небольших массивов (до 1000 элементов) можно использовать сортировку выбором, так как разница во времени невелика.
Для больших массивов (от 10000 элементов) сортировка слиянием предпочтительнее из-за ее эффективности.