Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Выполнил студент группы КС-36 Потапов Никита Александрович

Ссылка на репозиторий: (Ссылка на лабораторную в репозитории)

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 17.02.2025

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

Реализовать метод сортировки вставками. Реализовать проведения тестирования алгоритма сериями расчетов для измерения параметров времени.

* За один расчет выполняется следующие операции:

1. Генерируется массив случайных значений
2. Запоминается время начала расчета алгоритма сортировки
3. Выполняется алгоритм сортировки

(Во время выполнения измерить количество повторных прохождений по массиву. Во время выполнения измерить количество выполнения операций обмена значений.)

* Вычисляется время, затраченное на сортировку: текущее время - время начала
* Сохраняется время для одной попытки, после этого расчет повторяется до окончания серии.

1. Алгоритм вычисляется 8 сериями по 20 раз за серию.
2. Алгоритм в каждой серии вычисляется для массива размером M. (1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000)
3. Массив заполняется значения числами с плавающей запятой в интервале от -1 до 1.
4. Для серии запоминаются все времена, которые были замерены

* По полученным данным времени построить графики зависимости времени от числа элементов в массиве:

1. Совмещенный график наихудшего времени выполнения сортировки и сложности алгоритма указанной в нотации O большое. (Для построения графика вычисляется O большое для каждого размера массива. При этом при вычислении функции O(c \* g(N)) подбирается такая константа с, что бы при значении > 1000 график O(N) был выше графика наихудшего случая, но второй график на его фоне не превращался в прямую линию)
2. Совмещенный график среднего, наилучшего и наихудшего времени исполнения.
3. График среднего колличества обмена значений.
4. График повторных обходов массива.

* По результатам расчетов оформляется отчет по предоставленной форме, в отчете:
  + Приводится описание алгоритма.
  + Приводится описания выполнения задачи (Описание кода и специфических элементов реализации)
  + Приводятся выводы (Графики и их анализ)

# Описание метода/модели.

Алгоритм сортировки вставками (Insertion Sort) строит отсортированную последовательность постепенно, обрабатывая элементы входного массива по одному.

На каждом шаге алгоритм:

* Выбирает элемент из неотсортированной части массива.
* Сравнивает его с элементами отсортированной части (изначально состоящей из первого элемента), двигаясь справа налево.
* Сдвигает элементы отсортированной части вправо, чтобы освободить позицию для вставки.
* Вставляет текущий элемент на найденную позицию.

Процесс повторяется до полной обработки массива.

**Пример:**

Для массива [5, 2, 4, 6, 1] шаги будут выглядеть так:

После первой итерации: [2, 5, 4, 6, 1].

После второй: [2, 4, 5, 6, 1].

И так далее, пока не получим [1, 2, 4, 5, 6].

**Сложность алгоритма**

Лучший случай: O(n) — если массив уже отсортирован. Каждый элемент вставляется за O(1).

Средний и худший случаи: O(n²) — если элементы расположены в обратном порядке или случайно. Каждый элемент требует сравнения со всеми предыдущими.

**Потребление памяти**

Алгоритм использует O(1) дополнительной памяти (только для временных переменных). Данные сортируются непосредственно в исходном массиве.

**Преимущества:**

* Простота реализации.
* Эффективен для малых массивов или частично отсортированных данных.
* Стабильность: сохраняет порядок равных элементов.
* Адаптивность: время работы уменьшается, если массив почти упорядочен.

**Проблемы:**

* Низкая эффективность на больших массивах из-за квадратичной сложности.
* Не подходит для данных, хранящихся в медленной памяти (например, на диске), из-за большого числа операций сдвига.

# Выполнение задачи.

**Практическое описание решения**

Программа реализует алгоритм сортировки вставками (Insertion Sort) для массивов случайных чисел в интервале от -1 до 1.

**Основные задачи программы:**

* Генерация массивов случайных чисел заданного размера.
* Сортировка массивов с измерением времени выполнения, количества обменов (swaps) и количества проходов (passes).
* Сохранение результатов в текстовые файлы для последующего анализа и построения графиков.

**Используемый язык**

Программа написана на языке C++. Используются стандартные библиотеки:

* <iostream> для ввода-вывода.
* <vector> для работы с динамическими массивами.
* <random> для генерации случайных чисел.
* <chrono> для измерения времени выполнения.
* <fstream> для записи данных в файлы.

Программа состоит из следующих компонентов:

Функция generate\_random\_array: Генерирует массив случайных чисел типа double в интервале от -1 до 1. Использует генератор случайных чисел mt19937 и равномерное распределение uniform\_real\_distribution.

Функция insertion\_sort: Реализует алгоритм сортировки вставками. Принимает массив и два счетчика: swaps (количество обменов) и passes (количество проходов). В процессе сортировки обновляет значения счетчиков.

Функция main: Задает параметры для тестирования: количество попыток на серию (attempts\_per\_series) и размеры массивов (array\_sizes). Для каждого размера массива генерирует случайные данные, сортирует их и измеряет время выполнения, количество обменов и проходов. Результаты записываются в три файла: times.txt, swaps.txt, passes.txt.

**Проведенные тесты**

Программа проводит тестирование на массивах следующих размеров:

1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000.

Для каждого размера массива выполняется 20 попыток (задается параметром attempts\_per\_series).

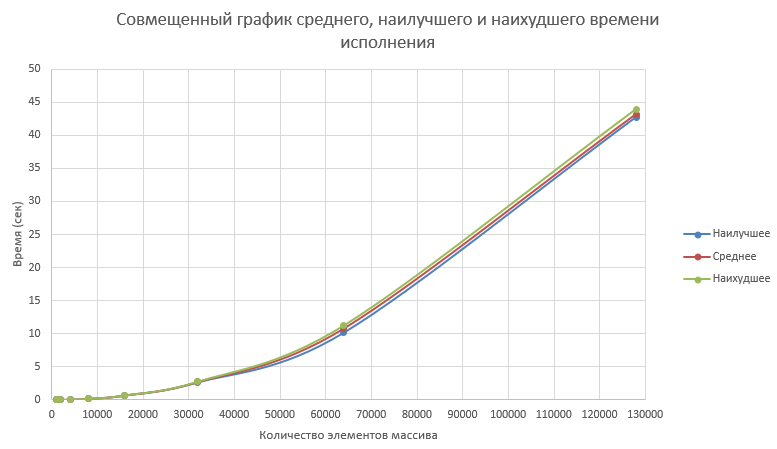
В процессе тестирования измеряются:

* Время выполнения сортировки (в секундах).
* Количество обменов (swaps) — сколько раз элементы массива менялись местами
* Количество проходов (passes) — сколько раз алгоритм проходил по массиву.

**Анализ результатов**

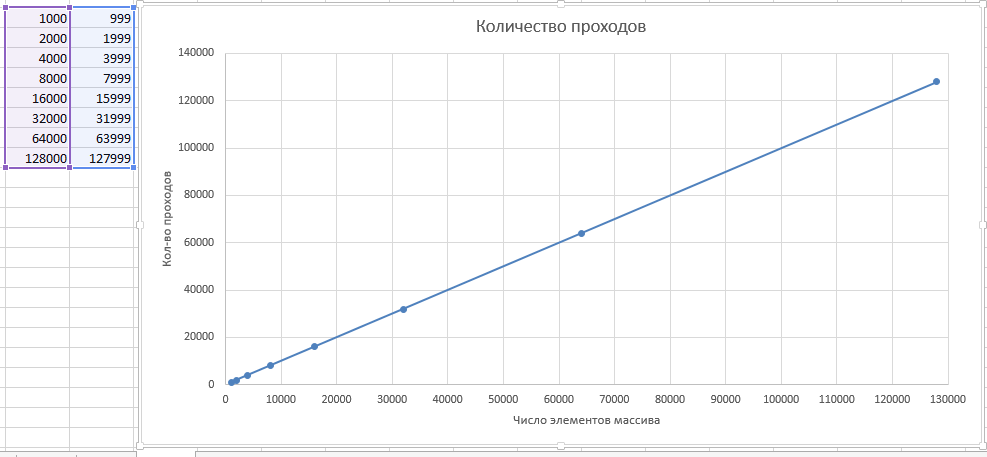
****

Количество обменов зависит от степени упорядоченности исходного массива. Для случайных данных количество обменов близко к n-(~10).

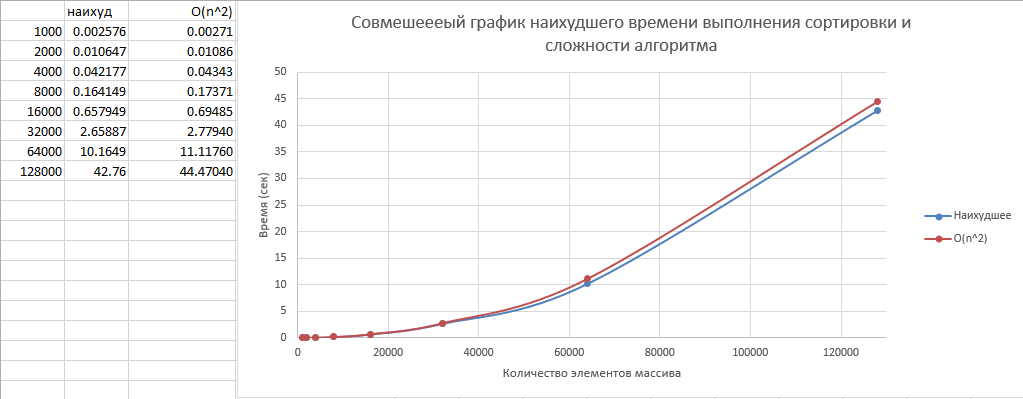




Время выполнения растет квадратично O(n²) с увеличением размера массива.



Количество проходов всегда равно n-1, где n — размер массива, так как проход выполняется для каждого элемента, начиная со второго.



# Заключение.

**Применимость алгоритма сортировки вставками**

Лабораторная работа позволила лучше понять принцип работы сортировки вставками. Алгоритм показал себя достаточно эффективным для небольших объемов данных (до 32 000 элементов). На массивах размером 32 000 и более элементов время выполнения значительно увеличилось, что соответствует теоретической сложности O(n²).

**Сложность реализации**

Алгоритм сортировки вставками легко реализуется. Его код интуитивно понятен и требует минимального количества строк. Программа была структурирована: генерация данных, сортировка и замеры времени были разделены на отдельные функции. Это упрощает чтение и модификацию кода.

**Впечатления от реализации**

Реализация алгоритма заняла минимум времени благодаря его простоте. Запись результатов в файлы (times.txt, swaps.txt, passes.txt) позволила легко анализировать данные и строить графики.

В целом, работа подтвердила, что сортировка вставками — это простой и полезный алгоритм для определенных задач, но его применение следует ограничить небольшими или частично упорядоченными наборами данных.