МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский**

**Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Прикладная математика и информатика»

**ОТЧЕТ**

по лабораторной работе

Тема:

**«Сортировки массивов»**

**Выполнил:**

студент группы 3824Б1ПМ4 Юсипов А. С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись  **Преподаватель:**

Мееров И. Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Нижний Новгород

2024

**Содержание:**

[**Введение** 2](file:///C:\Users\Арсений\AppData\Local\Packages\38833FF26BA1D.UnigramPreview_g9c9v27vpyspw\LocalState\4\documents\Отчет_Лацплес_lab5.docx#_Toc6275)

[**Постановка задачи** 3](file:///C:\Users\Арсений\AppData\Local\Packages\38833FF26BA1D.UnigramPreview_g9c9v27vpyspw\LocalState\4\documents\Отчет_Лацплес_lab5.docx#_Toc6276)

[**Описание алгоритмов** 3](file:///C:\Users\Арсений\AppData\Local\Packages\38833FF26BA1D.UnigramPreview_g9c9v27vpyspw\LocalState\4\documents\Отчет_Лацплес_lab5.docx#_Toc6277)

[**Описание программной реализации** 5](file:///C:\Users\Арсений\AppData\Local\Packages\38833FF26BA1D.UnigramPreview_g9c9v27vpyspw\LocalState\4\documents\Отчет_Лацплес_lab5.docx#_Toc6278)

[**Результаты экспериментов** 6](file:///C:\Users\Арсений\AppData\Local\Packages\38833FF26BA1D.UnigramPreview_g9c9v27vpyspw\LocalState\4\documents\Отчет_Лацплес_lab5.docx#_Toc6279)

[**Заключение** 7](file:///C:\Users\Арсений\AppData\Local\Packages\38833FF26BA1D.UnigramPreview_g9c9v27vpyspw\LocalState\4\documents\Отчет_Лацплес_lab5.docx#_Toc6280)

[**Приложение** 7](file:///C:\Users\Арсений\AppData\Local\Packages\38833FF26BA1D.UnigramPreview_g9c9v27vpyspw\LocalState\4\documents\Отчет_Лацплес_lab5.docx#_Toc6282)

# Введение

Сортировка является одной из важнейших задач в программировании и играет ключевую роль в обработке данных. В языке C, который славится своей производительностью и эффективностью, правильная реализация алгоритмов сортировки может существенно повлиять на время выполнения программы и её общий отклик.

Почему сортировка важна?

Упрощение поиска:

Сортировка данных позволяет значительно упростить процесс поиска. Например, если массив отсортирован, можно использовать алгоритмы поиска, такие как бинарный поиск, которые имеют гораздо более низкую временную сложность по сравнению с линейным поиском. Это позволяет быстро находить нужные элементы в больших объемах данных.

Оптимизация алгоритмов:

Многие алгоритмы, такие как алгоритмы слияния и динамического программирования, требуют предварительной сортировки данных для повышения своей эффективности. Например, алгоритм "Сортировка слиянием" использует метод «разделяй и властвуй», деля массив на более мелкие подмассивы и сортируя их, а затем объединяя обратно.

Обработка данных:

Сортировка — это важный этап в обработке данных, который используется в таких приложениях, как базы данных, системы управления и обработки данных, а также в веб-приложениях для оптимизации выборок на основе заданных критериев.

Сравнение и анализ:

При анализе данных часто необходимо сравнивать элементы между собой. Отсортировав данные, можно легко проводить различные операции сравнения, такие как нахождение дубликатов или выявление паттернов.

Реализация сортировки на языке C

В языке C существует множество алгоритмов сортировки, каждый из которых имеет свои преимущества и недостатки. Наиболее популярными являются:

Сортировка пузырьком (Bubble Sort): простой, но неэффективный для больших массивов, так как имеет временную сложность O(n^2).

Сортировка вставками (Insertion Sort): более эффективна для небольших массивов или массивов, которые уже частично отсортированы.

Сортировка выбором (Selection Sort): также имеет временную сложность O(n^2), но иногда проще в реализации.

Сортировка слиянием (Merge Sort): эффективный алгоритм с временной сложностью O(n log n), использующий подход «разделяй и властвуй».

Быстрая сортировка (Quick Sort): один из самых популярных и быстрых алгоритмов сортировки в среднем случае, со сложностью O(n log n).

# Постановка задачи

Задача состояла в создании программы, которая сортирует заранее сгенерированный массив с помощью трех сортировок: сортировка пузырьком, сортировка выбором, сортировка вставками. Далее программа должна вывести время за которое она отсортировала массив, мы же в свою очередь должны выяснить какой метод сортировки является наилучшим для определенного размера массива.

# Описание алгоритмов

1. **Сортировка пузырьком (Bubble Sort)**

**Описание:**

Сортировка пузырьком (Bubble Sort) — это один из наиболее простых и интуитивно понятных алгоритмов сортировки. Несмотря на свою простоту, он неэффективен для больших массивов, так как имеет временную сложность O(n²). Однако он может служить отличным введением в алгоритмы сортировки и основам работы с массивами.

**Принцип работы алгоритма пузырьковой сортировки:**

Сравнение соседних пар: Алгоритм проходит по массиву и сравнивает каждую пару соседних элементов.

Обмен элементов: Если первый элемент пары больше второго, элементы обмениваются местами (больший "всплывает" вверх по массиву).

Проход по массиву: Процесс продолжается до тех пор, пока весь массив не будет отсортирован. После каждой полной итерации, последний элемент становится на своё окончательное место, так как в конце находятся наибольшие элементы.

Повторение: Алгоритм повторяет этот процесс для оставшихся элементов, игнорируя уже отсортированные.

1. **Сортировка выбором (Selection Sort)**

**Описание:**

Сортировка выбором (Selection Sort) — это простой и интуитивно понятный алгоритм сортировки, который работает, выбирая наименьший (или наибольший) элемент из неотсортированной части массива и помещая его в начало (или конец) отсортированной части. Этот алгоритм имеет временную сложность O(n²), что делает его неэффективным для больших массивов, но он удобен для небольших объемов данных и вводит в основы работы с массивами и алгоритмами сортировки.

**Принцип работы алгоритма сортировки выбором:**

Инициализация: Алгоритм делит массив на две части — отсортированную и неотсортированную. Изначально, отсортированная часть пуста, а неотсортированная содержит весь массив.

Поиск минимального элемента: На каждом шаге алгоритм проходит по неотсортированной части массива и находит минимальный элемент.

Обмен местами: После нахождения минимального элемента, он меняется местами с первым элементом неотсортированной части массива, тем самым увеличивая размер отсортированной части.

Повторение: Процесс повторяется для оставшейся неотсортированной части, пока весь массив не будет отсортирован.

1. **Сортировка вставками (Insertion Sort)**

**Описание:**

Сортировка вставками (Insertion Sort) — это еще один простой алгоритм сортировки, который строит отсортированную последовательность по мере обработки элементов массива. Хотя данный алгоритм менее эффективен на больших объемах данных по сравнению с более продвинутыми алгоритмами (такими как быстрая сортировка или сортировка слиянием), он, тем не менее, имеет свои преимущества: простота реализации, устойчивость и хорошая производительность на небольших или частично отсортированных массивах.

**Принцип работы сортировки вставками:**

Алгоритм сортировки вставками работает по следующему принципу:

Инициализация: Представьте, что массив делится на две части — отсортированную и неотсортированную. Изначально отсортированной частью будет только первый элемент массива.

Обработка элементов: Вы берете следующий элемент из неотсортированной части массива и вставляете его в правильное место в отсортированной части. Это делается следующим образом:

Сохраняем значение текущего элемента (который будет вставлен).

Сравниваем его с элементами отсортированной части (идущими от конца к началу).

Сдвигаем элементы, если они больше текущего элемента, чтобы освободить место для вставки.

Вставка: Как только найдено правильное место, элемент вставляется между меньшим и большим значением.

Повторение: Процесс повторяется для всех элементов, пока весь массив не будет отсортирован.

# Описание программной реализации

В данной программе реализованы три алгоритма сортировки: сортировка пузырьком, сортировка выбором и сортировка вставками. Программа позволяет пользователю выбрать один из этих алгоритмов для сортировки массива случайных целых чисел. Ниже представлено подробное описание каждой части программы.

## 1. Подключение библиотек

**stdio.h**: Библиотека для ввода и вывода данных.

**stdlib.h**: Библиотека для работы с памятью и генерации случайных чисел.

**time.h**: Библиотека для работы с временем, используется для измерения времени выполнения сортировок.

## 2. Алгоритмы сортировки

**Сортировка пузырьком (bubble\_sort)**:

Проходит по массиву и сравнивает соседние элементы, меняя их местами, если они находятся в неправильном порядке. Процесс повторяется до тех пор, пока не будет выполнен полный проход без изменений.

**Сортировка выбором (search\_sort)**:

На каждой итерации находит наименьший элемент в неотсортированной части массива и перемещает его в начало отсортированной части.

**Сортировка вставками (insertion\_sort)**:

Строит отсортированный массив, вставляя каждый элемент в правильное положение относительно уже отсортированных элементов.

1. ***Генерация массива***

В консоли вводится размер сортируемого массива, сразу создается три его копии, для трех сортировок каждая.

Значения массива генерируются рандомно при помощи функции rand().

После, сгенерированный массив выводится на экран.

## 4. Основная функция

Копии массива создаются через указатель (int\*)malloc(size\*sizeof(int\*))

В конце память освобождается при помощи функции free

***5. Цикл выбора сортировки*** В этом блоке программа предлагает пользователю выбрать тип сортировки. В зависимости от выбора, массив mas копируется во временный массив copy с помощью функции memcpy, чтобы сохранить исходные данные для каждой сортировки.

## 6. Измерение времени выполнения

Для каждой сортировки используется функция clock() для измерения времени выполнения. Время выполнения каждой сортировки сохраняется в переменных time1, time2 и time3.

***7. Вывод результатов***

После завершения сортировок программа выводит отсортированный массив и время выполнения каждого алгоритма на экран.

## 8. Освобождение памяти

В конце программы освобождается память, выделенная для массива mas, с помощью функции free(), что предотвращает утечки памяти.

# Результаты экспериментов

100 элементов:

Bubble sort – 0 с

Search sort – 0 с

Insertion sort – 0 с

Результаты одинаковые, равные нулю.

10000 элементов:

Bubble sort – 0.191000 с

Search sort – 0.089000 с

Insertion sort – 0.045000 с

Результаты различаются, но не сильно.

100000 элементов:

Bubble sort – 23.408000 с

Search sort – 8.567000 с

Insertion sort – 4.561000 с

Самая долгая – Bubble sort, самая быстрая – Insertion sort

200000 элементов:

Bubble sort – 99.743000с

Search sort – 34.295000 с

Insertion sort – 18.371000 с

Bubble sort очень долго сортирует, по сравнению с другими видами сортировок.

# Заключение

Подводя итоги, можно сказать, что по крайней мере на моей модели компьютера, Bubble sort имеет самые большие показатели времени по сравнению со всеми другими видами сортировки.

# Приложение

https://github.com/Arseniy-1-1-1/Sortirovki