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:

**«Сортировка массивов разными способами»**

**Выполнил:**

студент группы 3824Б1ПМ4 Растунин.А. Д.

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

подпись  **Преподаватель:**

Куклин А.Е.

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

подпись

Нижний Новгород

2024

**Содержание:**

[**Введение** 2](#_Toc6275)

[**Постановка задачи** 3](#_Toc6276)

[**Описание алгоритмов** 3](#_Toc6277)

[**Описание программной реализации** 4](#_Toc6278)

[**Результаты экспериментов** 6](#_Toc6279)

[**Заключение** 7](#_Toc6280)

[**Литература** 7](#_Toc6281)

[**Приложение** 7](#_Toc6282)

# Введение

Сортировка — это ключевая операция в области программирования и компьютерных наук, играющая важную роль в обработке данных. Ее значимость можно рассмотреть с нескольких аспектов:  
  
1. Упрощение поиска: Многие алгоритмы поиска, такие как бинарный поиск, требуют, чтобы данные были отсортированы. Это позволяет значительно ускорить процесс поиска по сравнению с линейным методом.  
  
2. Улучшение производительности: Сортированные данные облегчают выполнение различных операций, таких как слияние, объединение и удаление дубликатов. Это становится особенно заметным при работе с большими массивами данных.  
  
3. Анализ данных: В статистике и аналитике сортировка является важным шагом для подготовки данных к анализу. Она позволяет выявить тенденции, осуществить группировку и необходимые вычисления.  
  
4. Оптимизация алгоритмов: Многие алгоритмы, основанные на сортировке, могут существенно улучшить производительность программ. Например, алгоритмы сортировки помогают в упорядочивании данных перед их обработкой, что может снизить время выполнения последующих операций.  
  
5. Применение в реальных задачах: Веб-приложения, базы данных и системы управления контентом часто требуют сортировки данных для представления пользователям. Например, пользователи могут сортировать товары по цене, популярности или рейтингу.  
  
Итак, сортировка — это не просто академическая концепция, а реальный инструмент, который активно используется в разработке программного обеспечения, повышая скорость и эффективность работы с данными. Компетенция в алгоритмах сортировки — важный навык для любого программиста.

В этом отчете рассматриваются три популярных алгоритма сортировки:

1. **Сортировка пузырьком (Bubble Sort)**
2. **Сортировка поиском (Selection Sort)**
3. **Сортировка вставками (Insertion Sort)**

# 

# Постановка задачи

Создать программу, которая сортирует заранее сгенерированный массив с помощью трех сортировок: сортировка пузырьком, сортировка поиском, 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 — это алгоритм сортировки, который работает путем многократного прохода по массиву, сравнивая соседние элементы и меняя их местами, если они находятся в неправильном порядке. Процесс повторяется до тех пор, пока не будет выполнен полный проход без изменений, что означает, что массив отсортирован.

**Принцип работы:**

* 1. Начинаем с первого элемента массива.
  2. Сравниваем текущий элемент со следующим.
  3. Если текущий элемент больше следующего, меняем их местами.
  4. Переходим к следующему элементу и повторяем шаги 2-3.
  5. После завершения прохода по массиву, повторяем процесс, пока не будет выполнен полный проход без изменений.

1. **Сортировка поиском (Search Sort)**

**Описание:**

Сортировка выбором — это алгоритм, который сортирует массив, находя наименьший элемент в неотсортированной части массива и перемещая его в начало отсортированной части. Этот процесс повторяется для всех элементов массива.

**Принцип работы:**

* 1. Разделите массив на отсортированную и неотсортированную части.
  2. На каждой итерации находите наименьший элемент в неотсортированной части.
  3. Меняйте местами найденный элемент с первым элементом неотсортированной части.
  4. Увеличивайте границу отсортированной части на один элемент и повторяйте процесс.

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

**Описание:**

Сортировка вставками — это алгоритм, который строит отсортированный массив, вставляя каждый новый элемент в правильное положение относительно уже отсортированных элементов. Этот алгоритм особенно эффективен для небольших массивов и частично отсортированных данных.

**Принцип работы:**

* 1. Начинаем с первого элемента, который считается отсортированным.
  2. Берем следующий элемент и сравниваем его с отсортированной частью.
  3. Вставляем элемент в правильное положение, сдвигая все большие элементы вправо.
  4. Повторяем процесс для всех элементов массива.

# Описание программной реализации

В данной программе реализованы три алгоритма сортировки: сортировка пузырьком, сортировка поиском 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.
* **cstring**: Библиотека для работы с функциями манипуляции строками и массивами (в данном случае используется для функции memcpy).

## 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.

Каждый из алгоритмов реализован в отдельной функции, принимающей массив и его размер в качестве аргументов.

***3. Основная функция***

1)Запрос размера массива у пользователя

2)Выделение памяти для массива array с помощью функции malloc. Здесь array = (int\*)malloc(lenght \* sizeof(int)); выделяет память для массива целых чисел размером lenght.

Использование sizeof(int) позволяет определить, сколько байт нужно выделить для массива целых чисел.

3)Для генерации случайных чисел используется функция rand(), инициализированная с помощью srand(time(NULL)), что обеспечивает разнообразие значений при каждом запуске программы.

4***)*** Цикл выбора сортировки.В этом блоке программа предлагает пользователю выбрать тип сортировки. В зависимости от выбора, массив array копируется во временный массив copy с помощью функции memcpy, чтобы сохранить исходные данные для каждой сортировки. Также на этом этапе происходит подсчет времени работы функции.

## 5. Освобождение памяти

В конце программы освобождается память, выделенная для массивов array и copy, с помощью функции free(), что предотвращает утечки памяти.

# Результаты экспериментов

Я проводил эксперименты над массивами, содержащими минимум 10000 элементов, так как при меньшем их содержании выводиться время, которое трудно сравнивать так как оно измеряется в сотых секундах.

10000 элементов:

Search sort – 0.38 с

Bubble sort – 0.2 с

Insertion sort – 0.1 с

50000 элементов:

Search sort – 10.8 с

Bubble sort – 4.9 с

Insertion sort – 2.6 с

99999 элементов:

Search sort – 46 с

Bubble sort – 19.22 с

Insertion sort – 10.2 с

Лучший результат в этом тесте показал Insertion sort.

# Заключение

Из результатов экспериментов видно, что присутствует разница иногда в долю секунды, иногда в пару секунд, но эту разницу можно считать незначительной. В итоге я считаю, что для маленьких массивов можно выбрать любую из этих трех сортировок, но для больших массивов стоит пользоваться более сложными сортировками.

# Литература

https://stackoverflow.com/questions/3557221/how-do-i-measure-time-in-c

# Приложение