

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  
Лацплес Г.П.

---

подпись

**Преподаватель:**  
Куклин А.Е.

---

подпись

Нижний Новгород  
2024

## Содержание:

Введение .....	2
Постановка задачи .....	2
Описание алгоритмов.....	2
Описание программной реализации.....	3
Результаты экспериментов .....	5
Заключение .....	6
Литература.....	6
Приложение .....	6

## Введение

Сортировка является одной из основных задач в области алгоритмов и структур данных. Она подразумевает упорядочивание элементов в массиве или списке по определенному критерию, например, по возрастанию или убыванию. Эффективные алгоритмы сортировки имеют важное значение в программировании, поскольку они используются во многих приложениях, таких как базы данных, поисковые системы и системы обработки данных.

В этом отчете рассматриваются три популярных алгоритма сортировки:

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

## 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 — это простой 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.

### 2. Сортировка выбором (Selection 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 повторите процесс.

## 3. 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 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.
- **cstring:** Библиотека для работы с функциями манипуляции строками и массивами (в данном случае используется для функции `memset`).

### 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. Генерация массива***

Функция `array_generating` заполняет массив случайными целыми числами в диапазоне от 0 до 999. Для генерации случайных чисел используется функция `rand()`, инициализированная с помощью `srand(time(NULL))`, что обеспечивает разнообразие значений при каждом запуске программы.

### ***4. Основная функция***

В основной функции происходит:

- Запрос размера массива у пользователя.
- Выделение памяти для массива `mas` с помощью функции `malloc`. Здесь `mas = (int*)malloc(size * sizeof(int))`; выделяет память для массива целых чисел размером `size`.  
Использование `sizeof(int)` позволяет определить, сколько байт нужно выделить для массива целых чисел.
- Проверка успешности выделения памяти. Если `mas` равно `NULL`, программа выводит сообщение об ошибке и завершает выполнение.

### ***5. Цикл выбора сортировки***

В этом блоке программа предлагает пользователю выбрать тип сортировки. В зависимости от выбора, массив `mas` копируется во временный массив `сор` 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 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

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:

Bubble sort – 0.1 с

Search sort – 0.1 с

Insertion sort – 0.097 с

Результаты практически одинаковые.

50000 элементов:

Bubble sort – 2.6 с

Search sort – 2.39 с

Insertion sort – 2.5 с

Лучший результат в этом тесте показал Search sort.

100000 элементов:

Bubble sort – 10.44 с

Search sort – 10.22 с

Insertion sort – 10.14 с

Лучший результат в этом тесте показал Insertion sort.

150000 элементов:

Bubble sort – 30.6 с

Search sort – 29.7 с

Insertion sort – 28.5 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**

[https://github.com/Velreiz/Sorting\\_lab](https://github.com/Velreiz/Sorting_lab)