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»**

**Выполнила:**

студентка группы 3824Б1ПМ4

Чиркунова А.Р.

подпись

**Преподаватель:**

Мееров И.Б.

подпись

Нижний Новгород  
2024

Содержание:

[**Введение** 3](#_Toc106028628)

[**Постановка задачи** 4](#_Toc106028629)

[**Описание алгоритмов**](#_Toc106028630) 5

[**Описание программной реализации**](#_Toc106028631) 7

[**Результаты экспериментов**](#_Toc106028632) 9

[**Заключение**](#_Toc106028633) 10

**Введение**

Сортировка в программировании помогает упорядочивать данные, организовывать к ним быстрый доступ, прогнозировать выполнение сложных операций и экономить ресурсы. Эффективные алгоритмы сортировки имеют важное значение в программировании, поскольку они используются во многих приложениях.

В данном отчете будут рассматриваться три основных алгоритма сортировки:

1. Сортировка пузырьком (Bubble Sort)

2. Сортировка выбором (Search Sort)

3. Сортировка вставками (Insertion Sort)

**Постановка задачи**

Цель лабораторной работы — написать код выполняющий сортировку массива разными способами (длина массива задаётся вводом с клавиатуры, числа в массиве генерируются случайным образом, пользователь сам выбирает нужную сортировку). Программа также должна засекать время, за которое выполняется определённая сортировка. Задача — выяснить какой метод сортировки массива эффективнее.

**Описание алгоритма**

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

Описание:

Сортировка пузырьком — простой алгоритм сортировки, который работает путём многократного перехода к 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; это 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 по массиву без изменений.

**2. Сортировка выбором (Search Sort)**

Описание:

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

Принцип работы:

1. Разделите массив на отсортированную и не отсортированную часть.

2. На каждой итерации находите наименьший элемент в не отсортированной части.

3. Меняйте местами найденный элемент с первым элементом не отсортированной части.

4. Увеличивайте границу отсортированной части на один элемент и повторяйте процесс.

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

Описание:

Сортировка вставками — 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. Начнём с первого элемента, который считается отсортированным.

2. Берём 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.

**2. Алгоритмы сортировки.**

Сортировка пузырьком (BubbleSort). 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 (SearchSort). На каждой итерации находит наименьший элемент в не отсортированной части массива и перемещает его в начало отсортированной части.

Сортировка вставками (InsertionSort). 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:

1. Запрос размера массива у пользователя.

2. 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.

3. 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 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 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**

Будем проводить эксперименты над массивами, содержащими не менее, чем 10000 элементов. Иначе время работы функции сортировки слишком незначительно.

**10000 элементов:**

Bubble Sort = 0,094 c

Search Sort = 0,037 c

Insertion Sort = 0,055 c

Результат: Сортировка выбором — самая эффективная.

**50000 элементов:**

Bubble Sort = 4,445 c

Search Sort = 0,669 c

Insertion Sort = 1,216 c

Результат: Сортировка выбором — самая эффективная.

**100000 элементов:**

Bubble Sort = 19,469 c

Search Sort = 2,504 c

Insertion Sort = 4,232 c

Результат: Сортировка выбором — самая эффективная.

**Заключение**

Из результатов эксперимента можно сделать выводы. Самая эффективная — сортировка выбором (Search Sort). На маленьких массивах можно пользоваться любой сортировкой, так как разница во времени слишком незначительна, но на массивах большой длины лучше использовать сортировку выбором, так как она работает быстрее. Сортировка вставками тоже работает довольно быстро и эффективно, но медленнее, чем сортировка выбором. Сортировка пузырьком на больших массивах работает очень долго; она неэффективна.