Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **«ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

по лабораторной работе № 3

Название: Анализ алгоритмов сортировки массовов

Дисциплина: Анализ алгоритмов

Студент ИУ7-52Б Короткая В. М.

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель Волкова Л.Л.

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Содержание

Введение

В этой лабороторной работе мы рассматриваем вопрос, который часто возникает в программировании - перемещение элементов в порядке возрастания или убывания. Можно легко представить, как бы усложнило жизнь пользование словарем, в котором слова расположены не в алфовитном порядке. Точно также, от порядка хранения элементов в памяти компьютера, зависит скорость выполнения и простота алгоритмов над этими элементами.

Вот одни из наиболее важных областей применения сортировок:

- решение задач групперирования, когда нужно собрать вместе все элементы с одинаковыми значениями некоторого признака;
- поиск общих элементов в двух и более массивах; если два и более массива рассортировать в одном и том же порядке, то можно найти одинаковые элементы за один последовательный просмотр без возвратов;
- поиск информации по значениям ключей.

Цель данной лабораторной работы заключается в изучении алгоритмов сортировки массивов. Рассматриваются алгоритмы сортировки пузырьком, вставками и Шейкер сортировка. Требуется расчитать и изучить трудоемкость и затрачиваемое каждым алгоритмом время.

Выделим следующие задачи:

- изучить работу алгоритмов сортировки;
- выполнить полную математическую оценку трудоемкости для алгоритмов сортировки с указанием лучшего и худшего случаев;
- реализовать три алгоритма сортировки;
- сравнить работу алгоритмов сортировок и сделать выводы.

1. Аналитическая часть

В данном разделе будут рассмотрены алгоритмы сортировки пузырьком, вставками и шейкерная сортировка. Также разобраны принципы работы этих алгоритмов.

1.1. Сортировка пузырьком (Bubble sort)

Сортировка пузырьком — это простейший и один из самых известных алгоритмов сортировки. Идея заключается в последовательном сравнении значений соседних элементов. Если текущий элемент больше следующего, меняем их местами. Алгоритм необходимо повторять до тех пор, пока массив не будет отсортирован.

Этот алгоритм считается учебным и почти не применяется на практике из-за низкой эффективности: он медленно работает на тестах, в которых маленькие элементы (их называют «черепахами») стоят в конце массива. Однако на нём основаны многие другие методы, например, шейкерная сортировка и сортировка расчёской.

1.2. Сортировка вставками (Insertion sort)

Сортировка вставками - алгоритм, при котором каждый последующий элемент массива сравнивается с предыдущими элементами (отсортированными) и вставляется в нужную позицию.

Общая идея алгоритма:

1. Сравниваем второй элемент с первым элементом массива и при необходимости меняем их местами. Условно эти элементы (первый и второй) будут являться отсортированным массивом, остальные элементы - неотсортированным.

2. Сравниваем следующий элемент из неотсортированного массива с элементами отсортированного и вставляем в нужную позицию.

3. Повторям шаг 2 до тех пор, пока в неотсортированном массиве не останется элементов.

1.3. Сортировка перемешиванием (Shaker)

Также известна как шейкерная или коктейльная сортировка.

Сортировка перемешиванием - это разновидность сортировки пузырьком. Отличие в том, что данная сортировка в рамках одной итерации проходит по массиву в обоих направлениях (слева направо и справа налево), тогда как сортировка пузырьком - только в одном направлении (слева направо).

Общая идея алгоритма:

1. Обход массива слева направо, аналогично пузырьковой - сравнение соседних элементов, меняя их местами, если левое значение больше правого. В итоге наибольшее число будет перемещено в конец массива.

2. Обход массива в обратном направлении (справа налево), начиная с элемента, который находится перед последним отсортированным. На этом этапе элементы также сравниваются между собой и меняются местами, чтобы наименьшее значение всегда было слева. В итоге наименьшее число будет перемещено в начало массива.

Вывод

Таким образом, были рассмотрены алгоритмы сортировки пузырьком, вставками и шейкерная сортировка. Также разобраны принципы работы этих алгоритмов.

Входными данными реализуемого ПО являются:

- размерность массива - целое число;

- целочисленный массив;

Выходными данными реализуемого ПО являеться результат алгоритмов сортировки т. е. отсортированный массив по возростанию.

Ограничением для реализуемого ПО является - размерность вводимого массива т. е. введеное число должно быть натуральным.

2. Конструкторская часть

В данном разделе будут сформированы требования к программе и составлены схемы алгоритмов. Также подсчитана трудоёмкость для каждого их алгоритмов.

2.1. Схемы алгоритмов

На рисунках 2.1 - 2.4 приведены схемы алгоритмов сортировки массива.

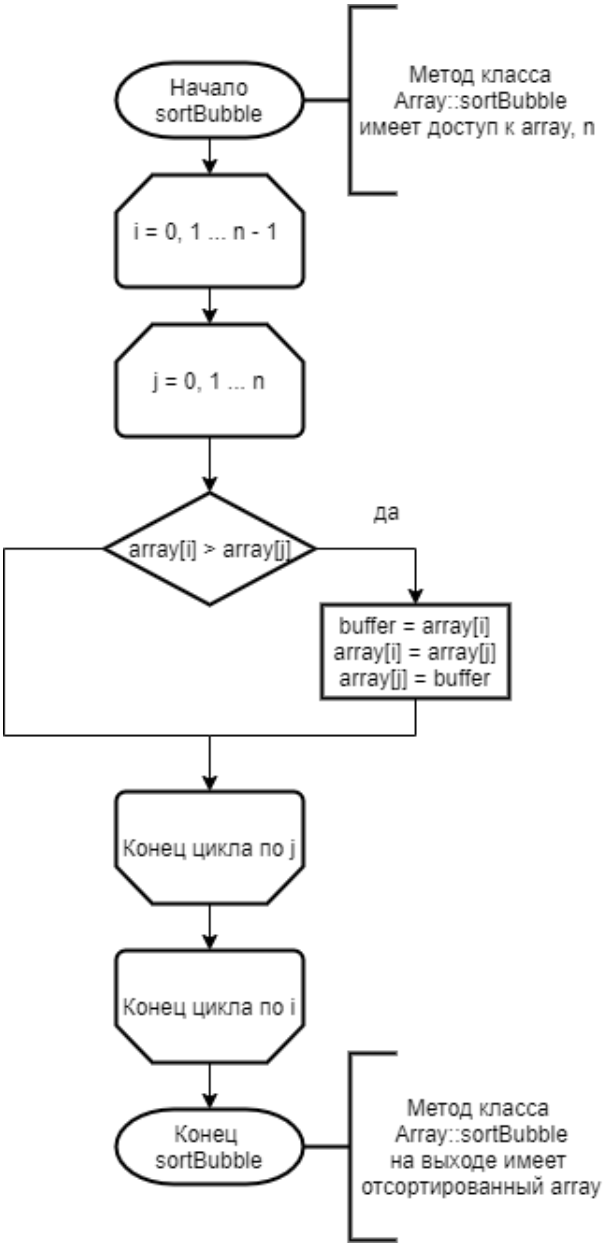

Рис. 2.1: Схема сортировки пузырьком

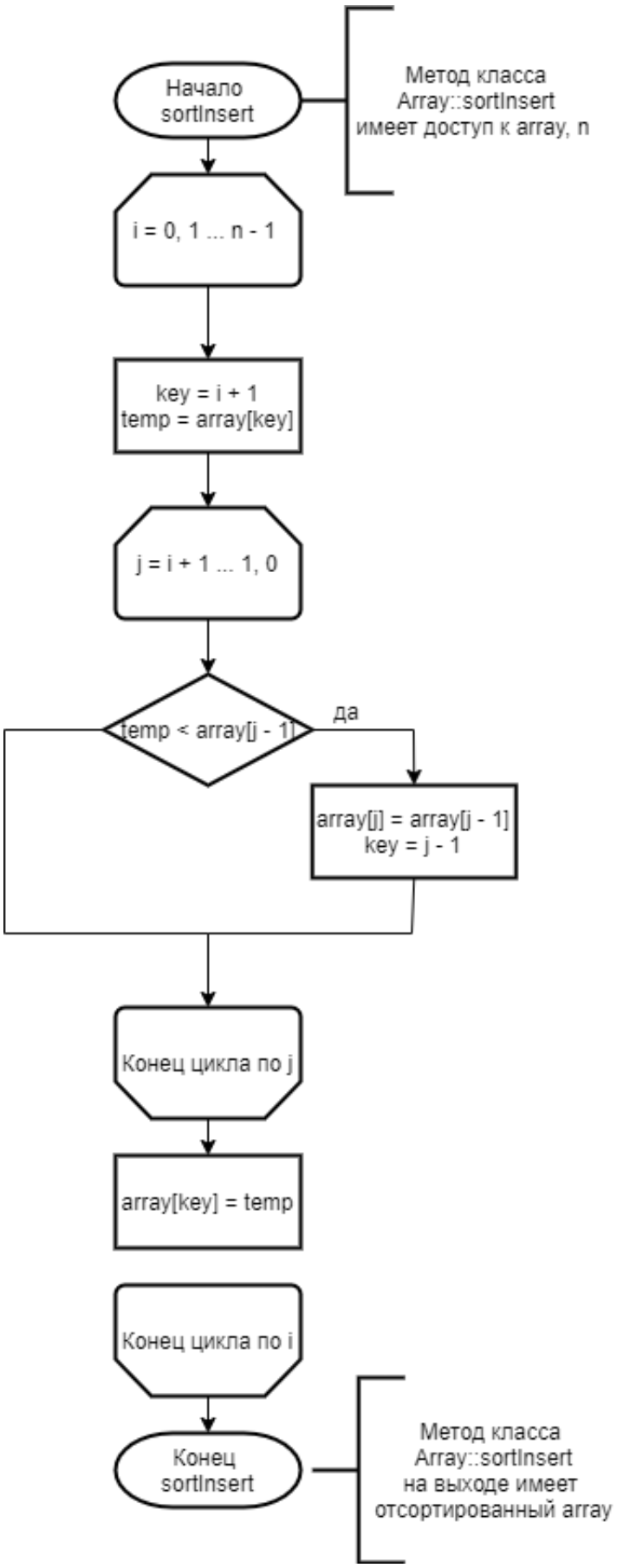

Рис. 2.2: Схема сортировки вставками

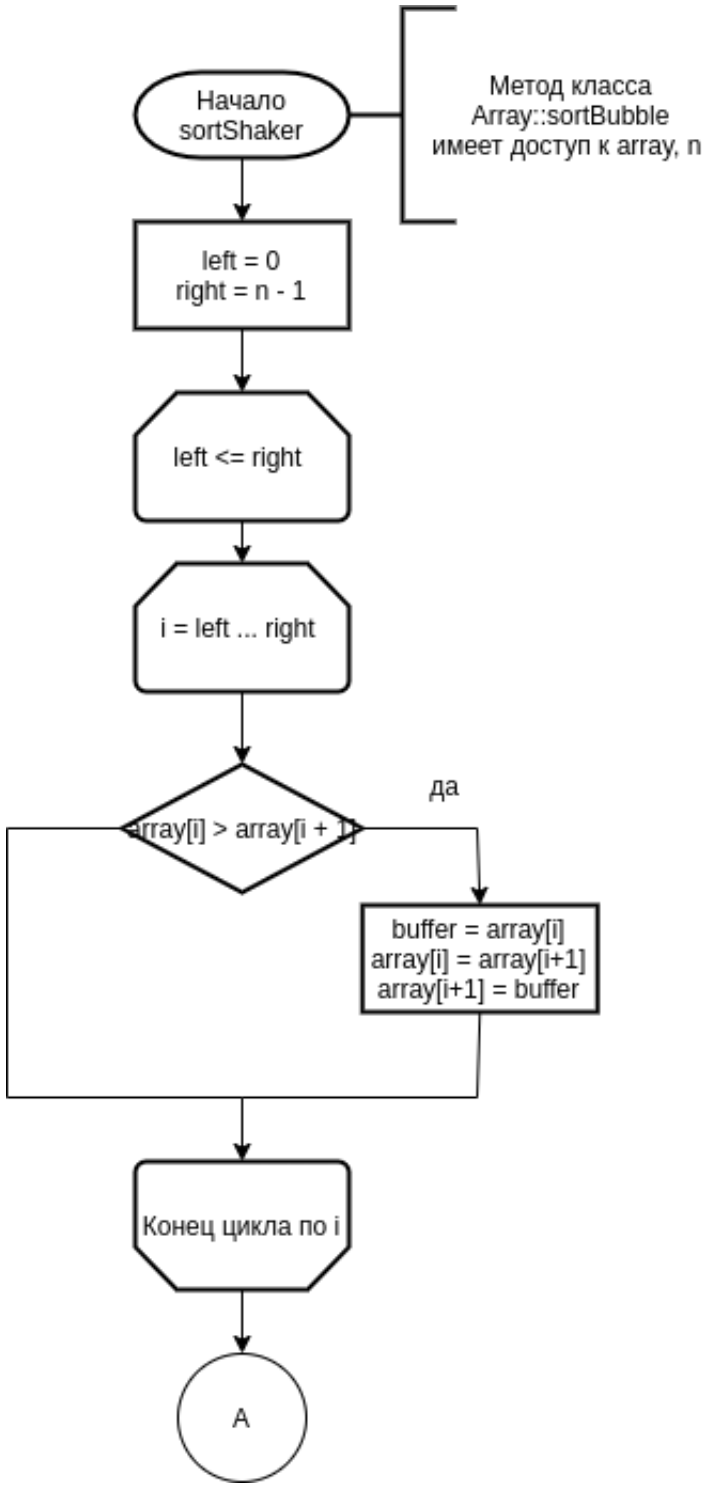

Рис. 2.3: Схема сортировки перемешиванием

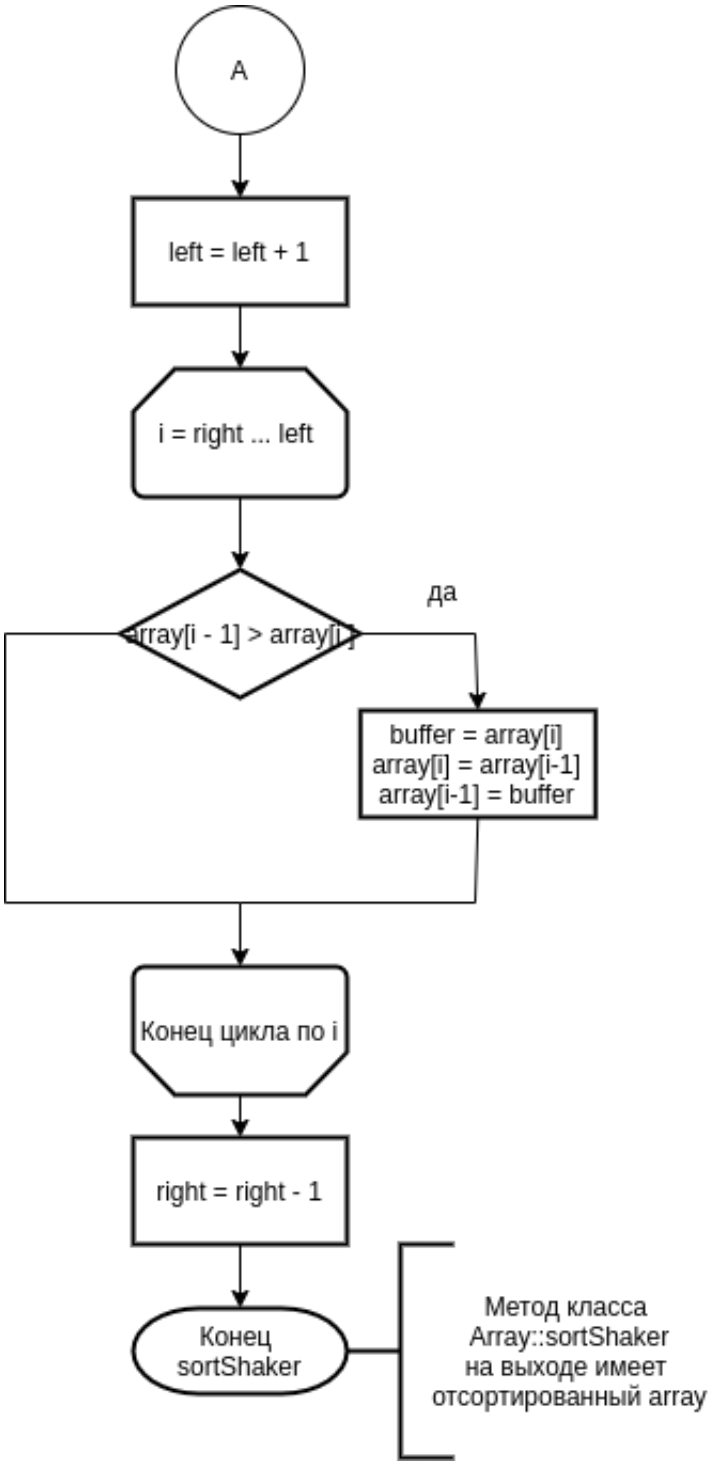

Рис. 2.4: Схема сортировки перемешиванием

2.2. Структура ПО

На рисунке 2.5 представлена диограмма классов.

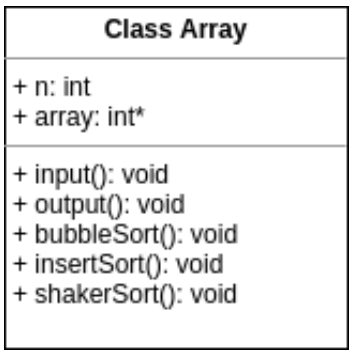

Рис. 2.5: Диаграмма классов реализуемого ПО

2.3. Тестирование

В рамках данной лабораторной работы были выделены следующие классы эквивалентности:

- входными данными является отсортированный по возрастанию массив;
- входными данными является отсортированный по убыванию массив;
- входными данными является не сортированный массив;

Для проверки работы программы будет осуществлено тестирование согласно классам эквивалентности.

2.4. Подсчет трудоемкости алгоритмов

Введем модель вычисления трудоемкости для оценки алгоритмов:

- базовые операции стоимостью 1 – $+, -, \cdot, /, =, ==, <=, >=, !=, +=, []$;
- оценка трудоёмкости цикла for от 0 до N с шагом 1 $F_{for} = 2+N\cdot(2+F_{body})$, где F_{body} – тело цикла;
- стоимость условного перехода примем за 0, стоимость вычисления условия остаётся.

Оценим трудоёмкость алгоритмов сортировки массива по коду программы.

Сортировка пузырьком

Лучший случай – $2 + 2 \cdot N + 3 \cdot N \cdot N$
Худший случай – $2 + 2 \cdot N + 6 \cdot N \cdot N$

Сортировка вставками

Лучший случай – $2 + 9 \cdot N + 7/2 \cdot N \cdot N$
Худший случай – $2 + 2 \cdot N + 11/2 \cdot N \cdot N$

Шейкерная сортировка

Лучший случай – $12 \cdot N \cdot N + 7 \cdot N + 5$
Худший случай – $28 \cdot N \cdot N + 7 \cdot N + 5$

Вывод

Таким образом, выше были сформированы требования к программе и составлены схемы алгоритмов. Также подсчитана трудоёмкость для каждого их алгоритмов.

3. Технологическая часть

В данном разделе будут реализованы функции алгоритмов сортировки массивов на языке C++.

3.1. Требования к программе

Для дальнейшего использования программы необходимо обеспечить консольный ввод размера массива, далее пользователю предостовляется выбор, ввести массив вручную или сгенерировать. С полученным массивом возможно произвести сортировку из предложенных(пузыркем, вставками и Шейкер). Также необходимо реализовать функцию подсчета процессорного времени, которое затрачивает функция.

3.2. Выбор языка программирования

В качестве языка программирования было решено выбрать C++, так как уже имеется опыт работы с библиотеками и инструментами языка, которые позволяют реализовать и провести исследования над алгоритмами сортировки массивов.

3.3. Сведенья о модулях программы

Программа состоит из:

main.cpp - главный файл программы, в котором распологается точка входа.

array.h - класс Array, который содежит алгоритмы сортировок

array.cpp - реализация методов класса Array

3.4. Реализация алгоритмов

В листингах 3.1 - 3.3 приведены реализации алгоритмов сортировки массивов на ЯП C++.

Листинг 3.1: реализация сортировки пузырьком

```
1    void Array::sortBubble()
2    {
3        for (int i = 0; i < n; i++)
4            for (int j = 0; j < n - 1; j++)
```

```
5                   if (array[j] > array[j + 1])
6                       swap(array[j], array[j + 1]);
7       }
```

Листинг 3.2: реализация сортировки вставками

```
1       void Array::sortInsert()
2       {
3           int key = 0;
4           int temp = 0;
5           for (int i = 0; i < n - 1; i++)
6           {
7               key = i + 1;
8               temp = array[key];
9               for (int j = i + 1; j > 0; j--)
10                  if (temp < array[j - 1])
11                  {
12                      array[j] = array[j - 1];
13                      key = j - 1;
14                  }
15              array[key] = temp;
16          }
17      }
```

Листинг 3.3: реализация сортировки перемешиванием

```
1       void Array::sortShaker()
2       {
3           int left = 0;
4           int right = n - 1;
5           int flag = 1;
6           while((left < right) && (flag > 0))
7           {
8               flag = 0;
9               for (int i = left; i < right; i++)
10                  if (array[i] > array[i + 1])
11                  {
```

```
12                      swap(array[i], array[i + 1]);
13                      flag = 1;
14                  }
15              right--;
16              for (int i = right; i > left; i--)
17                  if (array[i - 1] > array[i])
18                  {
19                      swap(array[i - 1], array[i]);
20                      flag = 1;
21                  }
22              left++;
23          }
24      }
```

Листинг 3.4: реализация функции обмена значений переменных (swap)

```
1           void Array::swap(int &a, int &b)
2           {
3                   int value = a;
4                   a = b;
5                   b = value;
6           }
```

Вывод

По итогу, были реализованы функции алгоритмов сортировки массивов на языке C++.

4. Исследовательская часть

В данном разделе сравним работу каждого алгоритма.

4.1. Примеры работы

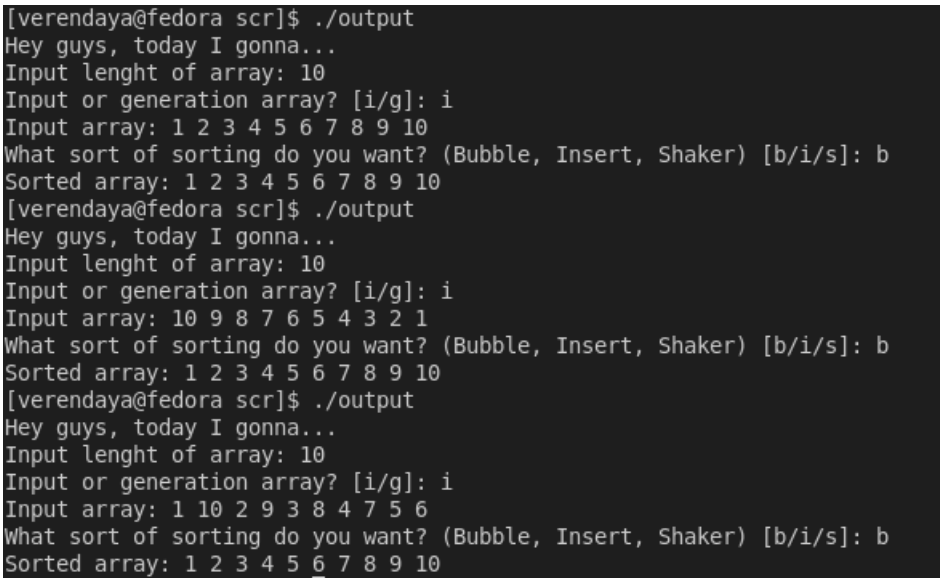

Рис. 4.1: Пример работы алгоритма пузырьком(BubbleSort).

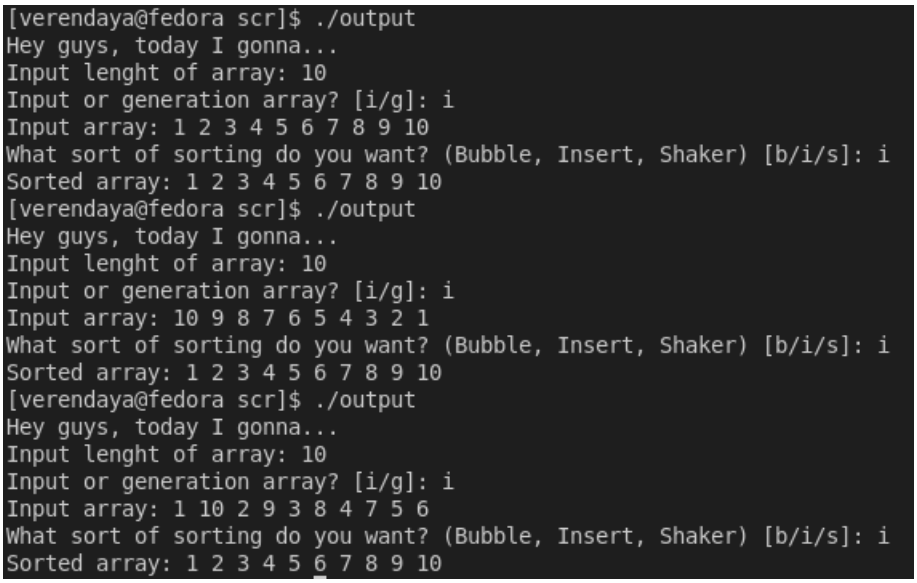

Рис. 4.2: Пример работы алгоритма вставками(InsertSort).

```
[verendaya@fedora scr]$ ./output
Hey guys, today I gonna...
Input lenght of array: 10
Input or generation array? [i/g]: i
Input array: 1 2 3 4 5 6 7 8 9 10
What sort of sorting do you want? (Bubble, Insert, Shaker) [b/i/s]: s
Sorted array: 1 2 3 4 5 6 7 8 9 10
[verendaya@fedora scr]$ ./output
Hey guys, today I gonna...
Input lenght of array: 10
Input or generation array? [i/g]: i
Input array: 10 9 8 7 6 5 4 3 2 1
What sort of sorting do you want? (Bubble, Insert, Shaker) [b/i/s]: s
Sorted array: 1 2 3 4 5 6 7 8 9 10
[verendaya@fedora scr]$ ./output
Hey guys, today I gonna...
Input lenght of array: 10
Input or generation array? [i/g]: i
Input array: 1 10 2 9 3 8 4 7 5 6
What sort of sorting do you want? (Bubble, Insert, Shaker) [b/i/s]: s
Sorted array: 1 2 3 4 5 6 7 8 9 10
```

Рис. 4.3: Пример работы алгоритма перемешиванием(ShakerSort).

4.2. Оценка затрачиваемого времени

Далее будут приведены графики сравнения работы алгоритмов для каждого класса эквивалентности.

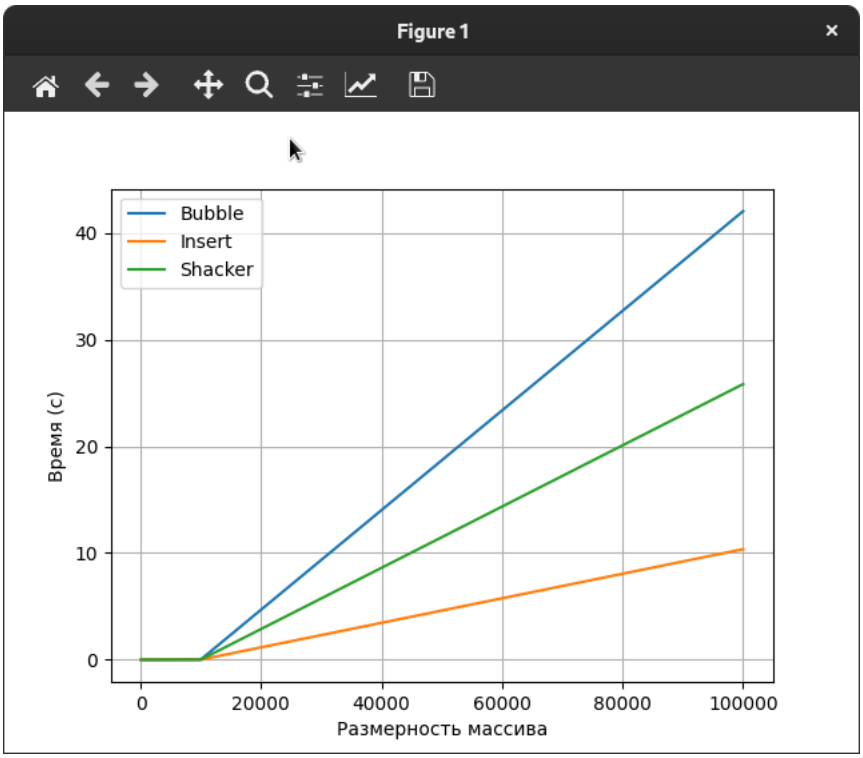

Рис. 4.4: Сравнение работы алгоритмов на не отсортированных массивах.

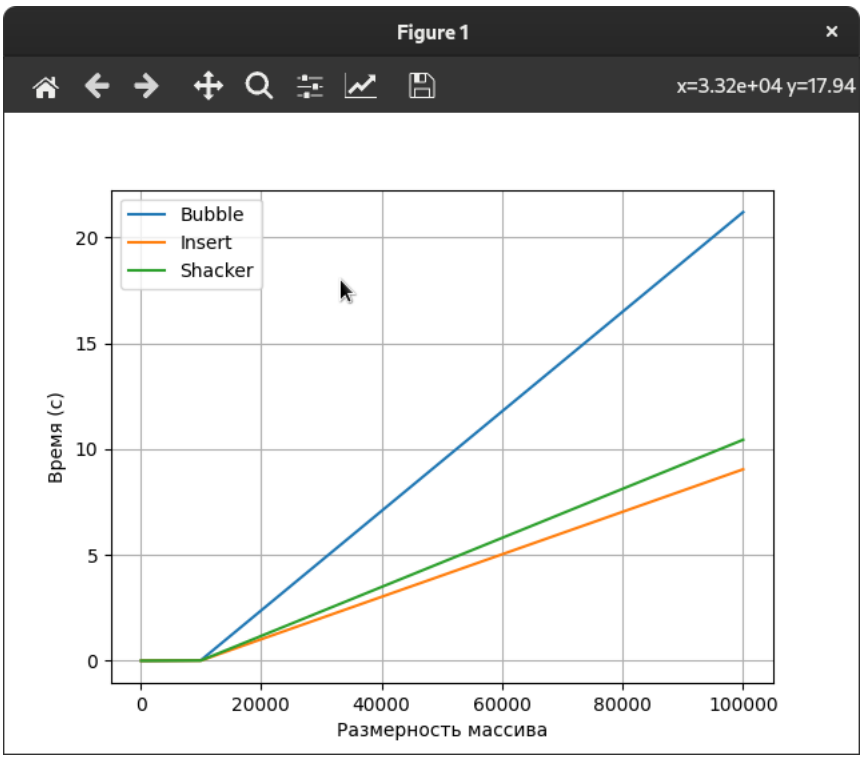

Рис. 4.5: Сравнение работы алгоритмов на отсортированных массивах по возростанию.

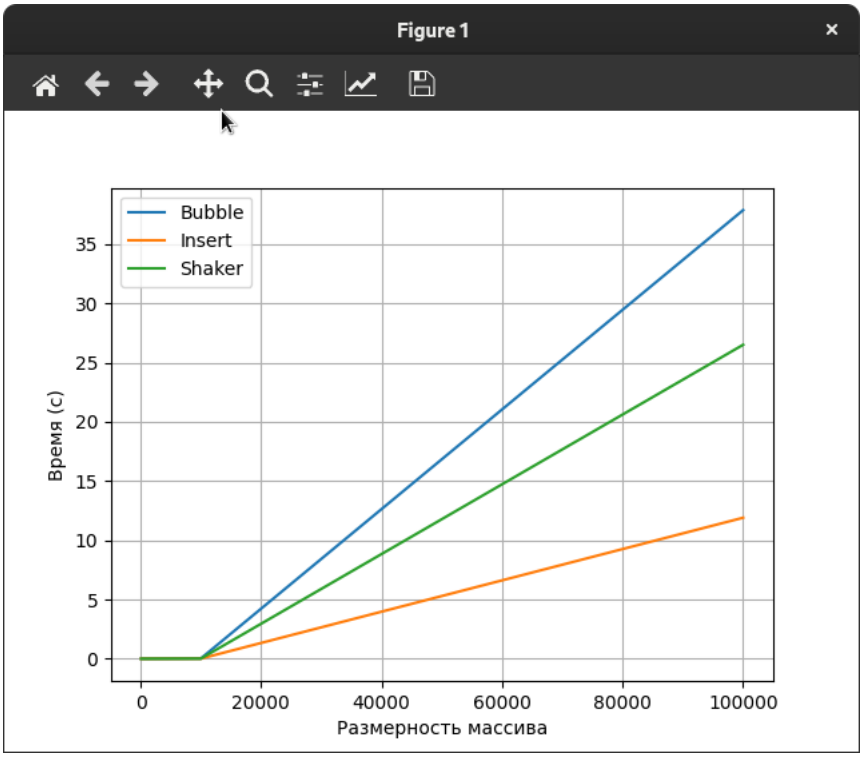

Рис. 4.6: Сравнение работы алгоритмов на отсортированных массивах по убыванию.

Вывод

Анализируя результаты замеров затрачиваемого времени можно сказать, что самым быстрым алгоритмом из представленных, при использова-

нии случайного заполнения, оказался алгоритм сортировки вставками, а алгоритм сортировки пузырьком и шейкерная сортировка являются разновидностями пузырьковой сортировки. Однако шейкерная сортировка работает в два раза быстрее пузырька.

Заключение

В ходе работы были изучены алгоритмы сортировки массивов. Реализованы 3 алгоритма, приведен программный код реализации алгоритмов сортировки. Была подсчитана трудоемкость каждого из алгоритмов, а также проведено сравнение алгоритмов по времени и трудоемкости. Показано, что наименее трудоемким и наименее затратным по времени алгоритмом является алгоритм сортировки вставками.

Цель работы достигнута, решены поставленные задачи. Получены практические навыки реализации алгоритмов сортировки массивов, а также проведена исследовательская работа по вычислению трудоемкости алгоритмов и анализу их временных характеристик.

Список литературы

1. Дж. Макконнел. Анализ алгоритмов. Активный обучающий подход. – М.: Техносфера, 2017. – 267с.

2. Шагбазян, Д.В. Алгоритмы сортировки. Анализ, реализация, применение: учебное пособие / Д.В. Шагбазян, А.А. Штанюк, Е.В. Малкина. – Нижний Новгород: Нижегородский госуниверситет, 2019. – 42 с.

3. Основы программирования на языках Си и C++ для начинающих[Электронный ресурс]. Режим доступа: http://cppstudio.com/ (дата обращения 10.10.2021)

4. LINUX.ORG.RU - Русскоязычная информация о ОС Linux[Электронный ресурс] Режим доступа://www.linux.org.ru/(дата обращения 25.10.2021)

5. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с. — ISBN 5-8459-0082-1.