



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

## О Т Ч Е Т

по лабораторной работе № 3

Название: Анализ алгоритмов сортировки массива

Дисциплина: Анализ алгоритмов

Студент

ИУ7-52Б

(Группа)

Сучков А.Д.

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Волкова Л.Л.

(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

# Оглавление

<b>Введение</b>	<b>3</b>
<b>1 Аналитическая часть</b>	<b>4</b>
1.1 Сортировка пузырьком . . . . .	4
1.2 Сортировка вставками . . . . .	4
1.3 Шейкерная сортировка . . . . .	5
1.4 Вывод . . . . .	5
<b>2 Конструкторская часть</b>	<b>6</b>
2.1 Требования к программе . . . . .	6
2.2 Схемы алгоритмов . . . . .	6
2.3 Подсчёт трудоёмкости алгоритмов . . . . .	6
2.4 Вывод . . . . .	7
<b>3 Технологическая часть</b>	<b>12</b>
3.1 Выбор языка программирования . . . . .	12
3.2 Реализация алгоритмов . . . . .	12
3.3 Оценка затрачиваемого времени . . . . .	14
3.4 Тестирование алгоритмов . . . . .	16
3.5 Вывод . . . . .	17
<b>4 Исследовательская часть</b>	<b>18</b>
4.1 Результаты экспериментов . . . . .	18
4.2 Вывод . . . . .	18
<b>Заключение</b>	<b>19</b>
<b>Список литературы</b>	<b>20</b>

## Введение

**Сортировка массива** – одна из самых популярных и востребованных операций над массивами. Массив – структура данных, которая хранит набор значений (элементов массива), идентифицируемых по индексу или набору индексов, принимающих целые (или приводимые к целому) значения из некоторого заданного непрерывного диапазона.

Алгоритмы сортировок реализуют упорядочивание элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, будет называться ключом сортировки. На практике в качестве ключа часто выступает число, в то время как остальные поля не участвуют в работе алгоритма.

С каждым годом объёмы хранилищ данных стремительно увеличиваются и поэтому от алгоритмов сортировок требуется максимальная производительность.

## 1. Аналитическая часть

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

Можно выделить следующие задачи лабораторной работы:

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

### 1.1. Сортировка пузырьком

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются  $N-1$  раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает – массив отсортирован.

При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма).

### 1.2. Сортировка вставками

Алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. [2]

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

### **1.3. Шейкерная сортировка**

Шейкерная сортировка – разновидность пузырьковой сортировки. [1] Анализируя метод пузырьковой сортировки, можно отметить два обстоятельства. Во-первых, если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, её можно исключить из рассмотрения. Во-вторых, при движении от конца массива к началу минимальный элемент «всплывает» на первую позицию, а максимальный элемент сдвигается только на одну позицию вправо. Эти две идеи приводят к следующим модификациям в методе пузырьковой сортировки.

Границы рабочей части массива (то есть части массива, где происходит движение) устанавливаются в месте последнего обмена на каждой итерации. Массив просматривается поочередно справа налево и слева направо.

### **1.4. Вывод**

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

## 2. Конструкторская часть

### 2.1. Требования к программе

Для дальнейшего тестирования программы необходимо обеспечить консольный ввод размера массива и его элементов, а также обеспечить выбор алгоритма сортировки. На выходе должны получить отсортированный массив. Также необходимо реализовать функцию подсчёта процессорного времени, которое могут затрачивать функции.

### 2.2. Схемы алгоритмов

На рисунках 2.1 - 2.5 приведены схемы алгоритмов сортировки массива.

### 2.3. Подсчёт трудоёмкости алгоритмов

Введём модель вычисления трудоёмкости 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 –  $28 \cdot N \cdot N + 7 \cdot N + 5$

## **2.4. Вывод**

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

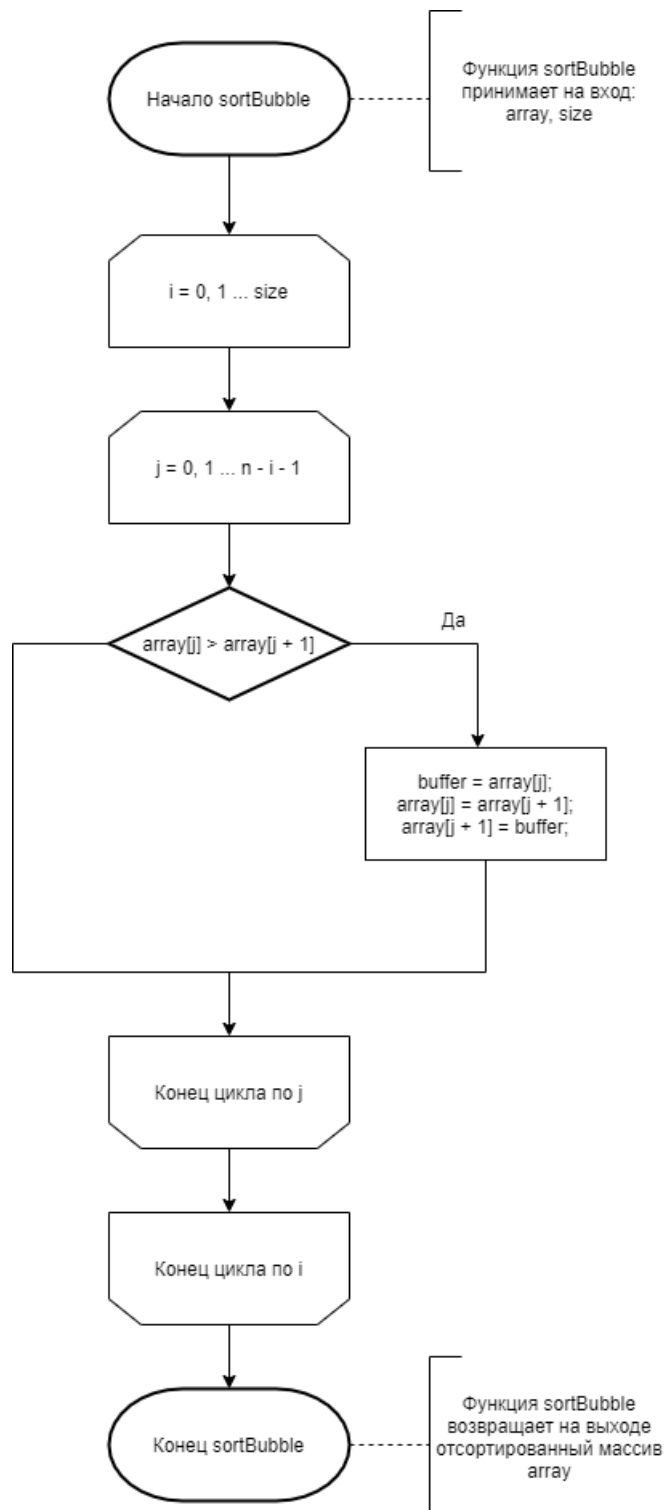


Рис. 2.1: Схема сортировки пузырьком



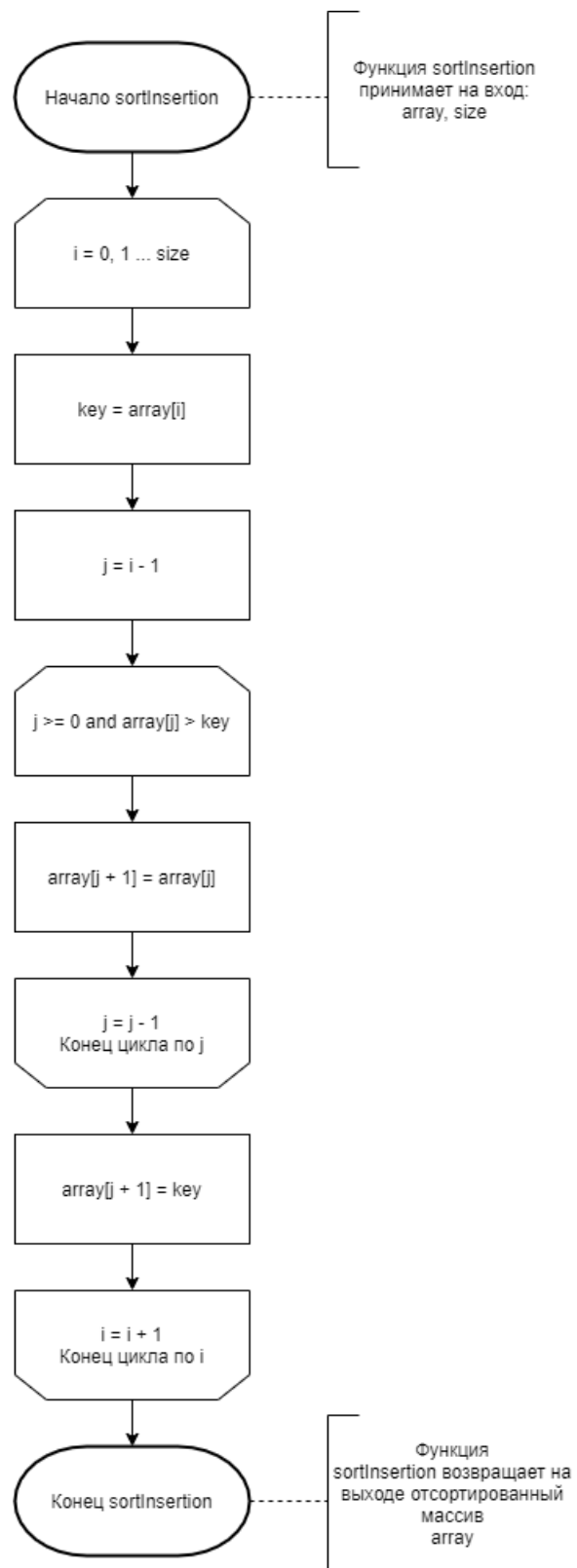


Рис. 2.2: Схема сортировки вставками

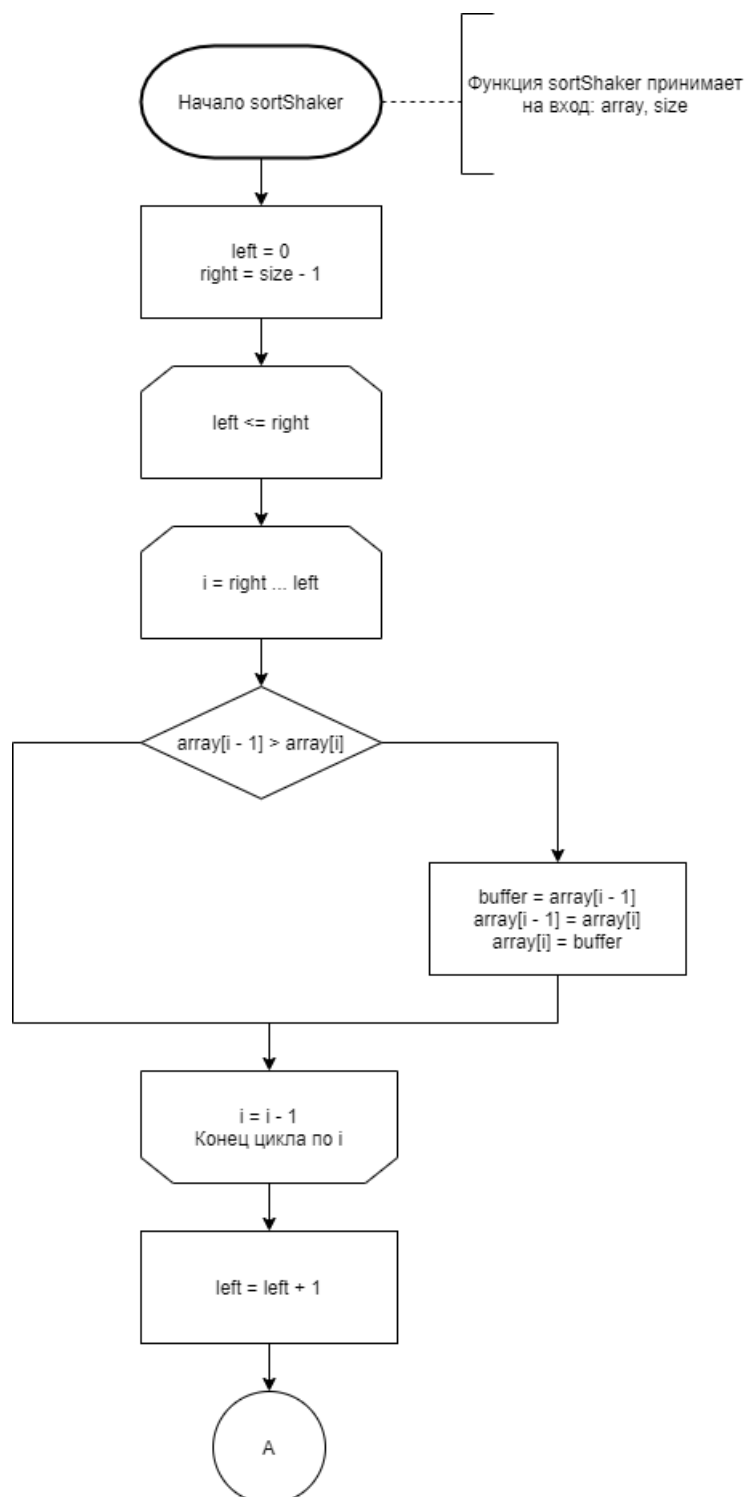


Рис. 2.3: Схема шейкерной сортировки, часть 1

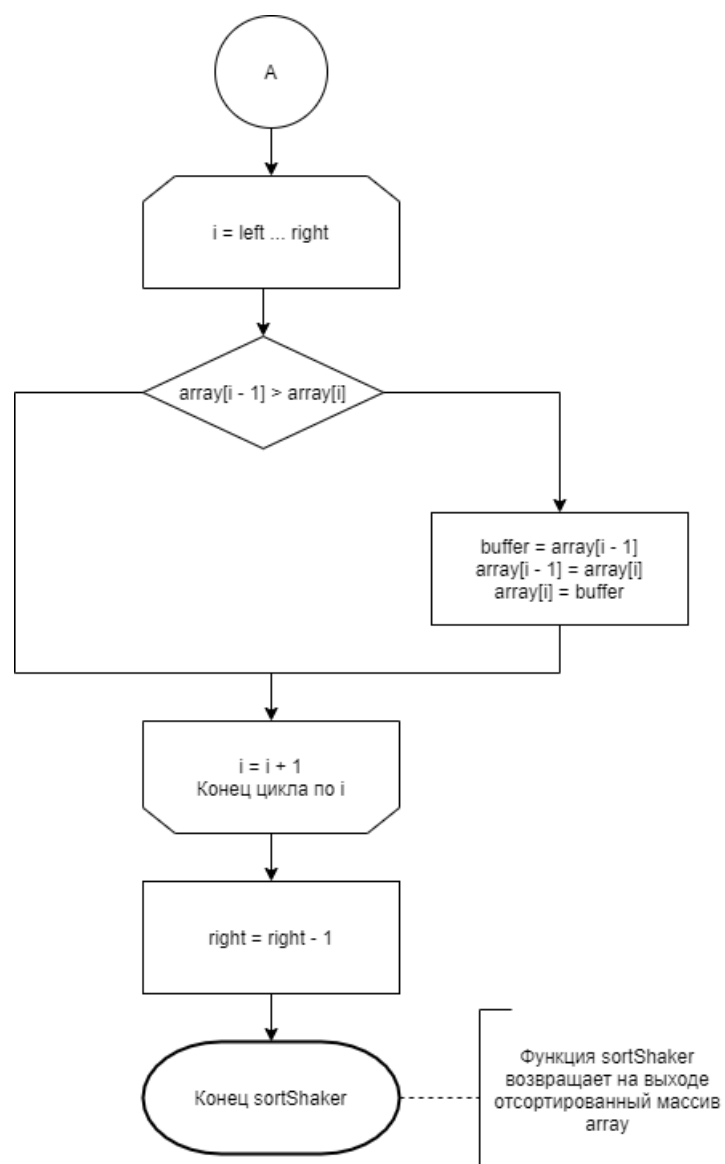


Рис. 2.4: Схема шейкерной сортировки, часть 2

## 3. Технологическая часть

### 3.1. 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.

### 3.2. Реализация алгоритмов

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

Листинг 3.1: реализация сортировки пузырьком

```
1 void sortBubble(arrayType& array, int size)
2 {
3     for (int i = 0; i < size; i++)
4     {
5         for (int j = 0; j < size - i - 1; j++)
6         {
7             if (array[j] > array[j + 1])
8                 sortSwap(array[j], array[j + 1]);
9         }
10    }
11 }
```

Листинг 3.2: реализация сортировки вставками

```
1 void sortInsertion(arrayType& array, int size)
2 {
3     int key;
4     int j;
5
6     for (int i = 1; i < size; i++)
7     {
8         key = array[i];
```

```

9
10         for (j = i - 1; j >= 0 && array[j] > key; j--)
11             array[j + 1] = array[j];
12
13         array[j + 1] = key;
14     }
15 }

```

Листинг 3.3: реализация сортировки вставками

```

1 void sortShaker(arrayType& array, int size)
2 {
3     int left = 1;
4     int right = size - 1;
5
6     while (left <= right)
7     {
8         for (int i = right; i >= left; i--)
9         {
10             if (array[i - 1] > array[i])
11                 sortSwap(array[i - 1], array[i]);
12         }
13
14         left++;
15
16         for (int i = left; i <= right; i++)
17         {
18             if (array[i - 1] > array[i])
19                 sortSwap(array[i - 1], array[i]);
20         }
21
22         right--;
23     }
24 }

```

### 3.3. Оценка затрачиваемого времени

Для замера процессорного времени выполнения алгоритмов используется библиотека `windows.h` и были написаны функции (листинг 3.4). Создаются массивы с размерностями 100, 200, 300, 1000, 10000, которые заполняются случайными числами с помощью функции генерации в листинге 3.5. Замеры проводились 100, 100, 100, 10, 1 раз соответственно размерам массивов с помощью функции в листинге 3.6.

Листинг 3.4: функции замера процессорного времени

```
1 double PCFreq = 0.0;
2 __int64 CounterStart = 0;
3
4 void StartCounter ()
5 {
6     LARGE_INTEGER li ;
7     if (!QueryPerformanceFrequency(&li ))
8         std::cout << "QueryPerformanceFrequency failed!\n";
9
10    PCFreq = double(li.QuadPart)/1000.0;
11
12    QueryPerformanceCounter(&li );
13    CounterStart = li.QuadPart;
14 }
15
16 double GetCounter ()
17 {
18     LARGE_INTEGER li ;
19     QueryPerformanceCounter(&li );
20     return double(li.QuadPart-CounterStart)/PCFreq;
21 }
```

Листинг 3.5: функция генерации массива со случайным наполнением

```
1 arrayType generateArray(int size)
2 {
3     arrayType newArray = createArray(size);
```

```

4
5     for (int i = 0; i < size; i++)
6     {
7         newArray[i] = rand() % 100 + 0;
8     }
9
10    return newArray;
11 }

```

Листинг 3.6: функция тестирования алгоритмов

```

1 void beginTimeTest()
2 {
3     int countSizes = 5;
4     int sizes [] = {100, 200, 300, 1000, 10000};
5     int count [] = {100, 100, 100, 10, 1};
6
7     void (*algorithms [])(arrayType&, int) = {sortBubble,
8                                                sortInsertion, sortShaker};
9
10    char algorithmNames[][100] = {"Bubble sort",
11                                   "Insert sort", "Shaker sort"};
12
13    for (int k = 0; k < 3; k++)
14    {
15        void (*sort)(arrayType&, int) = algorithms[k];
16
17        for (int i = 0; i < countSizes; i++)
18        {
19
20            arrayType testArray = generateArray(sizes[i]);
21
22            double finishTime = 0;
23
24            for (int j = 0; j < count[i]; j++)
25            {

```

```

26         StartCounter ();
27         sort (testArray ,  sizes [ i ] );
28         finishTime += GetCounter ();
29     }
30
31     finishTime /= count [ i ];
32
33     std :: cout << "\nFor " << algorithmNames [ k ]
34         << "\t→ Size - " << sizes [ i ] <<
35         "\tTime - " << finishTime ;
36
37     deleteArray ( testArray );
38 }
39
40 std :: cout << "\n_____";
41 }
42 }

```

### 3.4. Тестирование алгоритмов

На рисунках 3.1 - 3.3 приведены скриншоты интерфейса программы и тестирования, которые проводились в ручную.

```

>> Choose method for sort:
>> [1] - Bubble sort
>> [2] - Insertion sort
>> [3] - Shaker sort
>> [4] - Time test

>> [0] - Exit

>> You're choice: 1

>> Input array length: 5
>> Input elements by Enter:
0
0
3
1
0
>> Array: 0 0 0 1 3

```

Рис. 3.1: Тестирование сортировки пузырьком

Все тесты прошли успешно.



```
>> Choose method for sort:
>> [1] - Bubble sort
>> [2] - Insertion sort
>> [3] - Shaker sort
>> [4] - Time test

>> [0] - Exit

>> You're choice: 2

>> Input array length: 4
>> Input elements by Enter:
10
10
1
1
>> Array: 1 1 10 10
```

Рис. 3.2: Тестирование сортировки вставками

```
>> Choose method for sort:
>> [1] - Bubble sort
>> [2] - Insertion sort
>> [3] - Shaker sort
>> [4] - Time test

>> [0] - Exit

>> You're choice: 3

>> Input array length: 6
>> Input elements by Enter:
1
2
3
4
5
6
>> Array: 1 2 3 4 5 6
```

Рис. 3.3: Тестирование шейкерной сортировки

### 3.5. Вывод

По итогу, были реализованы функции алгоритмов сортировки массивов на языке C++, а также функции тестирования и подсчёта процессорного времени.

## 4. Исследовательская часть

Измерения процессорного времени проводятся при одинаковых размерах массивов 100, 200, 300, 1000, 10000.

### 4.1. Результаты экспериментов

Проведя измерения процессорного времени выполнения реализованных алгоритмов, можно составить таблицу 4.1.

Таблица 4.1: Результаты замеров процессорного времени выполнения алгоритмов сортировки в секундах

Название \ Размер	100	200	300	1000	10000
Сорт. пузырьком	$3.1 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$	$3.1 \cdot 10^{-4}$	$2.6 \cdot 10^{-3}$	0.341
Сорт. вставками	$8.1 \cdot 10^{-6}$	$5.9 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$	$7.0 \cdot 10^{-4}$	0.071
Шейкерная сорт.	$4.3 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$	$2.5 \cdot 10^{-3}$	0.309

### 4.2. Вывод

Анализируя результаты замеров затрачиваемого времени можно сказать, что самым быстрым алгоритмом, при использовании случайного заполнения, оказался алгоритм сортировки вставками, а алгоритм сортировки пузырьком и шейкерная сортировка являются разновидностями пузырьковой сортировки, следовательно результаты похожи. Однако на массивах большей длины выигрывает шейкерная сортировка.

## 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. Макконнелл. Анализ алгоритмов. Активный обучающий подход. – М.: Техносфера, 2017. – 267с.
2. Кормен Т. Алгоритмы: построение и анализ / Кормен Т. - Вильямс, 2014. - 198 с. - 219 с.