**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»

# Отчёт

## по лабораторной работе №3

**Название:** Алгоритмы сортировок

**Дисциплина:** Анализ алгоритмов

Студент ИУ7-55Б (Группа) _____ (Подпись, дата) И. Е. Афимин (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) Л.Л. Волкова (И.О. Фамилия)

*Москва, 2020*

# Содержание

# Введение

Алгоритм сортировки – это алгоритм для упорядочивания элементов в списке. Входом является последовательность из n элементов: $a_1, a_2, \ldots, a_n$ Результатом работы алгоритма сортировки является перестановка исходной последовательности $a'_1, a'_2, \ldots, a'_n$, такая что $a_1 \leqslant a_2 \leqslant \cdots \leqslant a_n$, где $\leqslant$ – отношение порядка на множестве элементов списка. Поля, служащие критерием порядка, называются ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

В данной работе рассматриваются три алгоритма:

1) сортировка пузырьком с флагом;
2) сортировка вставками;
3) сортировка выбором.

Целью данной лабораторной работы является реализация алгоритмов сортировки и исследование их трудоемкости.

Задачи данной лабораторной работы:

1) изучить алгоритмы сортировки пузырьком с флагом, вставками, выбором;
2) реализовать алгоритмы сортировки пузырьком с флагом, вставками, выбором;
3) дать оценку трудоёмкости в лучшем, произвольном и худшем случае (для двух алгоритмов сделать вывод трудоёмкости);
4) провести замеры процессорного времени работы для лучшего, худшего и произвольного случая.

# 1 Аналитический раздел

В данном разделе будут рассмотрены основные теоритические понятия алгоритмов сортировок пузырьком с флагом, вставками, выбором.

## 1.1 Алгоритмы сортировок

Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке [3] .

### 1.1.1 Алгоритм сортировки пузырьком с флагом

Алгоритм сортировки пузырьком или метод простых обменов имеет следующий принцип работы:

1) прохождение по всему массиву;
2) сравнивание между собой пар соседних ячеек;
3) если при сравнении оказывается, что значение ячейки i больше, чем значение ячейки i + 1, то нужно поменять значения этих ячеек местами.

Алгоритм сортировки пузырьком с флагом является модификацией этого алгоритма. Идея состоит в том, что если при выполнении прохода методом пузырька не было ни одного обмена элементов массива, то это означает, что массив уже отсортирован и остальные проходы не требуются.

### 1.1.2 Алгоритм сортировки вставками

Алгоритм сортировки вставками, просматривает элементы входной последовательности по одному, и для каждого элемента, размещает его в подходящее место среди ранее упорядоченных элементов.

В начальный момент времени отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается в нужную позицию в отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан.

Сортировка методом вставок – простой алгоритм сортировки. Хотя этот метод сортировки намного менее эффективен, чем более сложные алгоритмы (такие как быстрая сортировка), у него есть ряд преимуществ

1) простота реализации;
2) эффективен на небольших наборах данных;
3) эффективен на частично отсортированных последовательностях;
4) является устойчивым алгоритмом (не меняет порядок элементов, которые уже отсортированы).

### 1.1.3 Алгоритм сортировки выбором

Алгоритм сортировки выбором работает следующим образом: находим наименьший элемент в массиве и обмениваем его с элементом находящимся на первом месте. Повторяем процесс – находим наименьший элемент в последовательности, начиная со второго элемента, и обмениваем со вторым элементном и так далее, пока весь массив не будет отсортирован. Этот метод называется сортировка выбором, поскольку он работает, циклически выбирая наименьший из оставшихся элементов.

Главным отличием сортировки выбором от сортировки вставками является, то что в сортировке вставками извлекается из неотсортированной части массива первый элемент (не обязательно минимальный) и вставляется на своё место в отсортированной части. В отличии от сортировки выбором, где ищется минимальный элемент в неотсортированной части, который вставляется в конец отсортированной части массива.

## 1.2 Вывод

Были рассмотрены алгоритмы сортировки пузырьком, вставками и выбором. Каждый имеет свою особенность, а конкретно сложность работы в лучшем/худшем случаях.

# 2 Конструкторская часть

В данном разделе будут рассмотрены требования к функциональности ПО, схемы алгоритмов и определены способы тестирования.

## 2.1 Требования к функциональности ПО

В данной работе требуется обеспечить следующую минимальную функциональность консольного приложения.

1) возможность подать на вход массив;
2) при массиве нулевой длины программа не должна аварийно завершаться;
3) корректная сортировка;
4) обеспечить вывод замеров времени работы каждого из алгоритмов в худшем, лучшем и произвольном случаях.

## 2.2 Тесты

Тестирование ПО будет проводиться методом чёрного ящика. Необходимо проверить работу системы на массивах различных длин.

## 2.3 Схемы алгоритмов

Ниже будут представлены схемы алгоритмов нахождениея произведения матриц:

1) Схема сортировки пузырьком (рисунок 2.1);
2) Схема сортировки вставками (рисунок 2.2);
3) Схема сортировки выбором (рисунок 2.3);

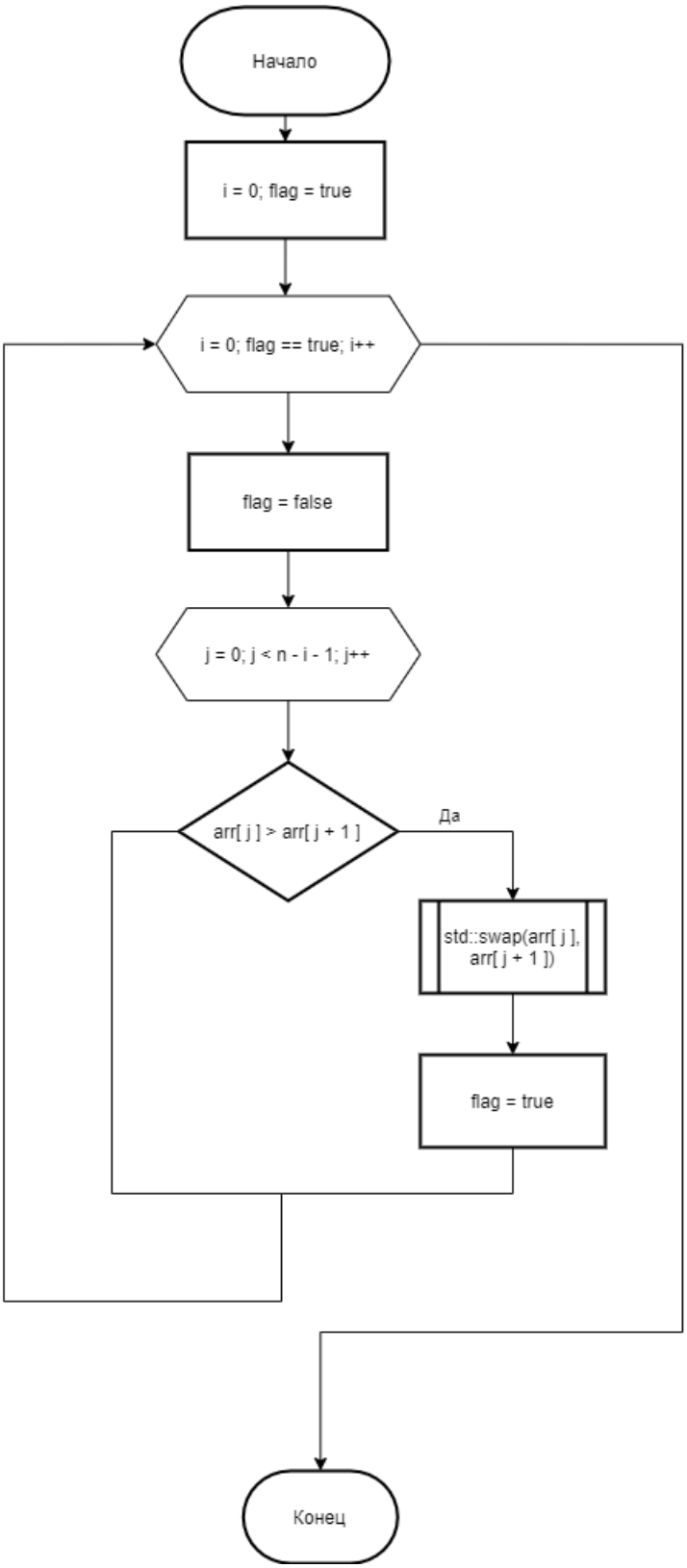


Рисунок 2.1 — Схема сортировки пузырьком

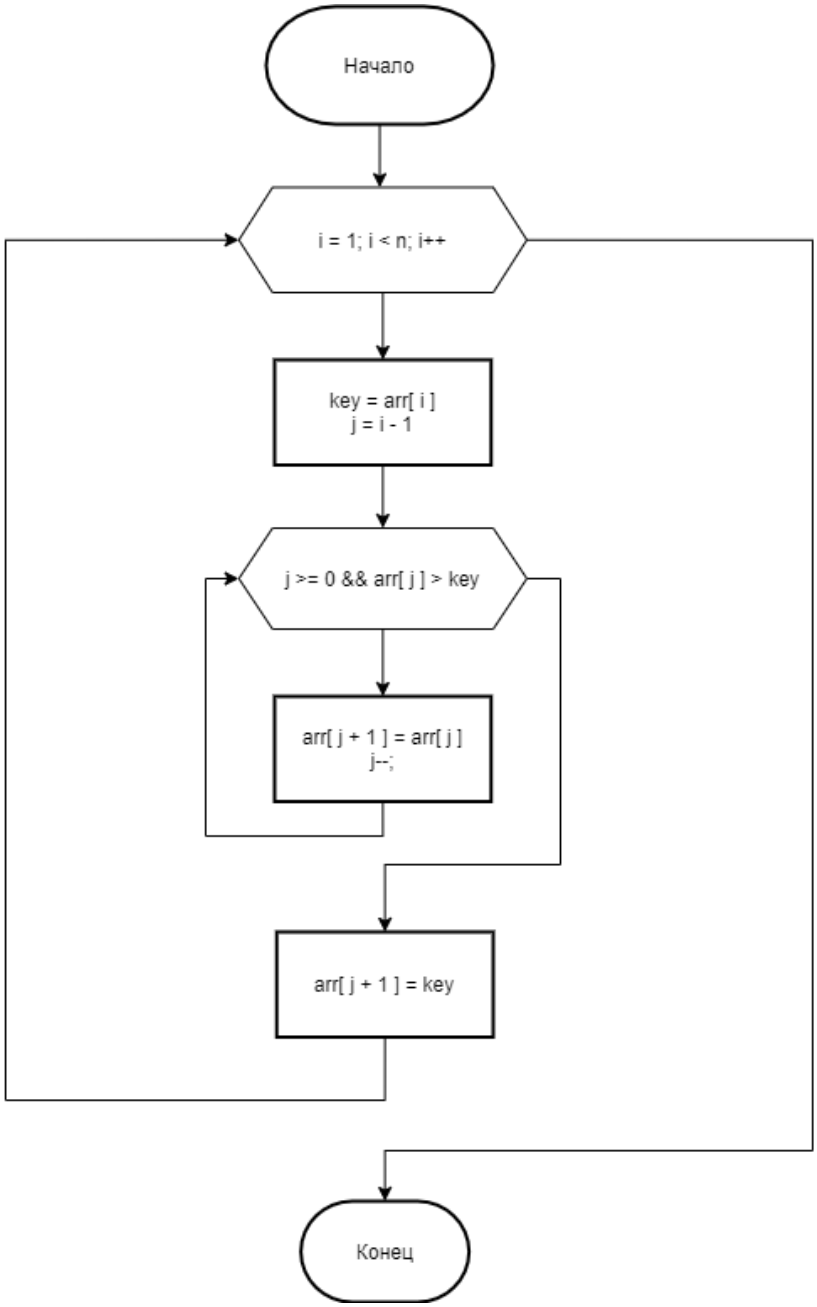


Рисунок 2.2 — Схема сортировки вставками

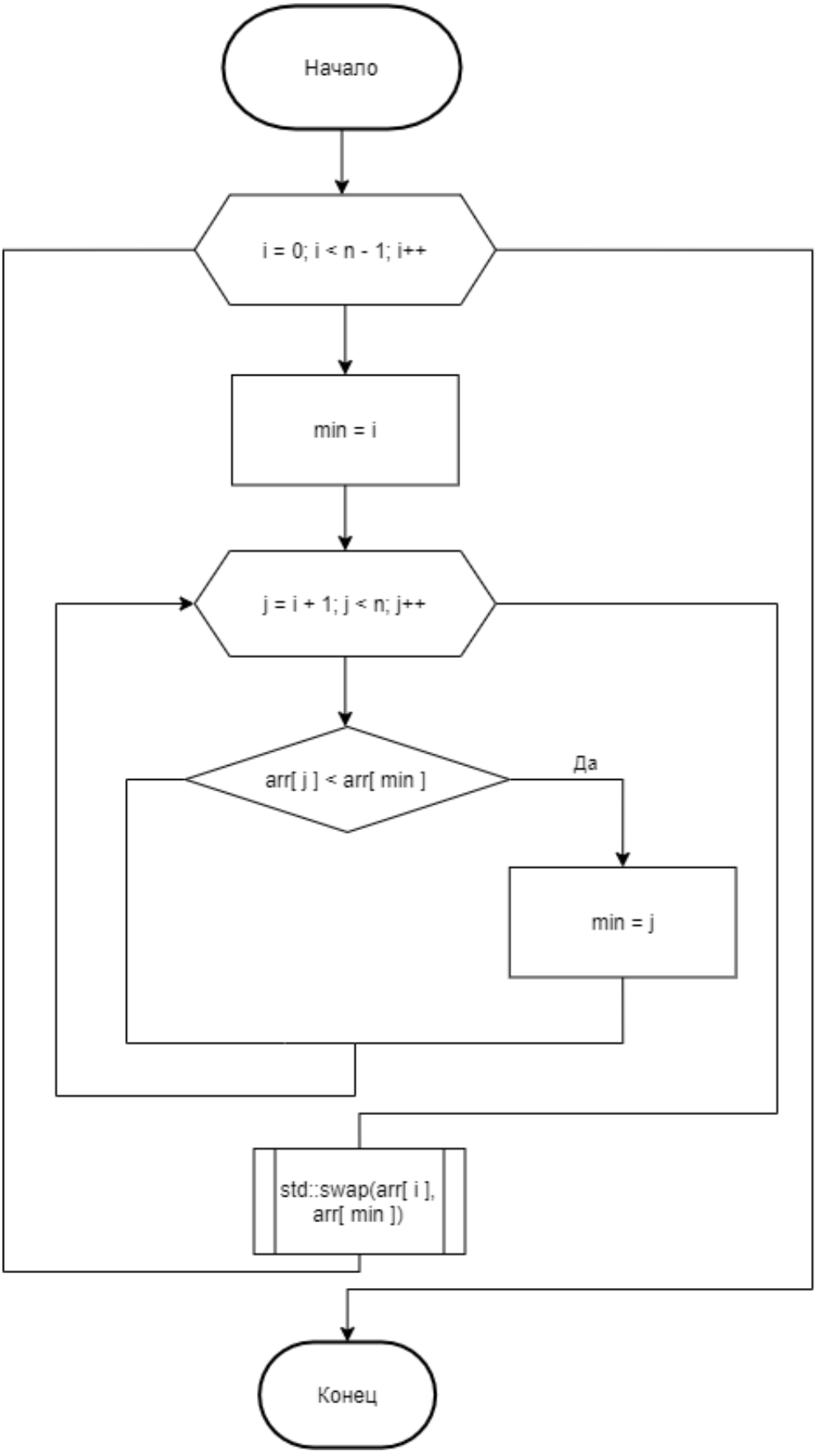


Рисунок 2.3 — Схема сортировки выбором

### 2.4 Трудоёмкость алгоритма

Трудоёмкость – количество работы, которую алгоритм затрачивает на обработку данных. Является функцией от длины входов алгоритма и позволяет оценить количество работы.

Введём модель вычисления трудоёмкости.

#### 2.4.1 Базовые операции

Ниже представлены базовые операции, стоимость которых единична:

1) $=, +, + =, -, - =, *, * =, /, / =, ++, --, \%$,
2) $<, \leqslant, ==, \neq, \geqslant, >$,
3) [ ].

#### 2.4.2 Условный оператор

```
if (условие) {
// тело A
}
else {
// тело B
}
```

Пусть трудоёмкость тела A равна $f_A$, а тела B $f_B$, тогда стоимость условного оператора можно найти по формуле (2.1):

$$f_{if} = f_{\text{условия}} + \begin{cases} min(f_A, f_B) - \text{лучший случай}, \\ max(f_A, f_B) - \text{худший случай} \end{cases} \tag{2.1}$$

#### 2.4.3 Цикл со счётчиком

```
for (int i = 0; i < n; i++) {
// тело цикла
}
```

Начальная инициализация цикла (int i = 0) выполняется один раз. Условие i < n проверяется перед каждой итерацией цикла и при входе в цикл – n + 1 операций. Тело цикла выполняется ровно n раз. Счётчик (i++) выполняется на каждой итерации, перед проверкой условия, т.е. n раз. Тогда, если трудоёмкость тела цикла равна $f$, трудоёмкость всего цикла определяется формулой (2.2)

$$f_{\text{цикла}} = 2 + n(2 + f) \tag{2.2}$$

### 2.4.4 Алгоритм сортировки пузырьком с флагом

**Лучший случай:** Массив отсортирован; не произошло ни одного обмена за 1 проход -> выходим из цикла

Трудоемкость:

$$f = 1 + 2 + (2 + 1 + 2 + (N - 1)(2 + 3) + 2) = 5N + 8 = O(N) \tag{2.3}$$

**Худший случай:** Массив отсортирован в обратном порядке; в каждом случае происходил обмен

Трудоемкость:

$$f = 1 + 2 + \sum_{i=1}^{N}(2 + 1 + 2 + (N - i)(2 + 4) + 1) = 3N^2 + 3N + 3 = O(N^2) \tag{2.4}$$

### 2.4.5 Алгоритм сортировки вставками

**Лучший случай:** отсортированный массив. При этом все внутренние циклы состоят всего из одной итерации.

Трудоемкость:

$$f = 1 + 2 + (N - 1)(2 + 3 + 2 + 2 + 1) = 10N - 7 = O \tag{2.5}$$

**Худший случай:** массив отсортирован в обратном нужному порядке. Каждый новый элемент сравнивается со всеми в отсортированной последовательности. Все внутренние циклы будут состоять из j итераций.

Трудоемкость:

$$f = 1 + 2 + (N - 1)(2 + 3 + \sum_{i=1}^{N-1}(2 + 2 + 1 + 4 + 1) + 1) = 10N^2 - 14N + 7 = O(N^2) \tag{2.6}$$

### 2.4.6 Алгоритм сортировки выбором

Трудоёмкость сортировки выбором в худшем и лучшем случаях совпадает и оценивается как $O(N^2)$.

## 2.5 Вывод

Сортировка пузырьком: лучший - $O(n)$, худший - $O(n^2)$

Сортировка вставками: лучший - $O(n)$, худший - $O(n^2)$

Сортровка выбором: лучший - $O(n^2)$, худший - $O(n^2)$

# 3 Технологическая часть

В данном разделе будут выбраны средства реплизации ПО, представлен листинг кода и будет показан метод тестирования.

## 3.1 Средства реализации

Для реализации программ я выбрал язык программирования C++, так как имею большой опыт работы с ним. Для замера процессорного времени была использована ассемблерная вставка [1].

## 3.2 Сведения о модулях программы

Программа состоит из:

— main.cpp - главный файл программы, в котором располагается точка входа в программу.
— main_test.cpp - файл с функциями тестирования и замера времени.
— sort_algorithms.cs - файл с реализациями 3-х алгоритмов сортировки.

## 3.3 Листинг программы

Ниже представлены листинги кода поиска растояния Левенштейна:

1) сортировка пузырьком (листинг 3.1);
2) сортировка вставками (листинг 3.2);
3) сортировка выбором (листинг 3.3).

Листинг 3.1 — Сортировка "Пузырьком"

```
1  void Bubble_Sort(IArr_t arr, int n)
2  {
3      int i = 0;
4      bool flag = true;
5
6      while (flag)
7      {
8          flag = false;
9          for (int j = 0; j < n - i - 1; j++)
10         {
11             if (arr[j] > arr[j + 1])
12             {
13                 std::swap(arr[j], arr[j + 1]);
14                 flag = true;
15             }
16         }
17
18         i++;
```

```
19      }
20  }
```

Листинг 3.2 — Сортировка вставками

```
1  void InserionSort(IArr_t arr, int n)
2  {
3      int key, j;
4      for (int i = 1; i < n; i++)
5      {
6          key = arr[i];
7          j = i - 1;
8
9          while (j >= 0 && arr[j] > key)
10         {
11             arr[j + 1] = arr[j];
12             j--;
13         }
14
15         arr[j + 1] = key;
16     }
17 }
```

Листинг 3.3 — Сортировка выбором

```
1  void Selection_Sort(IArr_t arr, int n)
2  {
3      for (int i = 0; i < n - 1; i++)
4      {
5          int min = i;
6          for (int j = i + 1; j < n; j++)
7          {
8              if (arr[j] < arr[min])
9              {
10                 min = j;
11             }
12         }
13
14         std::swap(arr[i], arr[min]);
15     }
16 }
```

### 3.4 Тестирование

В таблице 3.1 отображён возможный набор тестов для тестирования методом чёрного ящика. В столбцах Пузырёк, Вставки, Выбором представлены соответсвенно результаты

сортировки массива, полученные алгоритмом сортировки пузырьком, алгоритмом сортировки вставками и выбором.

Таблица 3.1 — Таблица тестовых данных

| № | Массив | Пузырёк | Вставки | Выбором |
|---|---|---|---|---|
| 1 | 1 2 3 4 | 1 2 3 4 | 1 2 3 4 | 1 2 3 4 |
| 2 | 1 4 3 2 | 1 2 3 4 | 1 2 3 4 | 1 2 3 4 |
| 3 | 4 3 2 1 | 1 2 3 4 | 1 2 3 4 | 1 2 3 4 |
| 4 | 5 | 5 | 5 | 5 |

### 3.5 Вывод

В данном разделе были представлены реализации алгоритмов сортировки пузырьком, вставками и выбором. А также представлен, успешно пройденный, набор тестов.

# 4 Экспериментальный раздел

В данном разделе будут проведены эксперименты для проведения сравнительного анализа трёх алгоритмов по затрачиваемому процессорному времени в зависимости от длины массива и степени его отсортированности.

## 4.1 Сравнительный анализ на основе замеров времени работы алгоритмов

В рамках данного проекта были проведёны следующие эксперименты:

1) сравнение времени работы алгоритмов в лучшем случае (графики 4.1, 4.2);
2) сравнение времени работы алгоритмов в худшем случае (график 4.3);
3) сравнение времени работы алгоритмов в произвольном случае (график 4.4).

Для построения графиков указанные массивы генерировались размерами от 1 до 10000.

Тестирование проводилось на ноутбуке с процессором Intel(R) Core(TM) i5-8250U CPU 1.60 GHz под управлением Windows 10 с 8 Гб оперативной памяти.

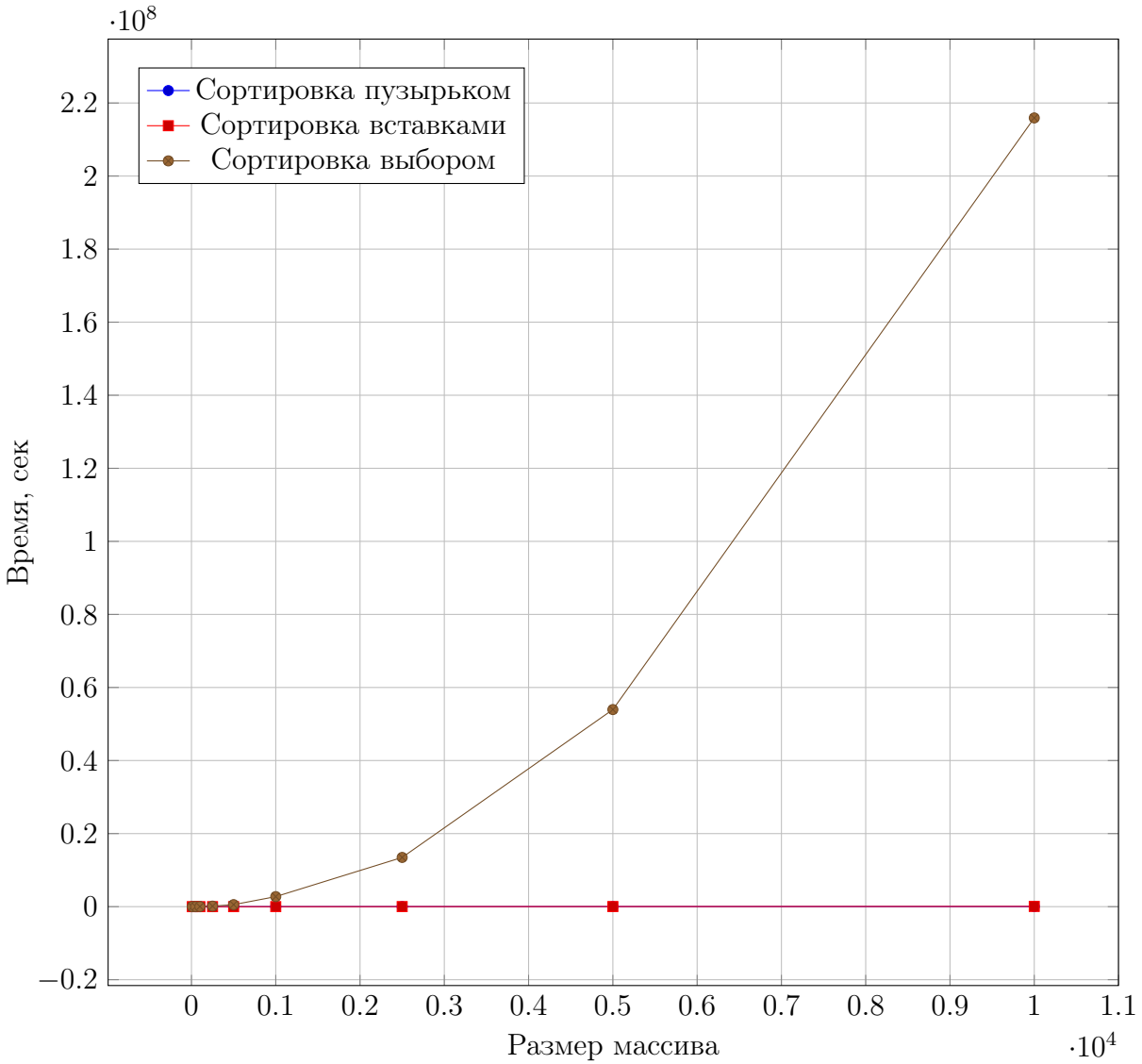


Рисунок 4.1 — График зависимости времени работы реализации алгоритмов сортировки в лучшем случае

### 4.2 Вывод

В ходе экспериментов по замеру времени работы было установлено, что в лучшем случае, когда массив отсортирован, сортировка выбором оказалась самой медленной. Алгоритм сортировки вставками оказался медленнее сортировки вставками. Сортировка пузырьком с флагом в лучшем случае работает быстрее всего.

В худшем случае самой медленной сортировкой является сортировка пузырьком с флагом, а сортировка выбором является самой быстрой (более, чем в 2 раза быстрее сортировки с флагом).

В произвольном случае время работы сортировок вставками и выбором сопоставимо. Самой медленной является сортировка пузырьком с флагом.

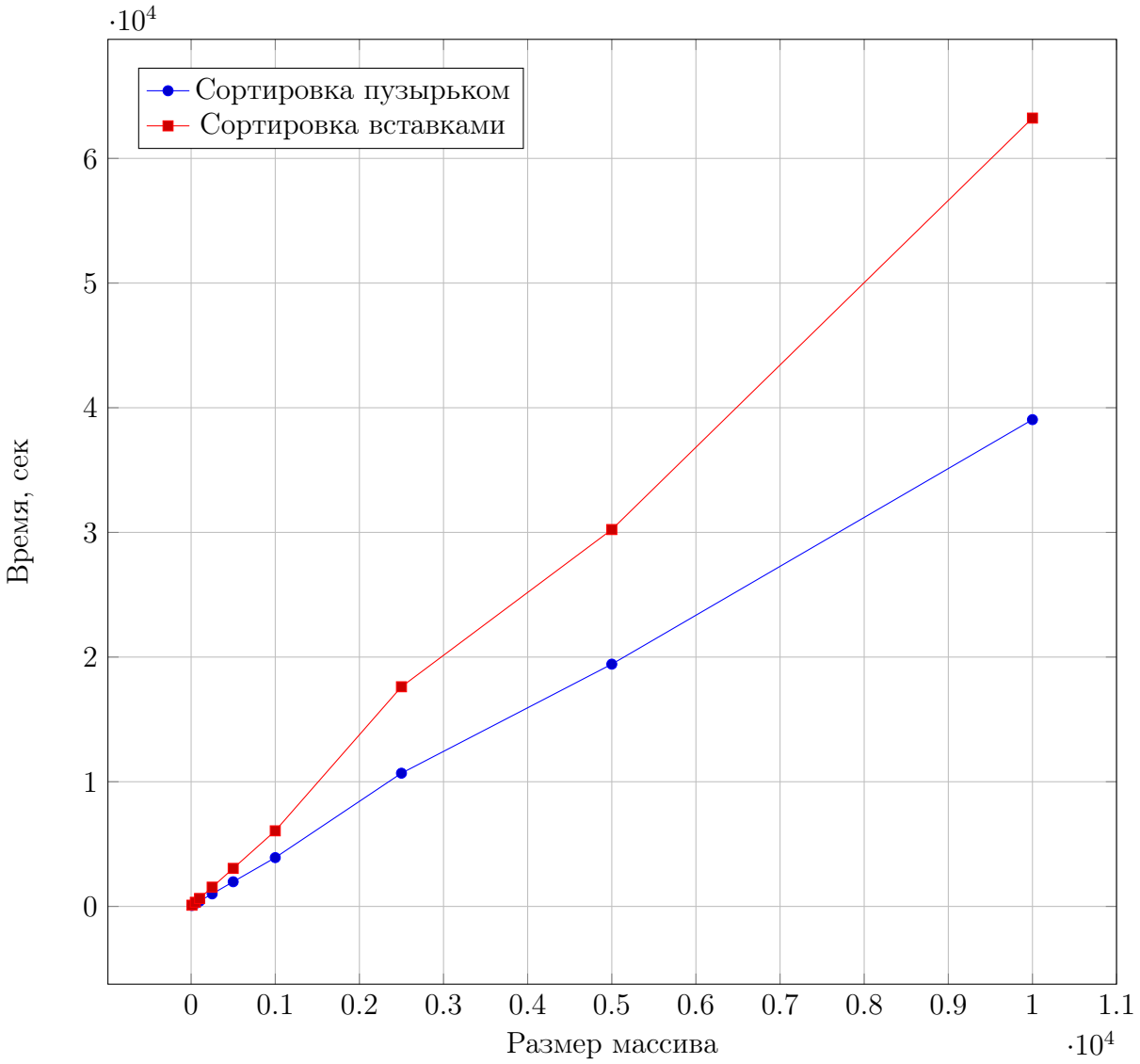


Рисунок 4.2 — График зависимости времени работы реализации алгоритмов сортировки пузырьком и вставками в лучшем случае

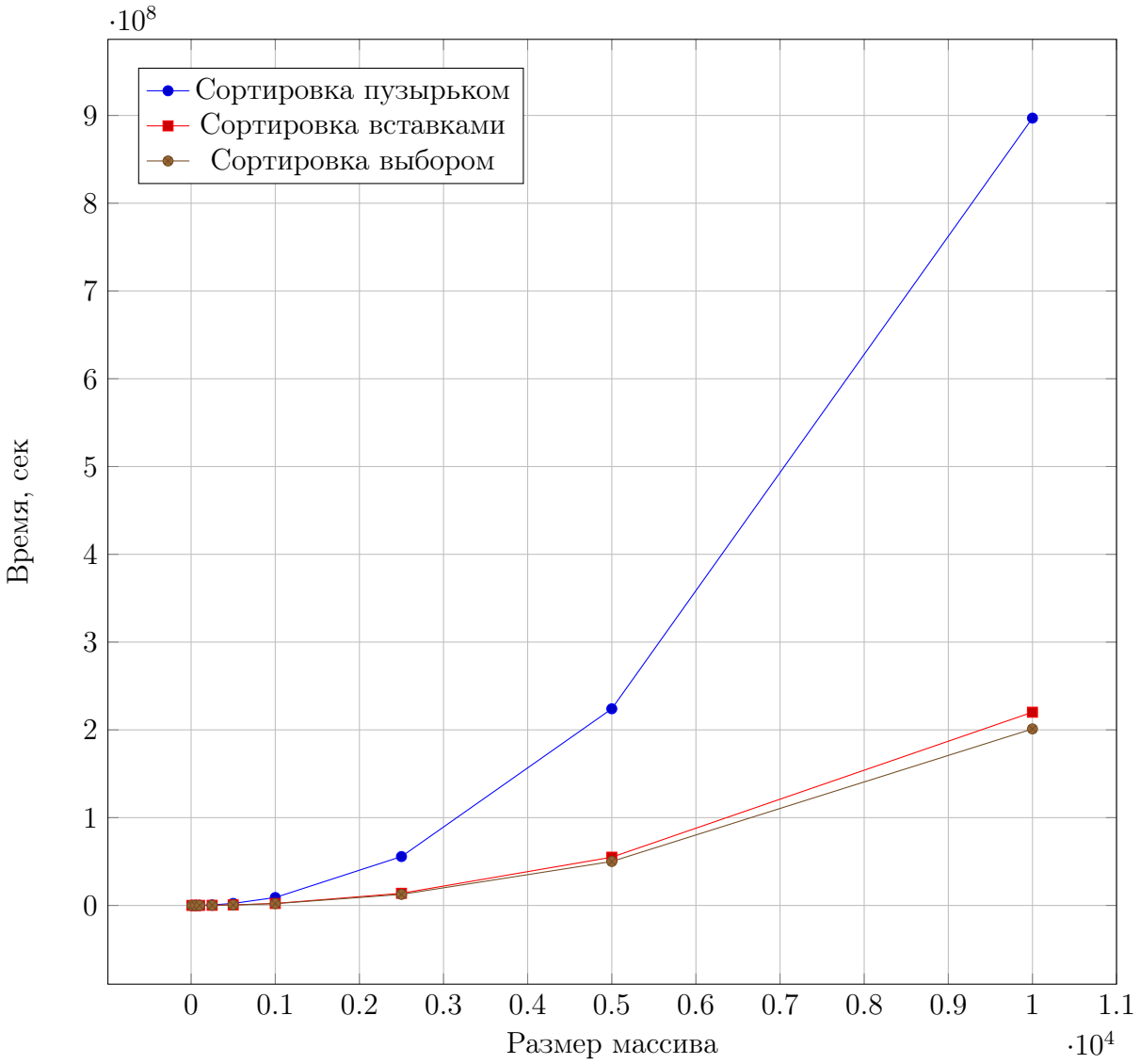


Рисунок 4.3 — График зависимости времени работы реализации алгоритмов сортировки в худшем случае

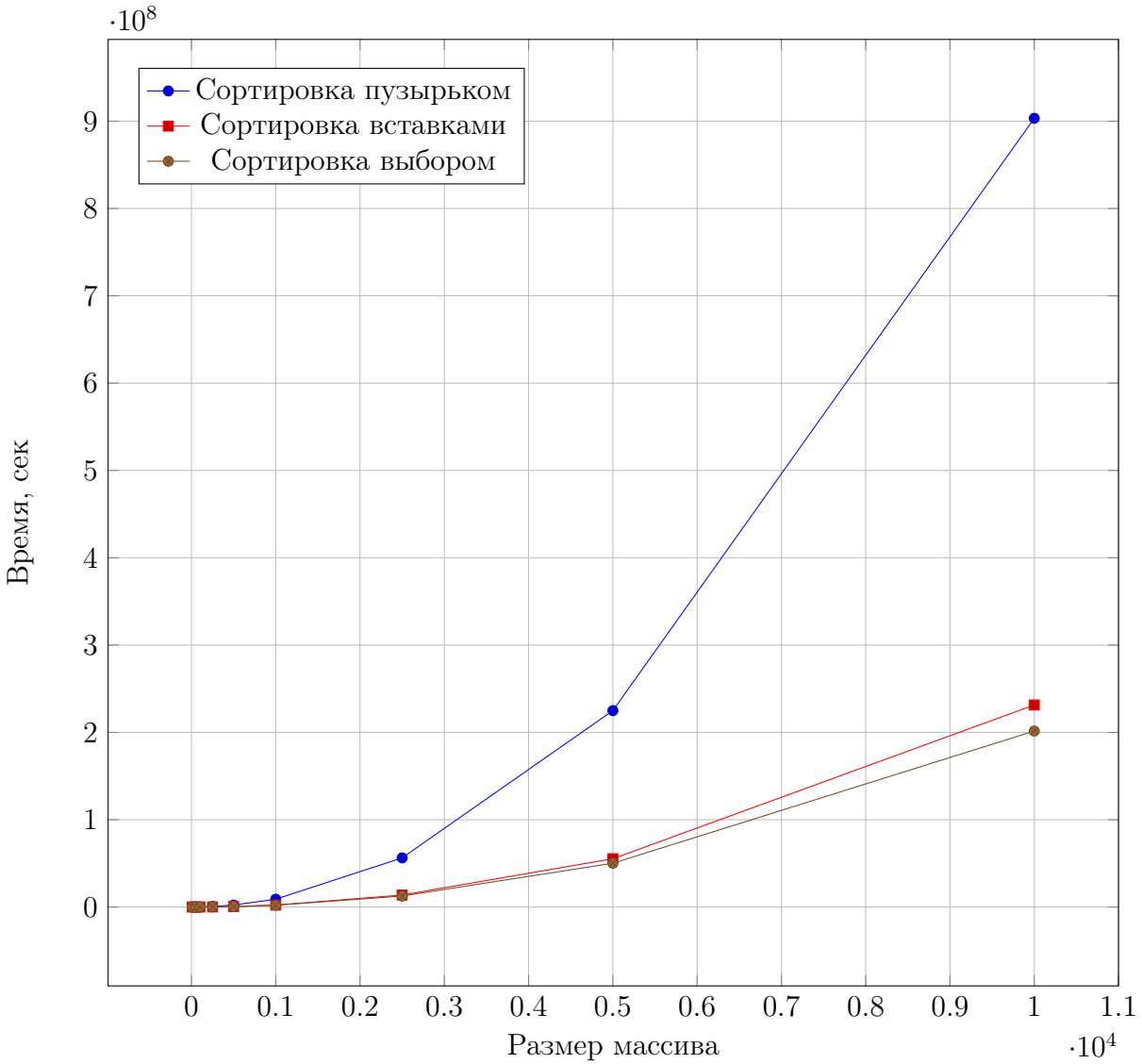


Рисунок 4.4 — График зависимости времени работы реализации алгоритмов сортировки в произвольном случае

# 5 Заключение

В ходе работы были изучены и реализованы алгоритмы сортировки (пузырьком с флагом, вставки и выбором). А также дана оценка их трудоемкости и замерено процессорное время работы алгоритмов.

В ходе экспериментов по замеру времени работы было установлено, что в лучшем случае, когда массив отсортирован, сортировка выбором оказалась самой медленной. Алгоритм сортировки вставками оказался быстрее сортировки выбором. Сортировка пузырьком с флагом в лучшем случае работает быстрее всего.

В худшем случае самой медленной сортировкой является сортировка пузырьком с флагом, а сортировка выбором является самой быстрой (более, чем в 2 раза быстрее сортировки с флагом).

В произвольном случае время работы сортировок вставками и выбором сопоставимо. Самой медленной является сортировка пузырьком с флагом.

## Список использованных источников

1. Ассемблерные вставки в AVR-GCC. // [Электронный ресурс]. Режим доступа: http://we.easyelectronics.ru/AVR/assemblernye-vstavki-v-avr-gcc.html, (дата обращения: 15.10.2020).

2. C/C++: как измерять процессорное время. // [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/282301/, (дата обращения: 15.10.2020).

3. Sorting algorithm. // [Электронный ресурс]. Режим доступа: *https* : *//en.wikipedia.org/wiki/Sorting*$_a$*lgorithm*, (дата обращения: 15.10.2020).