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 о лабораторной работе №3 по дисциплине ”Анализ алгоритмов” на тему ”Алгоритмы сортировки”

Студент Коняев Е.А

Группа ИУ7-53Б

Преподаватели Волкова Л.Л., Строганов Ю.В.

Дата сдачи отчета_____

Оценка (баллы) _____

Москва — 2022 г.

Оглавление

Введение

Алгоритмы сортировки — это набор инструкций, которые принимают массив или список в качестве входных данных и упорядочивают элементы в определенном порядке. Сортировка чаще всего осуществляется в числовом или алфавитном (или лексикографическом) порядке и может быть в порядке возрастания (A-Z, 0-9) или убывания (Z-A, 9-0).

Сортировки важны потому что они часто могут уменьшить сложность решаемой задачи, а потому алгоритмы сортировки очень важны в компьютерных науках. Эти алгоритмы имеют прямое применение в алгоритмах поиска, алгоритмах баз данных, методах разделяй и властвуй, алгоритмах структуры данных и многом другом.

При выборе алгоритма сортировки необходимо учитывать:

1. количиество сортируемых элементов;
2. количество доступной памяти;
3. возможность увеличения коллекции с сортируемыми элементами.

Эти факторы могут определить, какой алгоритм будет работать лучше всего в каждой ситуации. Некоторым алгоритмам может потребоваться много места или памяти для запуска, в то время как другим, при том, что они не являются самыми быстрыми, не требуется много ресурсов для запуска.

Именно из-за разнообразия алгоритмов сортровок, их особенностей и областей применения, данная работа была посвещена анализу некоторых из этих алгоритмов.

Глава 1

Аналитическая часть

1.1 Цели и задачи

Целью данной работой является исследования алгоритмов сортировок пузырьком, вставками и выбором. Для этого в ходе исследования алгоритмов требуется решить следующие задачи:

1. изучить и рассмотреть выбранный алгоритмы сортировок;
2. построить блок-схемы выбранных алгоритмов;
3. реализовать каждый из алгоритмов;
4. рассчитать их трудоемкость;
5. эксперитментально оценить временные характеристики алгоритмов;
6. сделать вывод на основании проделанной работы.

1.2 Сортировка пузырьком

Сортировка пузырьком — это базовый алгоритм для упорядочивания массива чисел или других элементов в правильном порядке. Метод работает, проверяя каждый набор соседних элементов в строке слева направо, меняя их позиции, если они не в порядке (не в порядке возрастания/убывания).

Алгоритм будет просматривать два элемента за раз, переставлять те, которые еще не в порядке возрастания/убывания, слева направо, а затем продолжать циклически проходить всю последовательность N - 1 раз, где N - кол-во элементов массива.

Существует так же модифицированная версия данного алгоритма, где процесс прохода по всем элементам массива продолжается до тех пор, пока алгоритм не посмотрит весь набор чисел и не найдет двух элементов, которые нужно поменять местами.

1.3 Сортировка вставками

В ходе данного алгоритма происходит прохождение по сортируемому массиву слева направо и обрабатывае по очереди каждого элемента. Слева от очередного элемента увеличивается отсортированная часть массива, справа по мере процесса сортировки постепенно уменьшается неотсортированная часть. В отсортированной части массива ищется точка вставки для очередного элемента. Сам элемент отправляется в буфер, в результате чего в массиве появляется свободная ячейка — это позволяет сдвинуть элементы и освободить точку вставки.

1.4 Сортировка выбором

Идея данного метода сортировки состоит в том, чтобы создавать отсортированную последовательность путем присоединения к ней одного элемента за другим в правильном порядке. Можно строить готовую последовательность, начиная с левого конца массива. Алгоритм состоит из n последовательных шагов. На очередном шаге выбираем наименьший из элементов, начинаю текущим и заканчивая последним, и меняем его местами с очередным элементом.

Таким образом, на последнем шаге вся последовательность, кроме последнего элемента оказывается отсортированной, а последний элемент в свою очередь стоит на последнем месте корректно потому, что все меньшие элементы уже ушли влево.

1.5 Вывод

В данном разделе были рассмотрены 3 алгоритма сортировки: пузырьком, вставками и выбором.

Глава 2

Конструкторская часть

2.1 Схемы алгоритмов

На рисунках 2.1, 2.2 и 2.3 показаны схемы алгоритмов сортировки пузырьком, выбором и вставками соответственно.

2.2 Вывод

В данном разделе на основе теоретических данных, полученных в аналитическом разделе были разработаны схемы алгоримов трех рассматриваемых алгоритмов сортировок.

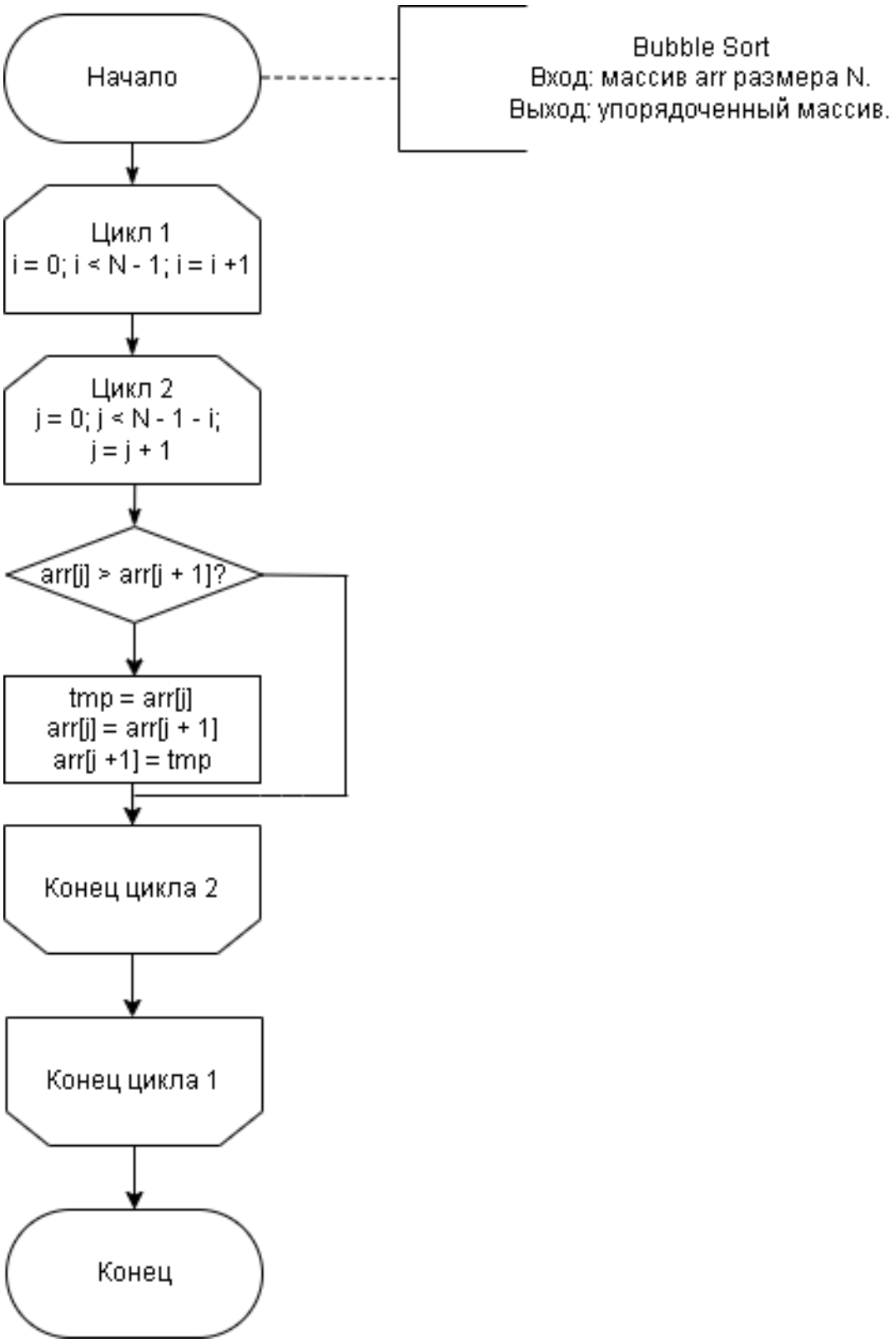


Рис. 2.1: Схема сортировки пузырьком

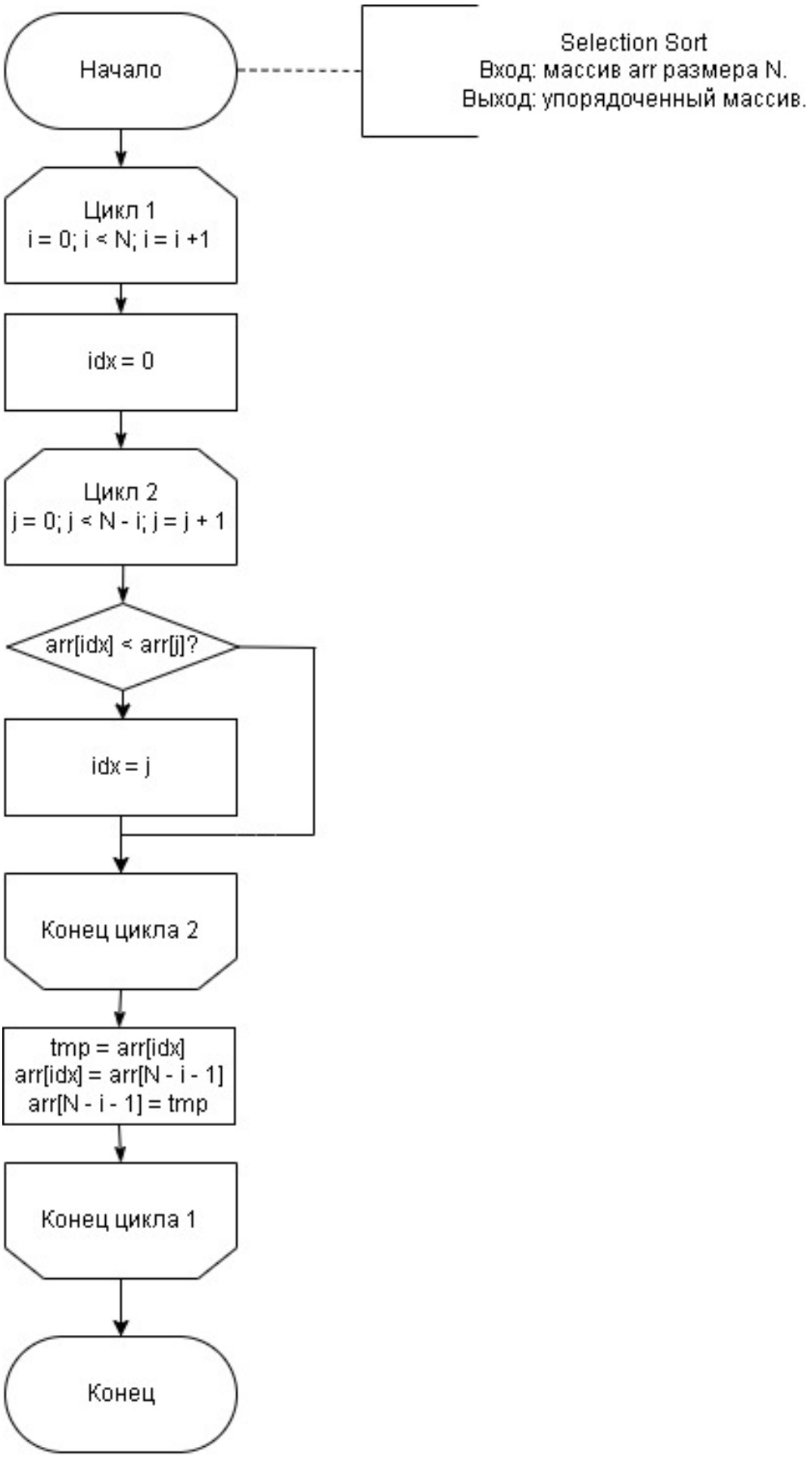


Рис. 2.2: Схема сортировки выбором

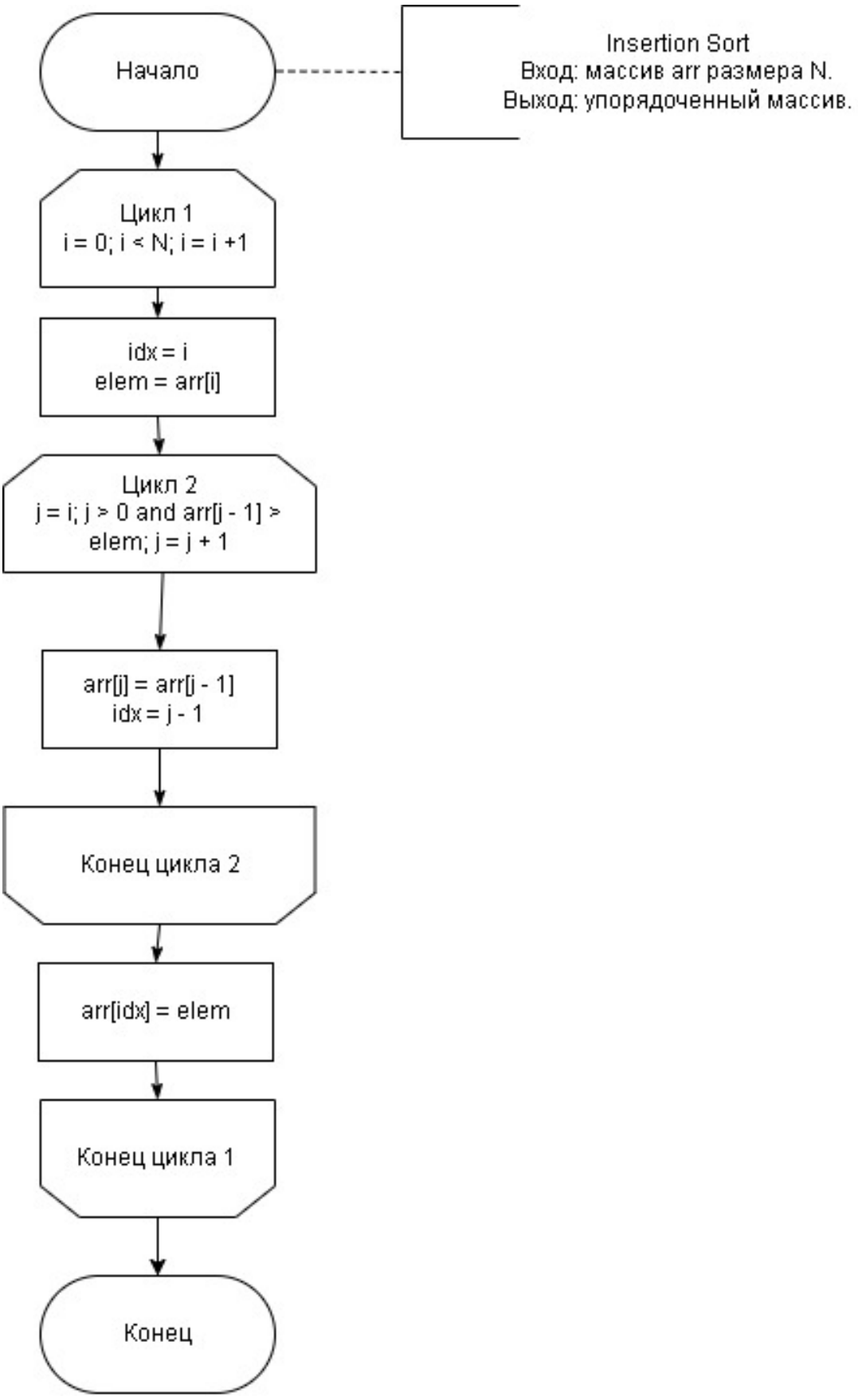


Рис. 2.3: Схема сортировки вставками

Глава 3

Технологическая часть

В данном разделе преведены требования к ПО, обоснования выбора языка программирования, среды разработки, приведен способ замера времени выполнения, а так же приведены листинги кода.

3.1 Требования к ПО

В программе должна присутсвовать возможность :

1. указания размера входного массива, а так же его элементов;
2. сортировки входного массива одним из трех рассматриваемых способов сортировок;
3. проверки временных характеристик на разных типах входных массивов (упорядочен по возрастанию/убыванию, заполнен случайными числами), для рассматриваемых способов сортировок.

3.2 Выбор языка программирования и среды разработки

Для реализации трех алгоритмов сортировок был выбор язык C, так как я уже знаком с данным языком и поставленная задача будет решена максимально быстро.

Средой разработки был выбран CLion. Данный выбор обусловлен тем, что данная среда на сегодняшний момент является по-моему мнению самой удобной IDE для разработки под C/C++, в силу того, что предоставляет мощные инструменты для отладки кода и его быстрого написания.

3.3 Выбор библиотеки и способа для замера времени

Для замера времени выполнения сортировок использовалась стандартная функция библиотеки <time.h> языка C — clock(), которая замеряет процессорное время. Если измерить время перед началом выполнения алгоритма, и после его окончания, то можно получить время выполнения функции. Реализация данной функции привидена в списке литературы[1].

Посколько все процессорное время не отдается какоЙ-либо одной задаче (явление вытеснения процессов из ядра, квантование процессорного времени), то усреднить результаты вычислений: замерить совокупное время выполнения реализации алгоритма N раз и вычислить среднее время выполнения.

3.4 Листинги кода

В листингах 3.1,3.2,3.3 приведены листинги алгоритмов сортировок пузырьком, выбором и вставками соответсвенно.

```
1     void bubbleSort(int *arr, size_t n)
2     {
3       for (int i = 0; i < n - 1; ++i)
4       for (int j = 0; j < n - 1 - i; ++j) {
5         if (arr[j] > arr[j + 1])
6         {
7           int tmp = arr[j];
8           arr[j] = arr[j + 1];
9           arr[j + 1] = tmp;
10        }
11      }
12    }
```

Листинг 3.1: Функция сортировки массива пузырьком

```
1     void selectionSort(int *arr, size_t n)
2     {
3       for (int i = 0; i < n; ++i) {
4         int idxToSwap = 0;
5         for (int j = 0; j < n - i; ++j) {
6           if (arr[idxToSwap] < arr[j]) {
7             idxToSwap = j;
8           }
9         }
10        int tmp = arr[idxToSwap];
11        arr[idxToSwap] = arr[n - i - 1];
12        arr[n - i - 1] = tmp;
13      }
14    }
```

Листинг 3.2: Функция сортировки массива выбором

```
1     int insertionSort(int *arr, size_t n)
2     {
3       for (int i = 0; i < n; ++i) {
4         int idxInsert = i;
5         int insertElement = arr[i];
6         for (int j = i; j > 0 && arr[j - 1] > insertElement; --j) {
7           arr[j] = arr[j - 1];
8           idxInsert = j - 1;
9         }
10        arr[idxInsert] = insertElement;
11      }
12    }
```

Листинг 3.3: Функция сортировки массива вставками

3.5 Тестирование алгоритмов

В таблице 3.1 приведены тесты для функций, реализующих алгоритмы сортировки. Все тесты пройдены успешно.

Входной массив	Результат	Ожидаемый результат
$[10, 20, 30, 40, 50]$	$[10, 20, 30, 40, 50]$	$[10, 20, 30, 40, 50]$
$[50, 40, 30, 20, 10]$	$[10, 20, 30, 40, 50]$	$[10, 20, 30, 40, 50]$
$[-15, -21, -37, -24, -54]$	$[-54, -37, -24, -21, -15]$	$[-54, -37, -24, -21, -15]$
$[4]$	$[4]$	$[4]$
Пустой массив	Пустой массив	Пустой массив

Таблица 3.1: Тестирование функций

3.6 Вывод

В данном разделе были разработаны исходные коды трёх алгоритмов сортировки: пузырьком, выбором и вставками.

Глава 4

Экспериментальная часть

4.1 Технические характеристики

Ниже приведены технические характеристики устройства, на котором было проведено тестирование ПО:

1. Операционная система: Windows-10, 64-bit.
2. Оперативная память: 16 GB.
3. Процессор: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, 1992 МГц, 4 ядра, 8 логических процессоров

4.2 Время выполнения алгоритмов

Таблица 4.1: Таблица времени выполнения сортировок на данных, отсоритрованных по возрастанию (в мс)

Размер	bubble	selection	insertion
100	0.02	0.02	0
1000	1.02	1.26	0
10000	102.5	117.5	0

График зависимости времени сортировок от кол-ва элементов на упорядоченном массиве

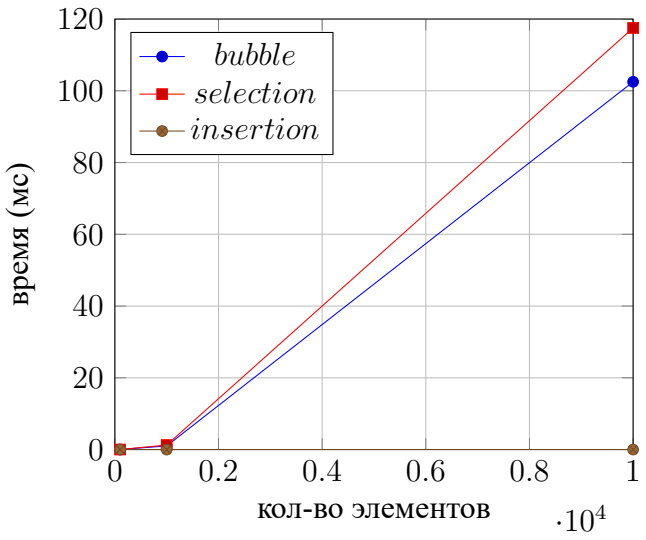


Таблица 4.2: Таблица времени выполнения сортировок на данных, отсоритрованных по убыванию (в мс)

Размер	bubble	selection	insertion
100	0.02	0.02	0.02
1000	1.94	1.04	1.48
10000	187.8	103.3	139.9

График зависимости времени сортировок от кол-ва элементов на обратно упорядоченном массиве

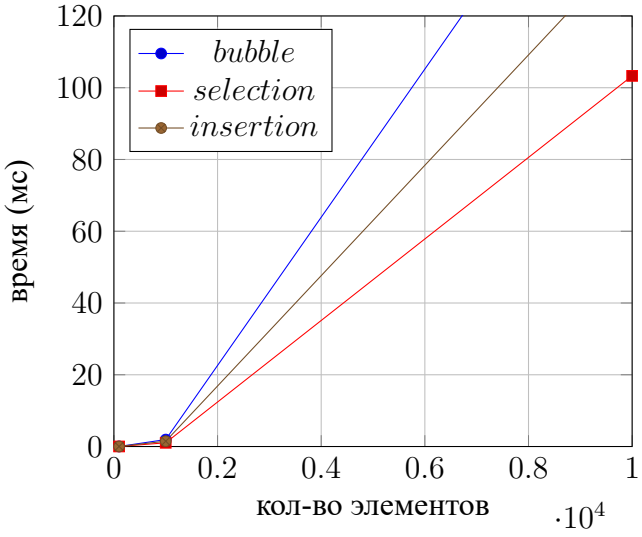
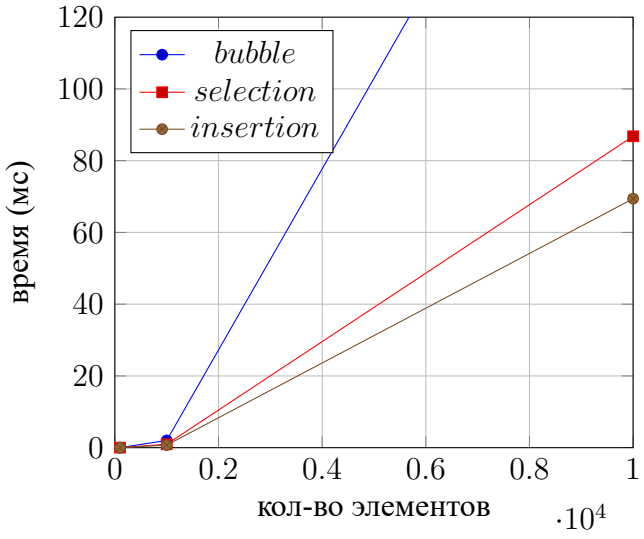


Таблица 4.3: Таблица времени выполнения сортировок на случайных данных (в мс)

Размер	bubble	selection	insertion
100	0.02	0.02	0.02
1000	2.02	0.96	0.72
10000	229.1	86.8	69.4

График зависимости времени сортировок от кол-ва элементов на массиве случайных чисел



4.3 Вывод

Алгоритм сортировки вставками работает лучше сортировок выбором и пуззырьком на случайных и уже отсортированных числах. Сортировка вставками массива из 10000 элементов в 3.3 раза быстрее сортировки пузырьком и в 1.6 раз ьыстрее сортировки выбором. Следовательно, алгоритм сортировки вставками работает быстрее, чем алгоритм сортировки пузырьком и выбором на больших массивах данных.

Заключение

В рамках данной лабораторной работы:

1. были изучены и рассмотрены алгоритмы сортировок пузырьком, вставками и выбором;
2. были построены блок-схемы выбранных алгоритмов;
3. был реализован каждый из алгоритмов;
4. была рассчитана их трудоемкость;
5. были эксперитментально оценены временные характеристики алгоритмов;
6. были сделаны выводы на основании проделанной работы

На основании анализа трудоемкости алгоритмов в выбранной модели вычислений, было показано, что алгоритм сортировки вставками имеет наименьшую сложность в уже отсортированном массиве, а так же эффективнее работает на массиве размером 10000, обгоняя по скорости алгоритмы пузырьком и выбором.