МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника»

"___" _____ 20 г.

Заведующий кафедрой

_____М.А. Митрохин

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ: ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

(2024/2025 учебный год)

Максин Сергей Николаевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 5 лет

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., Митрохин М.А.

(должность, ученая степень, ученое звание, Ф.И.О.)

Руководитель практики к.т.н., доцент, Карамышева Н.С.

(должность, ученая степень, ученое звание)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Утвержден на заседании кафедры

«Вычислительная техника»

"___" _____ 20 г.

Заведующий кафедрой

_____М.А. Митрохин

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ: ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

(2024/2025 учебный год)

Максин Сергей Николаевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 5 лет

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., Митрохин М.А.

(должность, ученая степень, ученое звание, Ф.И.О.)

Руководитель практики к.т.н., доцент, Карамышева Н.С.

(должность, ученая степень, ученое звание

№ п/п	Планируемая форма работы во время практики	Количество часов	Календарные сроки проведения работы	Подпись руководителя практики от вуза
1	Выбор темы и разработка индивидуального плана проведения работ (распределенная практика)	26	20.06.25 – 24.06.25	
2	Подбор и изучение материала по теме работы (распределенная практика)	26	24.06.25 – 26.06.25	
3	Разработка алгоритма (распределенная практика)	26	26.06.25 – 28.06.25	
4	Описание алгоритма и программы (сосредоточенная практика)	30	04.07.25 – 09.07.25	
5	Тестирование (сосредоточенная практика)	30	09.07.25 – 12.07.25	
6	Получение и анализ результатов (сосредоточенная	38	12.07.25 – 14.07.25	

	практика)			
7	Оформление отчёта (сосредоточенная практика)	40	14.07.25 – 17.07.25	
	Общий объём часов	216		

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЧЁТ

О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ

(2024/2025 учебный год)

Максин Сергей Николаевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

Максин С.Н. выполнял практическое задание «Шейкерная сортировка». На первоначальном этапе были изучен и проанализирован алгоритм шейкерной сортировки, был выбран метод решения и язык программирования С, на котором была написана программа сортировки массива методом шейкерной сортировки. Также, оформил отчёт.

Специалист Максин С.Н_____ "___" _____ 2025 г.

Руководитель практики Карамышева Н.С._____ "___" _____ 2025 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЗЫВ

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ: ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

(2024/2025 учебный год)

Максин Сергей Николаевич

Направление подготовки (специальность) 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Наименование профиля подготовки (специализация) «Эксплуатация вычислительных машин, комплексов, систем и сетей»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 5 лет

Год обучения 1 семестр 2

Период прохождения практики с 20.06.2025 по 17.07.2025

Кафедра «Вычислительная техника»

В процессе выполнения практики Максин С.Н. решал следующие задачи: разработка алгоритм записи отсортированного массива в файл и алгоритм таймера сортировки массива.

За период прохождения практики были освоены основные понятия и технологии программирования, изучены основные инструменты языков C/C++. И получены следующие результаты: разработан алгоритм шейкерной сортировки, получены результаты работы алгоритма, сделаны выводы эффективности работы алгоритма шейкерной сортировки. Во время выполнения работы Максин С.Н. показал себя ответственным, добросовестным учеником, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике; программированию.

За выполнение работы Максин С.Н заслуживает оценки «_____».
Руководитель практики к.т.н., Карамышева Н.С. .« »_____2025 г.

Содержание

Введение

Сортировка данных на сегодняшний день при современном развитии компьютерных технологий является одним из наиболее распространенных процессов современной обработки данных. Задачи на сортировку данных встречаются очень часто в различных профессиональных сферах деятельности.

Алгоритмы сортировки очень широко распространяются практически во всех задачах обработки информации. Они образуют отдельный класс алгоритмов, применяются с целью осуществления последующего более быстрого поиска.

Важность сортировки основана на том факте, что на ее примере можно показать многие основные фундаментальные приемы и методы построения алгоритмов. Сортировка является хорошим примером огромного разнообразия алгоритмов, которые выполняют одну и ту же задачу. Кроме того, многие из них имеют определенные преимущества друг перед другом. За счет усложнения алгоритма можно добиться существенного увеличения эффективности и быстродействия алгоритма по сравнению с более простыми методами. Как правило, термин сортировка понимают, как процесс перестановки объектов некоторого множества в определенном порядке.

Шейкерная сортировка используется редко в реальных приложениях из-за своей неэффективности на больших данных. Она может быть полезна в учебных целях для понимания основных принципов сортировки и структур данных. В **р**еальных приложениях для сортировки массивов обычно применяют более эффективные алгоритмы, такие как быстрая сортировка, сортировка слиянием или сортировка пирамидой.

1 Постановка задачи

Поставленная задача: необходимо заполнить массив из n-ого количества элементов случайными числами. После этого выполнить сортировку вставками над данными, находящимися в массиве, записать отсортированные данные в файл, посчитать время выполнения и сортировки.

Использовать сервис GitHub для совместной работы. Создать и выложить коммиты, характеризующие действия, выполненные каждым участником бригады.

Оформить отчет по проведенной практике.

1.1 Достоинства алгоритма шейкерной сортировки

– алгоритм прост в понимании и реализации;

– алгоритм эффективен при работе с небольшими массивами, где большая часть элементов уже находится в правильном порядке;

– алгоритм производит небольшое количество обменов элементов.

1.2 Недостатки алгоритма шейкерной сортировки

– в отличие от более сложных алгоритмов, не является стабильным;

– высокая алгоритмическая сложность $O(n^2)$;

– не рекомендуется для сортировки больших массивов.

1.3 Типичные сценарии применения данного алгоритма

– использование в учебных целях для понимания основных принципов сортировки структур данных.

2 Выбор решения

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. При разработке языка Си был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. Си – это язык программирования общего назначения, хорошо известный своей эффективностью, экономичностью и переносимостью. Указанные преимущества Си обеспечивают хорошее качество разработки почти любого вида программного продукта.

В качестве среды программирования была выбрана программа Microsoft Visual Studio Code. Microsoft Visual Studio Code — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Для удобства совместной разработки был использован сервис GitHub. GitHub — сервис для управления личными и командными проектами. Проект динамично разрабатывается, регулярно расширяя функционал и возможности.

3. Описание программы

При запуске программы выводится сообщение: «Введите размер массива:» и пользователю предлагается ввести количество значений для сортировки.

После того как данные были введены, генерируется массив из случайных чисел, которые выводятся на терминал.

Далее над этими данными выполняется шейкерная сортировка, при которой массив перебирается сначала слева направо, а потом справа налево. При этом сравниваются соседние элементы и меняются местами так, чтобы оказаться в подходящем месте среди ранее упорядоченных элементов. Так происходит до тех пор, пока набор входных данных не будет исчерпан.

После этого отсортированный массив записывается в файл shakerSort.txt.

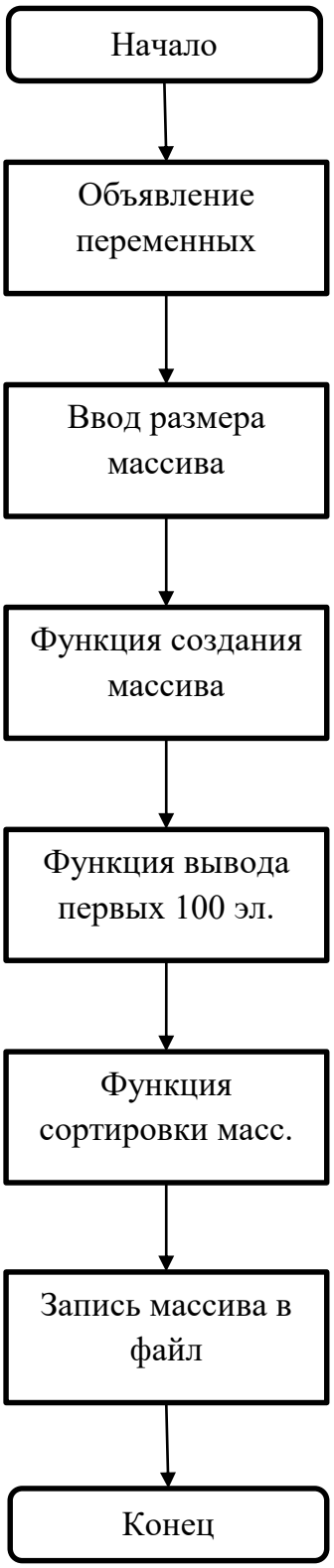
Программа так же осуществляет подсчет времени, затраченного на сортировку.

Подробный алгоритм работы программы и функции сортировки преставлен в подразделе 4.1 на рисунке 1.

Листинг программы приведен в приложении Б.

4 Схемы программы

4.1 Блок-схема программы



5. Описание интерфейса взаимодействия

Основной функционал программы реализуется через консольный интерфейс. Пользователь взаимодействует с программой посредством командной строки.

Основные элементы интерфейса:

5.1 Ввод размера массива

- программа запрашивает у пользователя размер массива через стандартный ввод

5.2 Вывод информации осуществляется через:

- Отображение первых 100 элементов исходного массива
- Информирование о завершении сортировки
- Вывод времени выполнения сортировки
- Сообщение об ошибках (при необходимости)

5.3 Автоматические действия программы:

- Генерация случайного массива чисел
- Сортировка массива методом шейкерной сортировки
- Запись отсортированного массива в файл

5.4 Порядок работы с программой:

1. Запуск программы
2. Ввод размера массива
3. Ожидание завершения генерации и сортировки
4. Просмотр результатов в консоли
5. Проверка файла shakerSort.txt для просмотра отсортированного массива

5.5 Обработка ошибок

- Программа предусматривает базовую проверку ошибок:
- Проверка успешности выделения памяти
- Проверка открытия файла для записи
- Вывод соответствующих сообщений при возникновении проблем.

6.Тестирование программы

6.1 Тестирование на разных наборах данных

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены в Приложении А на рисунках А.1-А.11.

Таблица 1 – Тестовый набор данных

	Размер массива size	Время выполнения сортировки в секундах
1	10000	0.105
2	20000	0.471
3	30000	1.159
4	40000	2.08
5	50000	3.335
6	60000	4.842
7	70000	6.604
8	80000	9.125
9	90000	11.107
10	100000	14.533
11	110000	16.589

6.2 Анализ полученных результатов тестирования (анализ работы алгоритма)

На основании анализа данных, полученных в результате тестирования алгоритма шейкерной сортировки, можно сделать вывод, что время, затраченное на работу программы относительно количества элементов увеличивается линейно, то есть с увеличением количества элементов пропорционально увеличивается время работы программы.

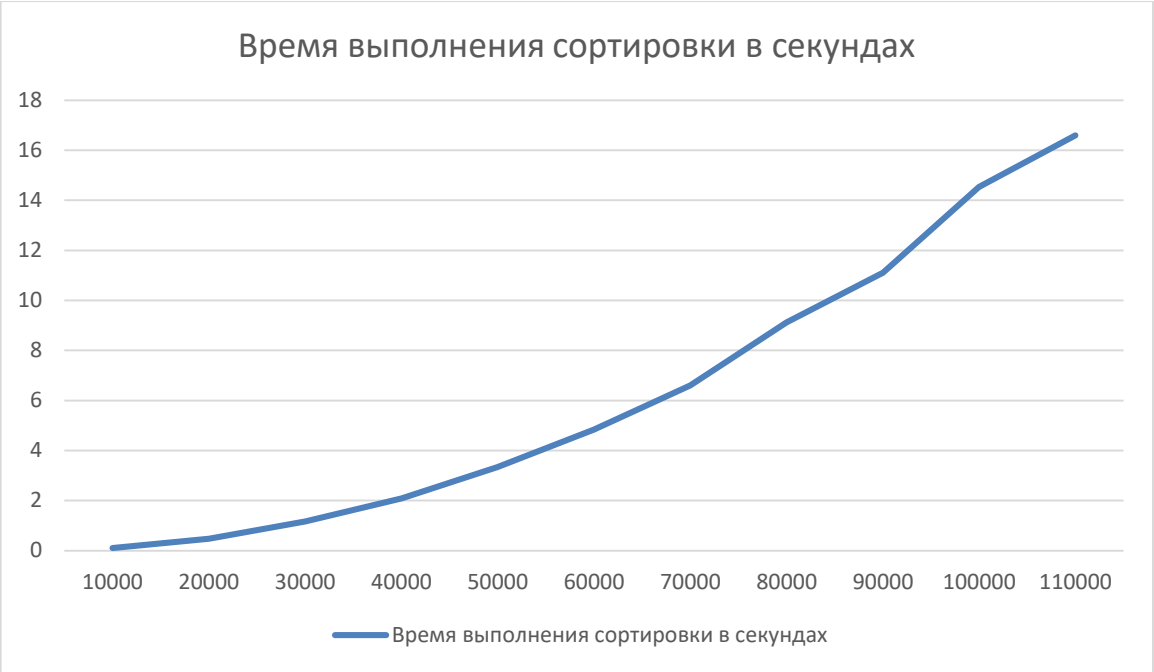


Рисунок 2 – Результаты тестирования

7 Отладка

В качестве среды разработки была выбрана программа Microsoft Visual Studio Code, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ.

Для отладки программы использовались точки остановки и пошаговое выполнение кода программы, анализ содержимого локальных переменных. Точки останова – это прерывание выполнения программы, при котором выполняется вызов отладчика. Отладчик является инструментом для поиска и устранения ошибок в программе, с помощью которого можно исследовать состояние программы.

Был использован метод бинарного поиска, он включает в себя разделение частей кода для упрощения процесса отладки. Это может быть особенно полезно, если причина ошибки находится в начале языка программирования, а фактическая ошибка ближе к концу.

Команда шаг с заходом (step into) выполняет следующую инструкцию в обычном пути выполнения программы, а затем приостанавливает выполнение программы, чтобы мы могли проверить состояние программы с помощью отладчика. Если выполняемый оператор содержит вызов функции, шаг с заходом заставляет программу перескакивать в начало вызываемой функции, где она приостанавливается.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. После завершения написания программы, мною были выявлены и исправлены ошибки.

8 Совместная работа

Во время работы над данной практикой наша бригада осуществляла совместную работу в GitHub.

Определили задачи проекта, назначили приоритет задачам. Разделили роли, назначили исполнителей задачам. Обсуждали выполнение задачи на доске. Корректировали статус задач по мере выполнения. Данная программа была загружена на компьютер, с помощью git clone <ссылка>.

Мной был разработан алгоритм записи отсортированного массива в файл и алгоритм таймера сортировки массива. Я загрузил этот код на репозиторий GitHub.

Ссылка на репозиторий: https://github.com/SurHeliko/cocktail-shaker-sort

Заключение

При выполнении данной работы были получены навыки совместной работы с помощью сервисов GitHub, навыки использования программы Git Bash. Был изучен алгоритм сортировки вставками.

Мною был написан алгоритм записи отсортированного массива в файл и алгоритм таймера сортировки массива. Было выполнено тестирование программы на разных наборах данных и отладка данной программы.

При выполнении практической работы были улучшены базовые навыки программирования на языке С. Улучшены навыки отладки, тестирования программ и работы со сложными типами данных.

В дальнейшем программу можно улучшить путем подключения упрощающих реализацию данной сортировки библиотек и улучшения графического интерфейса.

Список используемой литературы

1. ГОСТ 19.701 – 90 Схемы алгоритмов, программ, данных и систем.

2. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2-е издание.: Пер. с англ. – М.,2009.

3. Шейкерная сортировка [Электронный ресурс] – URL: https://ru.wikipedia.org (дата обращения: 30.06.2025 г)

Приложение А. Результаты тестирования программы

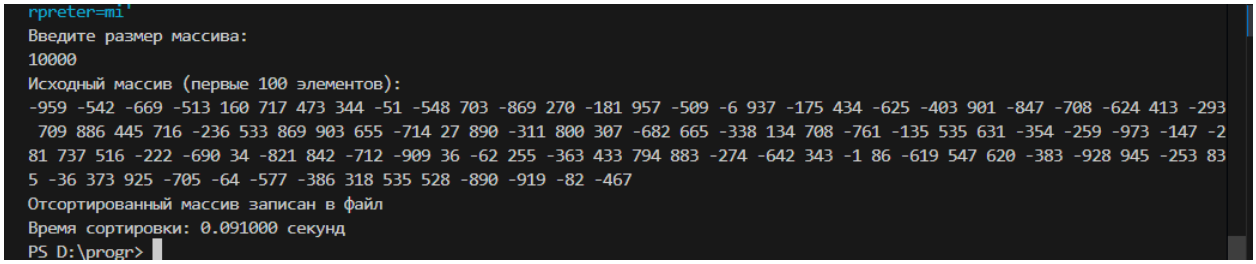


Рисунок А.1

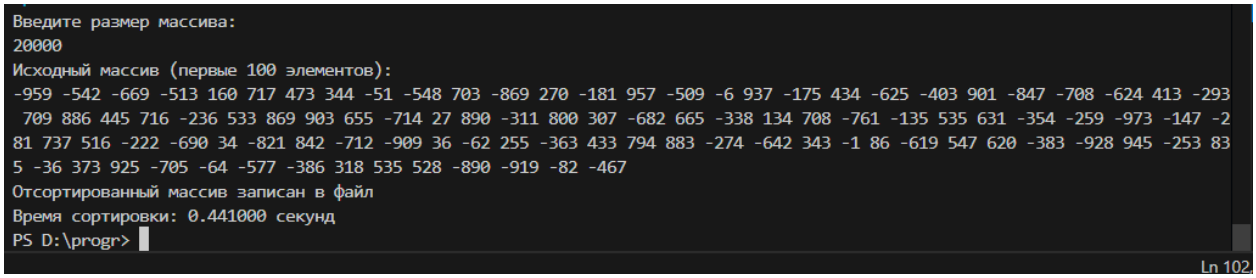


Рисунок А.2

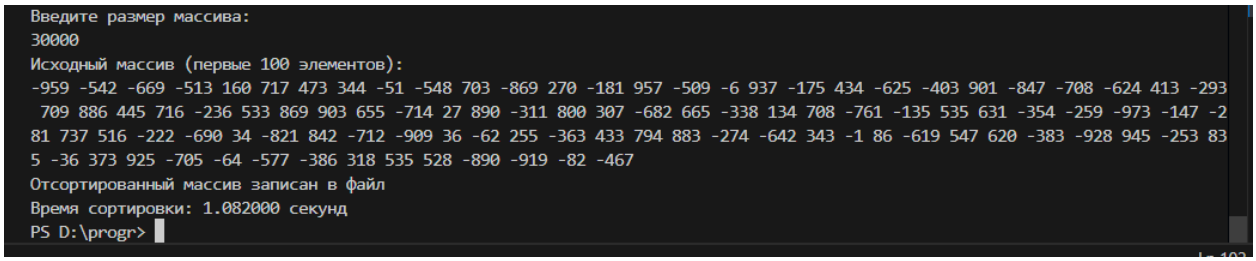


Рисунок А.3

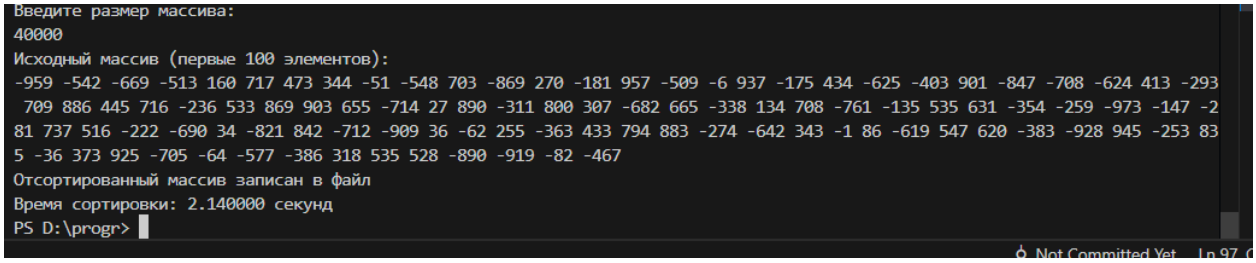


Рисунок А.4

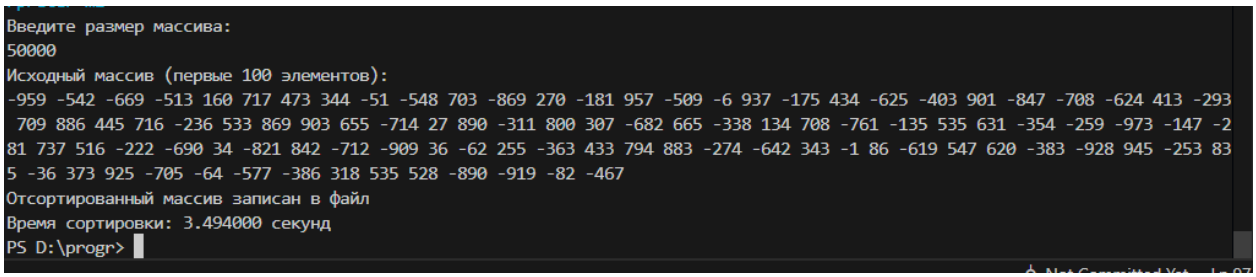


Рисунок А.5

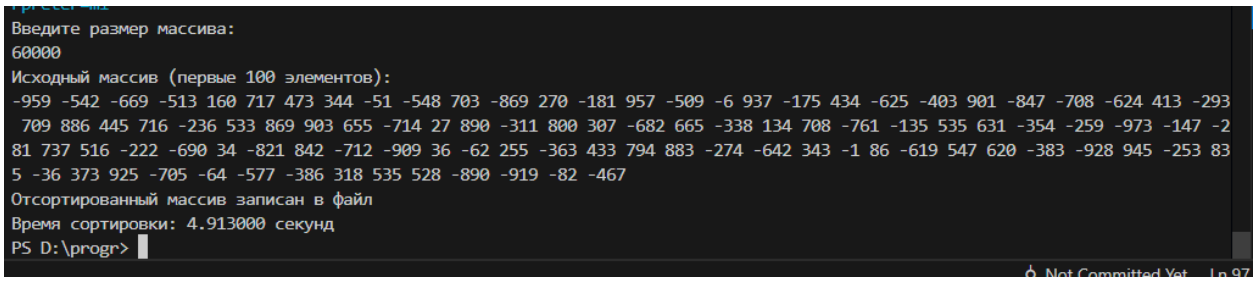


Рисунок А.6

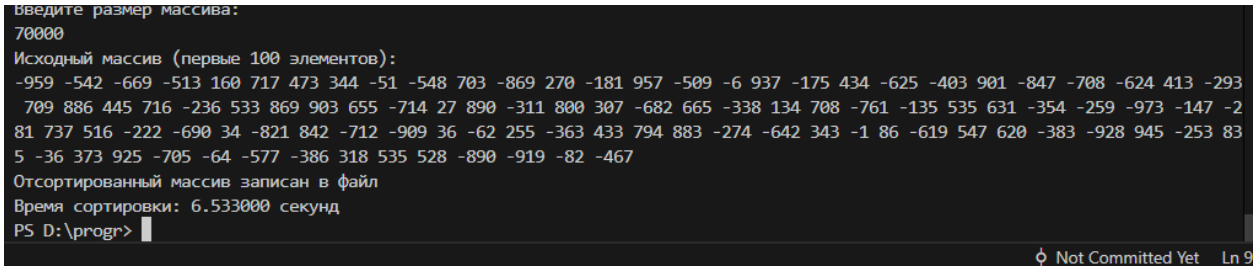


Рисунок А.7

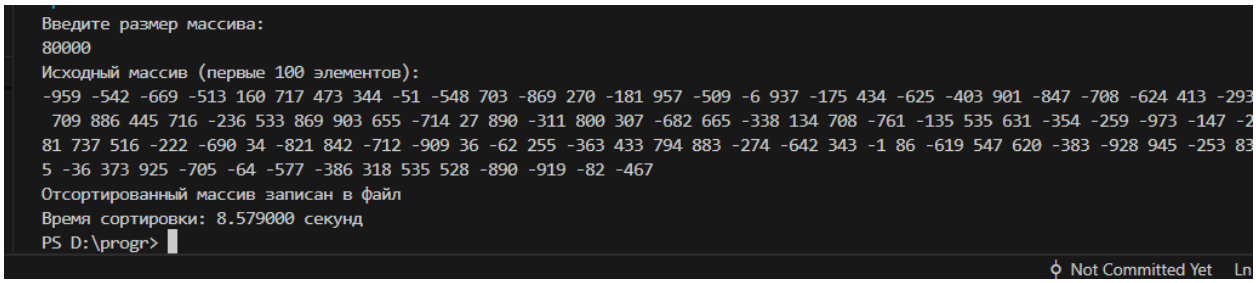


Рисунок А.8

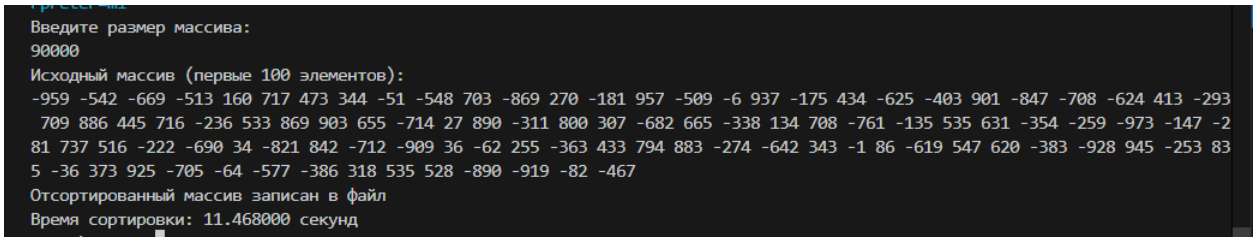


Рисунок А.9

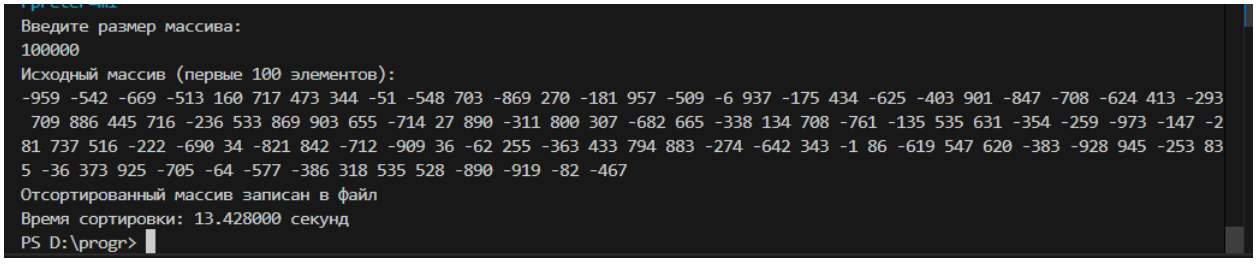


Рисунок А.10

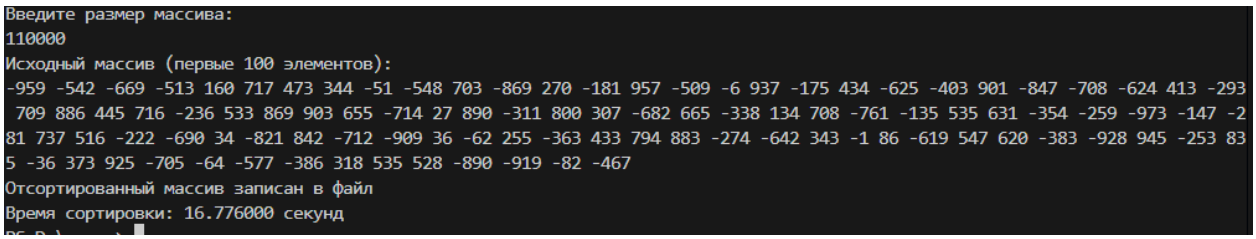

Введите размер массива:
110000
Исходный массив (первые 100 элементов):
-959 -542 -669 -513 160 717 473 344 -51 -548 703 -869 270 -181 957 -509 -6 937 -175 434 -625 -403 901 -847 -708 -624 413 -293
709 886 445 716 -236 533 869 903 655 -714 27 890 -311 800 307 -682 665 -338 134 708 -761 -135 535 631 -354 -259 -973 -147 -2
81 737 516 -222 -690 34 -821 842 -712 -909 36 -62 255 -363 433 794 883 -274 -642 343 -1 86 -619 547 620 -383 -928 945 -253 83
5 -36 373 925 -705 -64 -577 -386 318 535 528 -890 -919 -82 -467
Отсортированный массив записан в файл
Время сортировки: 16.776000 секунд

Рисунок А.11

Приложение Б. Листинг программы

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// сортировка
void shakerSort(int arr[], int size) {
    int left = 0;
    int right = size - 1;
    int swapped = 1;

    while (left < right && swapped) {
        swapped = 0;

        // проход слева направо
        for (int i = left; i < right; i++) {
            if (arr[i] > arr[i + 1]) {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
                swapped = 1;
            }
        }
        right--;

        // проход справа налево
        for (int i = right; i > left; i--) {
            if (arr[i] < arr[i - 1]) {
                int temp = arr[i];
```

```
                arr[i] = arr[i - 1];
                arr[i - 1] = temp;
                swapped = 1;
            }
        }
        left++;
    }
}

// вывод массива
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

// создание массива рандомных чисел
void initializeArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] = rand() % (1000 - -1000 + 1) + -1000;
    }
}

// запись массива в файл
void writeArrToFile(int arr[], int size, const char* filename) {
    FILE* file = fopen(filename, "w");
    if (file == NULL) {
        printf("Ошибка открытия файла для записи!\n");
```

```
        return;
    }

    for (int i = 0; i < size; i++) {
        fprintf(file, "%d ", arr[i]);
    }
    fprintf(file, "\n");

    fclose(file);
}

int main() {
    int size;
    printf("Введите размер массива:\n");
    scanf("%d", &size);

    //динамическое выделение памяти для массива
    int *arr = (int*)malloc(size * sizeof(int));
    if (arr == NULL) {
        printf("Ошибка выделения памяти!\n");
        return 1;
    }

    initializeArray(arr, size);

    printf("Исходный массив (первые 100 элементов):\n");
    printArray(arr, size > 100 ? 100 : size);

    clock_t start = clock();                // время до сортировки
```

```
    shakerSort(arr, size);


    clock_t stop = clock();                    // время после сортировки
    double time = (double)(stop - start) / CLOCKS_PER_SEC; // время
сортировки в секундах


    printf("Отсортированный массив записан в файл\n");
    printf("Время сортировки: %f секунд\n", time);


    writeArrToFile(arr, size, "shakerSort.txt");


    //освобождение памяти
    free(arr);


    return 0;
}
```