

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 (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКЕ
(2022/2023 учебный год)

Кулахметов Сабир Исхакович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

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

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

Руководитель практики д.т.н., профессор, Зинкин С.А.

(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

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ
(ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

_____ Кулахметов Сабир Исхакович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения _____ 1 _____ семестр _____ 2 _____

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

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

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

Руководитель практики д.т.н., профессор, Зинкин С.А.

(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	2	29.06.2023 - 29.06.2023	
2	Подбор и изучение материала по теме работы	15	30.06.2023 – 02.07.23	
3	Разработка алгоритма	43	02.07.23 – 06.07.23	
4	Описание алгоритма и программы	18	6.07.23 – 08.07.23	
5	Тестирование	5	08.07.23 – 08.07.23	
6	Получение и анализ результатов	10	08.07.23 – 10.07.23	
7	Оформление отчёта	15	10.07.23 – 12.07.2023	
	Общий объём часов	108		

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

ОТЧЁТ
О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ

(2022/2023 учебный год)

Кулахметов Сабир Исхакович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Кулахметов С.И. 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. Также, осуществил работу с файлами. Протестировал и отладил программу. Оформил отчёт.

Бакалавр Кулахметов С.И. _____ " ____ " _____ 2023
г.

Руководитель Зинкин С.А. _____ " ____ " _____ 2023 г.
практики

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

ОТЗЫВ

О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ

(2022/2023 учебный год)

Кулахметов Сабир Исхакович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств
вычислительной техники и автоматизированных систем»

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

Год обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

В процессе выполнения практики Кулахметов С.И. решал следующие задачи:
создание программы, анализ работы алгоритма.

За выполнение работы Кулахметов С. И. заслуживает оценки « ».

Руководитель практики д.т.н., профессор, Зинкин С.А. « » 2023 г.

Содержание

Введение.....	7
Постановка задачи	8
1.1 Преимущества	8
1.2 Недостатки.....	8
1.3 Типичные сценарии применения.....	9
Выбор решения	10
Описание программы.....	12
Схемы программы.....	15
4.1 Блок-схема программы	15
Тестирование программы	19
5.2 Анализ работы алгоритма	19
Отладка.....	20
Совместная работа	21
Заключение.....	22
Список используемой литературы	23
Приложение А.....	24
Приложение Б	29

Введение

Microsoft Visual Studio — это интегрированная среда разработки (IDE), созданная компанией Microsoft. Она используется для разработки различных программных приложений, включая настольные, веб-, мобильные и облачные приложения. Visual Studio предоставляет обширный набор инструментов и функций, которые помогают разработчикам на протяжении всего жизненного цикла разработки программного обеспечения.

Некоторые ключевые особенности Visual Studio включают:

1. Редактор кода: Visual Studio предлагает мощный редактор кода с поддержкой IntelliSense, поддерживает несколько языков программирования, включая C#, VB.NET, F#, C++, JavaScript и другие.
2. Инструменты отладки.
3. Система проектов.
4. Интегрированное тестирование.
5. Сотрудничество и контроль версий.
6. Расширяемость.
7. Разработка облачных приложений.

Язык программирования Си (C) является одним из наиболее популярных и влиятельных языков программирования. Он был разработан в начале 1970-х годов Деннисом Ритчи в Bell Labs и с тех пор стал широко используемым в индустрии разработки программного обеспечения.

Язык программирования Си остается популярным среди разработчиков, особенно в области системного программирования, встроенных систем и разработки низкоуровневого ПО. Он также является хорошим выбором для изучения основ программирования, поскольку множество концепций и подходов, применяемых в Си, переносятся на другие языки программирования.

Постановка задачи

1.1 Преимущества

- 1) Простота реализации: Шейкерная сортировка легко реализуется и понимается. Она не требует сложных структур данных или специальных операций.
- 2) Устойчивость: Алгоритм шейкерной сортировки является устойчивым, что означает, что элементы с одинаковыми значениями сохраняют свой относительный порядок после сортировки.
- 3) Хорошая производительность в некоторых случаях: Шейкерная сортировка может быть эффективна на частично отсортированных списках или списках с небольшим количеством элементов, так как она может обнаружить предварительно отсортированную часть и прекратить дальнейшую обработку.

1.2 Недостатки

- 1) Низкая эффективность в худшем случае: Шейкерная сортировка имеет квадратичную сложность времени в худшем случае, то есть время выполнения зависит от квадрата количества элементов. Это делает ее неэффективной для больших массивов данных.
- 2) Ограниченная эффективность на случайных данных: По сравнению с некоторыми более сложными алгоритмами сортировки, шейкерная сортировка может быть менее эффективной при работе с случайно упорядоченными данными или данными, содержащими большое количество повторяющихся значений.
- 3) Дополнительные затраты на перемещение элементов: Шейкерная сортировка требует обмена элементов путем их последовательного сравнения и обмена. Это может привести к дополнительным затратам на перемещение элементов в списке.
- 4) Ограниченный выбор алгоритма: В сравнении с другими сортировками, такими как быстрая сортировка или сортировка слиянием, шейкерная сортировка имеет ограниченные возможности и выбор алгоритмов для различных сценариев сортировки.

В целом, шейкерная сортировка имеет свои преимущества и недостатки, и ее эффективность зависит от размера списка, его упорядоченности и требований к производительности в конкретной задаче сортировки.

1.3 Типичные сценарии применения

Шейкерная сортировка может быть полезной в следующих типичных сценариях:

Частично отсортированные списки: Если вы имеете дело с частично отсортированными списками, где большая часть элементов уже находится близко к своим конечным позициям, шейкерная сортировка может быть эффективной. Она может быстро обнаружить уже отсортированную часть списка и сосредоточиться только на оставшихся элементах.

Малые массивы данных: Шейкерная сортировка может быть применена к небольшим массивам данных, где требуется простота реализации, а производительность не является основным фактором. В таких случаях преимущество простоты и понятности алгоритма может перевешивать его неоптимальную производительность.

Образцовые данные: Если у вас есть образцовые данные, в которых известен порядок элементов, шейкерная сортировка может быть применена для проверки правильности сортировки других алгоритмов. Вы можете использовать шейкерную сортировку для сравнения результатов с другими алгоритмами и убедиться, что они работают правильно.

Выбор решения

Алгоритм шейкерной сортировки (CocktailSort), также известный как сортировка перемешиванием или коктейльная сортировка, является улучшением алгоритма пузырьковой сортировки. Он выполняет сортировку элементов списка путем многократного прохода через список, меняя местами соседние элементы, если они находятся в неправильном порядке.

Краткое описание шейкерной сортировки:

1. Начинаем с исходного списка элементов, который нужно отсортировать.
2. Устанавливаем два указателя: один на начало списка (левый указатель), а другой на конец списка (правый указатель).
3. Пока левый указатель меньше правого указателя:
 - 1) Проходим по списку слева направо, сравнивая пары соседних элементов.
 - 2) Если текущий элемент больше следующего элемента, меняем их местами.
 - 3) После каждого прохода справа налево самый большой элемент перемещается в конец списка.
4. Уменьшаем правый указатель на 1 (так как самый большой элемент уже находится в конце списка).
5. Пока левый указатель меньше правого указателя:
 - 1) Проходим по списку справа налево, сравнивая пары соседних элементов.
 - 2) Если текущий элемент меньше предыдущего элемента, меняем их местами.
 - 3) После каждого прохода слева направо самый маленький элемент перемещаем в начало списка.
6. Увеличиваем левый указатель на 1 (так как самый маленький элемент уже находится в начале списка).
7. Повторяем шаги 3-6 до тех пор, пока левый указатель не станет больше или равным правому указателю.

После завершения алгоритма шейкерной сортировки, элементы списка будут упорядочены по возрастанию (или по другому заданному порядку), начиная с первого элемента и заканчивая последним элементом.

Описание программы

В программе для шейкерной сортировки подключены следующие заголовочные файлы: <stdio.h> – стандартный заголовочный файл ввода – вывода; <stdlib.h> - заголовочный файл, который содержит в себе функции, занимающиеся выделением памяти, контролем процесса выполнения программы, преобразованием типов в другие; <string.h> - заголовочный файл, содержащий функции для работы со строками, оканчивающимися на 0, и различными функциями работы с памятью; <locale.h> - заголовочный файл, который используется для задач, связанных с локализацией; <time.h> - заголовочный файл, содержащий типы функции для работы с датой и временем.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <time.h>
```

Два макроса отвечающих за количество сортируемых элементов и длину строки.

```
#define a 10000
#define b 100000
```

Далее подключаем русский язык и функции расчета времени, после выполнения работы алгоритма выводим размер массива и затраченное время на выполнение программы.

```
setlocale(LC_ALL, "RUS"); //подключаем русский язык
clock_t start_time, end_time;
double cpu_time_used;
start_time = clock();

end_time = clock(); // Записываем конечное время

cpu_time_used = ((double)(end_time - start_time)) /
CLOCKS_PER_SEC; // Вычисляем время выполнения в секундах
printf("Размер массива = %d\n", a);
printf("Время выполнения программы: %f секунд\n",
cpu_time_used);
```

После мы создаём метку, на которую переходим при наличии ошибки в записи на указанный файл, далее считываем отсортированный массив из файла.

```
label: //метка для перехода при ошибках

printf("Введите путь или имя файла с расширением: ");
scanf("%s", filename);

start_time = clock(); // Записываем начальное время

//<----- чтение из файла строки и её преобразование в
массив чисел ----->
if ((file = fopen(filename, "r")) == NULL) {
    printf("ошибка при открытии файла %s\n", filename);
    goto label; //переход пометке для совершения повторного
запроса
```

```

}

fgets(string, b, file);    //чтение чисел в качестве строки
printf("\t# Исходный массив #\n%s", string);

fclose(file);

file = fopen("result.txt", "w");
n++;
fprintf(file, "%d", array[0]);

for (int j = 1; j < n; j++) {
    fprintf(file, ", %d", array[j]);    //запись
отсортированных значений в файл
}

fclose(file);

//<----- вывод содержимого отсортированного массива ---
----->
printf("\t# Отсортированный массив #\n");
file = fopen("result.txt", "r");

fgets(string, b, file);
printf("%s", string);

fclose(file);

```

Алгоритм шейкерной сортировки описан следующим образом:

```

while (1) {
    while (string[i] != ',') {
        item[k] = string[i];
        k++;
        i++;
        if (string[i] == '\n') {break;}
    }
    array[n] = atoi(item);    //функция int atoi(char* )
преобразовывает строку в целое число
    if (string[i] == '\n') break;
    n++;
    i++;
    k = 0;
    memset(item, 0, 8); //обнуляем промежуточный массив
}
int right = n;    //правая граница массива
//<----- сортировка массива ----->
while(left < right) {
    //сортировка массива вправо
    for(int j = left; j < right; j++) {
        if(array[j] > array[j + 1]) {    //если первый
элемент больше второго, то меняем их местами
            //перестановка по правилу "Трёх стаканов"
            tmp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = tmp;

```

```

        last = j;
    }
}
right = last;    //двигаем правую границу к индексу
последнего обмена
//сортировка массива влево
for(int j = right - 1; j >= left; j--) {
    if(array[j] > array[j + 1]) {
        //перестановка по правилу "Трёх стаканов"
        tmp = array[j];
        array[j] = array[j + 1];
        array[j + 1] = tmp;
        last = j;
    }
}
left = last++;    //двигаем левую границу к индексу
последнего обмена
}

```

Схемы программы

4.1 Блок-схема программы

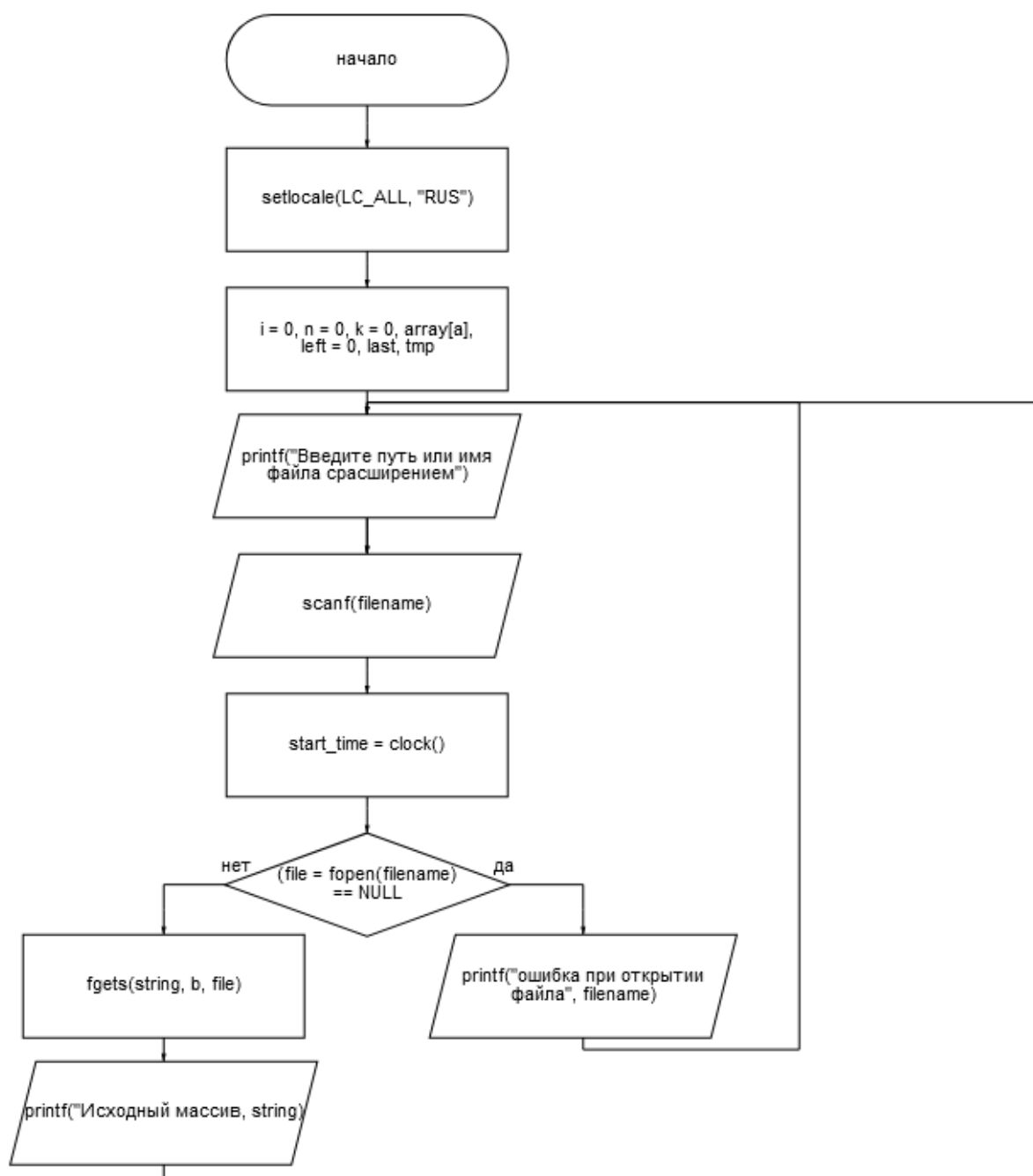
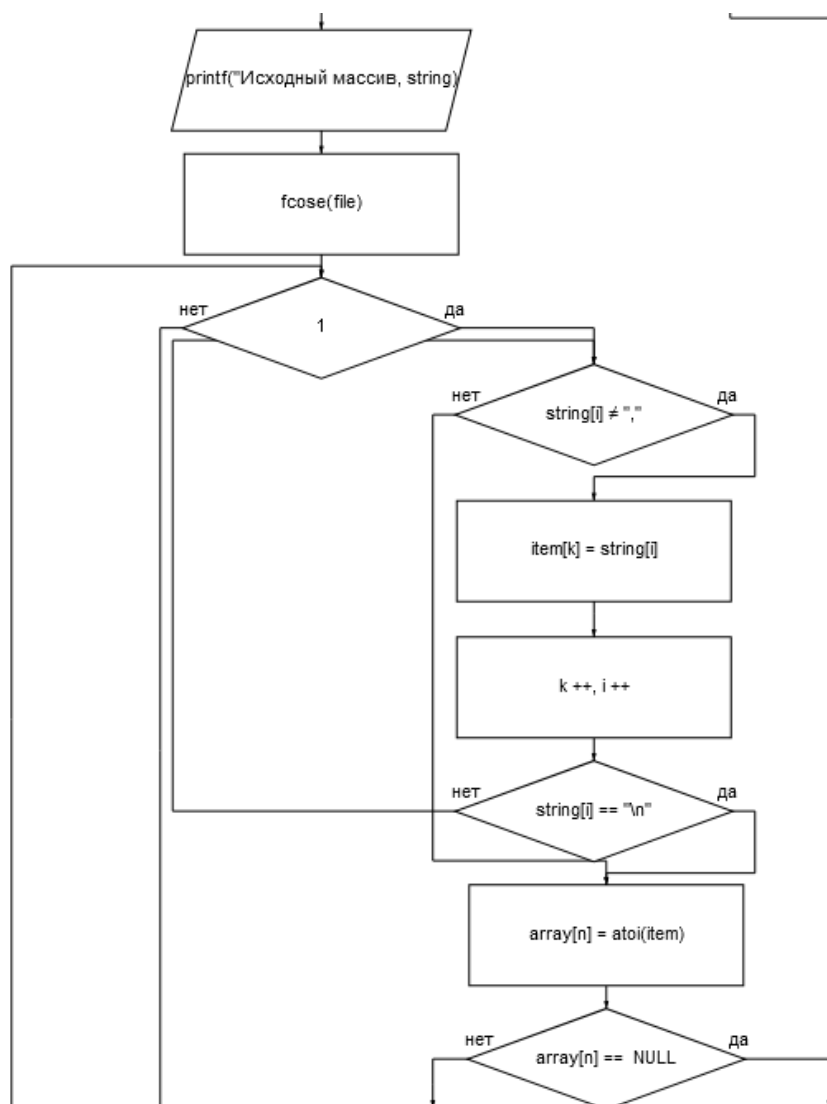
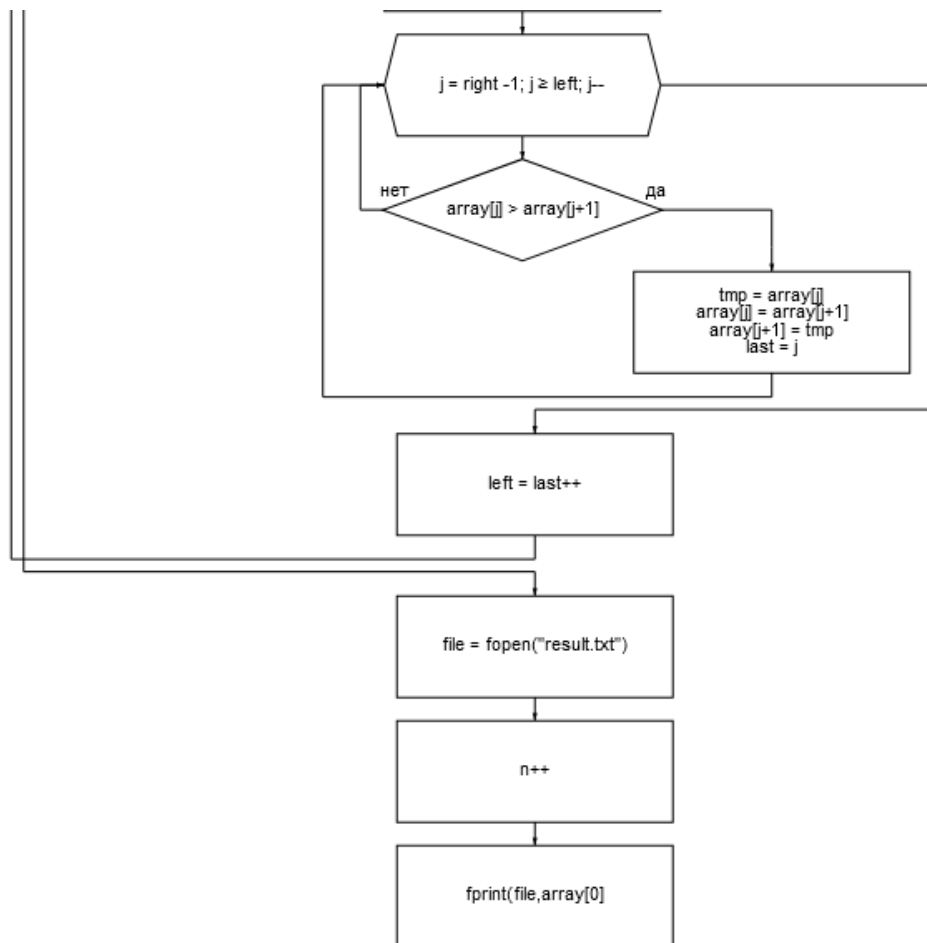
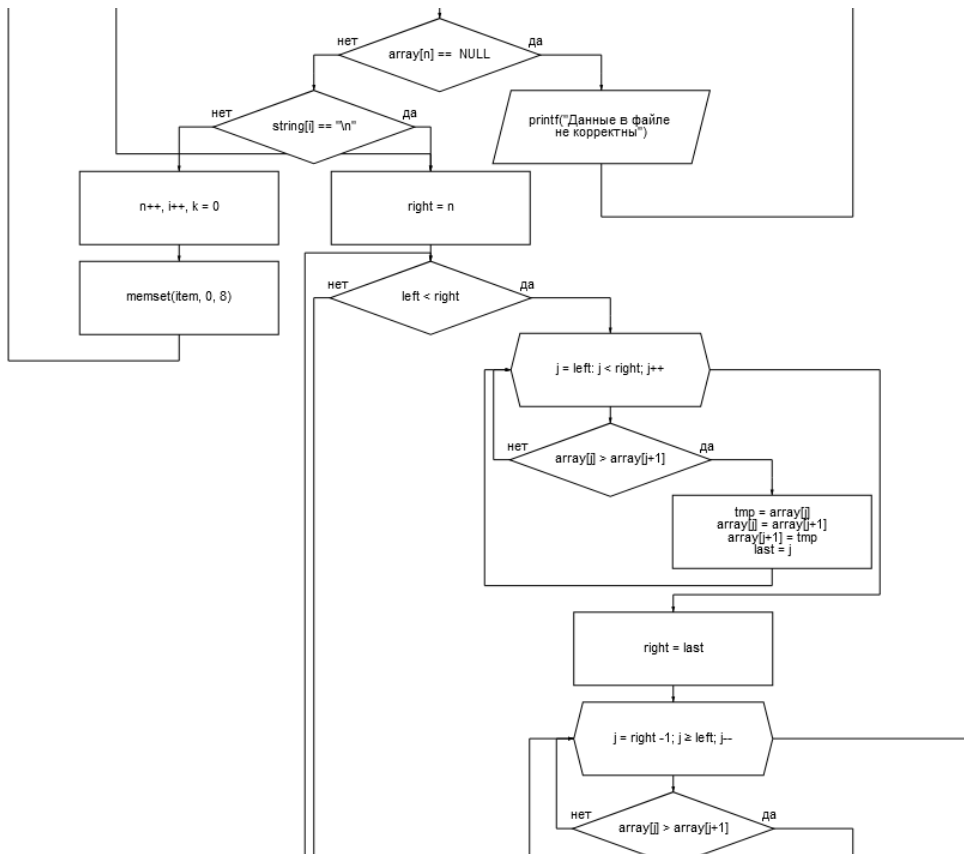


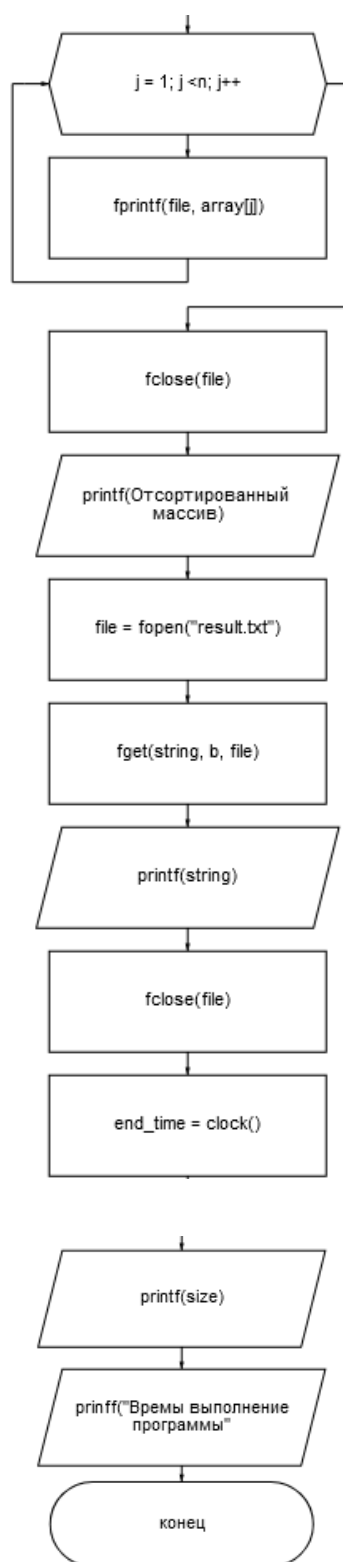
Рисунок 1 – Блок-схема программы



Продолжение Рисунка 1



Продолжение рисунка 1

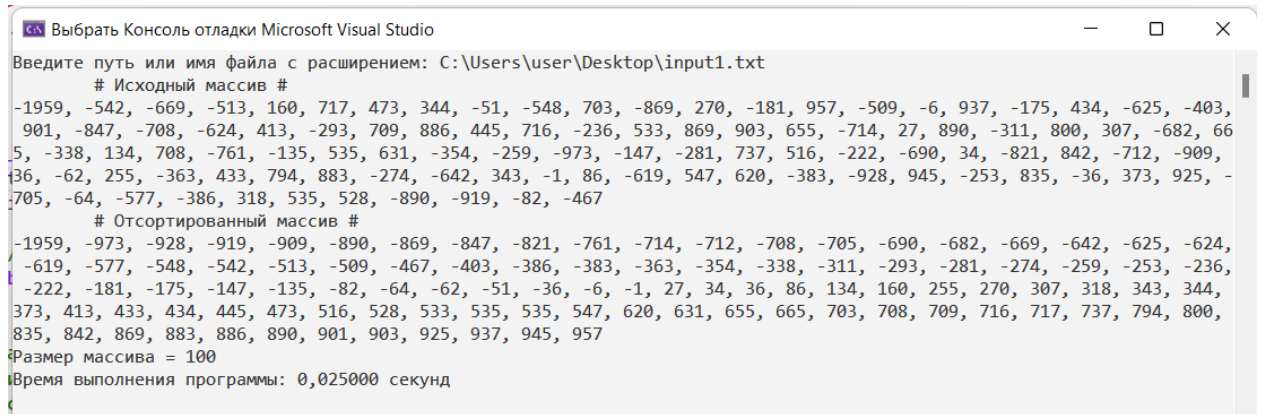


Продолжение рисунка 1

Тестирование программы

5.2 Анализ работы алгоритма

Анализ и реализация графического интерфейса



```
Выборать Консоль отладки Microsoft Visual Studio
Введите путь или имя файла с расширением: C:\Users\user\Desktop\input1.txt
# Исходный массив #
-1959, -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, 66
5, -338, 134, 708, -761, -135, 535, 631, -354, -259, -973, -147, -281, 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, 835, -36, 373, 925, -
705, -64, -577, -386, 318, 535, 528, -890, -919, -82, -467
# Отсортированный массив #
-1959, -973, -928, -919, -909, -890, -869, -847, -821, -761, -714, -712, -708, -705, -690, -682, -669, -642, -625, -624,
-619, -577, -548, -542, -513, -509, -467, -403, -386, -383, -363, -354, -338, -311, -293, -281, -274, -259, -253, -236,
-222, -181, -175, -147, -135, -82, -64, -62, -51, -36, -6, -1, 27, 34, 36, 86, 134, 160, 255, 270, 307, 318, 343, 344,
373, 413, 433, 434, 445, 473, 516, 528, 533, 535, 535, 547, 620, 631, 655, 665, 703, 708, 709, 716, 717, 737, 794, 800,
835, 842, 869, 883, 886, 890, 901, 903, 925, 937, 945, 957
Размер массива = 100
Время выполнения программы: 0,025000 секунд
```

Рисунок 2 – Результаты тестирования

Вывод данных осуществляется на консоль: изначальный массива данных, отсортированный массива данных, размер массива и время выполнения работы программы (в секундах).

Отладка

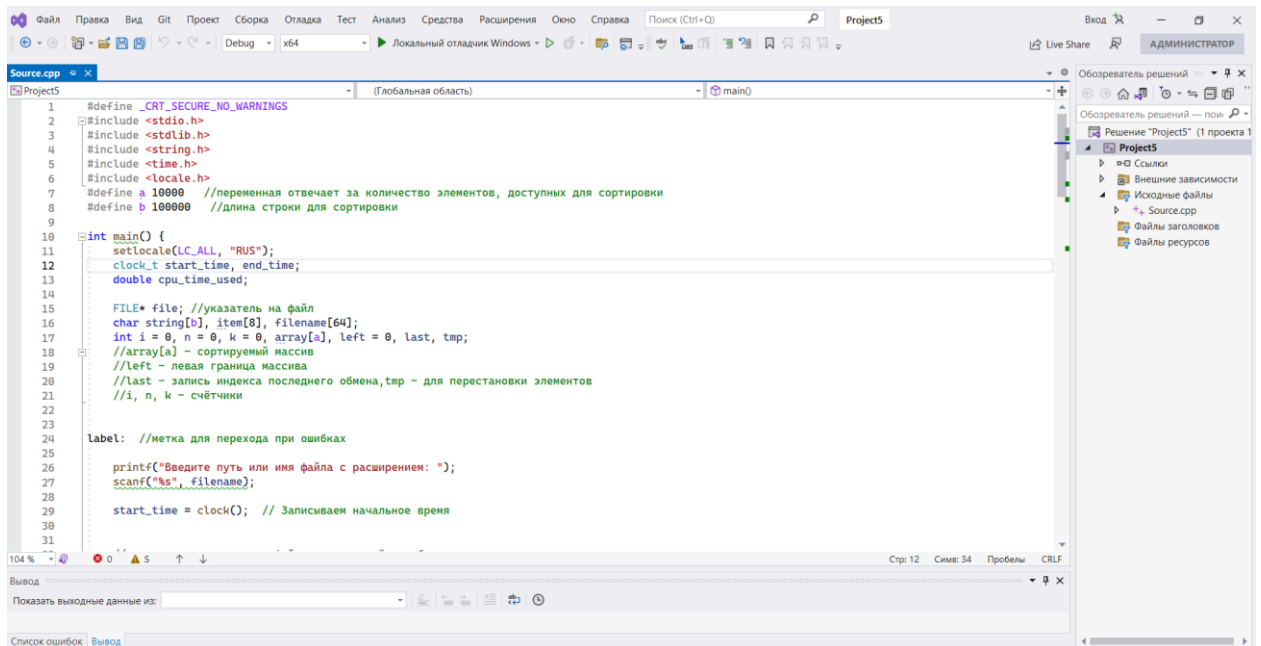


Рисунок 3 – Окно кода

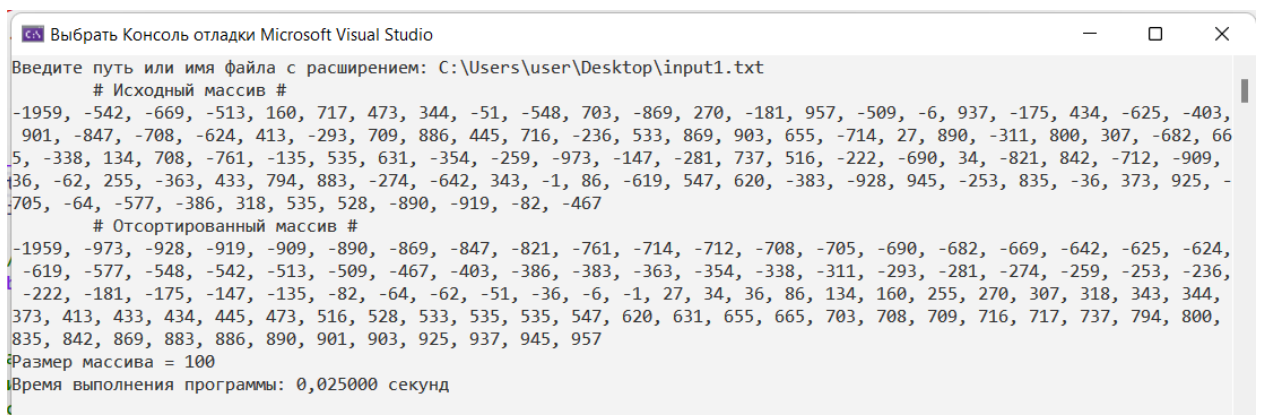




Рисунок 4 – Консоль отладки



Совместная работа

 **ЛетоПрактика** Общественный

Отключить 2

основной 2 отделения 0 тегов Перейти к файлу Добавить файл <> Код

 КулахметовС Финальный релиз 1.0 сбсб2е9 вчера 4 фиксации

 Источник.c	Финальный релиз 1.0	вчера
 main.c	Финальный релиз 1.0	вчера

Помогите людям, заинтересованным в этом репозитории, понять ваш проект, добавив файл README.

Добавить README

Рисунок 5 – Совместная работа Github

Заключение

В результате 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.

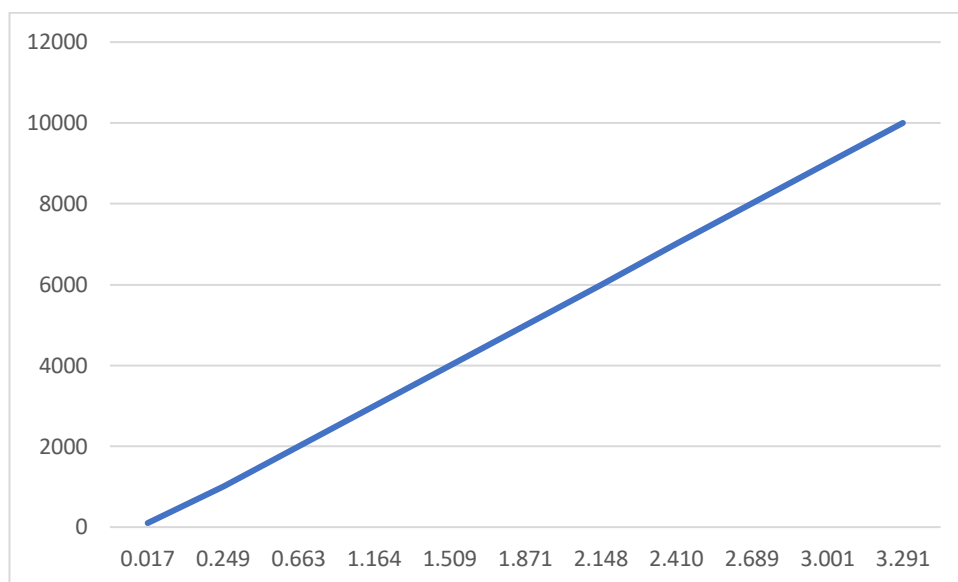


Рисунок 6 – Результаты тестирования

Список используемой литературы

1. Мельников Б. Ф. Алгоритмы – М.: БХВ, 2003. -192с.
2. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ - М.: МЦНМО, 2001. - 960 с.
3. Никлаус Вирт. Алгоритмы и структуры данных – М.:ДМК-Пресс,2016. -272с.
4. Харви Дейтел, Пол Дейтел. Как программировать на C/C++. М.: Бином, 2022г. – 1002с.

Приложение А

```
Выбрать Консоль отладки Microsoft Visual Studio

Введите путь или имя файла с расширением: C:\Users\user\Desktop\input1.txt

# Исходный массив #
-1959, -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, 66
5, -338, 134, 708, -761, -135, 535, 631, -354, -259, -973, -147, -281, 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, 835, -36, 373, 925, -
705, -64, -577, -386, 318, 535, 528, -890, -919, -82, -467

# Отсортированный массив #
-1959, -973, -928, -919, -909, -890, -869, -847, -821, -761, -714, -712, -708, -705, -690, -682, -669, -642, -625, -624,
-619, -577, -548, -542, -513, -509, -467, -403, -386, -383, -363, -354, -338, -311, -293, -281, -274, -259, -253, -236,
-222, -181, -175, -147, -135, -82, -64, -62, -51, -36, -6, -1, 27, 34, 36, 86, 134, 160, 255, 270, 307, 318, 343, 344,
373, 413, 433, 434, 445, 473, 516, 528, 533, 535, 535, 547, 620, 631, 655, 665, 703, 708, 709, 716, 717, 737, 794, 800,
835, 842, 869, 883, 886, 890, 901, 903, 925, 937, 945, 957
Размер массива = 100
Время выполнения программы: 0,021000 секунд
```

Рисунок А.1 – Результаты тестирования

```
Выбрать Консоль отладки Microsoft Visual Studio

7, -1289, -1277, -1256, -1241, -1132, -1091, -1068, -1058, -990, -990, -990, -960, -860, -839, -833, -831, -827, -801, -
788, -768, -713, -709, -687, -687, -686, -667, -664, -651, -643, -643, -640, -626, -504, -497, -491, -488, -486, -485, -
460, -445, -436, -424, -424, -399, -384, -358, -344, -343, -324, -267, -259, -242, -238, -219, -211, -211, -189, -186, -
168, -147, -146, -132, -106, -100, -95, -70, -47, -45, -39, -39, -29, 2, 8, 21, 40, 67, 75, 79, 92, 105, 112, 116, 144,
176, 187, 190, 195, 202, 211, 226, 285, 291, 302, 322, 332, 353, 383, 466, 522, 526, 548, 555, 585, 673, 694, 712, 770,
808, 813, 832, 835, 837, 876, 893, 899, 931, 973, 1000, 1002, 1008, 1020, 1023, 1059, 1075, 1100, 1106, 1110, 1114, 1124
, 1173, 1184, 1192, 1195, 1224, 1239, 1247, 1258, 1285, 1315, 1321, 1323, 1328, 1333, 1337, 1342, 1360, 1422, 1425, 1460
, 1462, 1477, 1478, 1511, 1538, 1555, 1585, 1600, 1626, 1635, 1672, 1701, 1782, 1817, 1833, 1840, 1903, 1927, 1933, 1942
, 1997, 2043, 2044, 2052, 2053, 2059, 2164, 2167, 2169, 2181, 2193, 2208, 2225, 2249, 2256, 2263, 2269, 2287, 2292, 2316
, 2317, 2355, 2382, 2390, 2392, 2403, 2423, 2438, 2455, 2508, 2524, 2529, 2550, 2590, 2603, 2608, 2623, 2636, 2637, 2661
, 2673, 2677, 2684, 2701, 2717, 2722, 2725, 2734, 2756, 2760, 2818, 2835, 2843, 2859, 2896, 2938, 2941, 2949, 2993, 3007
, 3022, 3030, 3031, 3061, 3064, 3142, 3169, 3186, 3261, 3290, 3357, 3401, 3452, 3458, 3584, 3648, 3653, 3694, 3829, 3931
, 3966, 3971, 3977, 3985, 3985, 4008, 4015, 4018, 4146, 4181, 4256, 4270, 4295, 4309, 4310, 4343, 4369, 4413, 4423, 4460
, 4474, 4485, 4604, 4606, 4625, 4688, 4700, 4736, 4771, 4798, 4887, 4893, 4902, 4924, 4932, 4945, 4955, 4962, 4989, 5006
, 5117, 5141, 5145, 5185, 5205, 5255, 5262, 5264, 5281, 5350, 5457, 5498, 5573, 5574, 5629, 5724, 5748, 5759, 5790, 5881
, 5890, 5944, 6036, 6085, 6105, 6118, 6139, 6142, 6202, 6215, 6279, 6282, 6303, 6413, 6423, 6512, 6519, 6541, 6549, 6565
, 6634, 6641, 6687, 6827, 6858, 6941, 6944, 6962, 6972, 7035, 7086, 7103, 7110, 7189, 7192, 7222, 7253, 7371, 7398, 7410
, 7421, 7437, 7451, 7505, 7546, 7549, 7578, 7673, 7713, 7773, 7807, 7825, 7841, 7861, 7864, 7870, 7913, 7958, 7964, 8007
, 8035, 8060, 8087, 8114, 8115, 8127, 8132, 8190, 8230, 8240, 8330, 8443, 8467, 8538, 8540, 8584, 8588, 8636, 8651, 8651
, 8662, 8678, 8696, 8716, 8756, 8762, 8786, 8787, 8823, 8875, 8896, 8935, 8958, 9008, 9037, 9072, 9090, 9156, 9169, 9187
, 9264, 9314, 9353, 9357, 9369, 9375, 9405, 9497, 9558, 9589, 9629, 9668, 9690, 9711, 9718, 9796, 9801, 9815, 9855, 9866
, 9895, 9912, 9923, 9954, 9976
Размер массива = 1000
Время выполнения программы: 0,251000 секунд
```

Рисунок А.2 – Результаты тестирования

```
Выбрать Консоль отладки Microsoft Visual Studio

629, 5667, 5678, 5680, 5697, 5705, 5721, 5734, 5759, 5760, 5770, 5783, 5824, 5824, 5828, 5874, 5898, 5928, 5951, 5990, 5
996, 6018, 6021, 6031, 6048, 6052, 6058, 6063, 6113, 6122, 6129, 6153, 6154, 6173, 6222, 6253, 6264, 6268, 6281, 6292, 6
299, 6302, 6303, 6308, 6362, 6363, 6418, 6423, 6428, 6431, 6434, 6439, 6443, 6463, 6477, 6488, 6489, 6500, 6504, 6517, 6
534, 6576, 6578, 6630, 6633, 6653, 6667, 6679, 6740, 6777, 6806, 6809, 6816, 6821, 6835, 6841, 6857, 6869, 6890, 6909, 6
924, 6946, 6948, 6949, 6962, 6969, 6979, 7007, 7008, 7032, 7065, 7067, 7080, 7088, 7109, 7124, 7152, 7157, 7171, 7196, 7
200, 7216, 7348, 7350, 7358, 7368, 7384, 7415, 7432, 7432, 7446, 7465, 7489, 7506, 7529, 7531, 7577, 7593, 7594, 7595, 7
611, 7624, 7644, 7649, 7663, 7666, 7681, 7726, 7753, 7753, 7756, 7813, 7830, 7870, 7892, 7918, 7936, 7938, 7967, 7982, 8
009, 8019, 8022, 8027, 8070, 8074, 8082, 8112, 8118, 8145, 8149, 8162, 8164, 8182, 8252, 8253, 8255, 8274, 8286, 8289, 8
296, 8297, 8315, 8315, 8318, 8321, 8323, 8350, 8396, 8423, 8433, 8442, 8458, 8464, 8470, 8476, 8486, 8489, 8503, 8510, 8
520, 8522, 8532, 8570, 8583, 8614, 8617, 8633, 8650, 8652, 8664, 8666, 8680, 8685, 8689, 8692, 8703, 8704, 8712, 8745, 8
761, 8768, 8791, 8801, 8869, 8888, 8962, 8978, 9011, 9022, 9033, 9082, 9087, 9105, 9141, 9151, 9152, 9168, 9170, 9174, 9
174, 9200, 9213, 9221, 9253, 9264, 9271, 9288, 9292, 9292, 9301, 9309, 9314, 9331, 9334, 9337, 9350, 9352, 9358, 9361, 9
421, 9426, 9447, 9458, 9486, 9492, 9510, 9541, 9556, 9560, 9565, 9577, 9614, 9617, 9621, 9643, 9657, 9658, 9704, 9734, 9
763, 9790, 9812, 9815, 9855, 9859, 9869, 9873, 9891, 9901, 9916, 9929, 9954, 9956, 9972, 9975, 9996, 10003, 10032, 10038
, 10080, 10093, 10106, 10109, 10114, 10145, 10180, 10188, 10189, 10191, 10195, 10212, 10221, 10227, 10238, 10271, 10300,
10303, 10333, 10422, 10470, 10523, 10524, 10527, 10556, 10568, 10648, 10674, 10693, 10695, 10714, 10771, 10774, 10807,
10814, 10833, 10836, 10838, 10877, 10900, 10910, 10932, 10971, 10974, 10974, 11001, 11003, 11020, 11060, 11063, 11068, 1
1101, 11107, 11111, 11111, 11115, 11163, 11172, 11185, 11196, 11202, 11226, 11240, 11244, 11276, 11286, 11310, 11311, 11
316, 11322, 11329, 11348, 11351, 11361, 11418, 11426, 11450, 11461, 11493, 11497, 11531, 11540, 11556, 11561, 11577, 115
91, 11601, 11627, 11647, 11673, 11754, 11758, 11762, 11783, 11810, 11812, 11818, 11907, 11923, 11928, 11934, 11947, 1198
2, 11998, 12029, 12060, 12075, 12168, 12170, 12179, 12182, 12196, 12209, 12226, 12257, 12270, 12356, 12372, 12391, 12404
, 12439, 12498, 12525, 12528, 12544, 12584, 12588, 12591, 12604, 12607, 12609, 12611, 12637, 12662, 12671, 12678, 12685,
12702, 12718, 12726, 12745, 12756, 12757
Размер массива = 2000
Время выполнения программы: 0,548000 секунд
```


Приложение Б

Листинг

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <locale.h>
#define a 10000 //переменная отвечает за количество элементов,
//доступных для сортировки
#define b 100000 //длина строки для сортировки

int main() {
    setlocale(LC_ALL, "RUS");
    clock_t start_time, end_time;
    double cpu_time_used;

    FILE* file; //указатель на файл
    char string[b], item[8], filename[64];
    int i = 0, n = 0, k = 0, array[a], left = 0, last, tmp;
    //array[a] - сортируемый массив
    //left - левая граница массива
    //last - запись индекса последнего обмена, tmp - для
    перестановки элементов
    //i, n, k - счётчики

    label: //метка для перехода при ошибках

    printf("Введите путь или имя файла с расширением: ");
    scanf("%s", filename);

    start_time = clock(); // Записываем начальное время

    //<----- чтение из файла строки и её преобразование в
    массив чисел ----->
    if ((file = fopen(filename, "r")) == NULL) {
        printf("ошибка при открытии файла %s\n", filename);
        goto label; //переход пометке для совершения повторного
запроса
    }

    fgets(string, b, file); //чтение чисел в качестве строки
    printf("\t# Исходный массив #\n%s", string);

    fclose(file);

    while (1) { //тут начинается магия
        while (string[i] != ',') {
            item[k] = string[i];
            k++;
            i++;
            if (string[i] == '\n') { break; }
        }

        array[n] = atoi(item); //функция int atoi(char* )
        преобразовывает строку в целое число
```

```

        if (array[n] == NULL) { //проверка на корректность
вводимых данных
            printf("Данные в файле не корректны!\nИсправьте их
или выберите другой файл!\n");
            goto label; //переход по метке в случае ошибки
        }

        if (string[i] == '\n') break;
        n++;
        i++;
        k = 0;
        memset(item, 0, 8); //обнуляем промежуточный массив
    }

    int right = n; //правая граница массива

    //<----- сортировка массива ----->
    while (left < right) {
        //сортировка массива вправо
        for (int j = left; j < right; j++) {
            if (array[j] > array[j + 1]) { //если первый
элемент больше второго, то меняем их местами
                //перестановка по правилу "Трёх стаканов"
                tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
                last = j;
            }
        }
        right = last; //двигаем правую границу к индексу
последнего обмена

        //сортировка массива влево
        for (int j = right - 1; j >= left; j--) {
            if (array[j] > array[j + 1]) {
                //перестановка по правилу "Трёх стаканов"
                tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
                last = j;
            }
        }
        left = last++; //двигаем левую границу к индексу
последнего обмена
    }

    //<----- запись полученного результата в файл -----
-->
    file = fopen("result.txt", "w");
    n++;
    fprintf(file, "%d", array[0]);

    for (int j = 1; j < n; j++) {
        fprintf(file, ", %d", array[j]); //запись
отсортированных значений в файл
    }

```

```

fclose(file);

//<----- вывод содержимого отсортированного массива ---
----->
printf("\t# Отсортированный массив #\n");
file = fopen("result.txt", "r");

fgets(string, b, file);
printf("%s", string);

fclose(file);

end_time = clock(); // Записываем конечное время

cpu_time_used = ((double)(end_time - start_time)) /
CLOCKS_PER_SEC; // Вычисляем время выполнения в секундах
printf("\n");
printf("Размер массива = %d\n", n);
printf("Время выполнения программы: %f секунд\n",
cpu_time_used);

return 0;
}

```