Оглавление

[Введение 2](#_Toc139309368)

[Постановка задачи 3](#_Toc139309369)

[Выбор решения 5](#_Toc139309370)

[Описание программы 7](#_Toc139309371)

[Схемы программы 9](#_Toc139309372)

[Тестирование программы 12](#_Toc139309373)

[Отладка 14](#_Toc139309374)

[Совместная разработка 15](#_Toc139309375)

[Заключение 17](#_Toc139309376)

[Список используемой литературы 18](#_Toc139309377)

[Приложение А 19](#_Toc139309378)

[Приложение Б 24](#_Toc139309379)

# Введение

Быстрая сортировка Шелла, часто называемая просто сортировкой Шелла, является эффективным и быстрым алгоритмом сортировки, предназначенным для упорядочивания элементов в массиве. Этот алгоритм был разработан Дональдом Шеллом в 1959 году и считается первым усовершенствованным алгоритмом сортировки.

Сортировка Шелла основана на концепции «сортировки вставками», но с добавлением сравнений и обменов элементов, находящихся на определенном расстоянии друг от друга. Идея заключается в том, чтобы предварительно упорядочить элементы с большими интервалами, а затем постепенно сокращать эти интервалы до значения 1. В результате элементы, находящиеся близко к своим истинным позициям, становятся уже частично отсортированными, что делает окончательную сортировку быстрее.

# Постановка задачи

По программе: Необходимо заполнить массив из n-ого количества элементов случайными числами, записать данные элементы в отдельный файл. После этого выполнить сортировку Шелла над данными, находящимися в массиве, записать отсортированные данные в другой файл.

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

Оформить отчет по проведенной практике.

**Достоинства алгоритма сортировки Шелла:**

* Высокая производительность: Сортировка Шелла обычно работает значительно быстрее простых алгоритмов сортировки, таких как сортировка пузырьком или сортировка вставками. Благодаря предварительной упорядоченности элементов на каждой итерации, она сокращает количество сравнений и обменов, что приводит к улучшению производительности.
* Активность: Сортировка Шелла имеет способность адаптироваться к специфическим свойствам входных данных. Различные интервалы сортировки могут использоваться для разных подмножеств элементов, что позволяет алгоритму справляться с различными типами данных и размерами массивов.
* Простота реализации: Алгоритм сортировки Шелла относительно прост в реализации и понимании. Он не требует использования сложных структур данных или рекурсии, что делает его доступным для разработчиков с разным уровнем опыта.

**Недостатки алгоритма сортировки Шелла:**

* Нестабильность: Сортировка Шелла не является стабильной сортировкой, что означает, что равные элементы могут изменить свой относительный порядок после сортировки. Если сохранение порядка равных элементов критически важно, то нужно использовать другой алгоритм сортировки.
* Неоднозначность выбора интервалов: Оптимальный выбор интервалов для сортировки Шелла не всегда очевиден. Различные интервалы могут приводить к различным производительностям алгоритма, и определение наилучшего интервала может требовать некоторого экспериментирования или анализа данных.
* Чувствительность к входным данным: Сортировка Шелла может быть менее эффективной на определенных типах данных или при особых распределениях значений. Например, если данные уже частично отсортированы или имеют особую структуру, то сортировка Шелла может потерять свои преимущества и не показывать оптимальную производительность.

**Типичные сценарии применения:**

* Сортировка списков интернет магазинов.
* Сортировка задач по срокам выполнения.
* Сортировка результатов спортивного соревнования.

# Выбор решения

Нашей бригадой было выбрано вести разработку в среде Microsoft Visual Studio на языке Си.

Язык программирования Си, разработанный в начале 1970-х годов в Bell Laboratories, является одним из самых влиятельных и широко используемых языков программирования в истории компьютеров. Си был создан Деннисом Ритчи для разработки операционной системы UNIX и с тех пор стал одним из наиболее популярных языков программирования.

Си известен своей простотой и эффективностью, что делает его предпочтительным выбором для разработки низкоуровневого программного обеспечения и системного программирования. Благодаря своей компактности и непосредственности, Си стал основным языком для разработки операционных систем, компиляторов, встроенных систем и других приложений, где требуется максимальная производительность.

Язык программирования Си известен своим низкоуровневым характером, который позволяет разработчикам иметь прямой контроль над аппаратными возможностями компьютера. Он предоставляет мощные средства для управления памятью, операциями с указателями и битовыми операциями, что делает его особенно полезным для создания эффективных и оптимизированных алгоритмов.

Кроме того, Си является основой для множества других языков программирования, таких как C++, C#, Objective-C и многих других. Базовые принципы Си, такие как структуры данных, управление памятью и процедурное программирование, остаются важными для понимания и работы с другими современными языками программирования.

Несмотря на свою длительную историю, Си остается активно используемым языком программирования в различных сферах, включая разработку операционных систем, встроенных систем, игр, научных вычислений и многих других областей. Си предлагает разработчикам гибкость, производительность и контроль, делая его неотъемлемым инструментом для множества проектов и задач.

После ввода n-ого количества элементов происходит их запись в файл input.txt и их сохранение в нём.

При выполнении алгоритма итоговый отсортированный список помещается в файл sorted.txt и сохраняется там.

# Описание программы

При запуске программы происходит вывод сообщения «Введите количество элементов: », после чего Пользователю необходимо вписать нужное количество сортируемых данных.

После того, как пользователь ввёл данные, генерируется массив из случайных чисел, эти числа записываются в файл «input.txt».

Затем идёт функция saveArrayToFile, с помощью которой данные записываются в созданный файл.

void saveArrayToFile(int arr[], int n, const char\* filename) {

// Открытие файла в режиме записи

FILE\* file = fopen(filename, "w");

// Проверка успешного открытия файла

if (file != NULL) {

// Запись каждого элемента массива в файл, разделенных пробелами

for (int i = 0; i < n; i++) {

fprintf(file, "%d ", arr[i]);

}

// Закрытие файла после записи

fclose(file);

}

}

Далее вызывается функция shellSort, которая сортирует данные, находящиеся в массиве.

В конце программы отсортированные данные записываются в другой файл «sorted.txt».

В процессе тестировки в программу вводится таймер, позволяющий совершать отладку и тестировку.

// Засечение времени начала сортировки с помощью 'clock' для измерения времени выполнения сортировки

clock\_t start\_time = clock();

…

// Засечение времени окончания сортировки с помощью 'clock' для измерения времени выполнения

clock\_t end\_time = clock();

…

// Вычисление времени выполнения сортировки

double execution\_time = (double)(end\_time - start\_time) / CLOCKS\_PER\_SEC;

// Вывод времени выполнения на консоль

printf("Время выполнения сортировки: %.4f сек\n", execution\_time);

Подробный алгоритм работы программы и функций сортировки и записи массивов в файлы представлен в пункте 5 на рисунке 1, рисунке 2 и рисунке 3.

# Схемы программы



Рисунок 1. Схема функции shellSort.



Рисунок 2. Схема функции saveArrayToFile.

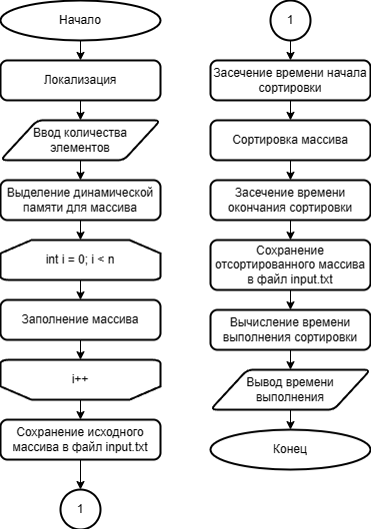


Рисунок 3. Схема функции main.

# Тестирование программы

Тестовый набор данных представлен в таблице 1.

Снизу представлена диаграмма 1, на которой показана линейная зависимость времени от количества данных.

Результаты тестирования приведены в Приложении А на рисунках

А.5 - А.14.

**Таблица 1.** Тестовый набор данных на диапазоне от 10000 до 100000.

|  |  |  |
| --- | --- | --- |
|  | Кол-во данных | Время выполнения сортировки (сек) |
| 1 | 10000 | 0,002 |
| 2 | 20000 | 0,007 |
| 3 | 30000 | 0,011 |
| 4 | 40000 | 0,020 |
| 5 | 50000 | 0,018 |
| 6 | 60000 | 0,022 |
| 7 | 70000 | 0,036 |
| 8 | 80000 | 0,033 |
| 9 | 90000 | 0,038 |
| 10 | 100000 | 0,042 |

**Диаграмма 1.** Тестирование программы на диапазоне от 10000 до 100000.



В результате выполнения тестирования было выявлено время работы данного алгоритма и получен конечный результат, описанный в таблицах.

Тестирование данной программы производилось с помощью встроенных строчек кода, реализующих таймер, засекающий время работы программы.

Коннов А.Д. выполнил тестировку и полученные результаты внёс в таблицу 1.

Тестирование показало, что при увеличении количества элементов массива увеличивается время, необходимое для их сортировки, что даёт линейную зависимость, показанную на диаграмме 1.

# Отладка

В качестве среды разработки была выбрана программа Microsoft Visual Studio , 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 глобальных и локальных переменных.

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

# Совместная разработка

Во время работы над данной практикой, нашей бригадой осуществлялась совместная работа в GitHub.

Алгоритм сортировки Шелла был написан владельцем репозитория – Ю. Митрошиным.

После написания программы, она была выгружена на удаленный репозиторий GitHub в отдельную ветку под названием main..

(См. Рисунок 4)

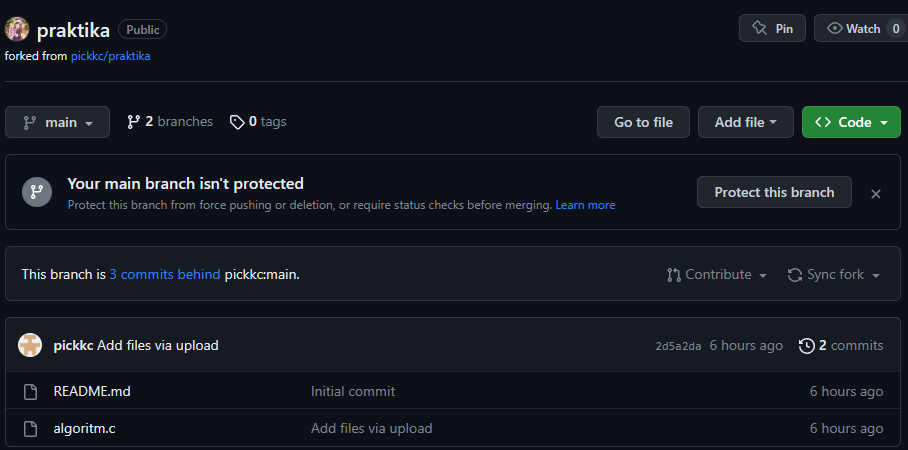


Рисунок 4.

После этого, второй участник – Коннов А.Д. загрузил данную программу себе на компьютер и добавил в нее алгоритм, считающий время работы программы в секундах, запись массива в файлы и написал основную функцию программы. После чего создал собственную ветку, и выгрузил обновленный код программы на удаленный репозиторий GitHub. Также был выгружены отдельно от основной программы добавленные в неё новые алгоритм и основная функция.

(См. Рисунок 5)

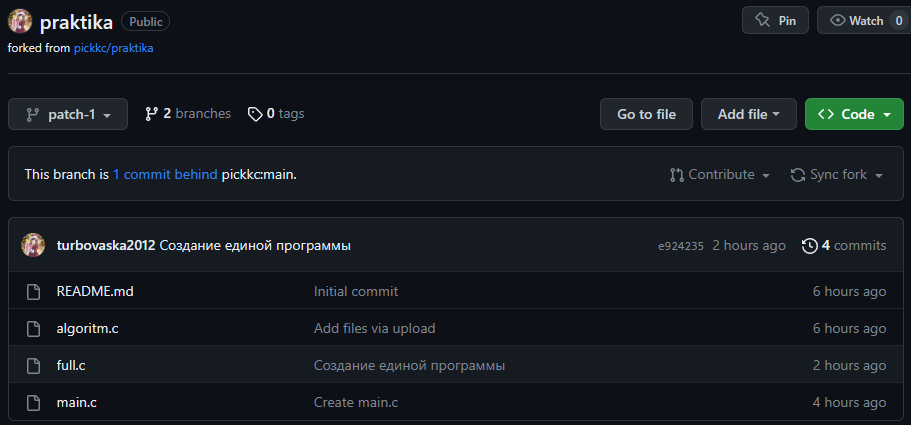


Рисунок 5.

Ссылка на удаленный репозиторий:

<https://github.com/pickkc/praktika>

# Заключение

Нашей бригадой был изучен алгоритм быстрой сортировки Шелла. Коннов А.Д. добавил к алгоритму время сортировки, запись массива в файлы, выполнил тестирование и отладку данной программы.

Так же при выполнении практической работы были улучшены наши базовые навыки программирования на языках С/C++. 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. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.

2. А.А. Тюгашев. Языки программирования. Учебное пособие. 2018 г.

4. [Деннис Ритчи](https://yandex.ru/search/?clid=2456108&text=%D0%94%D0%B5%D0%BD%D0%BD%D0%B8%D1%81%20%D0%A0%D0%B8%D1%82%D1%87%D0%B8&lr=49&noreask=1&ento=0oCglydXcxOTM5MjUYAioKcnV3MTU5ODU4MWou0K_Qt9GL0Log0L_RgNC-0LPRgNCw0LzQvNC40YDQvtCy0LDQvdC40Y8g0KHQuHIM0JDQstGC0L7RgNGLlfhdsQ), [Брайан Керниган](https://yandex.ru/search/?clid=2456108&text=%D0%91%D1%80%D0%B0%D0%B9%D0%B0%D0%BD%20%D0%9A%D0%B5%D1%80%D0%BD%D0%B8%D0%B3%D0%B0%D0%BD&lr=49&noreask=1&ento=0oCglydXcyNDY3MjAYAioKcnV3MTU5ODU4MWou0K_Qt9GL0Log0L_RgNC-0LPRgNCw0LzQvNC40YDQvtCy0LDQvdC40Y8g0KHQuHIM0JDQstGC0L7RgNGLGjVc_A). Язык программирования Си. 1978 г.

5. Солдатенко, И.С. Основы программирования на языке Си: учеб. Пособие. 2017г.

8. [Кормен, Т.](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%BC%D0%B5%D0%BD,_%D0%A2%D0%BE%D0%BC%D0%B0%D1%81), [Лейзерсон, Ч.](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%B9%D0%B7%D0%B5%D1%80%D1%81%D0%BE%D0%BD,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7_%D0%AD%D1%80%D0%B8%D0%BA), [Ривест, Р.](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D0%B2%D0%B5%D1%81%D1%82,_%D0%A0%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%9B%D0%B8%D0%BD%D0%BD" \o "Ривест, Рональд Линн), [Штайн, К.](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D0%B0%D0%B9%D0%BD,_%D0%9A%D0%BB%D0%B8%D1%84%D1%84%D0%BE%D1%80%D0%B4" \o "Штайн, Клиффорд) Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. — 2-е изд.  2005.

# Приложение А

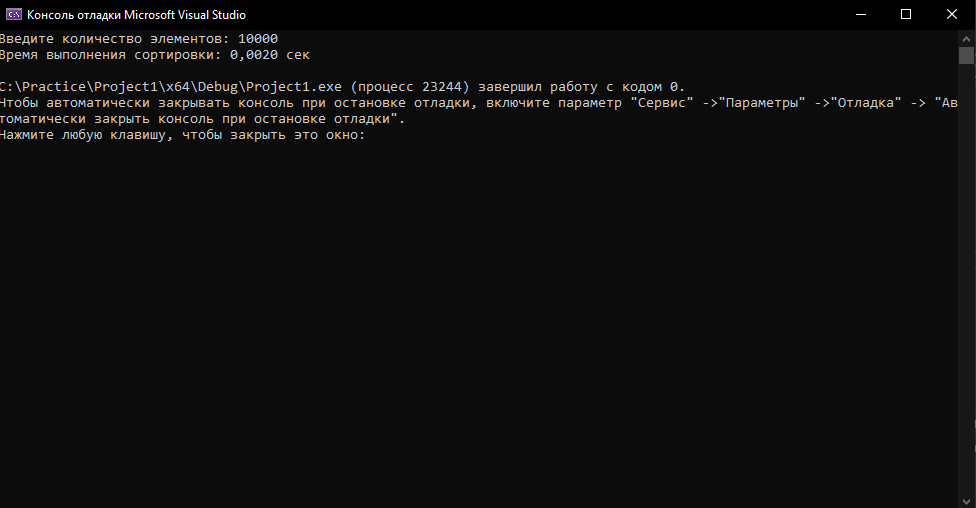


Рисунок А.5.

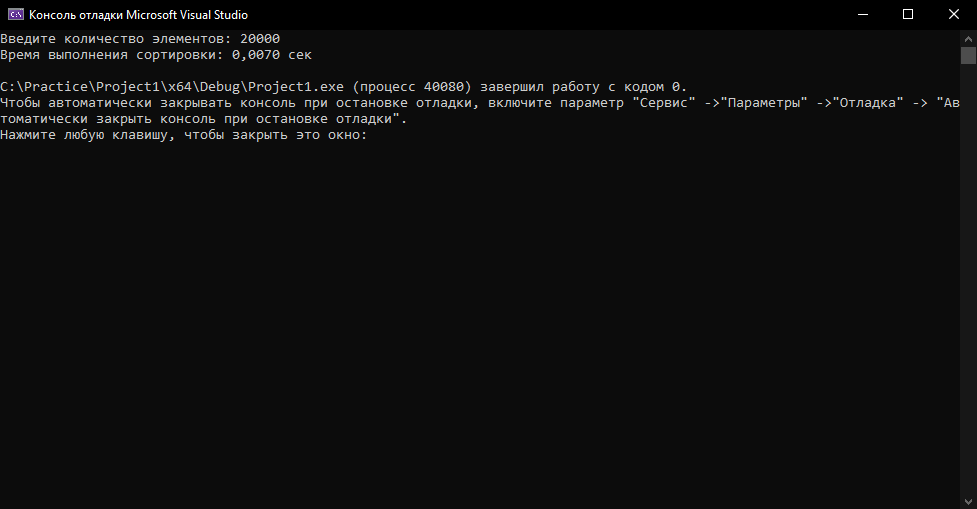


Рисунок А.6.

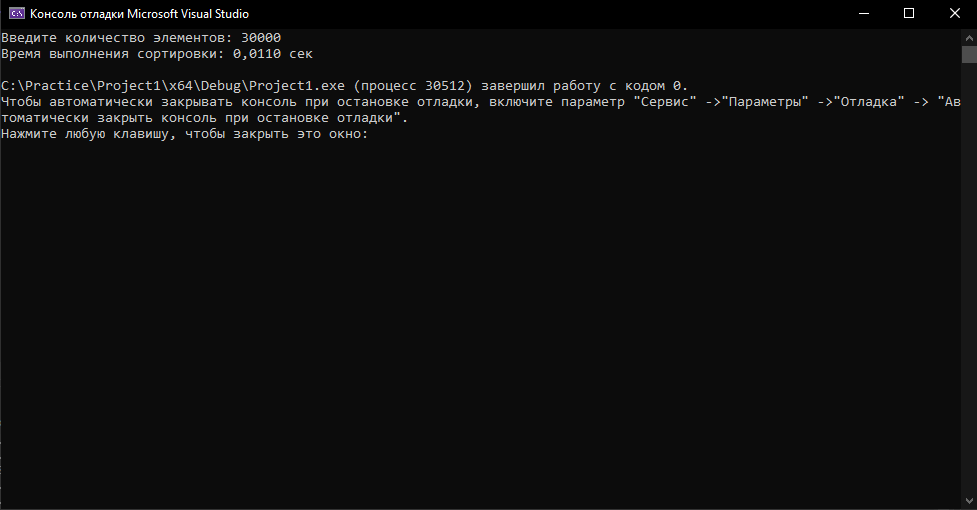


Рисунок А.7.

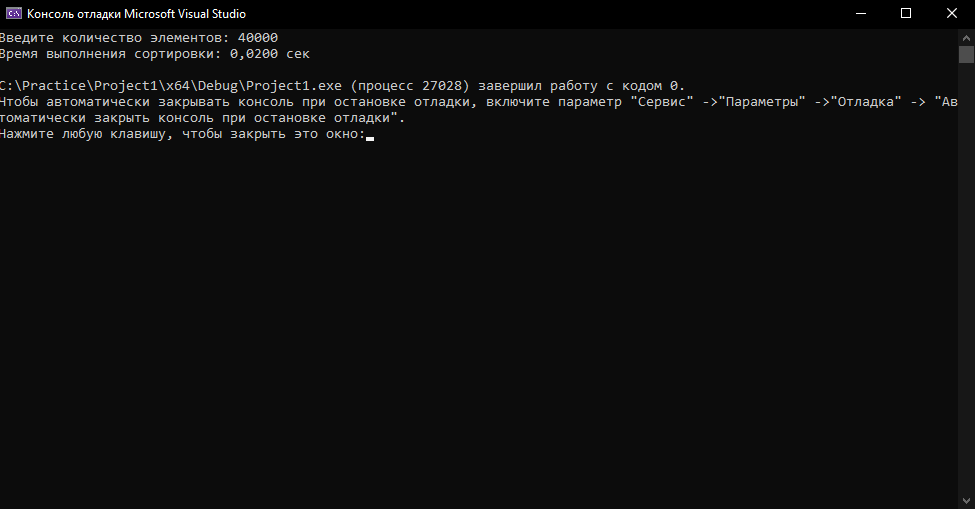


Рисунок А.8.

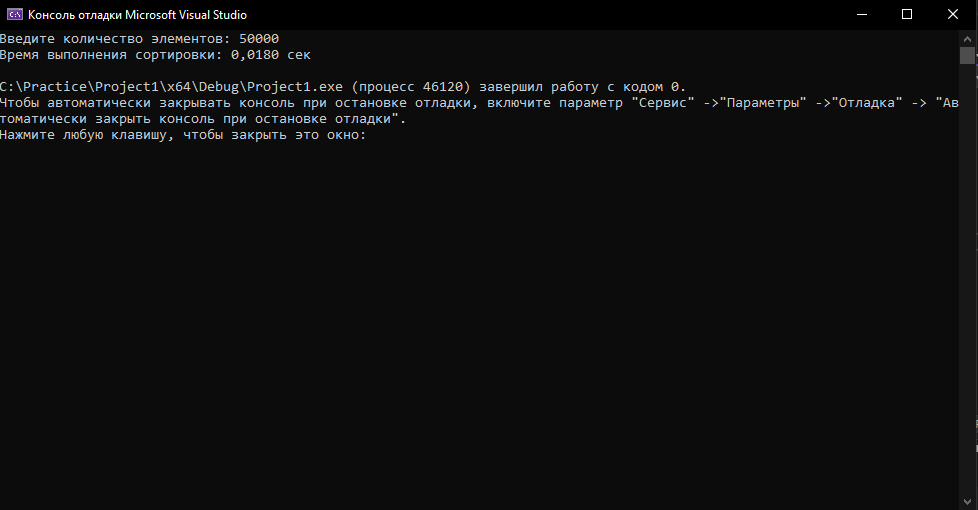


Рисунок А.9.

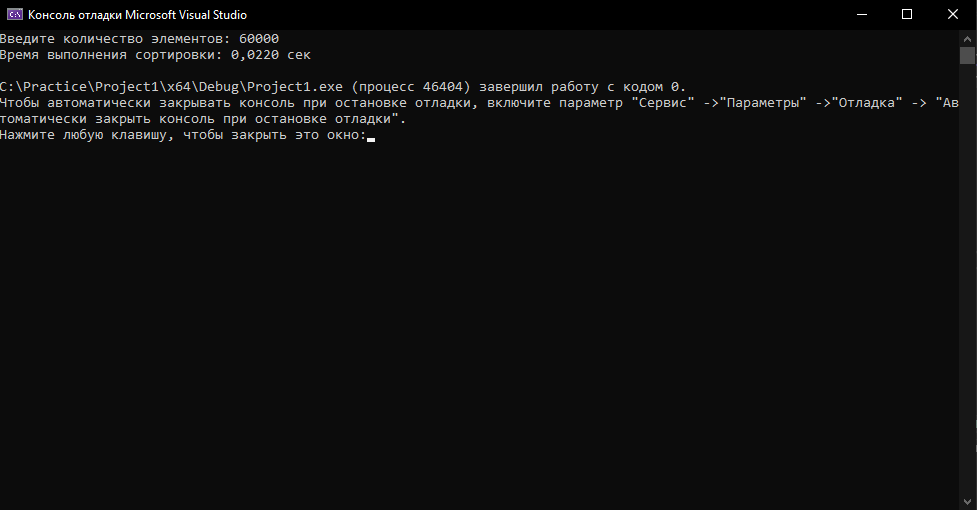


Рисунок А.10.

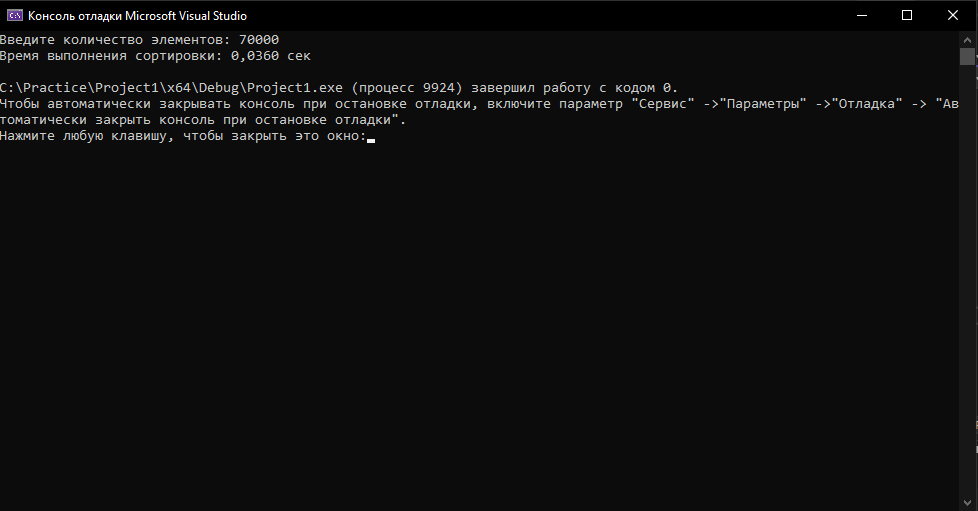


Рисунок А.11.



Рисунок А.12.

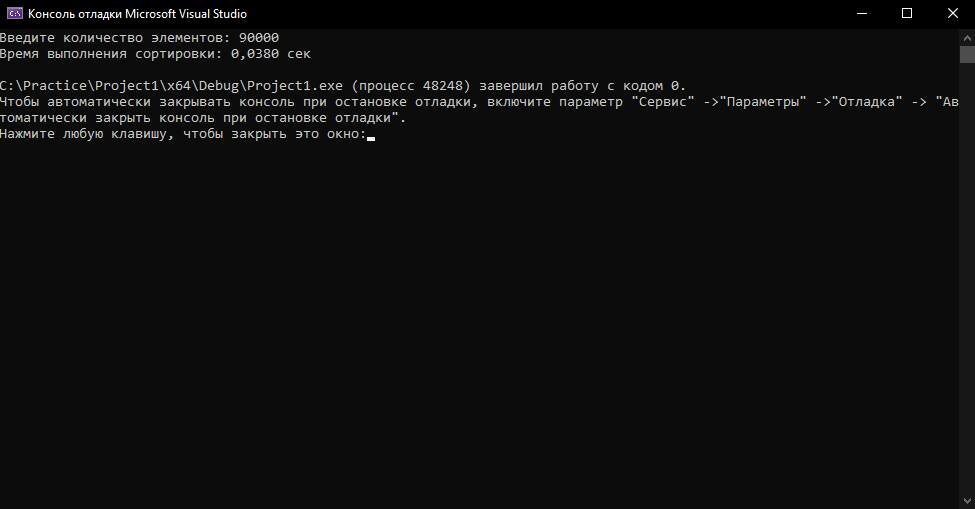


Рисунок А.13.

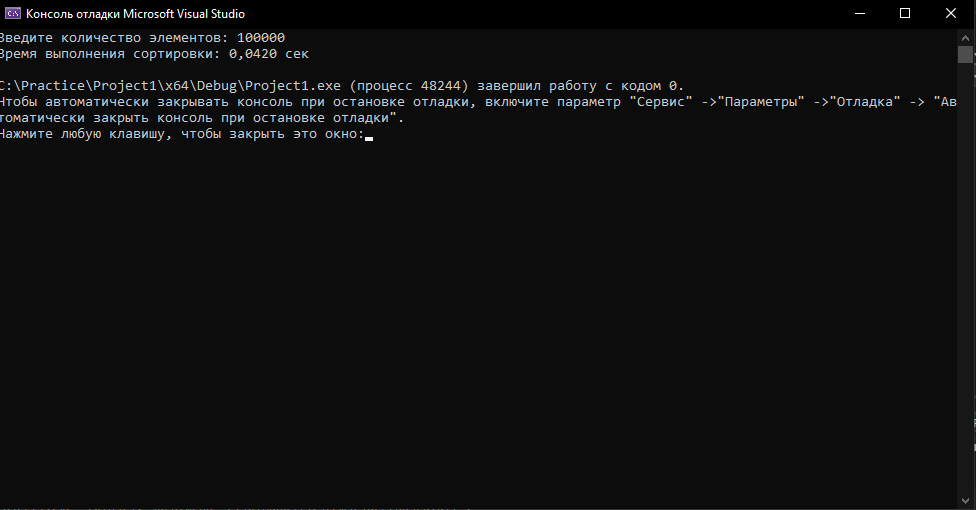


Рисунок А.14.

# Приложение Б

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

// Реализация алгоритма сортировки Шелла

void shellSort(int arr[], int n) {

// Определение начального расстояния между элементами, начиная с максимально возможного расстояния

int gap;

for (gap = n / 2; gap > 0; gap /= 2) {

// Выполнение сортировки вставкой с заданным расстоянием

for (int i = gap; i < n; i++) {

int temp = arr[i];

int j;

// Сравнение элементов с заданным расстоянием и выполнение шага вставки

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {

arr[j] = arr[j - gap];

}

arr[j] = temp;

}

}

}

// Функция для сохранения массива в файл

void saveArrayToFile(int arr[], int n, const char\* filename) {

// Открытие файла в режиме записи

FILE\* file = fopen(filename, "w");

// Проверка успешного открытия файла

if (file != NULL) {

// Запись каждого элемента массива в файл, разделенных пробелами

for (int i = 0; i < n; i++) {

fprintf(file, "%d ", arr[i]);

}

// Закрытие файла после записи

fclose(file);

}

}

// Основная функция

int main() {

// Установка локализации на "Rus" для поддержки русского языка

setlocale(LC\_ALL, "Rus");

// Инициализация генератора случайных чисел с использованием текущего времени

srand(time(0));

// Переменная для хранения количества элементов в массиве

int n;

// Приглашение пользователя ввести количество элементов

printf("Введите количество элементов: ");

// Считывание введенных данных и сохранение в 'n'

scanf("%d", &n);

// Динамическое выделение памяти для массива с помощью 'malloc'

int\* arr = (int\*)malloc(n \* sizeof(int));

// Заполнение массива случайными целыми числами в диапазоне от 0 до 100000

for (int i = 0; i < n; i++) {

arr[i] = rand() % 100001;

}

// Сохранение исходного массива в файл с именем "input.txt"

saveArrayToFile(arr, n, "input.txt");

// Засечение времени начала сортировки с помощью 'clock' для измерения времени выполнения сортировки

clock\_t start\_time = clock();

// Сортировка массива с помощью алгоритма сортировки Шелла

shellSort(arr, n);

// Засечение времени окончания сортировки с помощью 'clock' для измерения времени выполнения

clock\_t end\_time = clock();

// Сохранение отсортированного массива в файл с именем "sorted.txt"

saveArrayToFile(arr, n, "sorted.txt");

// Вычисление времени выполнения сортировки

double execution\_time = (double)(end\_time - start\_time) / CLOCKS\_PER\_SEC;

// Вывод времени выполнения на консоль

printf("Время выполнения сортировки: %.4f сек\n", execution\_time);

// Освобождение выделенной памяти для массива с помощью 'free'

free(arr);

return 0;

}