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)

                                         \_\_Байков Алексей Владимирович

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 обучения1семестр 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)

                                            Байков Алексей Владимирович

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)

                                            Байков Алексей Владимирович

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 обучения1семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Байков А.В выполнял практическое задание «Сортировка вставкой». На первоначальном этапе был изучен и проанализирован алгоритм вставочной сортировки, был выбран метод решения и язык программирования С, на котором была написана работа с файлами. Оформлен отчет.

Бакалавр Байков А.В \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

                                            Байков Алексей Владимирович

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 обучения1семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

В процессе выполнения практики Байков А.В решал следующие задачи: корректная работа с файлами, координация проекта.

За период выполнения практики были освоены основные понятия и технологии сортировки вставкой. Во время выполнения работы Байков А.В показал себя ответственным, добросовестным учеником, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и сортировке.

За выполнение работы Байков А.В заслуживает оценки «\_\_\_\_\_\_».

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

Содержание

[Введение…………………………………………………………………………...](#_gjdgxs) 7

[1 Постановка задачи…………………………………………………………….... 9](#_1fob9te)

[1.1 Достоинства алгоритма…………………………………………………….. 9](#_3znysh7)

[1.2 Недостатки алгоритма……………………………………………………… 9](#_2et92p0)

[1.3 Типичные сценарии применения……………………………………….... 10](#_tyjcwt)

[2 Выбор решения………………………………………………………………... 11](#_3dy6vkm)

[3 Описание программы…………………………………………………………. 13](#_1t3h5sf)

[4 Схема программы……………………………………………………………... 19](#_2s8eyo1)

[5 Тестирование программы…………………………………………………….. 20](#_3rdcrjn)

[6 Совместная разработка……………………………………………………….. 22](#_26in1rg)

[Заключение………………………………………………………………………. 23](#_lnxbz9)

[Список используемой литературы……………………………………………... 24](#_35nkun2)

[Приложение А………………………………………………………………….... 25](#_3j2qqm3)

[Приложение Б Листинг…………………………………………………………. 29](#_1y810tw)

# 

# **Введение**

В настоящее время сортировка данных является одним из наиболее распространенных процессов обработки информации при современном развитии компьютерных технологий. Задачи на сортировку данных возникают в различных профессиональных сферах деятельности и требуют эффективных алгоритмов.

Алгоритмы сортировки образуют отдельный класс алгоритмов и используются практически во всех задачах обработки информации. Они настолько тесно связаны друг с другом, что образуют отдельный класс алгоритмов. Алгоритмы сортировки обычно применяются для упорядочивания данных с целью более эффективного последующего поиска. Например, в словарях слова упорядочены по алфавиту для удобства использования.

Важность сортировки заключается в том, что на ее примере можно показать многие основные фундаментальные приемы и методы построения алгоритмов. Сортировка является примером огромного разнообразия алгоритмов, которые выполняют одну и ту же задачу, и многие из них имеют определенные преимущества перед другими. Усложнение алгоритма может дать значительное увеличение эффективности и быстродействия по сравнению с более простыми методами. Термин "сортировка" обычно понимается как процесс перестановки объектов некоторого множества в определенном порядке.

Сортировка вставками — это простой и эффективный алгоритм, используемый для сортировки заданного списка элементов в порядке возрастания или убывания. Он подпадает под категорию алгоритмов сортировки на основе сравнения.

Основная идея сортировки вставками состоит в том, чтобы разделить список на два подсписка: отсортированный подсписок и несортированный подсписок. Элементы входной последовательности рассматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.[[1]](https://docs.google.com/document/d/199mgpE83Kv7qJ7_5PmFX64jZ48YfC3ncLob_IgIeubU/edit#bookmark=id.35nkun2)

Алгоритм сортировки получает на вход последовательность из n чисел: a1, a2, ..., an, которые также называются ключами. Эти числа представлены в виде массива с n элементами. После сортировки алгоритм должен вернуть перестановку исходной последовательности a1', a2', ..., an', где выполняется следующее условие: a1' <= a2' <= ... <= an'.[[2]](https://docs.google.com/document/d/199mgpE83Kv7qJ7_5PmFX64jZ48YfC3ncLob_IgIeubU/edit#bookmark=id.1ksv4uv)

Алгоритм сортировки вставками начинает работу с пустой отсортированной последовательности. На каждом шаге алгоритма выбирается один элемент из входных данных и помещается на свою позицию в отсортированной последовательности. Это продолжается до тех пор, пока все элементы входных данных не будут отсортированы.

Для сортировки вставками не требуется использования дополнительной памяти, кроме постоянной величины для хранения одного элемента, так как перестановка выполняется в пределах входного массива. В результате работы алгоритма входной массив преобразуется в требуемую выходную последовательность элементов. [[3]](https://docs.google.com/document/d/199mgpE83Kv7qJ7_5PmFX64jZ48YfC3ncLob_IgIeubU/edit#bookmark=id.44sinio)

# **1 Постановка задачи**

Необходимо заполнить массив из n-ого количества элементов псевдослучайными числами, если у пользователя нет файла, а также записать данные элементы в отдельный файл, иначе заполнить массив числами из пользовательского файла. После этого выполнить сортировку вставками над данными, находящимися в массиве, записать отсортированные данные в другой файл, посчитать время выполнения сортировки. Использовать сервис GitHub для совместной работы. Оформить отчет по проделанной практике.

В рамках работы мне необходимо реализовать корректную работу с файлами в зависимости от условий наличия файла.

## **1.1 Достоинства алгоритма**

а) Эффективность для небольших наборов данных;

б) Простота алгоритма;

в) Сортировка на месте: сортировка вставками работает с входным массивом, не требуя дополнительного места в памяти;

г) Онлайн-сортировка. Сортировка вставками хорошо подходит для сценариев онлайн-сортировки, когда элементы постоянно добавляются в массив в режиме реального времени;

д) Простота реализации на разных языках программирования.

## **1.2 Недостатки алгоритма**

а) Сортировка вставками имеет временную сложность O(n^2), где n представляет количество элементов в списке. В худшем случае алгоритму необходимо сравнить каждый элемент в не отсортированном подмножестве со всеми элементами в отсортированном;

б) Не подходит для распараллеливания. Природа сортировки вставками, когда элементы сравниваются и вставляются по одному, затрудняет распараллеливание процесса сортировки;

в) Отсутствие адаптивности: сортировка вставками не использует преимущества любого существующего порядка в наборе данных.

## **1.3 Типичные сценарии применения**

В целом, сортировка вставками лучше всего подходит для небольших наборов данных, частично отсортированных данных, сценариев онлайн-сортировки, сортировки связанных списков и в образовательных целях.

# **2 Выбор решения**

В рамках разработки программы будет использована среда Microsoft Visual Studio 2022 и язык программирования C.

Microsoft Visual Studio 2022, последняя версия популярной интегрированной среды разработки (IDE), представляет собой значительный шаг вперед в средствах разработки программного обеспечения.

Одним из важных аспектов Visual Studio 2022 является расширенная поддержка кроссплатформенной разработки. Эта универсальность позволяет разработчикам охватить более широкую пользовательскую базу и максимизировать потенциальное влияние своих программных решений.

Что касается производительности, Visual Studio 2022 использует множество оптимизаций для улучшения общего опыта разработки. Время запуска IDE значительно сократилось, что позволяет разработчикам быстро приступить к написанию кода без лишних задержек.

Visual Studio 2022 также представляет 64-разрядную версию IDE, которая представляет собой существенное улучшение по сравнению с 32-разрядной предшественницей. Используя преимущества 64-разрядных вычислений, такие как увеличенный объем памяти и повышенная производительность, разработчики могут с легкостью выполнять более сложные проекты и работать с большими кодовыми базами.

Кроме того, Microsoft приложила значительные усилия для расширения возможностей отладки Visual Studio 2022. Обновленный отладчик обеспечивает повышенную надежность, улучшенную диагностику и расширенную поддержку различных языков программирования.

В результате разработчики могут использовать Visual Studio 2022 для создания передовых приложений, отвечающих требованиям современного программного обеспечения.

Язык программирования C является универсальным и широко используемым языком, который широко используется в различных областях компьютерных наук и разработки программного обеспечения.

Одной из ключевых характеристик C является его тесная связь с лежащей в его основе аппаратной архитектурой. C обеспечивает прямой доступ к памяти, позволяя программистам писать эффективный и оптимизированный код, который может в полной мере использовать аппаратные ресурсы. Этот низкоуровневый элемент управления обеспечивает точную настройку производительности, что делает язык C идеальным выбором для сред с ограниченными ресурсами.

Еще одной примечательной особенностью языка C является мощная поддержка модульного программирования. Язык предоставляет механизмы, такие как функции и структуры, которые упрощают организацию кода и его повторное использование. Благодаря использованию библиотек и файлов заголовков разработчики могут создавать повторно используемые компоненты, продвигая модульность кода и облегчая сотрудничество между программистами.

Портативность — еще одна сильная сторона языка программирования Си. Компиляторы C доступны для широкого спектра платформ, включая различные операционные системы и аппаратные архитектуры. Эта межплатформенная совместимость позволяет разработчикам писать код, который можно легко портировать и запускать в разных системах без существенных модификаций.

Кроме того, C повлиял на развитие многих других языков программирования, включая C++, Objective-C и Java.

В заключение, язык программирования C предлагает универсальный и мощный набор инструментов для разработки эффективного и переносимого программного обеспечения. [[4]](https://docs.google.com/document/d/199mgpE83Kv7qJ7_5PmFX64jZ48YfC3ncLob_IgIeubU/edit#bookmark=id.3j2qqm3)

# **3 Описание программы**

В программе для сортировки вставками подключены следующие заголовочные файлы: stdio.h для работы с вводом-выводом; time.h для работы со временем; locale.h для подключения русского языка, malloc.h для работы с динамической памятью; stdlib.h для работы со случайными числами.

#include <stdio.h>

#include <time.h>

#include <locale.h>

#include <malloc.h>

#include <stdlib.h>

Пользователь должен выбрать, сортировать собственный файл или случайно сгенерированный.

// Подтверждение или отрицание наличия файла

printf("Есть файл для сортировки? (Для подтверждения введите Y/y, если нет, то введите N/n, программа будет работать со случайными числами): ");

scanf("%c", &have\_file);

Если пользователь выбрал сортировку несобственного файла, то он попадет под первый сценарий, когда он вводит: имя для файла не отсортированного и отсортированного множества.

// Если введено не "Y\yes", то заполнением файла псевдослучайными числами

if (have\_file != 'Y' && have\_file != 'y')

{

// Имя файла для не отсортированного множества printf("В каком файле сохранить не отсортированное множество?: ");

scanf("%s", &name1);

printf("\n");

// Имя файла для отсортированного множества

printf("В каком файле сохранить отсортированное множество?: ");

scanf("%s", &name2);

Под указанный размер множества происходит расширение в памяти

// Расширение памяти

unsort\_mas = (int\*)malloc(size\_mas \* sizeof(int));

С помощью псевдослучайных чисел происходит заполнение файла отсортированного множества, в зависимости от типа минимального числа.

// Заполнение файла не отсортированным множеством

FILE\* file\_1 = fopen(name1, "w");

for (int i = 0; i < size\_mas; i++)

{

if (min\_num >= 0)

{

// Если минимальное число положительное fprintf(file\_1, "%5d", unsort\_mas[i] = rand() % max\_num);

}

else

{

// Если минимальное число отрицательное

fprintf(file\_1, "%5d", unsort\_mas[i] = min\_num + rand() % max\_num);

}

}

fclose(file\_1);

Следующий шаг заключался в том, чтобы считать числа из не отсортированного множества и записать в массив для дальнейшей обработки, игнорируя пробелы.

// Чтение из файла для сортировки

file\_1 = fopen(name1, "r");

// До конца файла

while ((buf = fgetc(file\_1)) != EOF)

{

for (int i = 0; i < size\_mas + 1; i++)

{

if (i == ' ')

{

// Без ' '

continue;

}

else {

fscanf(file\_1, "%5d", &unsort\_mas[i]);

}

}

}

fclose(file\_1);

Затем происходит запись отсортированного множества во 2 файл

// Запись в файл отсортированного множества

FILE\* file\_2 = fopen(name2, "w");

for (int i = 1; i < size\_mas + 1; i++)

{

fprintf(file\_2, "%5d", unsort\_mas[i]);

}

fclose(file\_2);

Если же пользователь в самом начале программы выбрал сортировку собственного файла, работает 2 сценарий. Он выбирает какой файл необходимо отсортировать.

// Второй сценарий, если у пользователя есть файл для сортировки

else

{

// Ввод имени пользовательского файла для сортировки

printf("Какой файл необходимо отсортировать?: ");

scanf("%s", &name1);

printf("\n");

Следующий шаг заключался в том, чтобы проверить существует ли такой файл, если он не найден, то пользователю предлагается 2 варианта, первый - это ввести “Yes”, чтобы закончить работу программы или второй - ввести любой другой символ, для того чтобы продолжить ввод имени файла, пока файл с таким именем не будет найден.

// Проверка на наличие файла

FILE\* file\_1 = fopen(name1, "r");

while (!file\_1)

{

char next[4]; // Выход из программы

printf("Ошибка: файл с таким именем не найден!");

printf("\n\n");

// Если пользователь захочет прекратить попытки ввода корректного имени файла

printf("Продолжить? (Введите \"Yes/yes\", если хотите продолжить): ");

scanf("%s", &next);

printf("\n");

// Если пользователь введет не "Yes\yes", то выход из программы

if (next[0] != 'Y' && next[0] != 'y')

{

return 1;

}

// Если после неудачной попытки, введет корректное имя файла

else

{

printf("Какой файл необходимо отсортировать?: ");

scanf("%s", &name1);

printf("\n");

file\_1 = fopen(name1, "r");

}

fclose(file\_1);

}

fclose(file\_1);

Следующий шаг, заключается в том, чтобы узнать размер пользовательского файла, для выделения памяти.

// Сохранение длины файла

file\_1 = fopen(name1, "r");

fseek(file\_1, 0L, SEEK\_END);

size\_mas = ftell(file\_1);

fclose(file\_1);

Если файл окажется пустым, то программа вернет ошибку.

// если файл пуст

if (size\_mas == 0)

{

printf("Ошибка: Файл пуст!");

printf("\n");

return 1;

}

Иначе, пользователю будет предложено ввести имя файла для сохранения отсортированного множества. После этого происходит выделение памяти.

else

{

// Ввод имени файла для сохранения отсортированного множества

printf("В каком файле сохранить отсортированное множество?: ");

scanf("%s", &name2);

// Расширение памяти

unsort\_mas = (int\*)malloc(size\_mas \* sizeof(int));

Следующий шаг заключается в том, чтобы считать числа из не отсортированного множества и записать в массив для дальнейшей обработки, игнорируя пробелы.

// Чтение из файла для сортировки

file\_1 = fopen(name1, "r");

// До конца файла

while ((buf = fgetc(file\_1)) != EOF)

{

for (int i = 0; i < size\_mas + 1; i++)

{

if (i == ' ')

{

// Без ' '

continue;

}

else {

fscanf(file\_1, "%5d", &unsort\_mas[i]);

}

}

}

fclose(file\_1);

Затем происходит запись отсортированного множества во 2 файл.

// Запись в файл отсортированного множества

FILE\* file\_2 = fopen(name2, "w");

for (int i = 1; i < size\_mas + 1; i++)

{

fprintf(file\_2, "%5d", unsort\_mas[i]);

}

fclose(file\_2);

Заключительным этапом в программе является освобождение памяти и возврат 0 в случае удачного завершения.

// Освобождение памяти

free(unsort\_mas);

return 0;

# 

# **4 Схема программы**



Рисунок 4.1 – Блок схема программы

# 

# **5 Тестирование программы**

Тестирование происходило для наиболее оптимального вывода таблиц и корректной работы с файлами.

Случай №1, когда файл пуст.



Рисунок 5.1 - Пустой файл

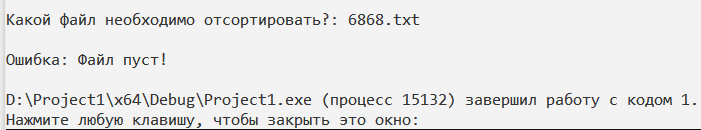


Рисунок 5.2 - Работа программы с пустым файлом

Программа отработала корректно, вернула 1

Случай №2, когда файла не существует.

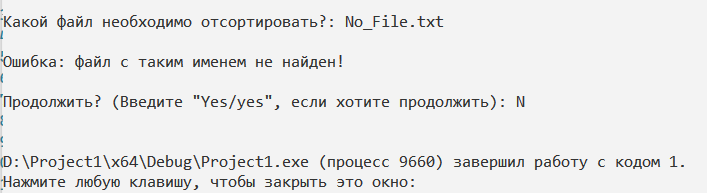


Рисунок 5.3 - Работа программы с несуществующим файлом

Программа отработала корректно, вернула 1



Рисунок 5.4 - Структура неотсортированного файла

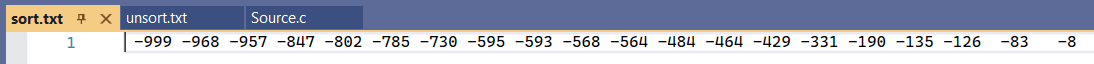


Рисунок 5.5 - Структура отсортированного файла



Рисунок 5.6 - Структура неотсортированного файла



Рисунок 5.7 - Структура отсортированного файла



Рисунок 5.8 - Структура неотсортированного файла



Рисунок 5.9 - Структура отсортированного файла

# **6 Совместная разработка**

При разработке программы использовалась среда GitHub для обменивания промежуточными данными. На рисунке 7 показана схема, которая получилась при совместной разработке.

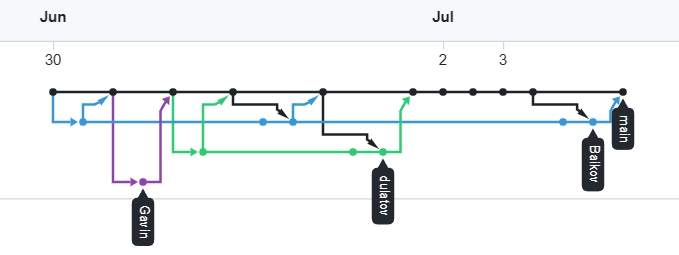


Рисунок 7 - Схема совместной разработки

# **Заключение**

В ходе выполнения практической работы удалось создать программу, использующую алгоритм сортировки вставками для эффективного упорядочения множества неопределенного размера. Это достижение стало возможным благодаря командной работе, чему способствовало использование платформы GitHub.

# **Список используемой литературы**

1. Microsoft Developer Blogs : https://devblogs.microsoft.com/visualstudio/visual-studio-2022/

1. Кнут Д. Э. 5.2 Внутренняя сортировка // Искусство программирования. Том 3. Сортировка и поиск, 2007. — Т. 3. — 832 с.
2. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. 2.1. Сортировка вставкой // Алгоритмы: построение и анализ, 2013. — С. 38—45.

1. Ахо А. В., Хопкрофт Д. Э., Ульман Д. Д. Структуры данных и алгоритмы, 2000. — С. 231.

# **Приложение А**

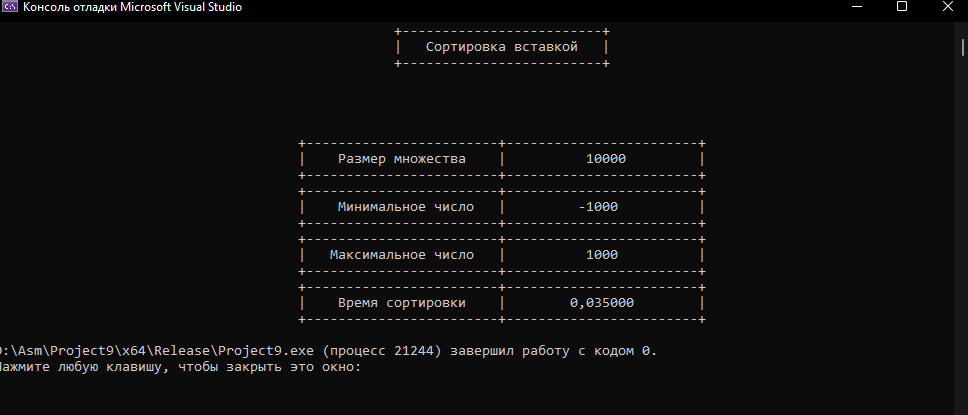


Рисунок А.1

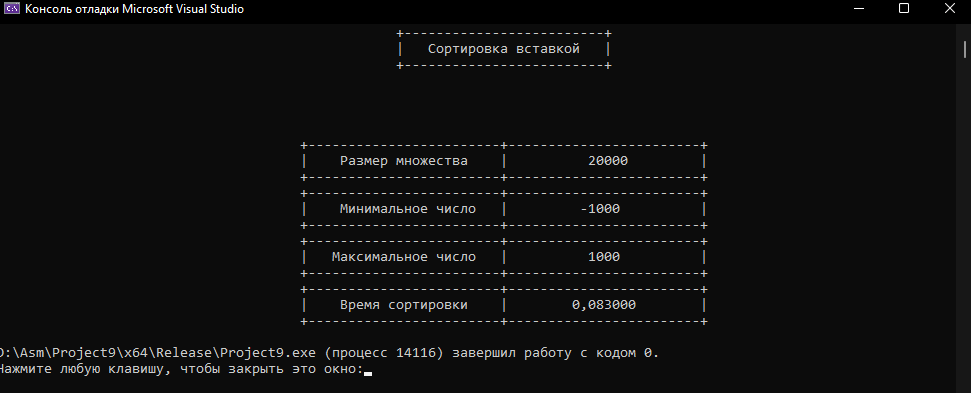


Рисунок А.2

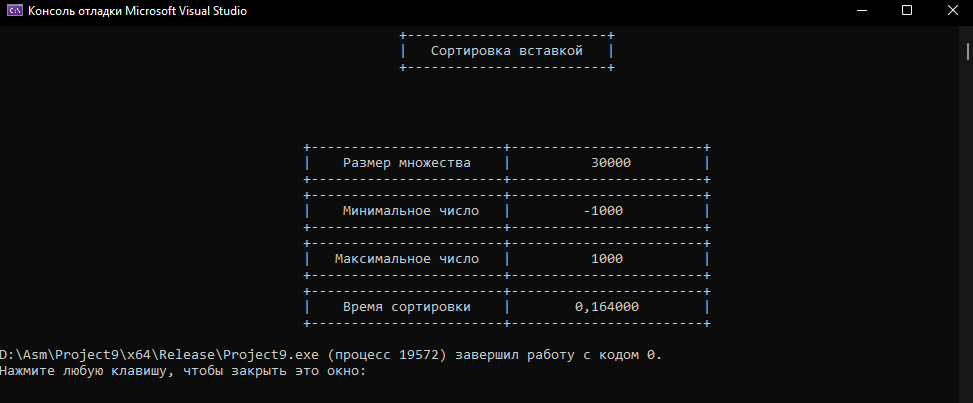


Рисунок А.3

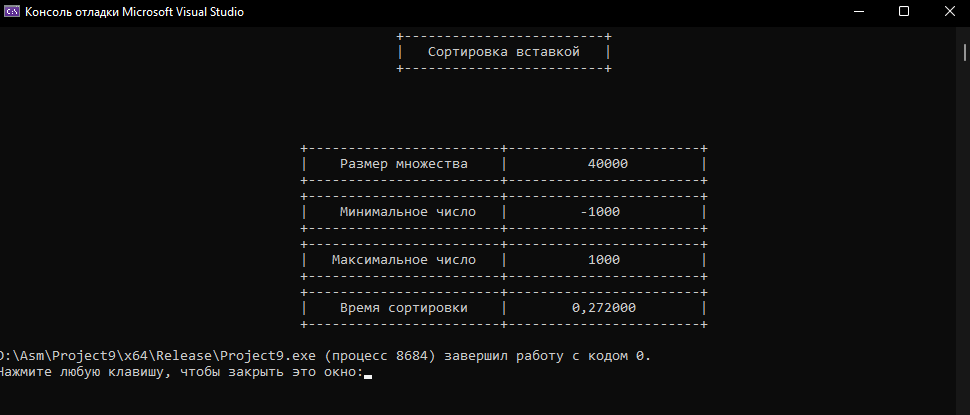


Рисунок А.4

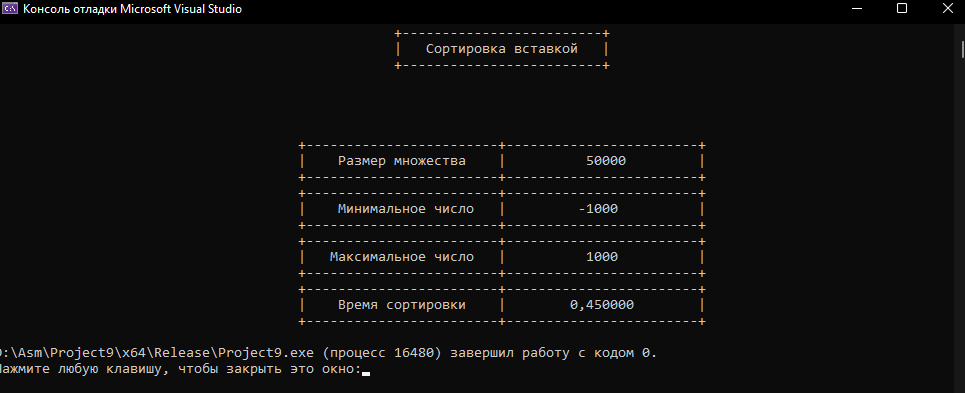


Рисунок А.5

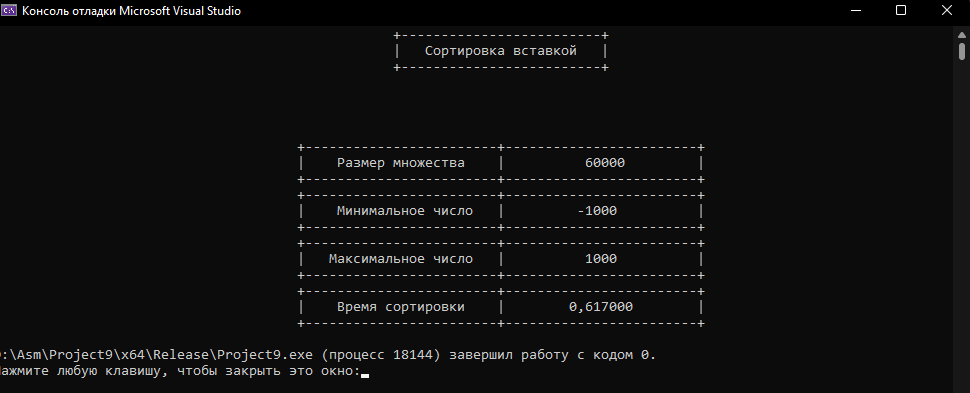


Рисунок А.6

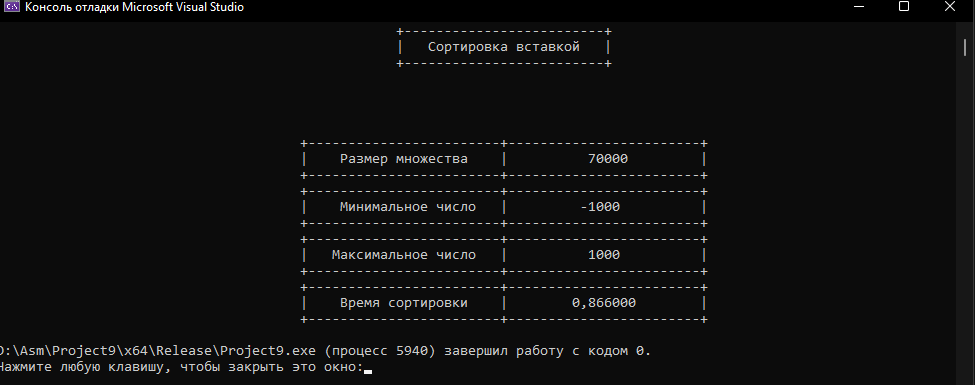


Рисунок А.7

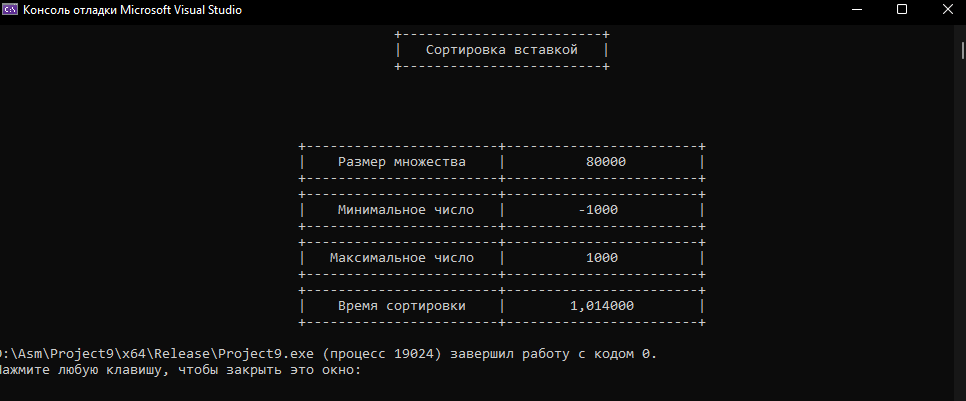


Рисунок А.8

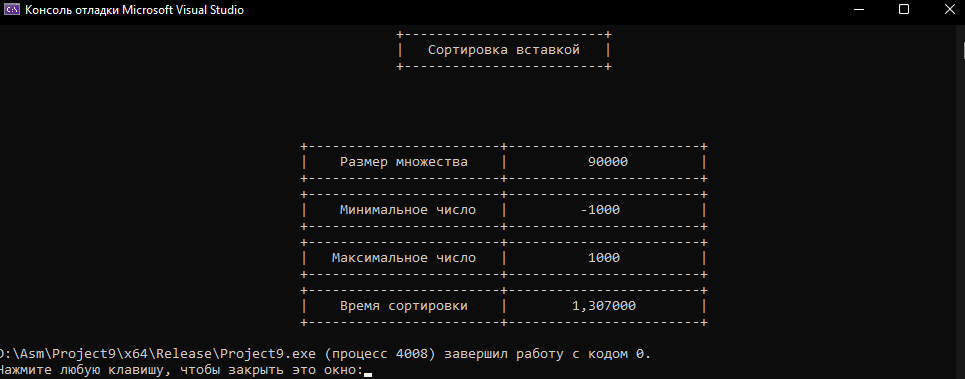


Рисунок А.9

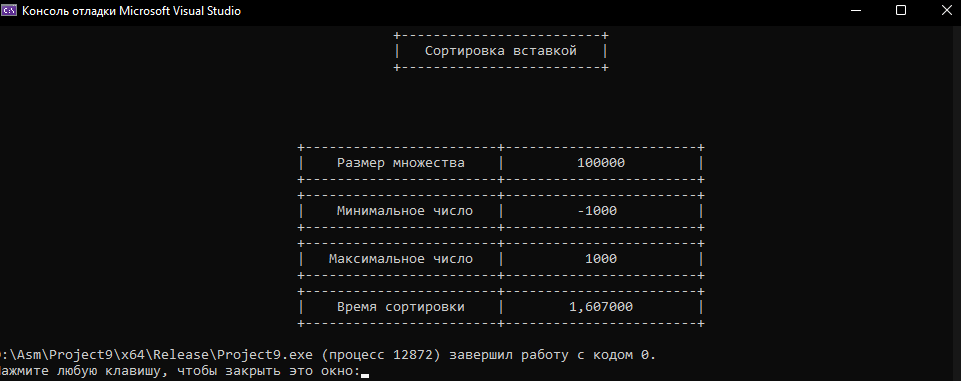


Рисунок А.10

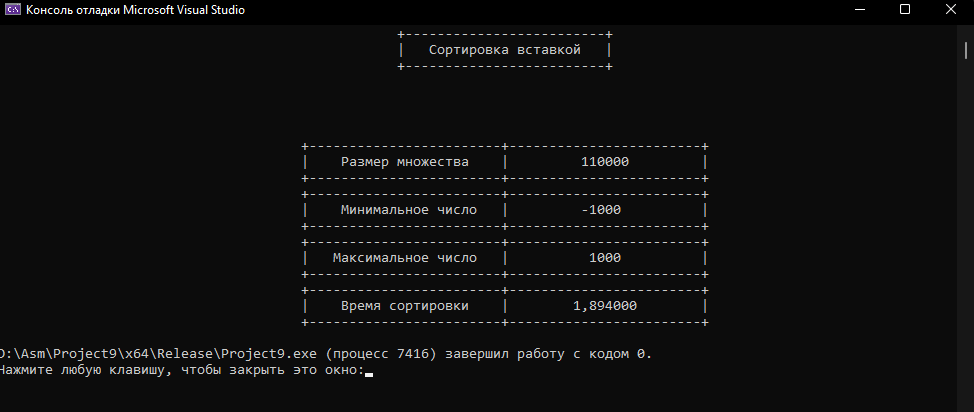


Рисунок А.11

# **Приложение Б Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <locale.h>

#include <malloc.h>

#include <stdlib.h>

// Функция, выводящая таблицу с названием программы

void print\_name()

{

printf("\t\t\t\t\t\t +-------------------------+\n");

printf("\t\t\t\t\t\t | Сортировка вставкой |\n");

printf("\t\t\t\t\t\t +-------------------------+\n");

printf("\n\n\n\n");

}

// Функция, выводящая таблицу с введенными данными пользователя и итоговое время сортировки

void last\_table(int size\_mas, int max\_num, int min\_num, int time\_end)

{

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t | Размер множества | %14d |\n", size\_mas);

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t | Минимальное число | %13d |\n", min\_num);

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t | Максимальное число | %13d |\n", max\_num);

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t | Время сортировки | %14lf(c) |\n", (double)time\_end / CLOCKS\_PER\_SEC);

printf("\t\t\t\t +------------------------+------------------------+\n");

}

// Функция сортировки

void insertion\_sort(int size\_mas, int\* unsort\_mas)

{

// Сортировка

for (int i = 1; i < size\_mas + 1; i++)

{

// Текущие расположение

int j = i;

// Пока слева есть место и элемент слева больше элемента справа

while (j > 0 && unsort\_mas[j - 1] > unsort\_mas[j])

{

// Смена мест

int tmp = unsort\_mas[j - 1];

unsort\_mas[j - 1] = unsort\_mas[j];

unsort\_mas[j] = tmp;

j--;

}

}

}

// Основная функция

int main()

{

// Подключение случайных чисел

srand(time(NULL));

// Подключение русского языка

setlocale(LC\_ALL, "RUS");

// Вывод первой таблицы

print\_name();

int size\_mas; // Размер множества

int\* unsort\_mas; // Указатель для расширение памяти

int max\_num; // Минимальное число

int min\_num; // Максимальное число

char have\_file; // Для проверки на наличие пользавотельского файла

char buf; // Буфер

char name1[20]; // Имя файла не отсортированного множества

char name2[20]; // Имя файла отсортированного множества

// Подтверждение или отрицание наличия файла

printf("Есть файл для сортировки? (Для подтверждения введите Y/y, если нет, то введите N/n, программа будет работать со случайными числами): ");

scanf("%c", &have\_file);

printf("\n");

// Если введено не "Y\yes", то заполнением файла псевдослучайными числами

if (have\_file != 'Y' && have\_file != 'y')

{

// Имя файла для не отсортированного множества

printf("В каком файле сохранить не отсортированное множество?: ");

scanf("%s", &name1);

printf("\n");

// Имя файла для отсортированного множества

printf("В каком файле сохранить отсортированное множество?: ");

scanf("%s", &name2);

printf("\n");

// Ввод размера множества

printf("Размер множества: ");

scanf("%d", &size\_mas);

printf("\n");

// Ввод нижнего предела множества

printf("Минимальное число: ");

scanf("%d", &min\_num);

printf("\n");

// Ввод верхнего предела множества

printf("Максимальное число: ");

scanf("%d", &max\_num);

printf("\n");

// Расширение памяти

unsort\_mas = (int\*)malloc(size\_mas \* sizeof(int));

// Заполнение файла не отсортированным множеством

FILE\* file\_1 = fopen(name1, "w");

for (int i = 0; i < size\_mas; i++)

{

if (min\_num >= 0)

{

// Если минимальное число положительное

fprintf(file\_1, "%5d", unsort\_mas[i] = rand() % max\_num);

}

else

{

// Если минимальное число отрицательное

fprintf(file\_1, "%5d", unsort\_mas[i] = min\_num + rand() % max\_num);

}

}

fclose(file\_1);

// Начало отсчета времени работы сортировки

clock\_t time\_start = clock();

// Чтение из файла для сортировки

file\_1 = fopen(name1, "r");

// До конца файла

while ((buf = fgetc(file\_1)) != EOF)

{

for (int i = 0; i < size\_mas + 1; i++) {

if (i == ' ') {

// Без ' '

continue;

}

else {

fscanf(file\_1, "%5d", &unsort\_mas[i]);

}

}

}

fclose(file\_1);

// Вызов функции, которая отвечает за алгоритм сортировки

insertion\_sort(size\_mas, unsort\_mas);

// Запись в файл отсортированного множества

FILE\* file\_2 = fopen(name2, "w");

for (int i = 1; i < size\_mas + 1; i++)

{

fprintf(file\_2, "%5d", unsort\_mas[i]);

}

fclose(file\_2);

// Конец отсчета времени

clock\_t time\_end = clock() - time\_start;

// Очистка консоли

system("cls");

// Вывод первой таблицы

print\_name();

// Вывод второй таблицы

last\_table(size\_mas, max\_num, min\_num, time\_end);

}

// Второй сценарий, если у пользователя есть файл для сортировки

else

{

// Ввод имени пользовательского файла для сортировки

printf("Какой файл необходимо отсортировать?: ");

scanf("%s", &name1);

printf("\n");

// Проверка на наличие файла

FILE\* file\_1 = fopen(name1, "r");

while (!file\_1)

{

char next[4]; // Выход из программы

printf("Ошибка: файл с таким именем не найден!");

printf("\n\n");

// Если пользователь захочет прекратить попытки ввода корректного имени файла

printf("Продолжить? (Введите \"Yes/yes\", если хотите продолжить): ");

scanf("%s", &next);

printf("\n");

// Если пользователь введет не "Yes\yes", то выход из программы

if (next[0] != 'Y' && next[0] != 'y')

{

return 1;

}

// Если после неудачной попытки, введет корректное имя файла

else

{

printf("Какой файл необходимо отсортировать?: ");

scanf("%s", &name1);

printf("\n");

file\_1 = fopen(name1, "r");

}

fclose(file\_1);

}

fclose(file\_1);

// Сохранение длины файла

file\_1 = fopen(name1, "r");

fseek(file\_1, 0L, SEEK\_END);

size\_mas = ftell(file\_1);

fclose(file\_1);

// если файл пуст

if (size\_mas == 0)

{

printf("Ошибка: Файл пуст!");

printf("\n");

return 1;

}

else

{

// Ввод имени файла для сохранения отсортированного множества

printf("В каком файле сохранить отсортированное множество?: ");

scanf("%s", &name2);

// Расширение памяти

unsort\_mas = (int\*)malloc(size\_mas \* sizeof(int));

// Начало отсчета времени работы сортировки

clock\_t time\_start = clock();

// Чтение из файла для сортировки

file\_1 = fopen(name1, "r");

// До конца файла

while ((buf = fgetc(file\_1)) != EOF)

{

for (int i = 0; i < size\_mas + 1; i++) {

if (i == ' ') {

// Без ' '

continue;

}

else {

fscanf(file\_1, "%d", &unsort\_mas[i]);

}

}

}

fclose(file\_1);

// Вызов функции отвечающий за сортировку

insertion\_sort(size\_mas, unsort\_mas);

// Запись в файл отсортированного множества

FILE\* file\_2 = fopen(name2, "w");

for (int i = 1; i < size\_mas + 1; i++)

{

fprintf(file\_2, "%5d", unsort\_mas[i]);

}

fclose(file\_2);

// Конец отсчета времени

clock\_t time\_end = clock() - time\_start;

// Очистка консоли

system("cls");

// Вывод первой таблицы

print\_name();

// Вывод времени потраченного на сортировку

printf("\t\t\t\t +------------------------+------------------------+\n");

printf("\t\t\t\t | Время сортировки | %14lf(c) |\n", (double)time\_end / CLOCKS\_PER\_SEC);

printf("\t\t\t\t +------------------------+------------------------+\n");

}

}

// Освобождение памяти

free(unsort\_mas);

return 0;

}