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**

(2023/2024 учебный год)

Кузин Максим Евгеньевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

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

Год обучения 1 семестр 2

Период прохождения практики с 25.06.2024 по 08.07.2024

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, 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

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

(2023/2024 учебный год)

Кузин Максим Евгеньевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года Год обучения 1 семестр 2 Период прохождения практики с 25.06.2024 по 08.07.2024

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., 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 | 2 | 25.06.24 -  25.06.24 |  |
| 2 | Подбор и изучение материала по теме  работы | 15 | 26.06. 24 –  28.06.24 |  |
| 3 | Разработка  алгоритма | 43 | 01.07.24 –  03.07.24 |  |
| 4 | Описание алгоритма и  программы | 18 | 03.07.24– 04.07.24 |  |
| 5 | Тестирование | 5 | 04.07.24 –  05.07.24 |  |
| 6 | Получение и  анализ результатов | 10 | 05.07.24 –  08.07.24 |  |
| 7 | Оформление  отчёта | 15 | 05.07.24 –  08.07.24 |  |
|  | **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)

Кузин Максим Евгеньевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

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

Год обучения 1 семестр 2

Период прохождения практики с 25.06.2024 по 08.07.2024

Кафедра «Вычислительная техника»

Кузин М.Е. выполнял практическое задание «Сортировка пузырьком». На первоначальном этапе были изучен и проанализирован алгоритм сортировки пузырьком, был выбран метод решения и язык программирования С++, на котором была написана программа сортировки массива пузырьком. Также, осуществил подсчёт времени выполнения сортировки. Протестировал и отладил программу. Оформил отчёт.

Бакалавр Кузин М.Е. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Руководитель Карамышева Н.С. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 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)

Кузин Максим Евгеньевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интелект»

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

Год обучения 1 семестр 2

Период прохождения практики с 25.06.2024 по 08.07.2024

Кафедра «Вычислительная техника»

В процессе выполнения практики Кузин М.Е. решала следующие задачи: изучение алгоритма сортировки пузырьком, анализ работы алгоритма, сравнение существующих методов сортировки, произведение подсчёта времени выполнения сортировки, тестирование и отладка программы.

За период выполнения практики были освоены основные понятия и технологии сортировки пузырьком. Во время выполнения работы Кузин М.Е. показала 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 Кузин М. Е. заслуживает оценки « ».

Руководитель практики к.т.н., доцент, Карамышева Н.С. « » 2024 г.

# Содержание

[Введение 2](#_bookmark0)

1. [Постановка задачи 3](#_bookmark1)
   1. [Достоинства алгоритма сортировки вставками 3](#_bookmark2)
   2. [Недостатки алгоритма сортировки вставками 3](#_bookmark3)
   3. [Типичные сценарии применения данного алгоритма 3](#_bookmark4)
2. [Выбор решения 4](#_bookmark5)
3. [Описание программы 5](#_bookmark6)
4. [Схемы программы 6](#_bookmark7)
   1. [Блок-схема программы 6](#_bookmark8)
5. [Тестирование программы 7](#_bookmark9)
   1. [Тестирование на разных наборах данных 7](#_bookmark10)
   2. [Анализ полученных результатов тестирования (анализ работы алгоритма) 7](#_bookmark11)
6. [Отладка 9](#_bookmark12)
7. [Совместная разработка 10](#_bookmark13)

[Заключение 14](#_bookmark14)

[Список используемой литературы 15](#_bookmark15)

[Приложение А 16](#_bookmark16)

[Приложение Б Листинг 19](#_bookmark17)

# Введение

Сортировка данных играет огромную роль в современном мире, особенно в контексте информационных технологий и аналитики. 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.

Существует множество алгоритмов сортировки, таких как быстрая сортировка, сортировка слиянием, пузырьковая сортировка и другие. Они используются для упорядочивания данных по различным критериям.

Пузырьковая сортировка — один из наиболее широко известных алгоритмов сортировки. Суть алгоритма заключается в последовательном просмотре массива от конца к началу или от начала к концу и сравнении каждой пары элементов между собой. При этом «неправильное» расположение элементов устраняется путём их перестановки. Процесс просмотра и сравнения элементов повторяется до просмотра всего массива.

Особенностью данного метода является сравнение не каждого элемента со всеми,

а сравнение в парах соседних элементов. В процессе исполнения алгоритма более «лёгкие» элементы массива постепенно «всплывают».

# Постановка задачи

Поставленная задача: необходимо заполнить массив из n-ого количества элементов случайными числами, записать данные элементы в отдельный файл. После этого выполнить сортировку пузырьком над данными, находящимися в массиве, записать отсортированные данные в другой файл, посчитать время выполнения при сортировке.

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

Оформить отчет по проведенной практике.

## Достоинства алгоритма сортировки пузырьком

* легко понять и реализовать;
* эффективна для небольших массивов;
* устойчива, то есть не меняет порядок равных элементов;
* не требует дополнительной памяти.

## Недостатки алгоритма сортировки вставками

* очень медленная, так как выполняет много лишних сравнений и обменов;
* имеет квадратичную сложность времени выполнения;
* неэффективна для больших объёмов данных.

## Типичные сценарии применения данного алгоритма

* Применение внутри более эффективных алгоритмов. Пузырьковая сортировка лежит в основе некоторых более эффективных алгоритмов, например, шейкерной сортировки, пирамидальной сортировки и быстрой сортировки;
* В компьютерной графике. Пузырьковая сортировка популярна благодаря возможности обнаруживать мелкие ошибки в почти отсортированных массивах и исправлять её только с линейной сложностью. Например, она используется в алгоритме заполнения полигонов;

# Выбор решения

Для написания данной программы будет использован язык программирования С++. Этот язык является распространённым языком программирования. С++ – это язык программирования общего назначения, хорошо известный своей эффективностью, экономичностью и переносимостью. Указанные преимущества С++ обеспечивают хорошее качество разработки почти любого вида программного продукта.

В качестве среды программирования была выбрана программа Microsoft Visual Studio. Microsoft Visual Studio — 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

— крупнейший сервис для хостинга IT-проектов и их совместной разработки. Удобство совместной разработки на GitHub заключается в следующих возможностях:

* Несколько разработчиков могут одновременно работать над одним проектом. Это позволяет сократить время разработки и повысить качество кода.
* Хранение и управление кодом в одном месте. Все проекты хранятся в репозиториях, что облегчает работу с ними и упрощает их управление.
* Отслеживание изменений. GitHub позволяет отслеживать все изменения, внесённые в проект, и возвращаться к предыдущим версиям кода в случае необходимости.
* Инструменты для эффективной организации командной работы. Пользователи могут легко создавать задачи, делиться кодом, вносить изменения и обсуждать различные аспекты проекта.
* Разрешение конфликтов. При работе нескольких участников над одним и тем же файлом могут возникать конфликты. GitHub позволяет легко определить места расхождения, а также предоставляет инструменты для их урегулирования.
* Работа как в социальной сети. У каждого разработчика есть свой профиль, где отображается его активность, публичные репозитории, подписчики и другая информация. Разработчик может подписываться на других участников, видеть их активность и следить за их работой.

# Описание программы

Данная программа написана на C++/CLI, используюет Windows Forms для создания графического пользовательского интерфейса. Программа выполняет несколько функций, связанных с генерацией случайных чисел, сортировкой и записью результатов в файл.

В начале кода происходят подключения необходимых библиотек и пространств имен.

Определена константа dimension и объявлен массив arr размером dimension.

Функция RandArr заполняет массив случайными числами от –n до n и отображает их в элементе управления numbers.

Функция readfile записывает содержимое массива в файл по указанному пути.

Функция BubbleSort выполняет сортировку массива методом пузырька, замеряет время выполнения и отображает результат сортировки в элементе управления numbers.

Обработчики событий кнопок button1\_Click и button2\_Click вызывают соответствующие функции при нажатии на кнопки.

Обработчик события checkBox1\_CheckedChanged отображает или скрывает метку label1 в зависимости от состояния флажка.

Обработчик события listBox1\_SelectedIndexChanged реагирует на изменение выбранных элементов в списке, но в данном фрагменте кода не содержит действий.

Эта программа является графическим приложением, которое генерирует случайные числа, сортирует их методом пузырька, отображает время сортировки и записывает результаты в файлы.

Подробный алгоритм работы программы и функции сортировки представлен в подразделе 4.1 на рисунках 1 - 4.

Листинг программы приведен в приложении Б.

# Схемы программы

* 1. **Блок-схема программы**

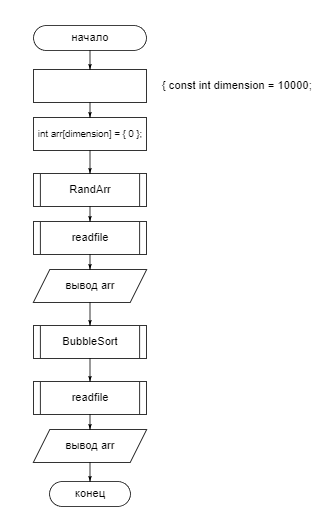


Рисунок 1 – Блок-схема программы

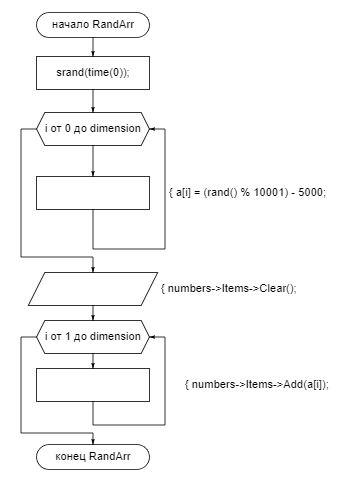


Рисунок 2 – Блок-схема функции RandArr

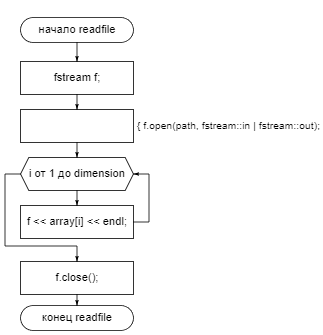


Рисунок 3 – Блок-схема функции readfile

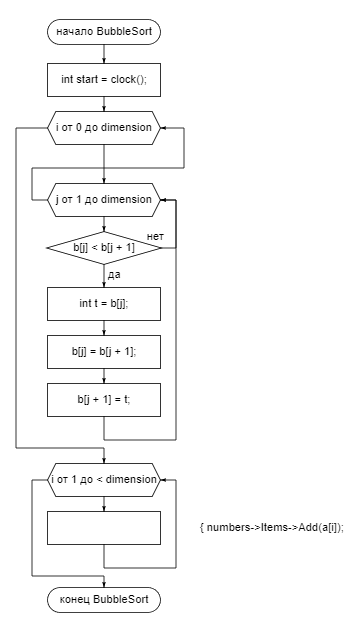


Рисунок 4 – Блок-схема функции BubbleSort

# Тестирование программы

## 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 | 0.265 |
| 2 | 20000 | 1.117 |
| 3 | 30000 | 2.563 |
| 4 | 40000 | 4.612 |
| 5 | 50000 | 7.266 |
| 6 | 60000 | 10.511 |
| 7 | 70000 | 14.374 |
| 8 | 80000 | 18.896 |
| 9 | 90000 | 23.973 |
| 10 | 100000 | 29.827 |
| 11 | 110000 | 36.014 |

## Анализ полученных результатов тестирования (анализ работы алгоритма)

На основании анализа данных, полученных в результате тестирования алгоритма сортировки пузырьком, 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.

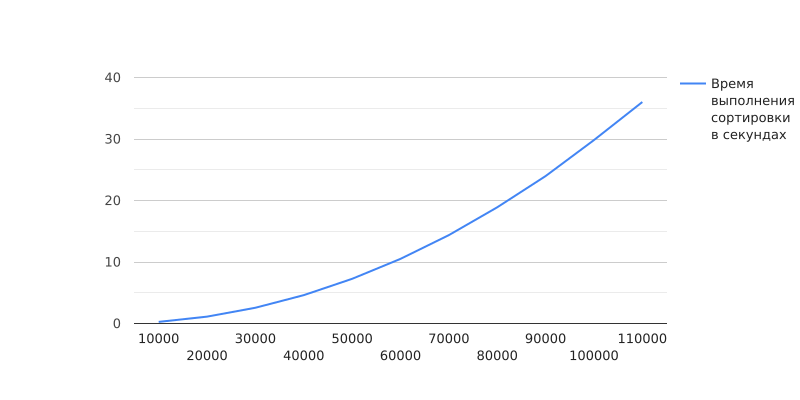


Рисунок 5 – Результаты тестирования

# 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 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.

# Совместная разработка

Для удобства совместной разработки был использован сервис Discord.Определили задачи проекта, назначили приоритет задачам.

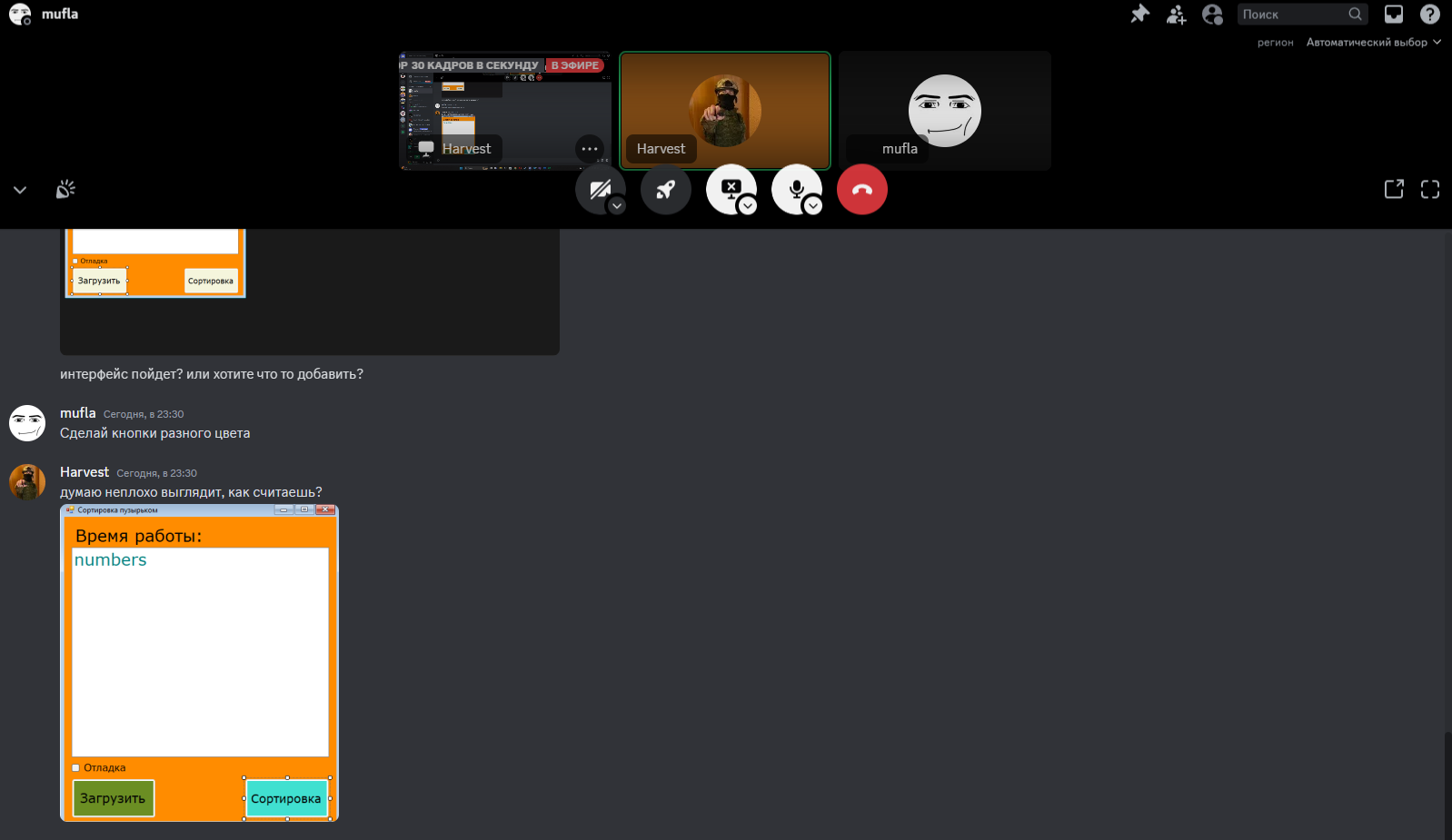


Рисунок 6–Обсуждение задач проекта

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

# Заключение

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

Мною был написан алгоритм, считающий количество перестановок элементов массива во время сортировки и время работы программы в секундах. 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. Сортировка вставками [Электронный ресурс] – URL: https://ru.wikipedia.org (дата обращения: 02.07.2023 г)

# Приложение А. Результаты тестирования программы

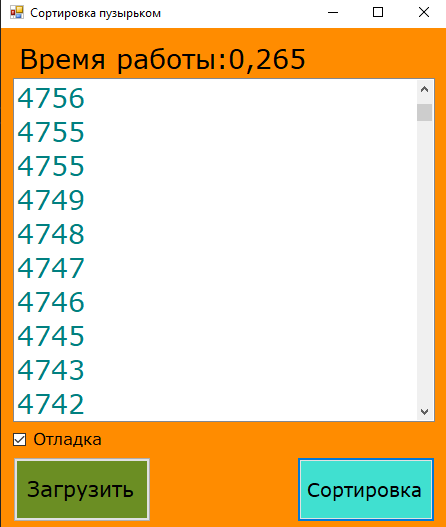
****

Рисунок А.1

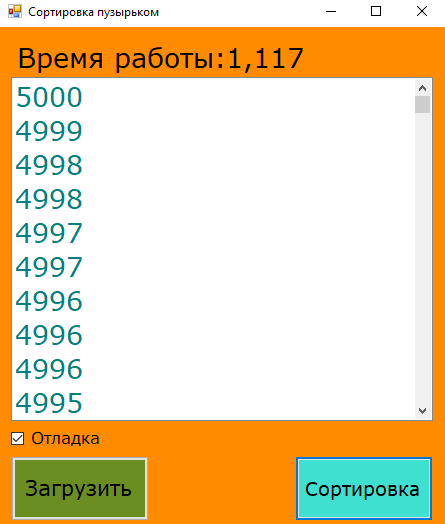


Рисунок А.2

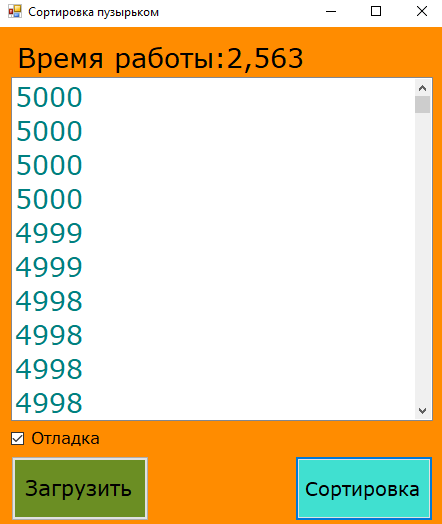


Рисунок А.3

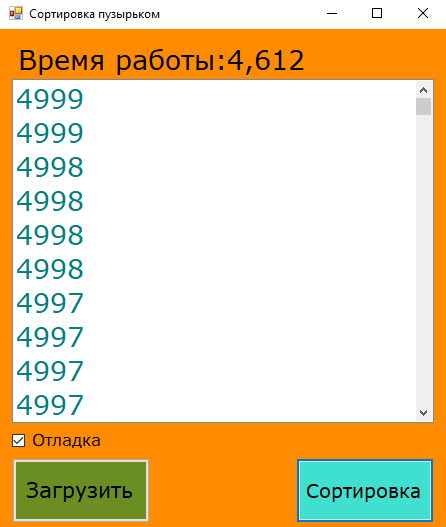


Рисунок А.4

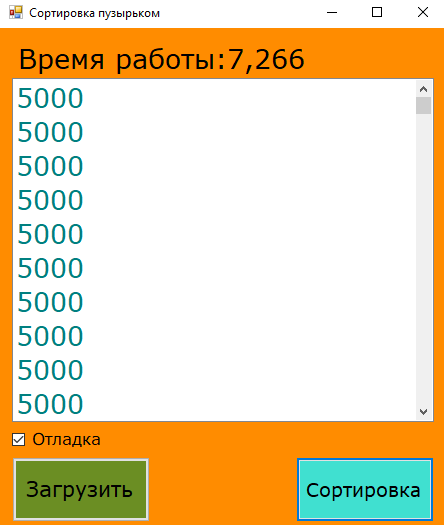


Рисунок А.5

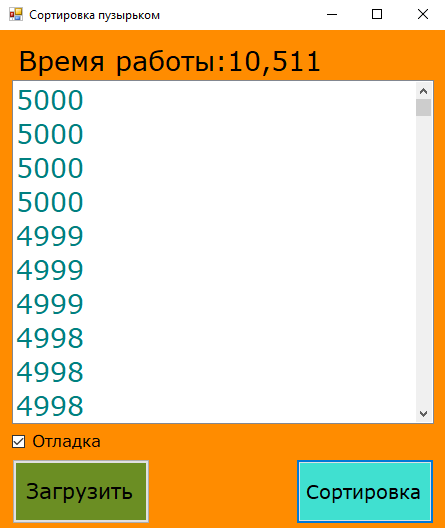


Рисунок А.6

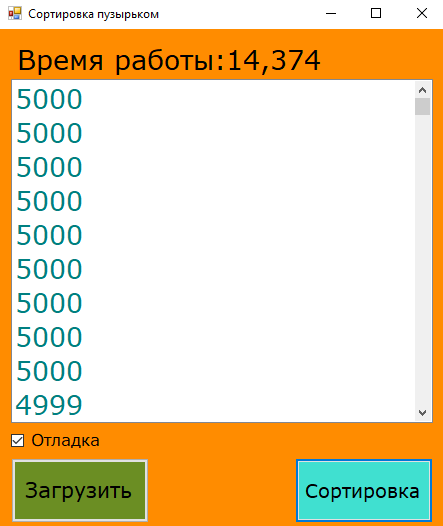


Рисунок А.7

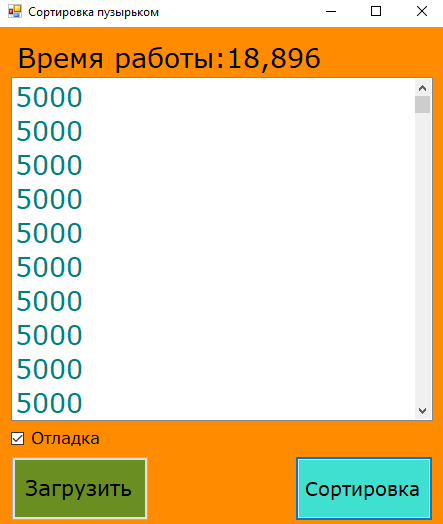


Рисунок А.8

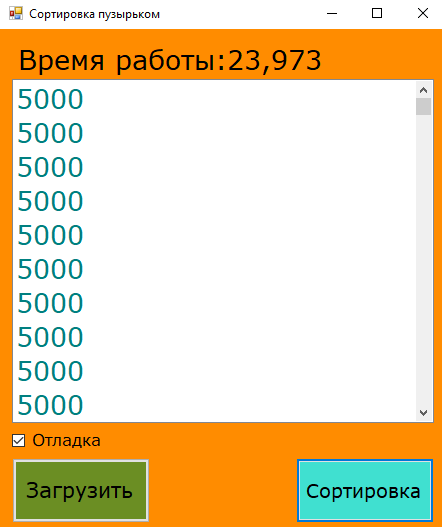


Рисунок А.9

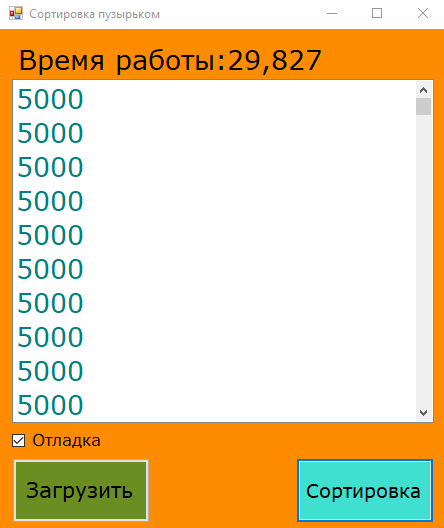


Рисунок А.10

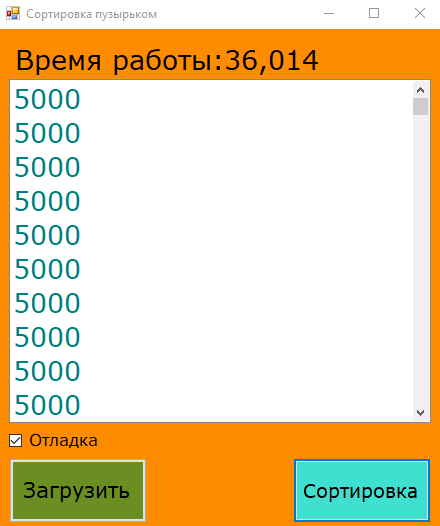


Рисунок А.11

# Приложение Б. Листинг программы

#define \_CRT\_SECURE\_NO\_WARNINGS #include <stdio.h>

#include <locale.h> #include <stdlib.h> #include <time.h> #include <conio.h>

int main()

{

setlocale(LC\_ALL, "Russian"); srand(time(NULL));

FILE\* f;

int m, size; int\* array; char ch;

do {

system("cls");

printf("МЕНЮ:\n");

printf("1 - Сортировка случайных значений по возрастанию\n"); printf("2 - Сортировка случайных значений по убыванию\n"); printf("3 - Сортировка возрастающих значений по убыванию\n"); printf("4 - Сортировка возрастающих значений по возрастанию\n");

printf("ecs - выход\n"); ch = \_getch();

switch (ch) { case '1':

//Сортировка случайных значений по возрастанию system("cls");

printf("Введите количество элементов массива: "); scanf("%d", &size);

printf("\n");

array = (int\*)malloc(size \* sizeof(int));

printf("Массив случайных чисел: 'input.txt'\n"); f = fopen("input.txt", "w");

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

{

array[i] = rand() - rand();

fprintf(f, "%d ", array[i]);

}

fclose(f);

printf("Отсортированный массив: 'output.txt'\n");

f = fopen("output.txt", "w");

time\_t start = clock(); //время до сортировки

long count = 0;

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

{

m = array[i];

for (int j = i - 1; j >= 0; j--)

{

if (array[j] < m) break;

array[j + 1] = array[j]; array[j] = m;

count++;

}

}

time\_t stop = clock(); //время после сортировки for (int i = 0; i < size; i++) {

fprintf(f, "%d ", array[i]);

}

fclose(f);

double time = (stop - start) / 1000.0; //время сортировки printf("\n");

printf("Время выполнения сортировки: "); printf("%lf\n", time); printf("Количество перестановок: "); printf("%o\n", count);

system("pause"); break;

}

switch (ch) {

case '2':

//Сортировка случайных значений по убыванию system("cls");

printf("Введите количество элементов массива: "); scanf("%d", &size);

printf("\n");

array = (int\*)malloc(size \* sizeof(int));

printf("Массив случайных чисел: 'input.txt'\n"); f = fopen("input.txt", "w");

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

{

array[i] = rand() - rand();

fprintf(f, "%d ", array[i]);

}

fclose(f);

printf("Отсортированный массив: 'output.txt'\n");

f = fopen("output.txt", "w");

time\_t start = clock(); //время до сортировки

long count = 0;

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

{

m = array[i];

for (int j = i - 1; j >= 0; j--)

{

if (array[j] > m) break;

array[j + 1] = array[j]; array[j] = m;

count++;

}

}

time\_t stop = clock(); //время после сортировки for (int i = 0; i < size; i++) {

fprintf(f, "%d ", array[i]);

}

fclose(f);

double time = (stop - start) / 1000.0; //время сортировки printf("\n");

printf("Время выполнения сортировки: "); printf("%lf\n", time); printf("Количество перестановок: "); printf("%o\n", count);

system("pause"); break;

}

switch (ch) { case '3':

//Сортировка возрастающих значений по убыванию system("cls");

printf("Введите количество элементов массива: "); scanf("%d", &size);

printf("\n");

array = (int\*)malloc(size \* sizeof(int));

printf("Массив случайных чисел: 'input.txt'\n"); array[0] = rand();

f = fopen("input.txt", "w"); for (int i = 0; i < size; i++)

{

array[i + 1] = array[i] + rand() % 100 + 100; fprintf(f, "%d ", array[i]);

}

fclose(f);

printf("Отсортированный массив: 'output.txt'\n");

f = fopen("output.txt", "w");

time\_t start = clock(); //время до сортировки

long count = 0;

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

{

m = array[i];

for (int j = i - 1; j >= 0; j--)

{

if (array[j] > m) break;

array[j + 1] = array[j]; array[j] = m;

count++;

}

}

time\_t stop = clock(); //время после сортировки for (int i = 0; i < size; i++) {

fprintf(f, "%d ", array[i]);

}

fclose(f);

double time = (stop - start) / 1000.0; //время сортировки printf("\n");

printf("Время выполнения сортировки: "); printf("%lf\n", time); printf("Количество перестановок: "); printf("%o\n", count);

system("pause"); break;

}

switch (ch) {

case '4':

//Сортировка возрастающих значений по возрастанию system("cls");

printf("Введите количество элементов массива: "); scanf("%d", &size);

printf("\n");

array = (int\*)malloc(size \* sizeof(int));

printf("Массив случайных чисел: 'input.txt'\n");

array[0] = rand();

f = fopen("input.txt", "w"); for (int i = 0; i < size; i++)

{

array[i + 1] = array[i] + rand() % 100 + 100; fprintf(f, "%d ", array[i]);

}

fclose(f);

printf("Отсортированный массив: 'output.txt'\n");

f = fopen("output.txt", "w");

time\_t start = clock(); //время до сортировки

long count = 0;

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

{

m = array[i];

for (int j = i - 1; j >= 0; j--)

{

if (array[j] < m) break;

array[j + 1] = array[j]; array[j] = m;

count++;

}

}

time\_t stop = clock(); //время после сортировки for (int i = 0; i < size; i++) {

fprintf(f, "%d ", array[i]);

}

fclose(f);

double time = (stop - start) / 1000.0; //время сортировки printf("\n");

printf("Время выполнения сортировки: "); printf("%lf\n", time); printf("Количество перестановок: "); printf("%o\n", count);

system("pause"); break;

}

} while (ch != 27);

}