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)

                                               Дасаев Тимур Зягитович

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)

                                               Дасаев Тимур Зягитович

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)

                                            Дасаев Тимур Зягитович

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»

Дасаев Т.З. выполнял практическое задание «Сортировка слиянием». На первоначальном этапе были изучен и проанализирован алгоритм сортировки слиянием, был выбран метод решения и язык программирования С, на котором была написана программа сортировки массива методом слияния. Также написал программу сортировки. Протестировал программу. Оформил отчёт.

Бакалавр Дасаев Т.З. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

                                            Дасаев Тимур Зягитович

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 Дасаев Т.З. решал следующую задачу: написание алгоритма сортировки.

За период выполнения практики были освоены основные понятия и технологии сортировки слиянием, реализован метод работы с файлами. Во время выполнения работы Дасаев Т.З. показал себя ответственным, добросовестным студентом, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и сортировке.

За выполнение работы Дасаев Т.З. заслуживает оценки «\_\_\_\_\_\_».

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

**Содержание**

[Введение 7](#_Toc139362608)

[1 Постановка задачи 8](#_Toc139362609)

[1.1 Достоинства алгоритма 8](#_Toc139362610)

[1.2 Недостатки алгоритма 8](#_Toc139362611)

[1.3 Типичные сценарии применения 8](#_Toc139362612)

[2 Выбор решения 9](#_Toc139362613)

[3 Описание программы 10](#_Toc139362614)

[4 Схемы программы 11](#_Toc139362615)

[4.1 Блок-схема программы 11](#_Toc139362616)

[4.2 Блок-схема алгоритма 13](#_Toc139362617)

[5 Тестирование программы 18](#_Toc139362622)

[5.1 Тестирование на разных наборах данных 18](#_Toc139362623)

[5.2 Анализ полученных результатов 18](#_Toc139362624)

[6. Отладка 20](#_Toc139362626)

[Заключение 24](#_Toc139362627)

[Список используемой литературы 25](#_Toc139362628)

[Приложение А 26](#_Toc139362629)

[Приложение Б (Листинг) 30](#_Toc139362630)

# Введение

Язык программирования Си разрабатывался в период с [1969](https://ru.wikipedia.org/wiki/1969) по [1973 годы](https://ru.wikipedia.org/wiki/1973_%D0%B3%D0%BE%D0%B4) в лабораториях [Bell Labs](https://ru.wikipedia.org/wiki/Bell_Labs), и к 1973 году на этот язык была переписана большая часть [ядра](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) [UNIX](https://ru.wikipedia.org/wiki/Unix), первоначально написанного на ассемблере [PDP-11](https://ru.wikipedia.org/wiki/PDP-11)/20. Название языка стало логическим продолжением старого языка «[Би](https://ru.wikipedia.org/wiki/%D0%91%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))», многие особенности которого были положены в основу.

По мере развития язык сначала стандартизировали как [ANSI C](https://ru.wikipedia.org/wiki/ANSI_C), а затем этот стандарт был принят комитетом по международной стандартизации [ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) как ISO C, ставший также известным под названием C90. В стандарте [С99](https://ru.wikipedia.org/wiki/C99) язык получил новые возможности, такие как массивы переменной длины и встраиваемые функции. А в стандарте [C11](https://ru.wikipedia.org/wiki/C11) в язык добавили реализацию потоков и поддержку атомарных типов. Однако с тех пор язык развивается медленно, и в стандарт C18 попали лишь исправления ошибок стандарта C11.

Язык Си разрабатывался как язык системного программирования, для которого можно создать однопроходный компилятор. Стандартная библиотека также невелика. Как следствие данных факторов — компиляторы разрабатываются сравнительно легко. Поэтому данный язык доступен на самых различных платформах. К тому же, несмотря на свою низкоуровневую природу, язык ориентирован на переносимость.

**Сортировка слиянием** — алгоритм сортировки, который упорядочивает списки(или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

# Постановка задачи

Разработать программу, сортирующую массив, указанного размера,

методом слияния. Исходный массив считать из файла, а отсортированный записать в файл. Протестировать программу на наличие ошибок.

## Достоинства алгоритма

* Работает даже на структурах данных последовательного доступа.
* Хорошо сочетается с [подкачкой](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB_%D0%BF%D0%BE%D0%B4%D0%BA%D0%B0%D1%87%D0%BA%D0%B8) и [кэшированием](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88-%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) памяти.
* Неплохо работает в [параллельном](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) варианте: легко разбить задачи между процессорами поровну, но трудно сделать так, чтобы другие процессоры взяли на себя работу, в случае если один процессор задержится.
* Не имеет «трудных» входных данных.
* Устойчивая - сохраняет порядок равных элементов (принадлежащих одному классу эквивалентности по сравнению).

## Недостатки алгоритма

* На «почти отсортированных» массивах работает столь же долго, как на хаотичных. Существует вариант сортировки слиянием, который работает быстрее на частично отсортированных данных, но он требует дополнительной памяти, в дополнении ко временному буферу, который используется непосредственно для сортировки.
* Требует дополнительной памяти по размеру исходного массива.

## Типичные сценарии применения

* сортировка списков абитуриентов
* сортировка списка цен товаров магазина
* сортировка списка участников турнира
* сортировка списка большого объёма данных

# Выбор решения

Для реализации метода сортировки была выбрана среда Microsoft Visual Studio.

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Функциональная структура среды включает в себя:

- редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода;

- отладчик кода;

- редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

- веб-редактор;

- дизайнер классов;

- дизайнер схем баз данных.

# Описание программы

В программе для сортировки слиянием подключены следующие заголовочные файлы: *stdio.h* – заголовочный файл стандартной библиотеки языка Си, содержащий определения макросов, константы и объявления функций и типов;*locale.h*– заголовочный файл для консоли на русском языке; *stdlib.h –* заголовочный файл стандартной библиотеки языка Си, который содержит в себе функции, занимающиеся выделением памяти, контролем процесса выполнения программы, преобразованием типов и другие*; time.h –* заголовочный файл стандартной библиотеки языка программирования C, содержащий типы и функции для работы с датой и временем*;*

Далее идет функция Merge(), в которой будет производиться соединение двух отсортированных массивов. В ней я сначала объявляю переменный k, i, j и присваиваю k и i значение переменной low, а j присваиваем значение переменной mid с добавлением единицы.

Дальше идет цикл while, который сортирует две части исходного массива по возрастанию. Если arr[i]<=arr[j], то мы записываем первую часть массива в новый массив aux, в котором временно будут находится отсортированные значения. Иначе же в массив aux запишется элемент второй половины массива. Затем копируем массив aux в исходный массив mas

После идет функция mergesort(), которая делит пополам массив, до тех пор, пока массив не поделится на массивы размером 2 или 1. Потом же массив объединяется функцией Merge() и в итоге массив будет отсортирован

# Схемы программы

# Блок-схема программы

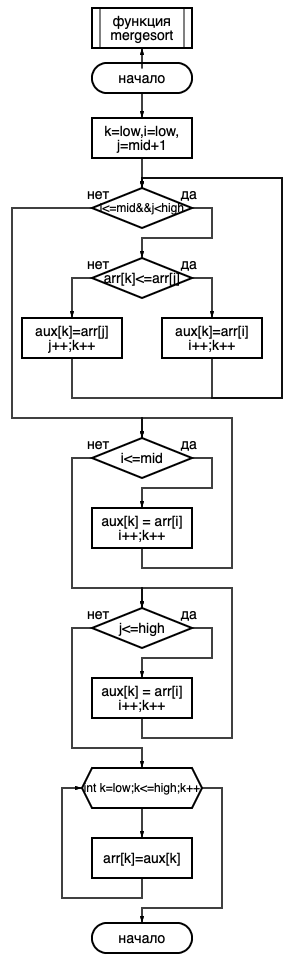
****

Рисунок 1 – Блок-схема программы

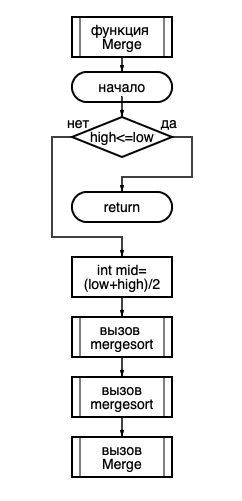
****

Рисунок 2 – Блок-схема программы

## 

# 4.2 Блок-схема алгоритма

## 

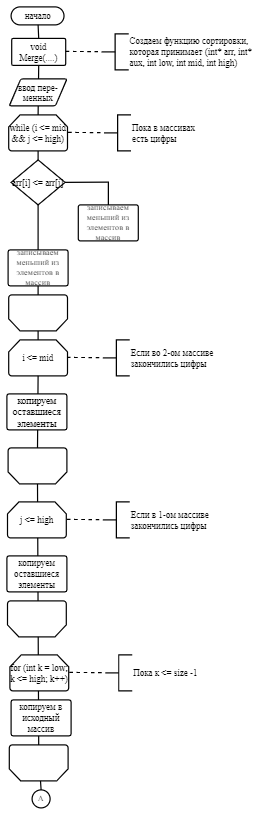


Рисунок 3 – Блок-схема алгоритма

## 

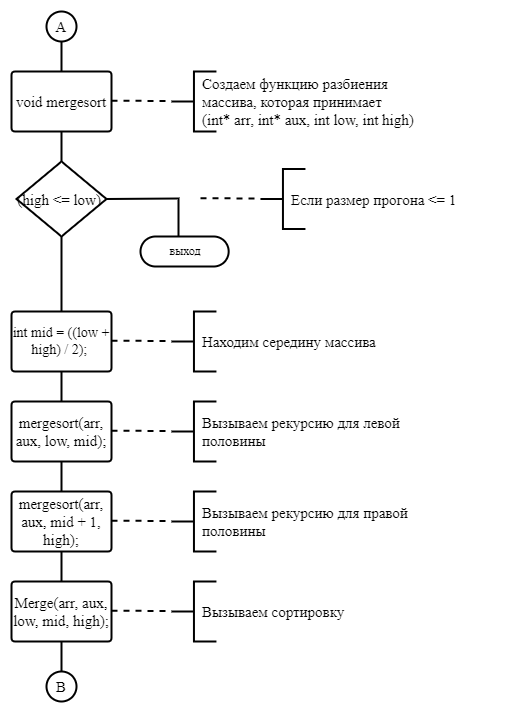


Рисунок 4 – Блок-схема алгоритма

## 

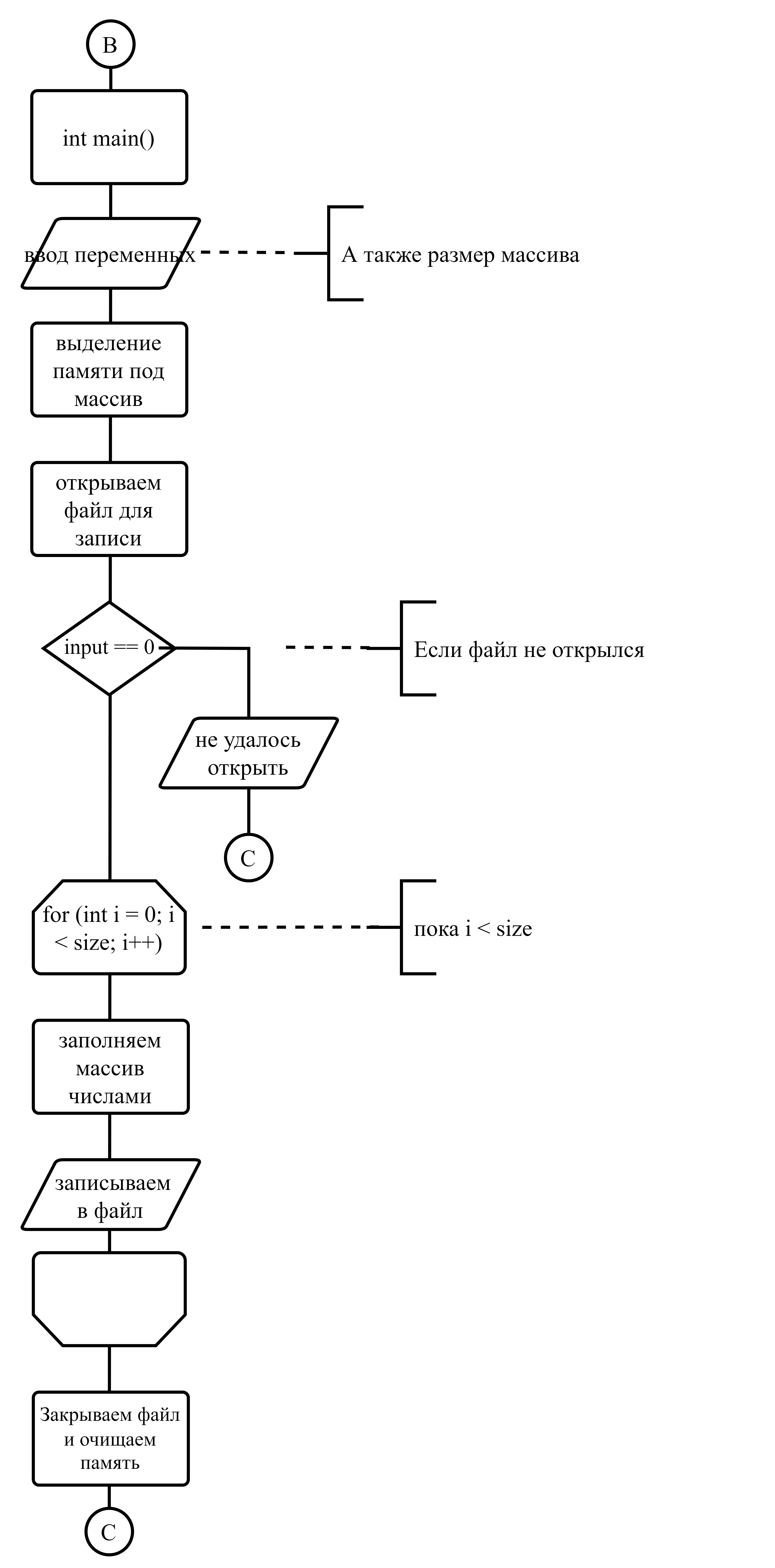


Рисунок 5 – Блок-схема алгоритма

## 

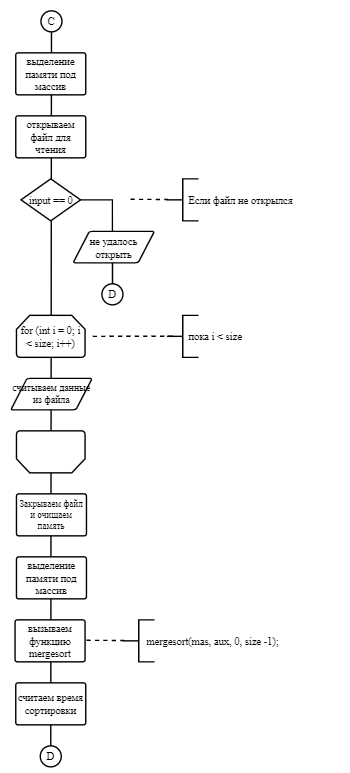


Рисунок 6 – Блок-схема алгоритма

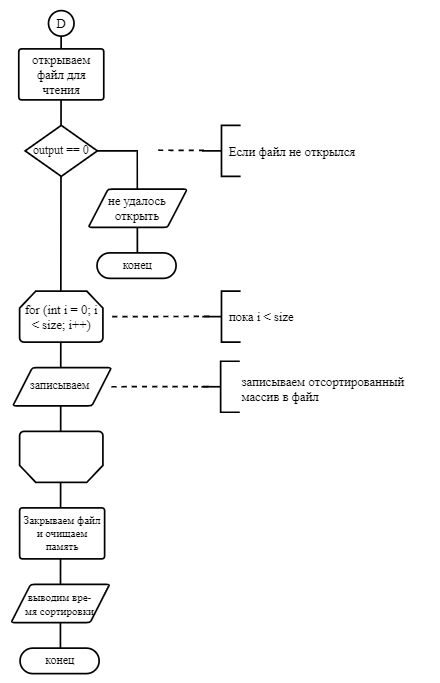


Рисунок 7 – Блок-схема алгоритма

# Тестирование программы

# Тестирование на разных наборах данных

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены в Приложении А на рисунках 16 - 25.

Таблица 1 – Тестовый набор данных

|  |  |  |
| --- | --- | --- |
| № теста | Размер массива size | Время выполнения сортировки в секундах |
| 1 | 10000 | 0.3084 |
| 2 | 20000 | 0.6206 |
| 3 | 30000 | 0.8168 |
| 4 | 40000 | 0.9318 |
| 5 | 50000 | 1.0345 |
| 6 | 60000 | 1.4409 |
| 7 | 70000 | 1.5490 |
| 8 | 80000 | 1.4483 |
| 9 | 90000 | 1.8713 |
| 10 | 100000 | 2.0149 |

# 

# Анализ полученных результатов

На основании анализа данных, полученных в результате тестирования алгоритма сортировки слиянием, можно сделать вывод, что время, затраченное на работу программы относительно количества элементов увеличивается по формуле N \* log N, где N – это размер общего массива. Т.е. сортировка массива выполниться за N операций.

Рисунок 8 – диаграмма работы программы



# 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.

1. **Совместная работа**

Создал репозиторий на gitHub’е. https://github.com/t1muurr/practice.git.

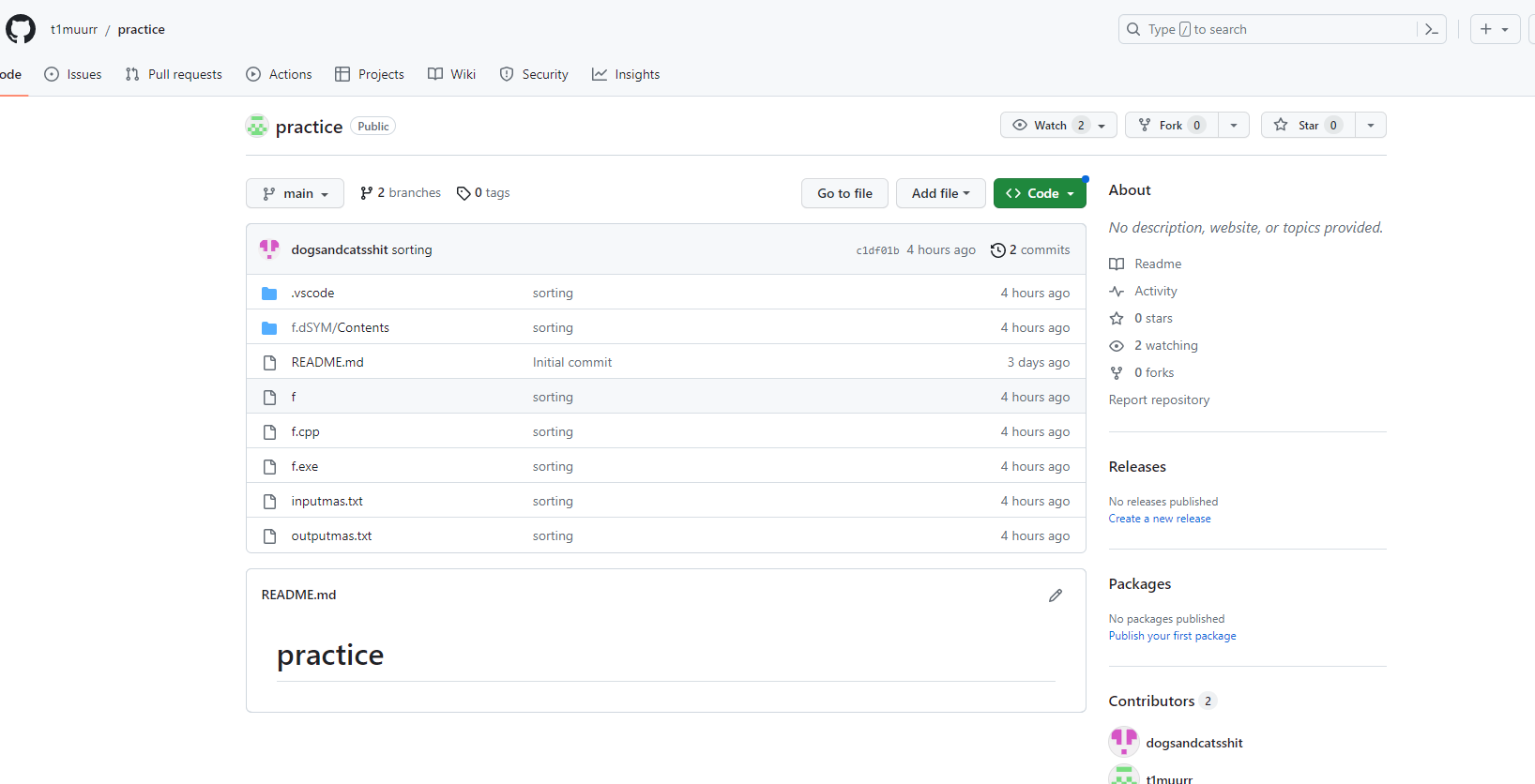


Рисунок 9 – скриншот GitHub(репозитория)

Создал папку и клонировал туда репозиторий.

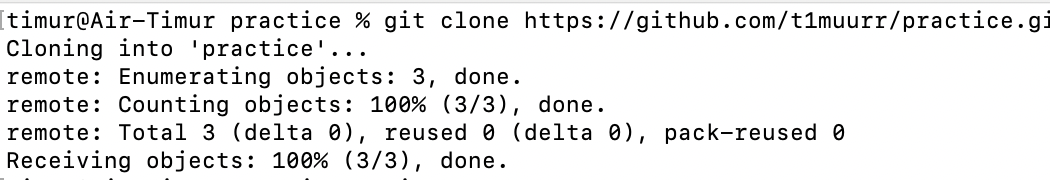


Рисунок 10 – скриншот GitBash

Добавил файлы, сделал коммит.



Рисунок 11 – скриншот GitBash

Изменил файлы, сделал коммит.

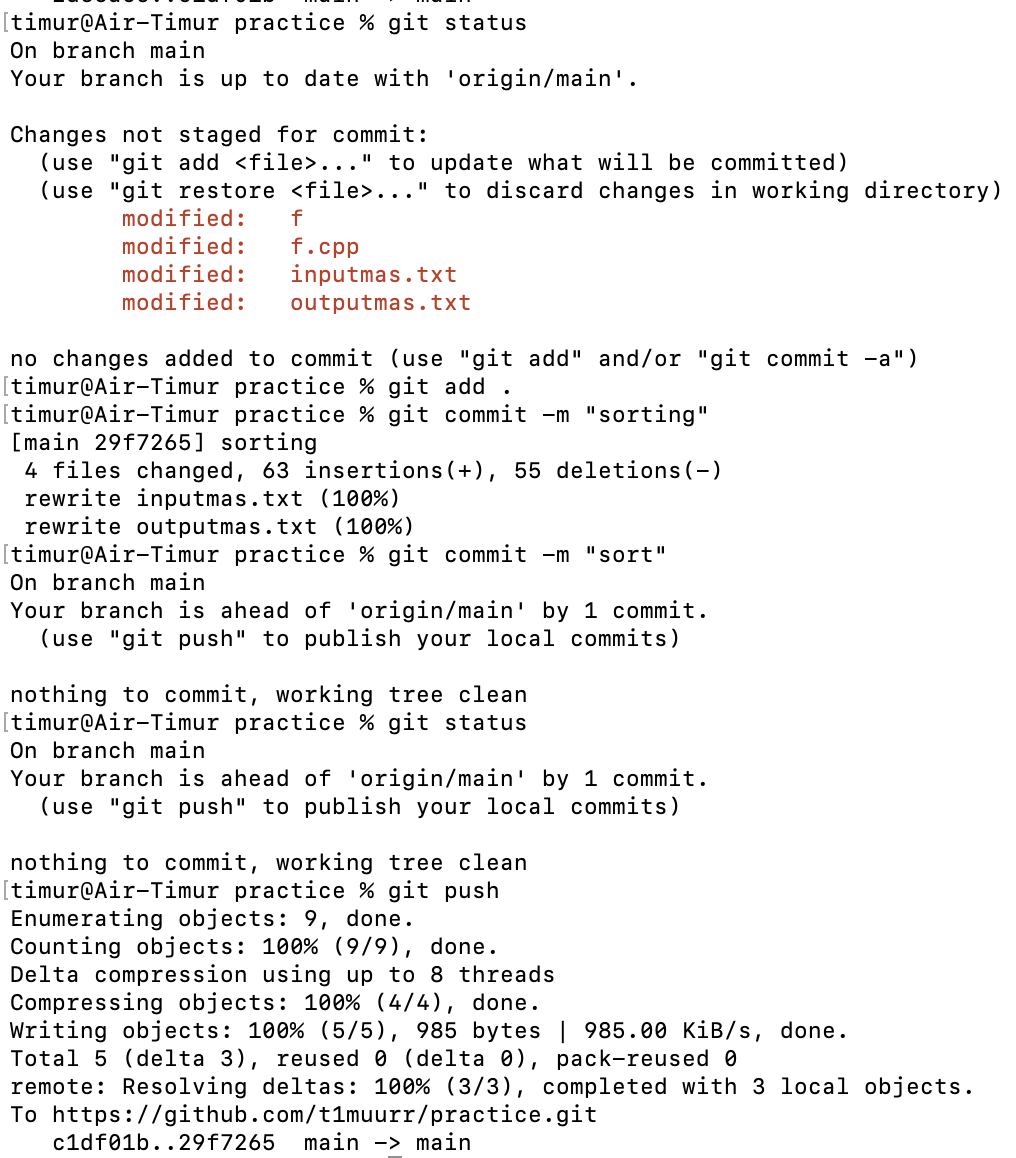


Рисунок 12 – скрин GitBash

Изменил файлы, добавил отчет, сделал их отслеживаемыми. Создал коммит. Отправил эту ветку main на gitHub в ветку main со всеми файлами.

Проверил на gitHub’е.

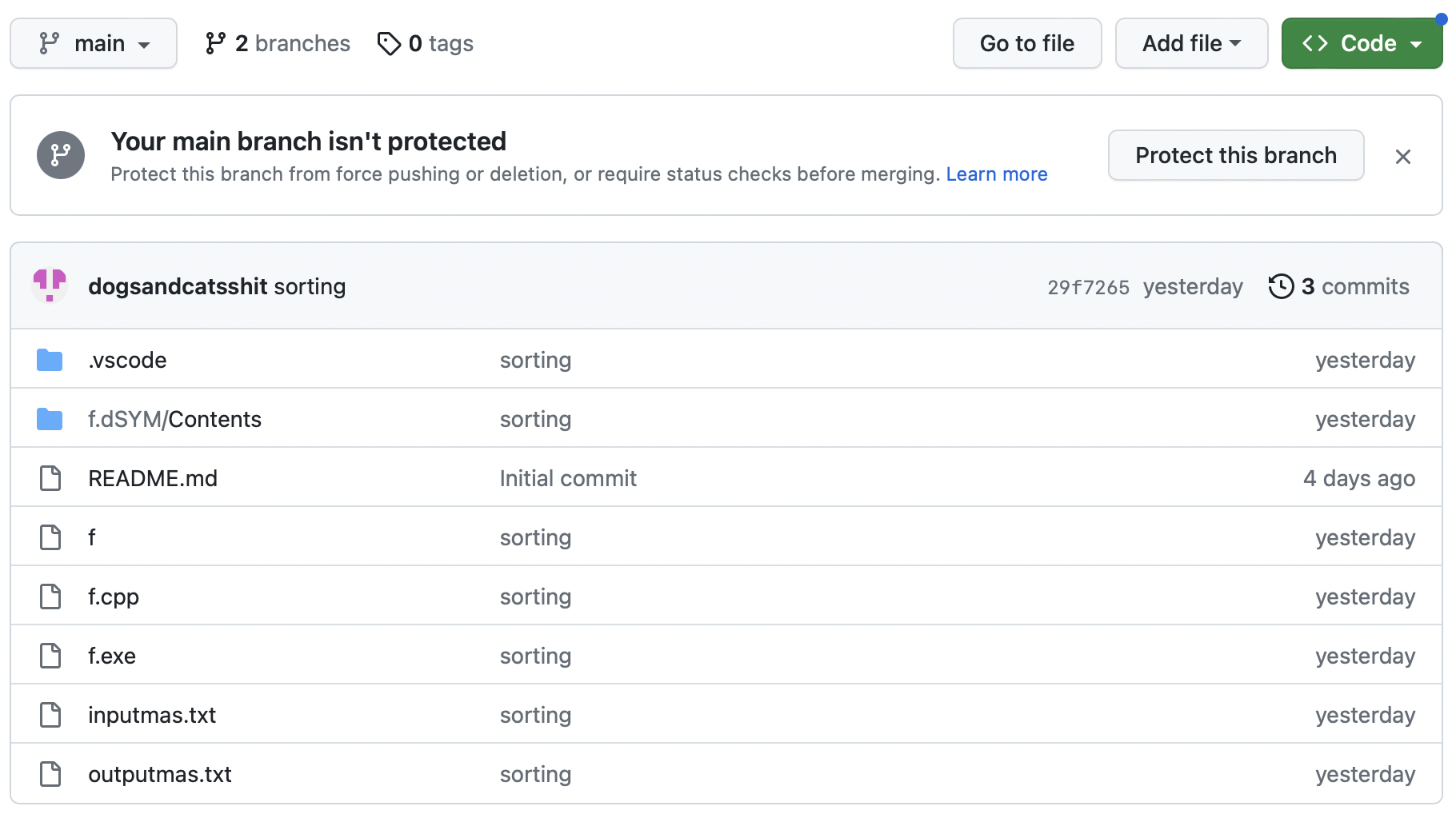


Рисунок 13 – скриншот GitHub(репозитория)

# Заключение

Нашей бригадой были получены навыки совместной работы с помощью сервиса GitHub, навыки использования программы Git Bash. Нами так же был изучен алгоритм сортировки слиянием. Изосин М.А. написал программу, выполняющую считывание массива из файла и запись отсортированного массива в файл, выполнил тестирование программы, выполнил отладку программы и оформил отчет по данной практике. Дасаев Т. З. написал программу, выполняющую данную сортировку над массивом псевдослучайных чисел и оформил отчет 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. Можно повысить максимальную разность чисел.

# Список используемой литературы

* + 1. Сортировка Слиянием [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Сортировка\_слиянием
    2. И. В. Красиков, И.Е. Красикова. Алгоритмы. Просто как дважды два / И. В. Красиков, И. Е. Красикова. - М. : Эксмо, 2007. - 256 с.
    3. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. СПб.: Невский диалект, 2001. 352с.
    4. Царев, Р. Ю. Структуры и алгоритмы обработки данных. Поиск и сортировка данных / Р.Ю. Царев. Красноярск: ИПЦ КГТУ, 2005. 60с.

# Приложение А

Результаты работы программы.

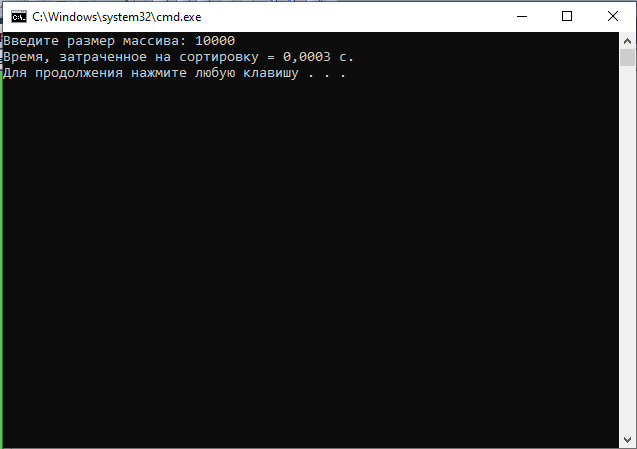


Рисунок 14 – скриншот результата работы программы

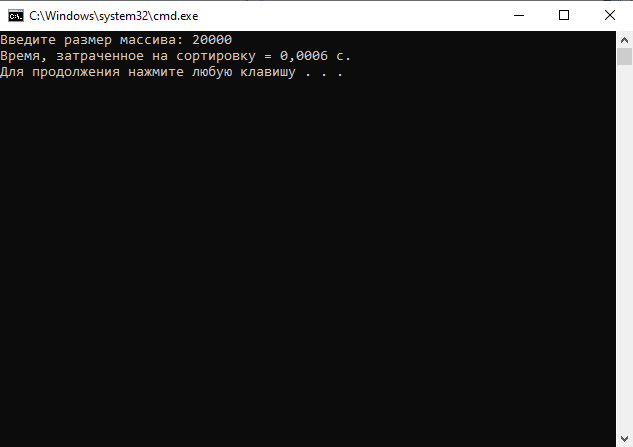


Рисунок 15 – скриншот результата работы программы

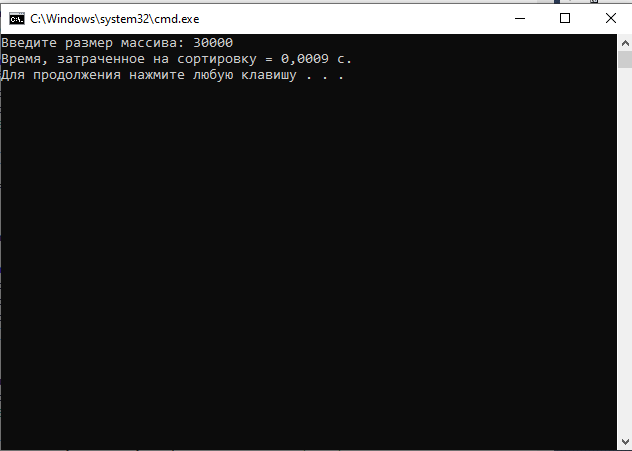


Рисунок 16 – скриншот результата работы программы

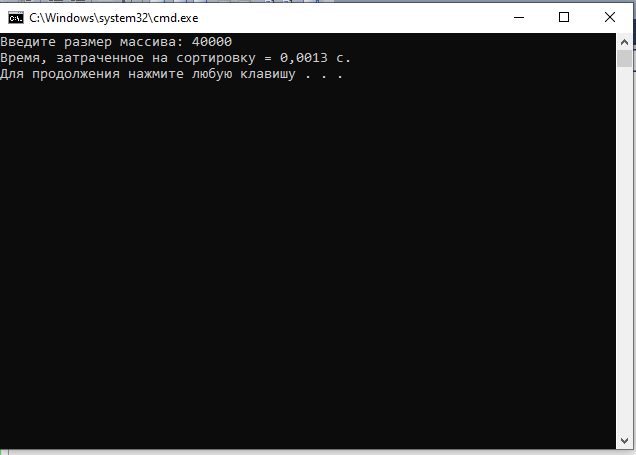


Рисунок 17 – скриншот результата работы программы

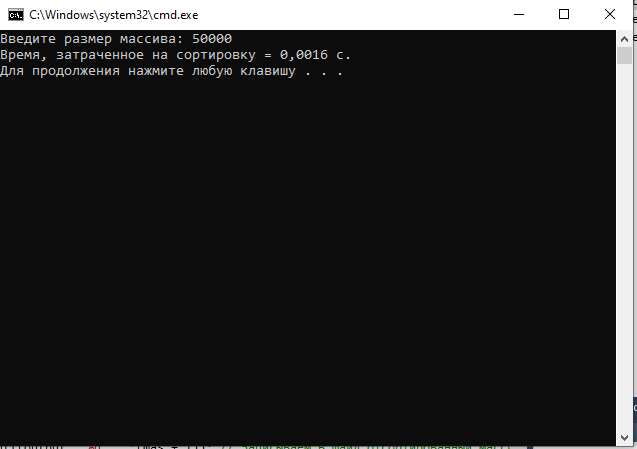


Рисунок 18 – скриншот результата работы программы

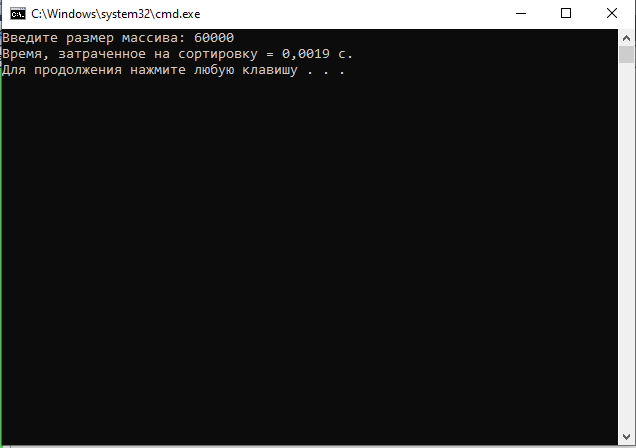


Рисунок 19 – скриншот результата работы программы

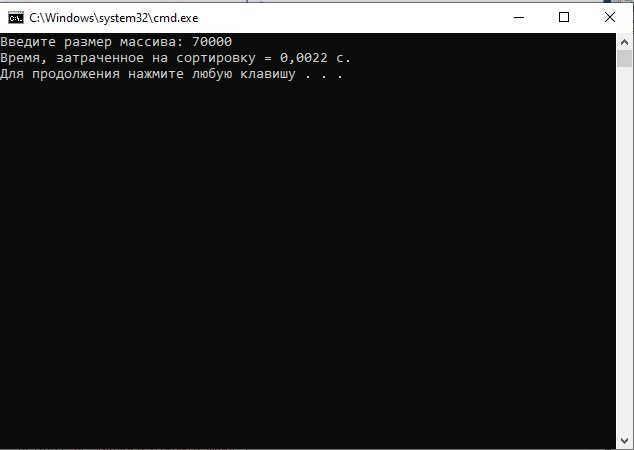


Рисунок 20 – скриншот результата работы программы

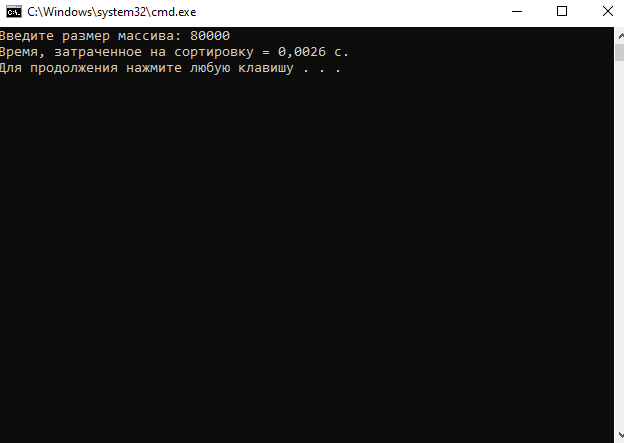


Рисунок 21 – скриншот результата работы программы

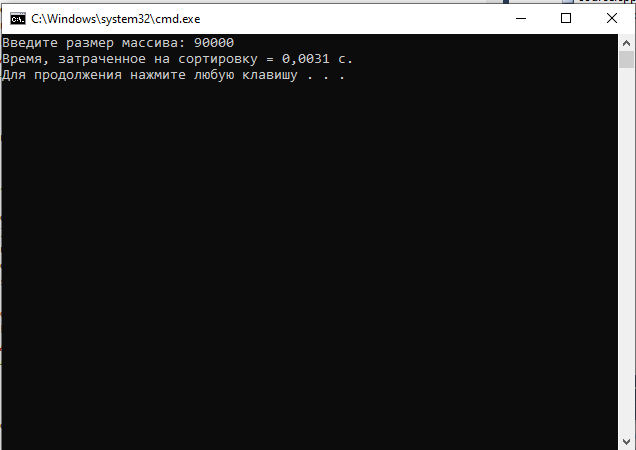


Рисунок 22 – скриншот результата работы программы

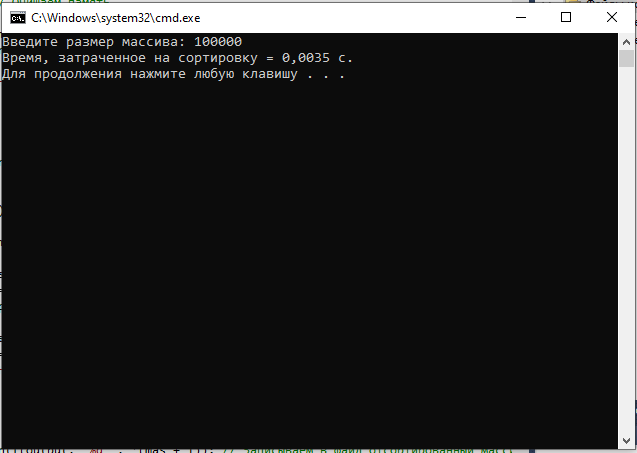


Рисунок 23 – скриншот результата работы программы

# Приложение Б (Листинг)

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

void Merge(int arr[], int\* aux, int low, int mid, int high)

{

int k = low, i = low, j = mid + 1;

// пока есть элементы в левом и правом рядах

while (i <= mid && j <= high)

{

if (arr[i] <= arr[j])

{

aux[k] = arr[i];

i++; k++;

}

else

{

aux[k] = arr[j];

j++; k++;

}

}

// копируем оставшиеся элементы

while (i <= mid)

{

aux[k] = arr[i];

i++; k++;

}

// копируем оставшиеся элементы

while (j <= high)

{

aux[k] = arr[j];

j++; k++;

}

// Вторую половину копировать не нужно (поскольку остальные элементы

// уже находятся на своем правильном месте во вспомогательном массиве)

// копируем обратно в исходный массив, чтобы отразить порядок сортировки

for (int k = low; k <= high; k++)

{

arr[k] = aux[k];

}

}

// Сортируем массив `arr[low…high]`, используя вспомогательный массив `aux`

void mergesort(int arr[], int\* aux, int low, int high)

{

// базовый вариант

if (high <= low)

{ // если размер прогона <= 1

return;

}

// найти середину

int mid = ((low + high) / 2);

// рекурсивно разделяем прогоны на две половины до тех пор, пока размер прогона не станет <= 1,

// затем объединяем их и возвращаемся вверх по цепочке вызовов

mergesort(arr, aux, low, mid); // разделить/объединить левую половину

mergesort(arr, aux, mid + 1, high); // разделить/объединить правую половину

Merge(arr, aux, low, mid, high); // объединить два полупрогона.

}

int main()

{

setlocale(LC\_ALL, "Rus"); // Консоль на русском

srand(time(0));

FILE\* input, \* output; // Указатели на файлы

int\* mas, \* aux; // Указатель на массив

int size; // Размер массива

float timer; // Переменная для подсчета времени сортировки

printf("Введите размер массива: ");

scanf("%d", &size); // Ввод размера массива

mas = (int\*)malloc(size \* sizeof(int)); // Выделение памяти под массив

input = fopen("inputmas.txt", "w"); // Открываем файл для записи

if (input == NULL) // Если файл не открылся

printf("Не удалось открыть файл");

else // Если файл открылся

{

for (int i = 0; i < size; i++) // Пока i меньше размера массива

{

\*(mas + i) = rand() % 2001 - 1000; // Заполняем массив числами диапащона [-1000:1000]

fprintf(input, "%d ", \*(mas + i)); // Записываем массив в файл

}

}

fclose(input); // Закрываем файл

free(mas); // Очищаем память

mas = (int\*)malloc(size \* sizeof(int)); // Выделение памяти под массив

input = fopen("inputmas.txt", "r"); // Открываем файл для чтения

if (input == NULL) // Если файл не открылся

printf("Не удалось открыть файл");

else // Если файл открылся

{

for (int i = 0; i < size; i++) // Пока i меньше размера массива

{

fscanf(input, "%d", &mas[i]); // Считываем данные из файла в массив

}

}

fclose(input); // Закрываем файл

aux = (int\*)malloc(size \* sizeof(int)); // Выделение памяти под массив

time\_t start = clock(); // Кладем в start нынешнее время

mergesort(mas, aux, 0, size - 1); // Вызываем функцию

time\_t stop = clock(); // Кладем в stop нынешнее время

timer = (stop - start) / 10000.0; // Рассчитываем время сортировки в секундах;

output = fopen("outputmas.txt", "w"); // Открываем файл для записи

if (output == NULL) // Если файл не открылся

printf("Не удалось открыть файл");

else // Если файл открылся

{

for (int i = 0; i < size; i++) // Пока i меньше размера массива

{

fprintf(output, "%d ", \*(mas + i)); // Записываем в файл отсортированный массив

}

}

fclose(output); // Закрываем файл

free(mas); // Очищаем память

free(aux); // Очищаем память

printf("Время, затраченное на сортировку = %0.4f с.\n", timer);

return 0;

}