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 обучения 1 семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

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

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

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

(должность, ученая степень, ученое звание)

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

Утвержден на заседании кафедры

«Вычислительная техника»

"___" _____ 20 г.

Заведующий кафедрой

_____М.А. Митрохин

ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ

(2022/2023 учебный год)

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 обучения 1 семестр 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 практики Зинкин С.А. _____ "___" _____ 2023 г.

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 обучения 1 семестр 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

Язык программирования Си разрабатывался в период с 1969 по 1973 годы в лабораториях Bell Labs, и к 1973 году на этот язык была переписана большая часть ядра UNIX, первоначально написанного на ассемблере PDP-11/20. Название языка стало логическим продолжением старого языка «Би», многие особенности которого были положены в основу.

По мере развития язык сначала стандартизировали как ANSI C, а затем этот стандарт был принят комитетом по международной стандартизации ISO как ISO C, ставший также известным под названием C90. В стандарте C99 язык получил новые возможности, такие как массивы переменной длины и встраиваемые функции. А в стандарте 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.

1 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.

1.1 Достоинства алгоритма

- Работает даже на структурах данных последовательного доступа.
- Хорошо сочетается с подкачкой и кэшированием памяти.
- Неплохо работает в параллельном 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).

1.2 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.

1.3 Типичные сценарии применения

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

2 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.

3 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

4 Схемы программы

4.1 Блок-схема программы

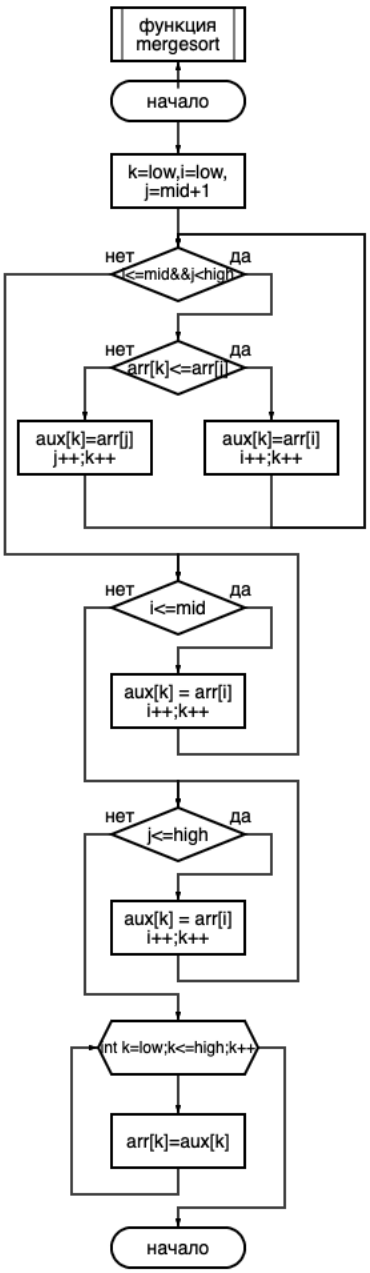

Рисунок 1 – Блок-схема программы

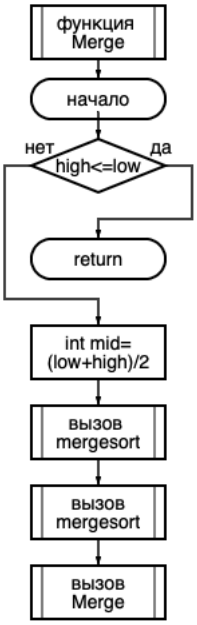

Рисунок 2 – Блок-схема программы

4.2 Блок-схема алгоритма

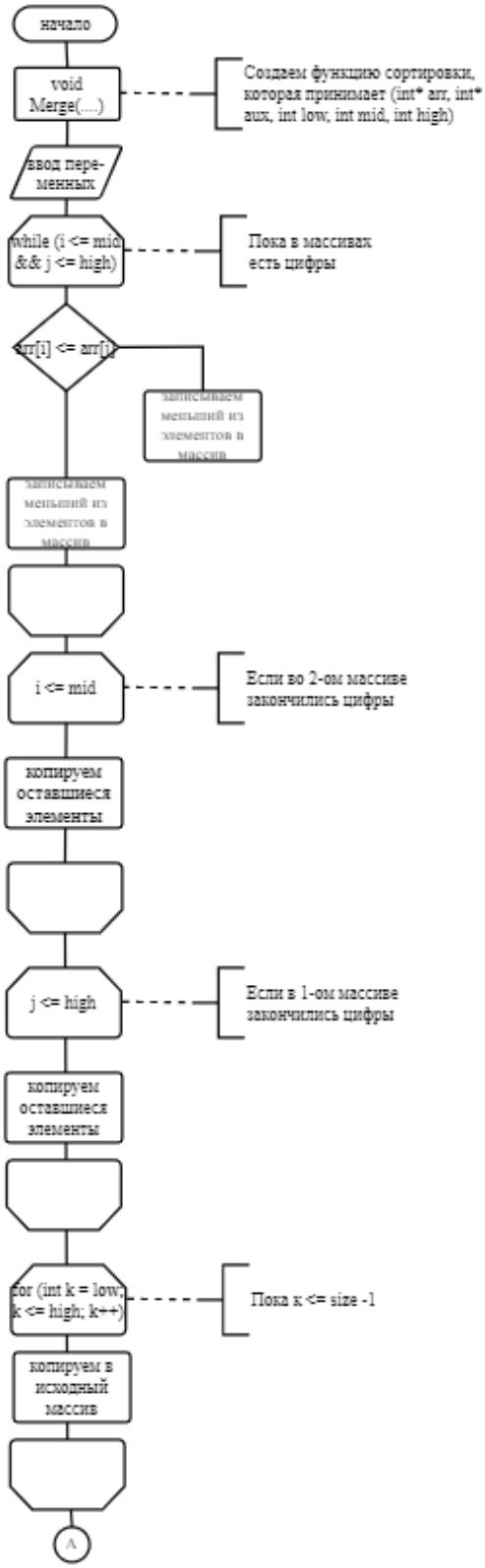

Рисунок 3 – Блок-схема алгоритма

Рисунок 4 – Блок-схема алгоритма

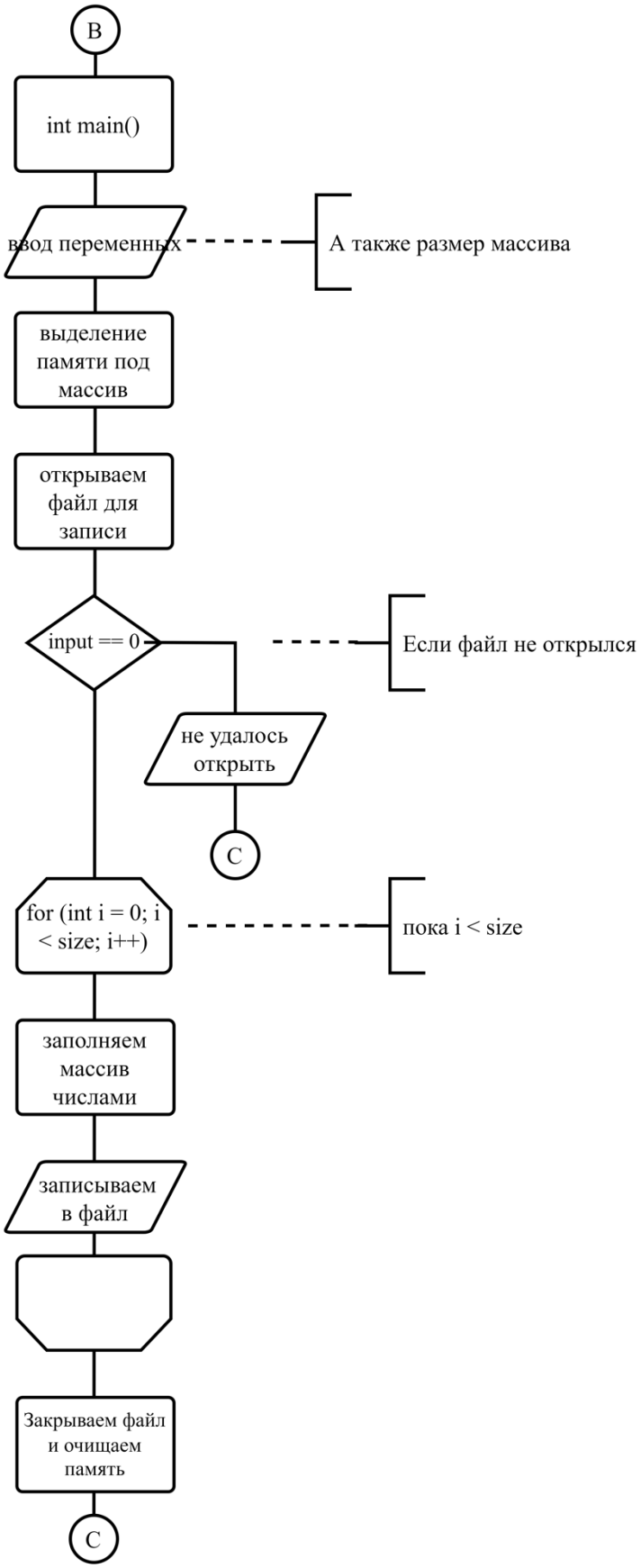

Рисунок 5 – Блок-схема алгоритма

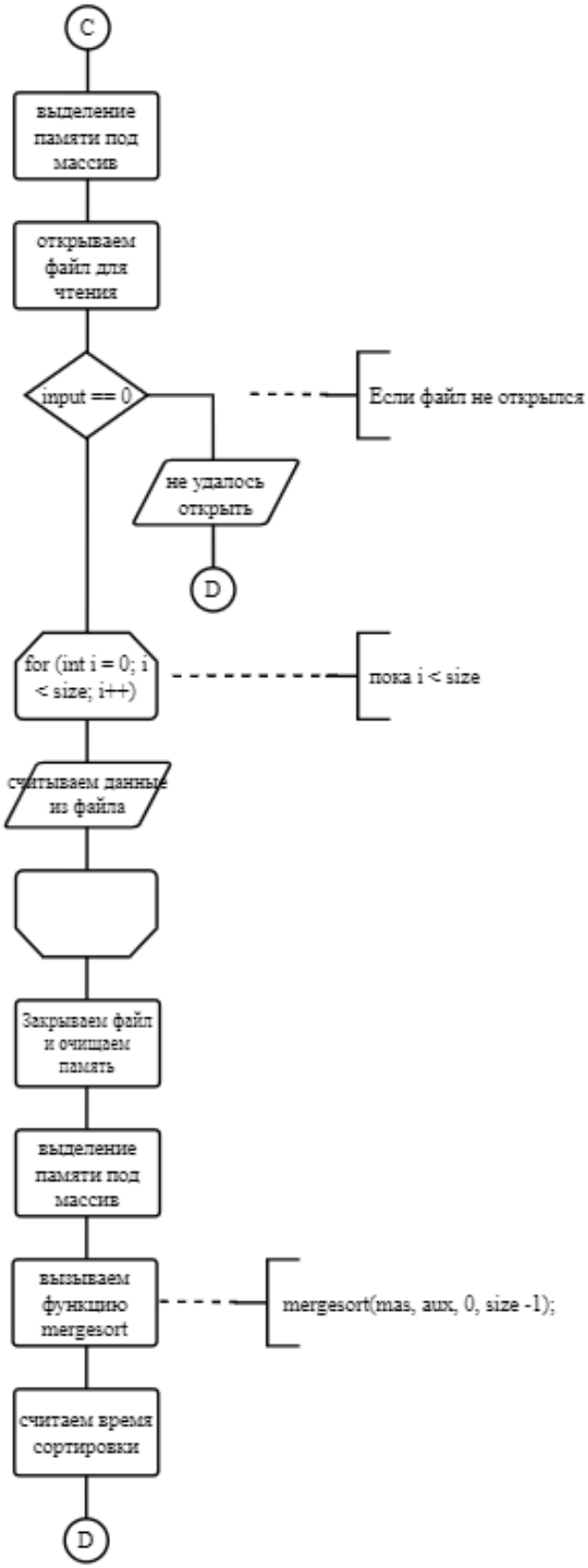

Рисунок 6 – Блок-схема алгоритма

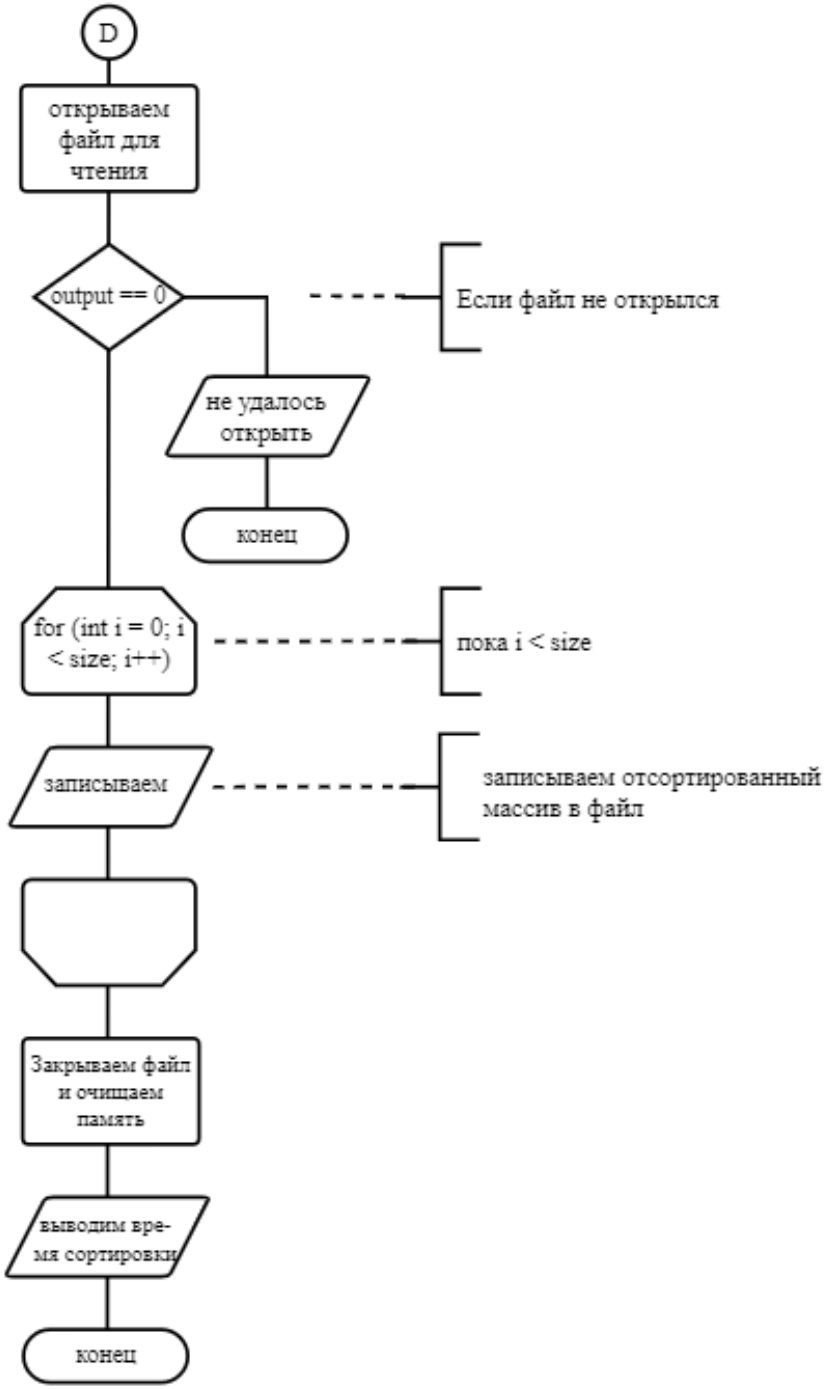

Рисунок 7 – Блок-схема алгоритма

5 Тестирование программы

5.1 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

5.2 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.

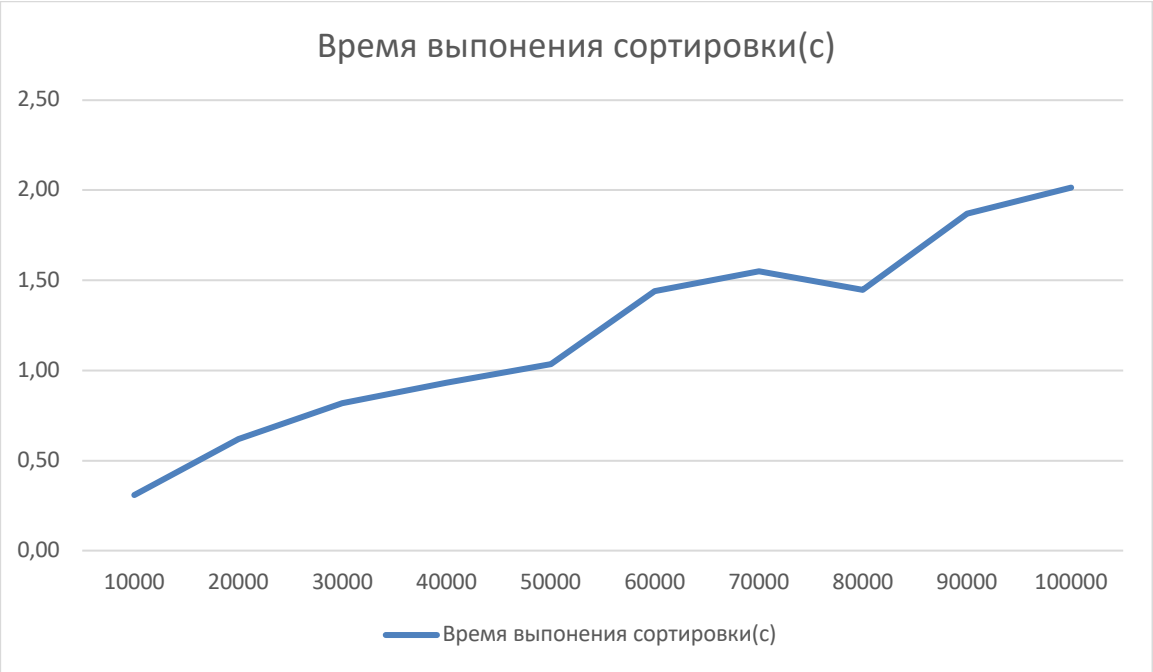

Рисунок 8 – диаграмма работы программы

6. 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.

7 Совместная работа

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

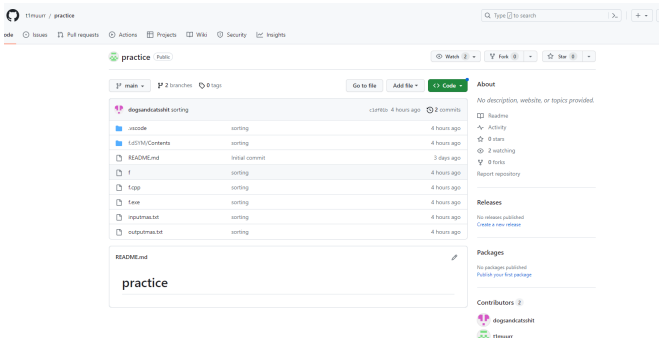

Рисунок 9 – скриншот GitHub(репозитория)

Создал папку и клонировал туда репозиторий.

```
timur@Air-Timur practice % git clone https://github.com/t1muurr/practice.gi
Cloning into 'practice'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Рисунок 10 – скриншот GitBash

Добавил файлы, сделал коммит.

```
timur@Air-Timur practice % git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
timur@Air-Timur practice % git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .vscode/
        f
        f.cpp
        f.dSYM/
        f.exe
        inputmas.txt
        outputmas.txt

nothing added to commit but untracked files present (use "git add" to track)
timur@Air-Timur practice % git add -
fatal: pathspec '-' did not match any files
timur@Air-Timur practice % git add .
timur@Air-Timur practice % git commit -m "sorting"
[main c1df01b] sorting
 8 files changed, 172 insertions(+)
 create mode 100644 .vscode/tasks.json
 create mode 100755 f
 create mode 100644 f.cpp
 create mode 100644 f.dSYM/Contents/Info.plist
 create mode 100644 f.dSYM/Contents/Resources/DWARF/f
 create mode 100755 f.exe
 create mode 100644 inputmas.txt
 create mode 100644 outputmas.txt
timur@Air-Timur practice % git push
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/t1muurr/practice.git/'
timur@Air-Timur practice % git remote -v
origin  https://github.com/t1muurr/practice.git (fetch)
origin  https://github.com/t1muurr/practice.git (push)
timur@Air-Timur practice % git push
Username for 'https://github.com': t1muurr
Password for 'https://t1muurr@github.com':
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (15/15), 1.75 MiB | 47.00 KiB/s, done.
Total 15 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/t1muurr/practice.git
   2deeac6..c1df01b  main -> main
```

Рисунок 11 – скриншот GitBash

Изменил файлы, сделал коммит.

```
timur@Air-Timur practice % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   f
        modified:   f.cpp
        modified:   inputmas.txt
        modified:   outputmas.txt

no changes added to commit (use "git add" and/or "git commit -a")
timur@Air-Timur practice % git add .
timur@Air-Timur practice % git commit -m "sorting"
[main 29f7265] sorting
 4 files changed, 63 insertions(+), 55 deletions(-)
 rewrite inputmas.txt (100%)
 rewrite outputmas.txt (100%)
timur@Air-Timur practice % git commit -m "sort"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
timur@Air-Timur practice % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
timur@Air-Timur practice % git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 985 bytes | 985.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/t1muurr/practice.git
   c1df01b..29f7265  main -> main
```

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.

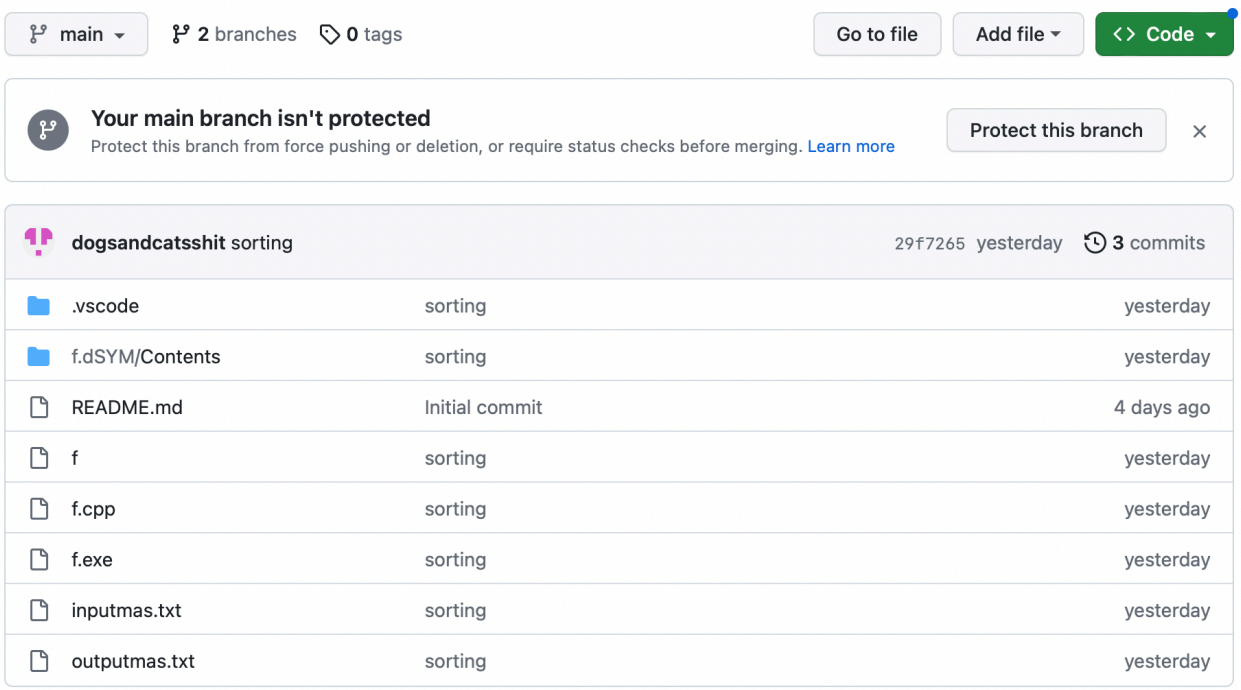

Рисунок 13 – скриншот GitHub(репозитория)

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 C/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.

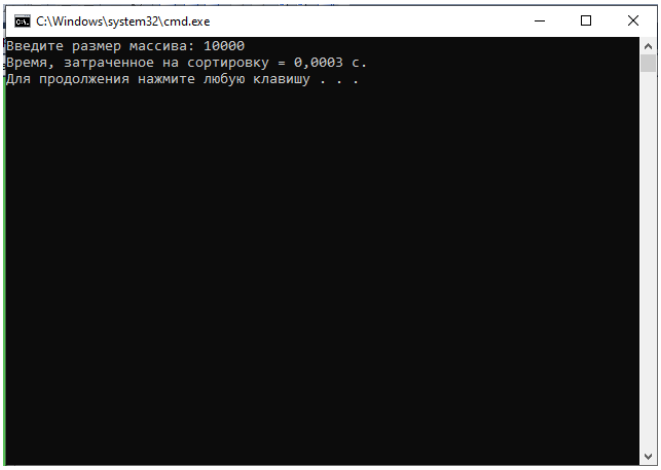

Рисунок 14 – скриншот результата работы программы

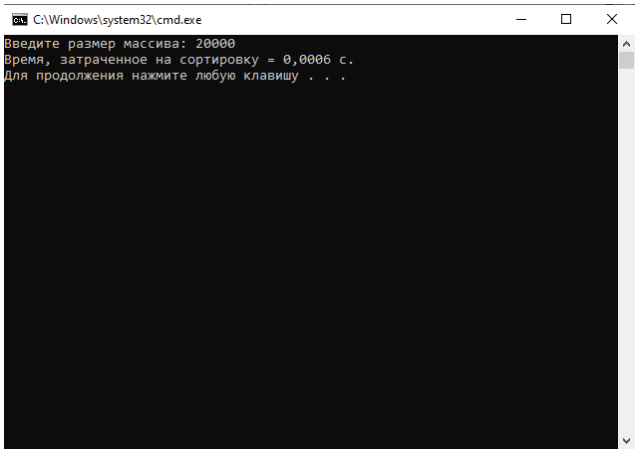

Рисунок 15 – скриншот результата работы программы

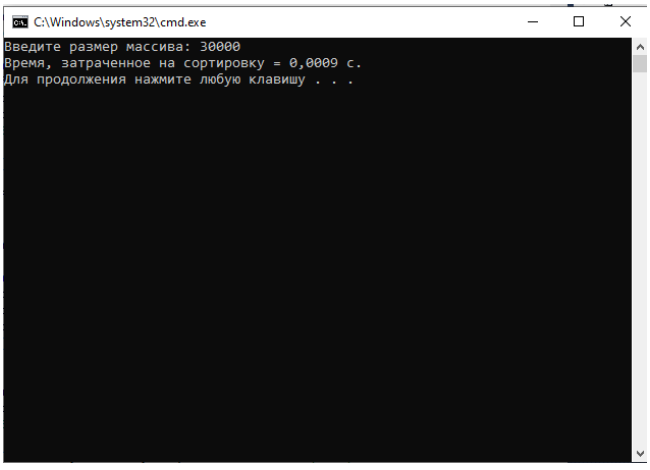

Рисунок 16 – скриншот результата работы программы

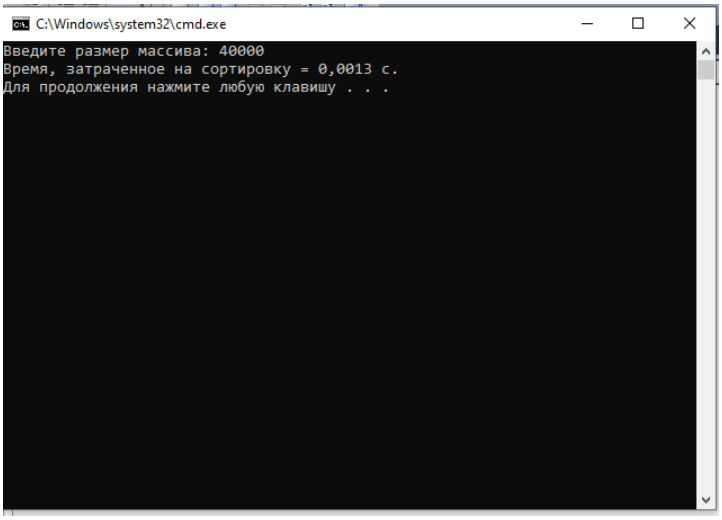

Рисунок 17 – скриншот результата работы программы

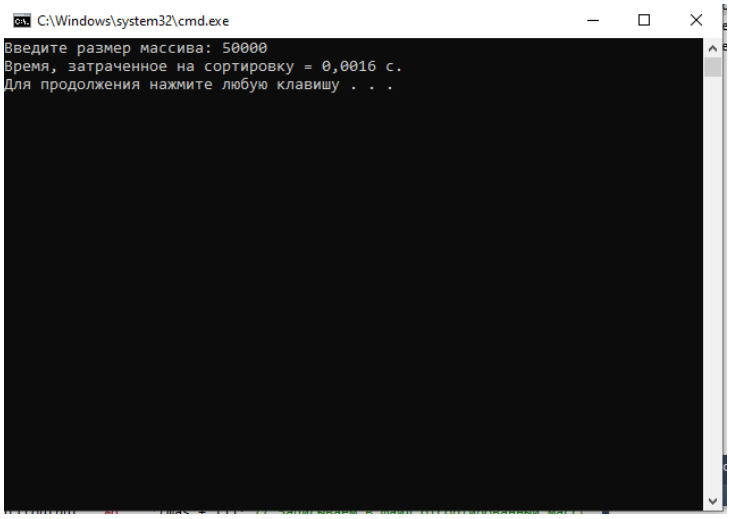

Рисунок 18 – скриншот результата работы программы

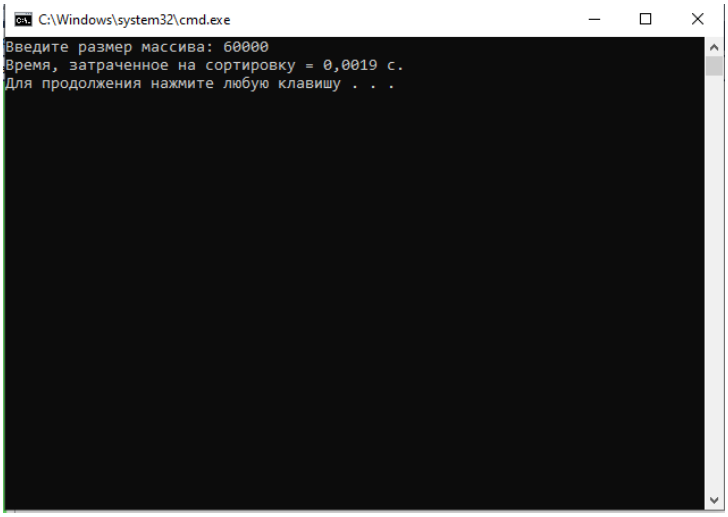

Рисунок 19 – скриншот результата работы программы

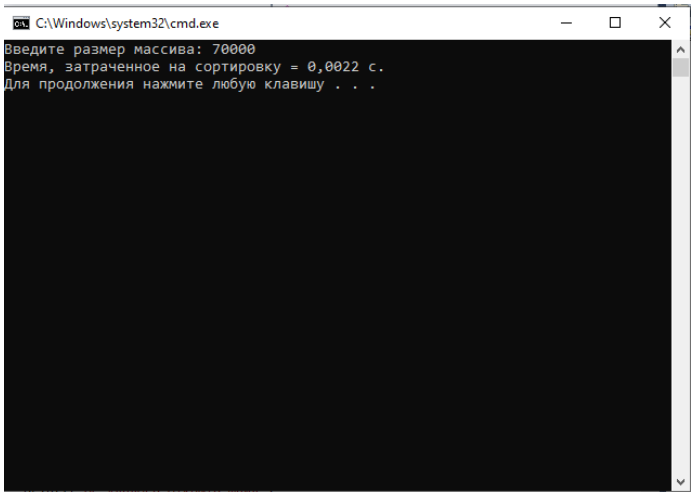

Рисунок 20 – скриншот результата работы программы

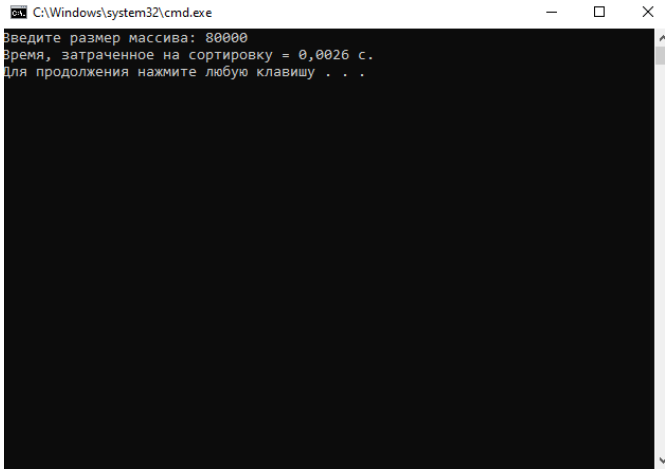

Рисунок 21 – скриншот результата работы программы

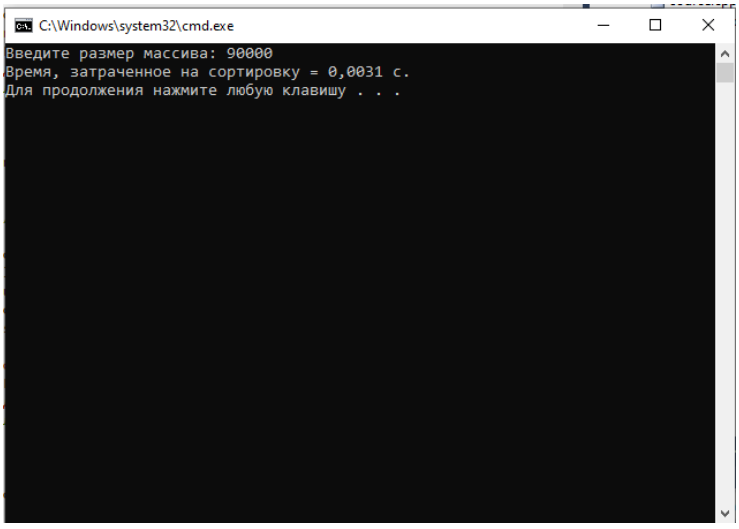

Рисунок 22 – скриншот результата работы программы

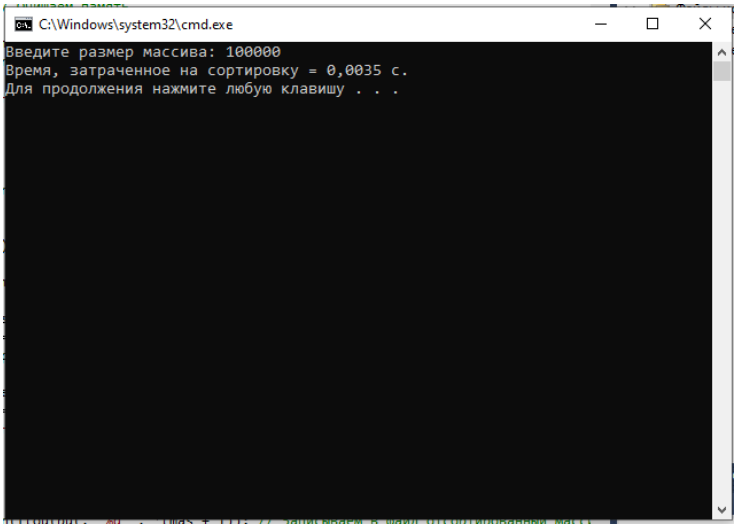

Рисунок 23 – скриншот результата работы программы

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;
    }
```