[Введение 2](#_Toc139316342)

[1 Постановка задачи 3](#_Toc139316343)

[1.1 Достоинства алгоритма 3](#_Toc139316344)

[1.2 Недостатки алгоритма 3](#_Toc139316345)

[1.3 Типичные сценарии применения 4](#_Toc139316346)

[2 Выбор решения 4](#_Toc139316347)

[3 Описание программы 4](#_Toc139316348)

[4 Схемы программы 5](#_Toc139316349)

[4.1 Блок-схема программы 5](#_Toc139316350)

[5 Тестирование программы 8](#_Toc139316351)

[5.1 Тестирование на разных наборах данных 8](#_Toc139316352)

[5.1 Анализ полученных результатов 8](#_Toc139316353)

[6. Отладка 9](#_Toc139316355)

[Заключение 12](#_Toc139316356)

[Список используемой литературы 13](#_Toc139316357)

[Приложение А 13](#_Toc139316358)

[Приложение Б (Листинг) 16](#_Toc139316359)

# 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" \o "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. Программы, соответствующие стандарту языка, могут компилироваться под различные архитектуры компьютеров.

Язык Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования как C++, C#, Java, PHP и др.

**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 обьема 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:

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

2) отладчик кода;

3) редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

4) веб-редактор;

5) дизайнер классов;

6) 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 основная функция int main(), в которой будет производиться работа с файлами.

Затем мы переводим консоль на русский язык командой *setlocale();*

Потом мы определяем указатели на файлы(inputmas.txt и outputmas.txt). С помощью scanf вводим необходимый нам размер массива. Выделяем память под массив с помощью malloc();

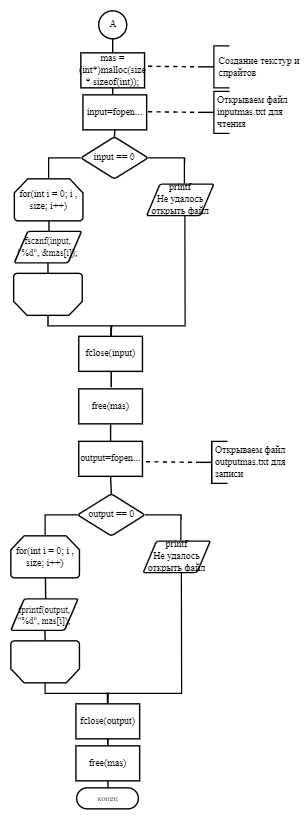
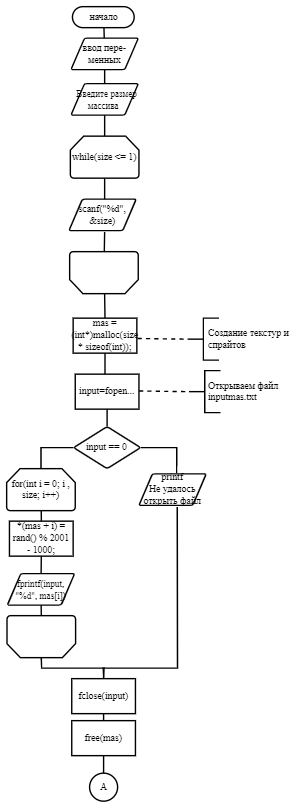
Далее мы открываем файл inputmas.txt. Проверяем на открытие, и если открылся, то заполняем массив псевдослучайными числами и записываем в него массив. Закрываем файл.

Затем открываем и считываем из этого файла массив элементов.

После этого мы открываем файл outputmas.txt. Проверяем на открытие, и если открылся, то записываем в него уже отсортированный массив.

# Схемы программы

# Блок-схема программы

****

# 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

|  |  |  |
| --- | --- | --- |
| № теста | Размер массива size | Время выполнения сортировки в секундах |
| 1 | 10000 | 0.0003 |
| 2 | 20000 | 0.0006 |
| 3 | 30000 | 0.0009 |
| 4 | 40000 | 0.0013 |
| 5 | 50000 | 0.0016 |
| 6 | 60000 | 0.0019 |
| 7 | 70000 | 0.0022 |
| 8 | 80000 | 0.0026 |
| 9 | 90000 | 0.0031 |
| 10 | 100000 | 0.0035 |

# 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.

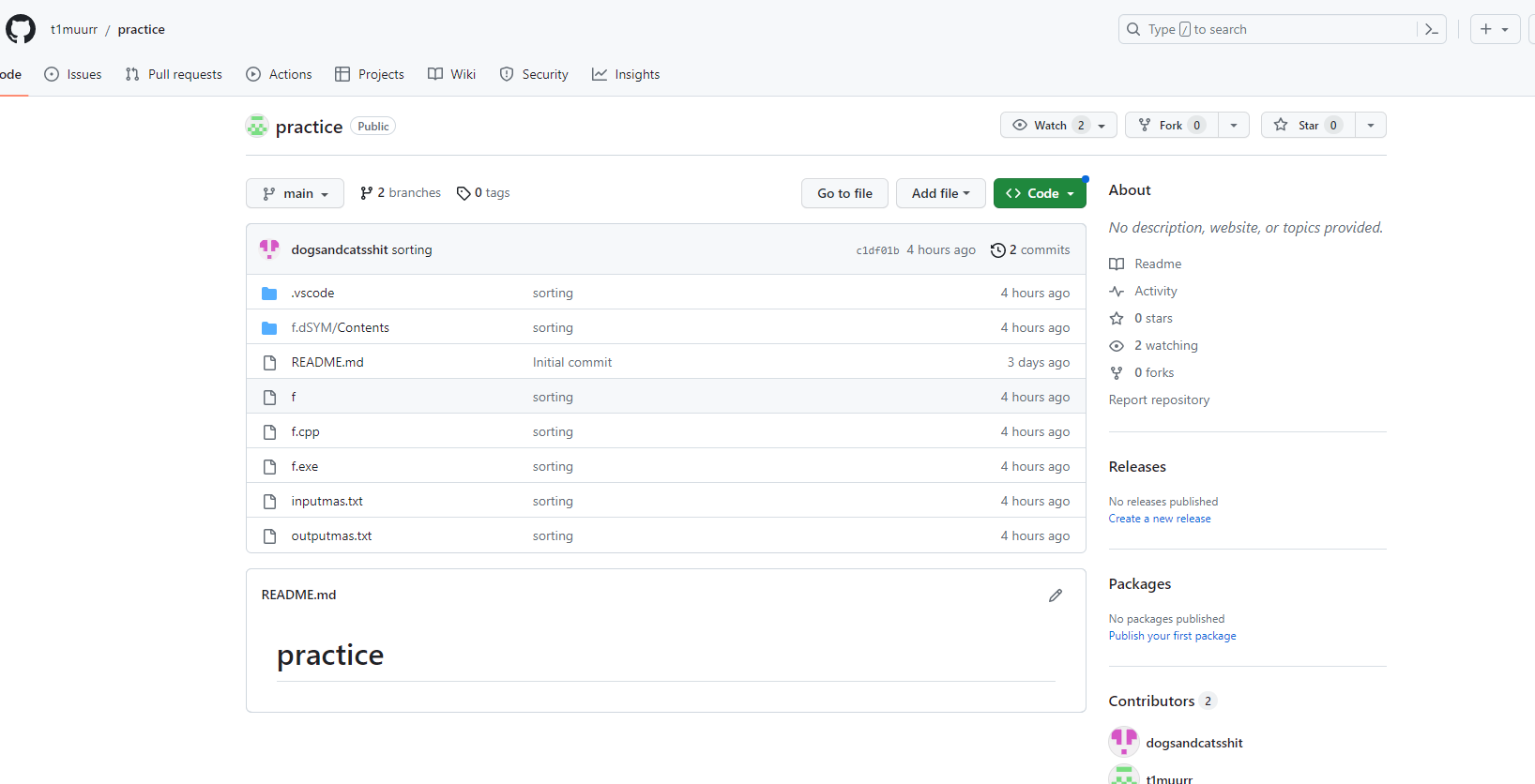


# Отладка

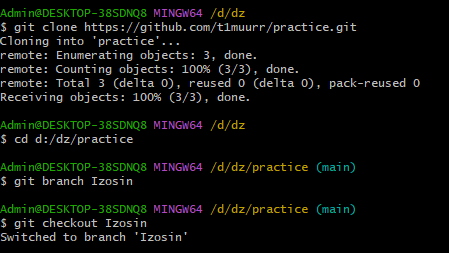
В качестве среды разработки была выбрана программа Microsoft Visual Studio , которая содержит в себе все необходимые средства для разработки и отладки модулей и программ. Для отладки программы использовались точки остановки и пошаговое выполнение кода программы, анализ содержимого глобальных и локальных 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**

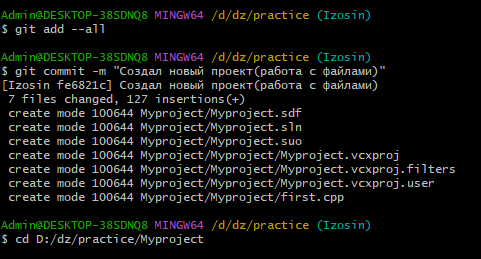
Создал репозиторий на gitHub’е. https://github.com/t1muurr/practice.git



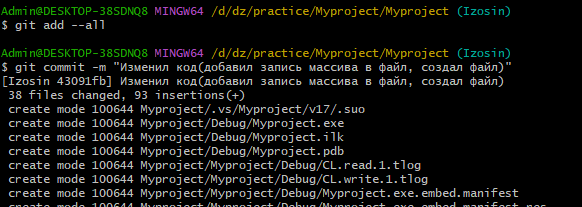
Создал папку и клонировал туда репозиторий. Затем создал отдельную ветку под своей фамилией Izosin. Перешел на нее. Тут будет проект моей части программы.



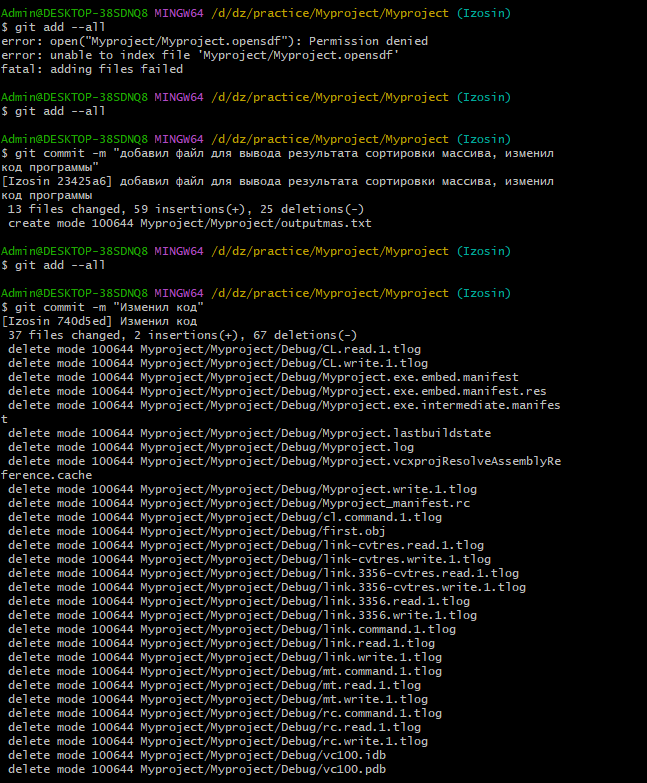
Добавил файлы, сделал коммит. Перешел в папку Myproject



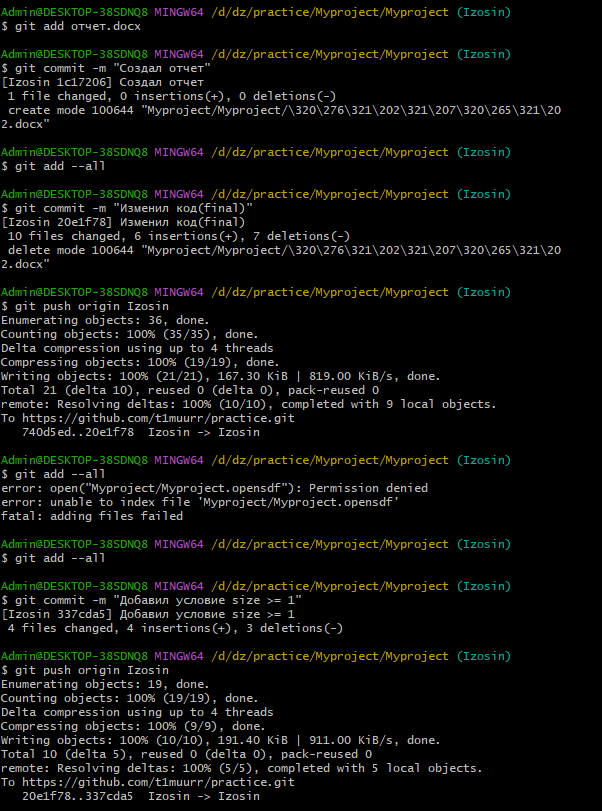
Изменил файлы, сделал коммит.



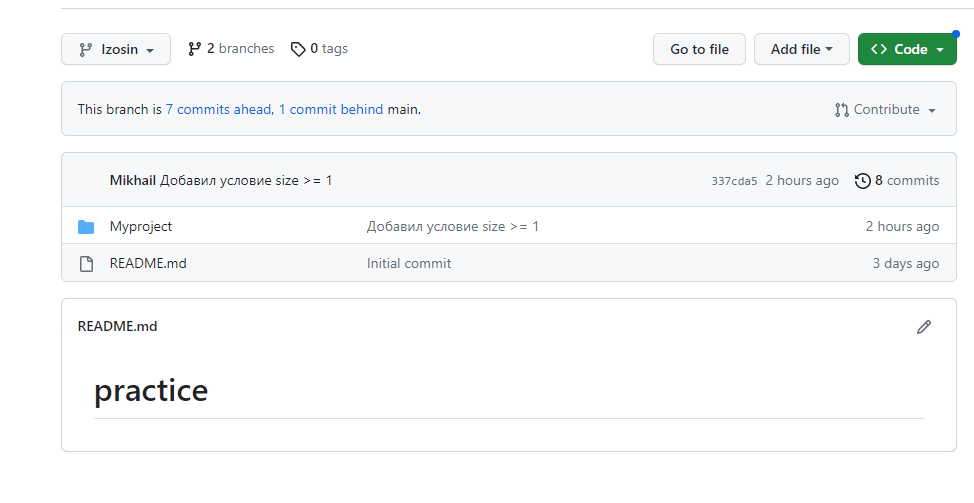
Опять изменил файлы, сделал коммиты.



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



Проверил на gitHub’е.



# Заключение

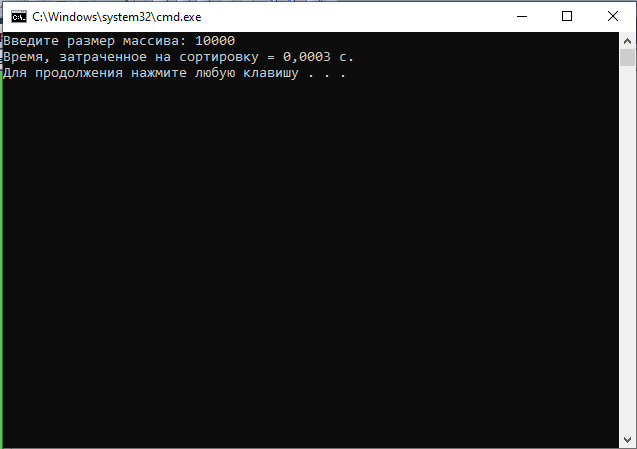
Нашей бригадой были получены навыки совместной работы с помощью сервиса GitHub, навыки использования программы Git Bash. Нами так же был изучен алгоритм сортировки слиянием. Изосин М.А. написал программу, выполняющую считывание массива из файла и запись отсортированного массива в файл, выполнил тестирование 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.

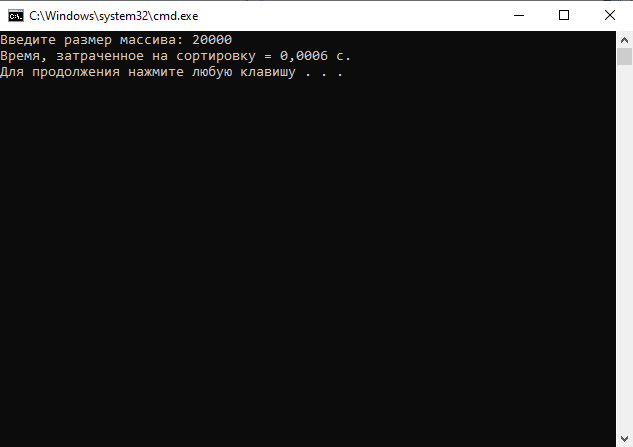
# Список используемой литературы

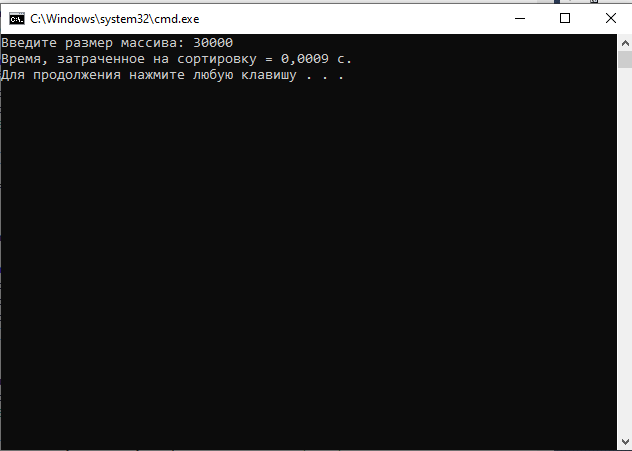
* + 1. Википедия – свободная энциклопедия. Сортировка слиянием. Ссылка: https://ru.wikipedia.org/wiki/Сортировка\_слиянием
    2. И. В. Красиков, И.Е. Красикова. Алгоритмы. Просто как дважды два / И. В. Красиков, И. Е. Красикова. - М. : Эксмо, 2007. - 256 с.
    3. Вирт, Н. Алгоритмы и структуры данных / 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.

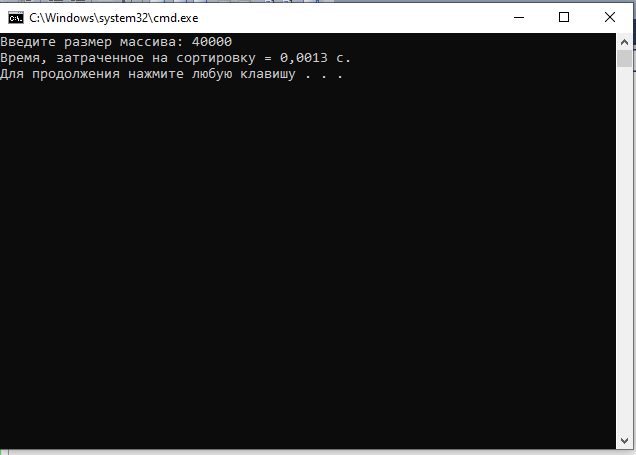
# Приложение А

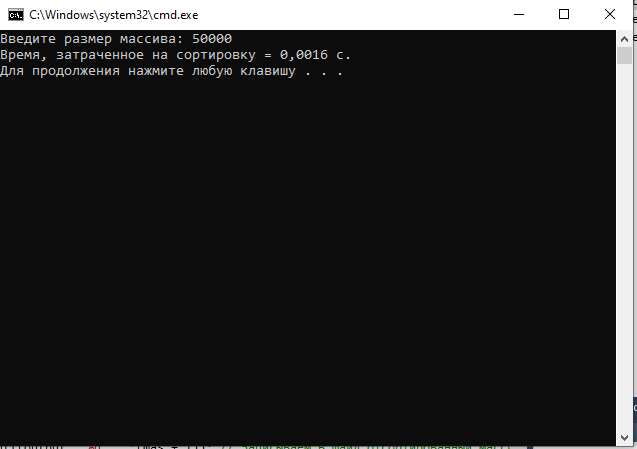
Результаты работы программы.

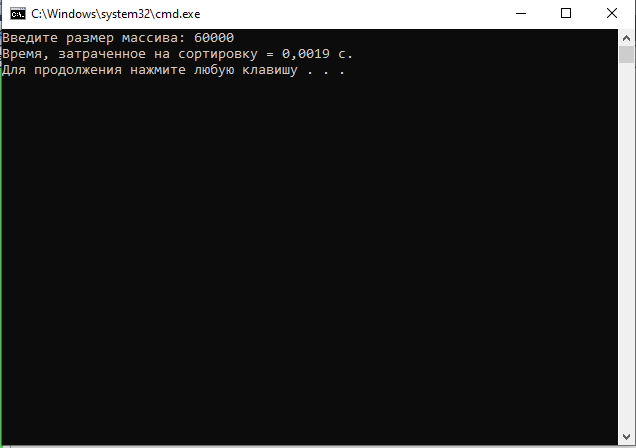


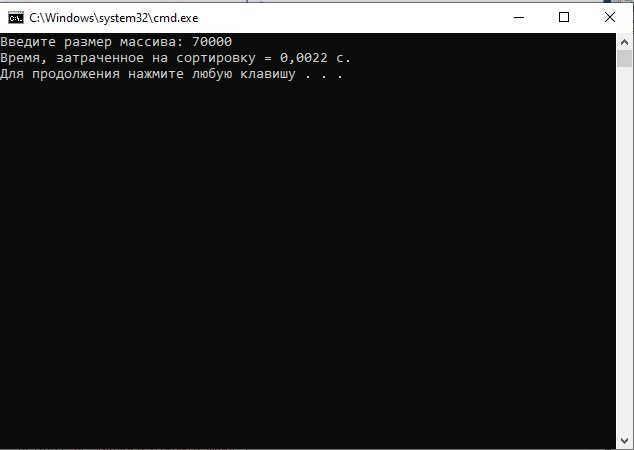


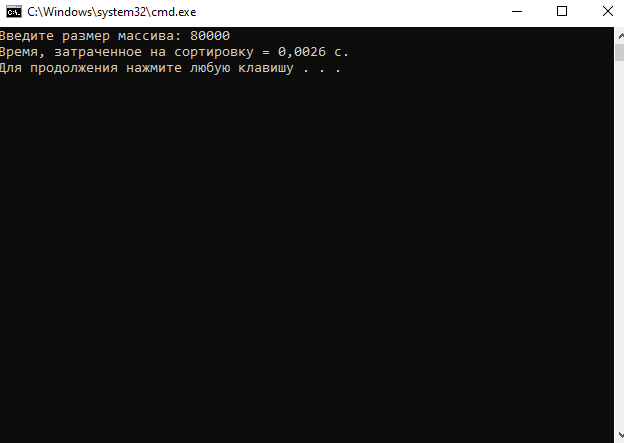


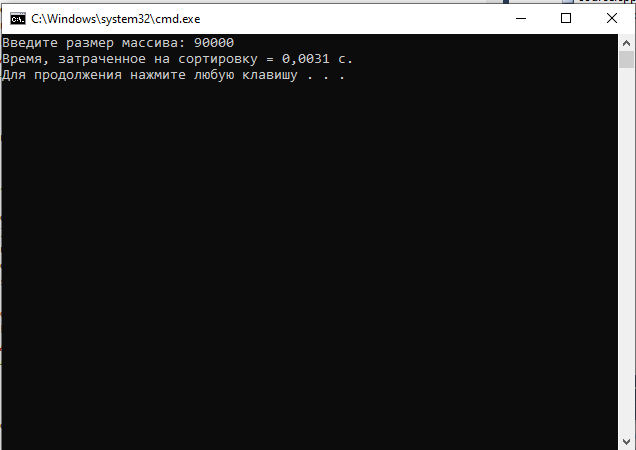


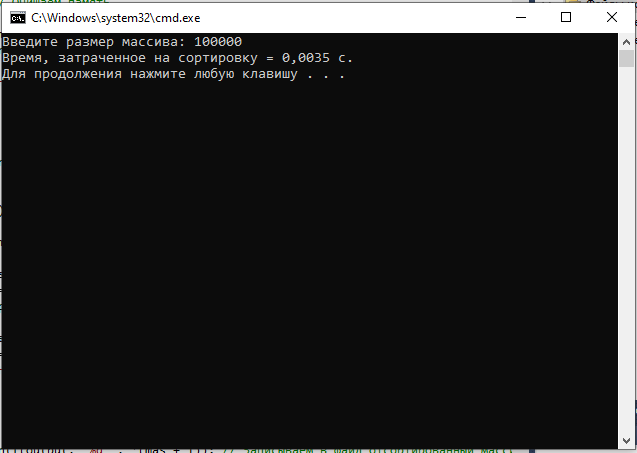












# Приложение Б (Листинг)

#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;

}