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

Утвержден на заседании кафедры

«Вычислительная техника»

"___" _____ 20 г.

Заведующий кафедрой

_____М.А. Митрохин

ОТЧЕТ ПО УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКЕ

(2022/2023 учебный год)

Самофалова Александра Владимировна

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)

Самофалова Александра Владимировна

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)

Самофалова Александра Владимировна

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»

Самофалова А.В. выполняла 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. Выявила достоинства и недостатки алгоритма, а также типичные сценарии применения. Выполнила блок-схему алгоритма. Оформила отчёт.

Бакалавр Самофалова А.В. _____ "___" _____ 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)

Самофалова Александра Владимировна

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 Самофалова А.В. решала следующие задачи: создание алгоритма сортировки слиянием, анализ алгоритма.

За период выполнения практики были освоены основные понятия и технологии сортировки слиянием. Во время выполнения работы Самофалова А.В. 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 Самофалова А.В. заслуживает оценки «_____».

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

Содержание

Введение

Сортировка данных на сегодняшний день при современном развитии компьютерных технологий является одним из наиболее распространенных процессов современной обработки данных. Задачи на сортировку данных встречаются очень часто в различных профессиональных сферах деятельности.

Алгоритмы сортировки очень широко распространяются практически во всех задачах обработки информации. При этом они настолько тесно связаны друг с другом, что образуют отдельный класс алгоритмов. Алгоритмы сортировки, как правило, применяются с целью осуществления последующего более быстрого поиска. Например, трудно пользоваться словарями, если бы слова в них не были бы упорядочены по алфавиту.

Важность сортировки основана на том факте, что на ее примере можно показать многие основные фундаментальные приемы и методы построения алгоритмов. Сортировка является хорошим примером огромного разнообразия алгоритмов, которые выполняют одну и ту же задачу. Кроме того, многие из них имеют определенные преимущества друг перед другом. За счет усложнения алгоритма можно добиться существенного увеличения эффективности и быстродействия алгоритма по сравнению с более простыми методами. Как правило, термин сортировка понимают, как процесс перестановки объектов некоторого множества в определенном порядке. Цель сортировки - облегчить последующий поиск элементов в отсортированном множестве.

Сортировка слиянием основана на декомпозиции - массив разбивается на две примерно равные части, и после их рекурсивной сортировки выполняется операция слияния. Операция слияния двух отсортированных частей в одну состоит в том, что из каждой части выбирается по одному элементу, и меньший из них помещается в результирующий массив. Так продолжается до тех пор, пока не будет исчерпана одна из частей - в этом случае оставшаяся часть просто переносится в конец результирующего массива.

1. Постановка задачи

Необходимо реализовать и продемонстрировать работу алгоритма сортировки слиянием. Программа должна реализовывать алгоритм сортировки, выполнять считывание входных данных из файла и запись результатов в файл и отображать все данные в графическом интерфейсе.

Программа должна обрабатывать данные, предоставленные в корректно сформированных входных файлах.

Конечная программа не должна использовать реализацию алгоритмов сортировки, предоставляемые стандартными библиотеками, включая системные.

Решение задачи сортировки слиянием разделяется на три этапа:

- Сортируемый массив разбивается на две части примерно одинакового размера;
- Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом;
- Два упорядоченных массива половинного размера соединяются в один.

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 слиянием обычно реализуется таким образом, что она осуществляет последовательный доступ к данным. Поэтому данный метод сортировки применяется к структурам данных, доступ к элементам которых может осуществляться только последовательно, например, к спискам, потокам и так далее.

По тем же причинам слияние часто используется в качестве основы для сортировки на специализированных и высокопроизводительных машинах, поскольку именно последовательный доступ в подобного рода системах обработки данных является самым быстрым.

2. Выбор решения

Для написания программы был выбран язык программирования C. Язык C — это универсальный язык программирования, для которого характерны экономичность выражения, современный поток управления и структуры данных, богатый набор операторов. Язык C не является ни языком "очень высокого уровня", ни "большим" языком, и не предназначается для некоторой специальной области применения, но отсутствие ограничений и общность языка делают его более удобным и эффективным для многих задач, чем языки, предположительно более мощные.

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом. Данная среда разработки удобна для написания, редактирования, отладки и сборки кода, а затем для развертывания приложения. Помимо редактирования и отладки кода, Visual Studio включает компиляторы, средства завершения кода, систему управления версиями, расширения и многие другие

функции для улучшения каждого этапа процесса разработки программного обеспечения.

Исходный массив будет динамическим для того, чтобы можно было можно было изменить его размер.

Сортировка будет осуществлена методом нисходящего слияния. Исходная последовательность рекурсивно разбивается на половины, пока не получим подпоследовательности по 1 элементу. Из полученных последовательностей формируем упорядоченные пары методом слияния, затем - упорядоченные четверки и так далее.

3. Описание программы

Для определения размера будущего массива объявляется переменная size целого типа:

```
int size = 0; // инициализируем размер массива нулем
```

Затем у пользователя запрашивается размер массива, размер считывается и записывается в переменную size:

```
printf("Введите размер массива больше одного: ");
scanf("%d", &size); //запись размера массива
```

Под исходный массив выделяется память. Если память не была выделена, то на экран выводится сообщение об ошибке:

```
int* a = (int*)malloc(size * sizeof(int));//выделение памяти под динамический массив
if (a == NULL) {//вывод ошибки в случае, если память не была выделена
	printf("Ошибка выделения памяти");
	return;
}
```

Массив заполняется случайными числами от 0 до 9999:

```
for (int i = 0; i < size; i++) {
	a[i] = rand() % 10000;
}
```

Функция void mergeSort(int *a, int l, int r) выполняет сортировку массива слиянием.

Сначала в тексте функции идет проверка равенства левой и правой границы последовательности, то есть проверяется сомкнулись они или нет. Если

условие выполнилось, то последовательность состоит из одного элемента и его дальнейшая сортировка не требуется. Управление возвращается вызывающей функции с помощью команды return:

```
void mergeSort(int* a, int l, int r) {
    if (l >= r)//проверяем условие равенства левой и правой границы последовательности
    return;//управление возвращается вызывающей функции
```

Далее, если предыдущее условие не выполнилось, определяется середина последовательности:

```
    int mid = l + (r - l) / 2;// определяем середину массива
```

После этого рекурсивно вызываются функции сортировки для первой и второй половины последовательности. Таким образом последовательность делится на более мелкие последовательности до тех пор, пока каждая не станет равна одному:

```
    //рекурсивно вызываем функцию сортировки для каждой половины
    mergeSort(a, l, mid);//вызов функции сортировки для первой половины последовательности
    mergeSort(a, mid + 1, r);//вызов функции сортировки для левой половины последовательности
```

В переменные целого типа i и j записываются начало первой и второй последовательностей, сортировка и слияние которых будет выполняться:

```
    int i = l;// начало первой половины
    int j = mid + 1; // начало второй половины
```

Затем выделяется память для дополнительного массива, в который будет записываться отсортированная последовательность. Если память выделить не удалось, на экран выводится ошибка и функция передает управление вызывающей функции:

```
    int* tmp = (int*)malloc((r - l + 1) * sizeof(int));// выделение памяти под дополнительный массив, в котором формируется
    //отсортированная последовательность
    if (tmp == NULL) {//вывод ошибки в случае, если память не была выделена
        printf("Ошибка выделения памяти");
        return;
```

```
}
```

Потом идет сортировка последовательностей и их слияние. В цикле сравниваются элементы двух последовательностей, если элемент первой является меньшим и при этом индекс ее элемента не вышел за границу или же во второй последовательности не осталось элементов, то есть индекс ее вышел за границу, то в дополнительный массив записывается элемент первой последовательности. В ином случае в дополнительный массив записывается элемент второй последовательности:

```
for (int s = 0; s < r - l + 1; s++)//сортировка последоваельности
{
// записываем в формируемую последовательность меньший из элементов двух путей
// или остаток первого пути если j > r
if ((j > r) || ((i <= mid) && (a[i] < a[j]))) //проверка условия выхода за правую границу второй последовательности
//или являеся ли элемент первой половины последовательности меньшим
{
tmp[s] = a[i];//записываем в дополнительный массив элемент первой половины последовательности
i++;//инкрементируем i и продвигаемся по первой половине последовательности
}
else//если условие не выполняется
{
tmp[s] = a[j];//записываем в формируемую последовательность элемент второй половины последовательности
j++;// инкрементируем j и продвигаемся по второй половине последовательности
}
}
```

Далее элементы сформированного дополнительного массива записываются в исходный массив. Память, выделенная под дополнительный массив, освобождается:

```
// переписываем сформированную последовательность в исходный массив
for (int s = 0; s < r - l + 1; s++)
```

```
        a[l + s] = tmp[s];
    free(tmp);
}
```

4. Схемы программы

4.1 Блок-схема программы

Рисунок 1 – Схема функции main()

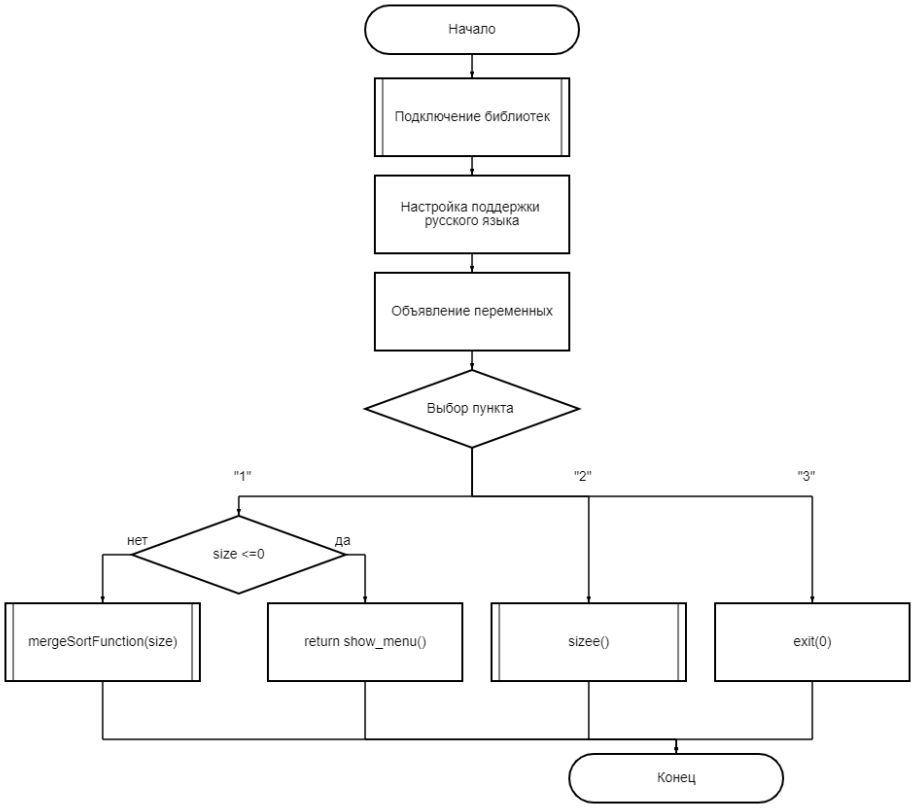


Рисунок 2 – Схема функции main()

4.2 Блок-схема алгоритма

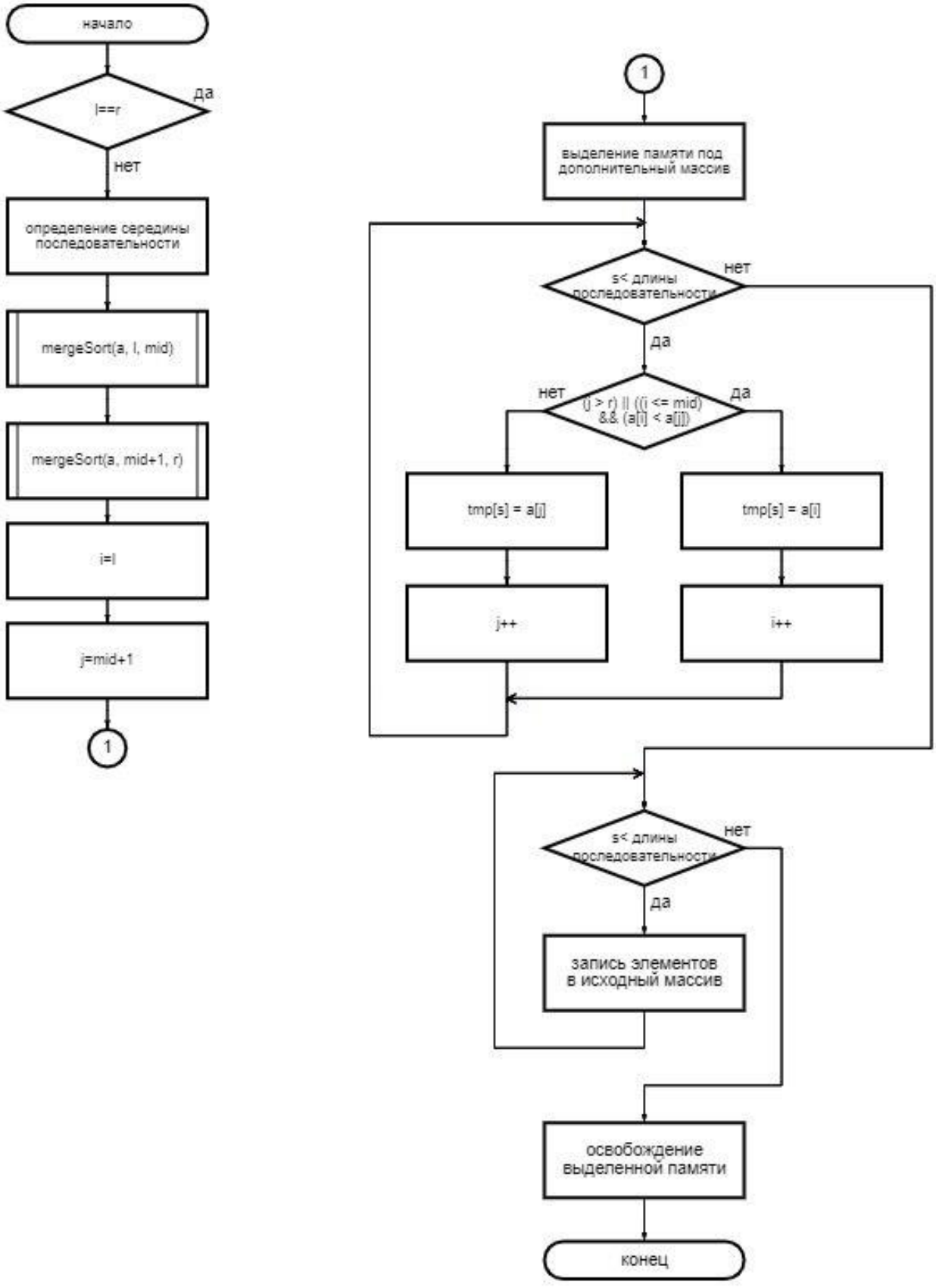


Рисунок 3 – Блок-схема алгоритма

5. Тестирование программы

Тестовый набор данных представлен в таблице 3. Результаты тестирования приведены в Приложении Б.

Таблица 3 – Тестовый набор данных

№ теста	Размер массива size	Время выполнения сортировки в секундах
1	10000	0.006
2	20000	0.014
3	30000	0.02
4	40000	0.023
5	50000	0.024
6	60000	0.027
7	70000	0.029
8	80000	0.032
9	90000	0.037
10	100000	0.04
11	110000	0.044

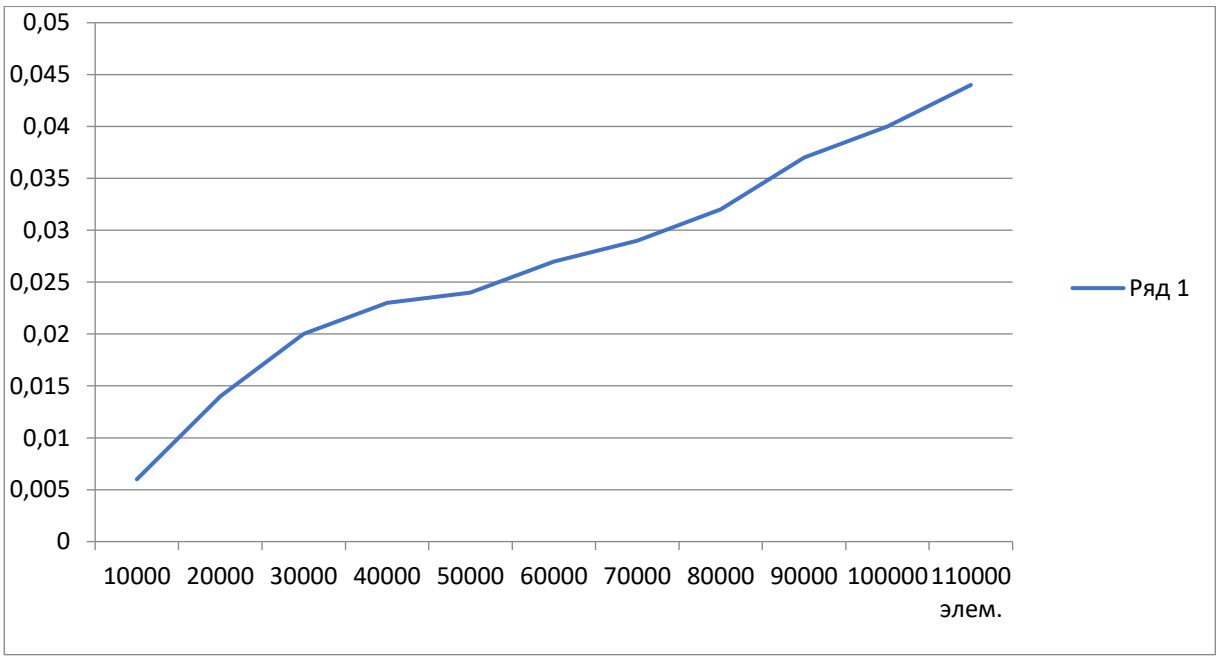


Рисунок 4 – Результаты тестирования

6. Отладка

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовалось несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных.

Для отладки нужно запустить приложение с отладчиком, подключенным к процессу приложения. Это можно сделать с помощью клавиши F5 или кнопки Начать отладку на панели инструментов отладки.

Точка останова указывает, где следует приостановить выполнение кода, чтобы проверить значения переменных или поведение памяти либо выполнение ветви кода. Точку останова можно задать, щелкнув в поле слева от строки кода.

С помощью клавиши F11 можно выполнить шаг с заходом при выполнении отладки. При запуске приложения с помощью клавиши F11 отладчик останавливается на первом выполняемом операторе, чтобы мы могли проверить содержимое переменных. Если оператор содержит вызов функции, то при шаге с заходом программа приостанавливает работу на первой строчке этой функции.

Клавиша F10 выполняет шаг без захода и позволяет выполнять отладку без захода в функции в коде.

Проверка содержимого переменных при проведении отладки осуществлялась с помощью окна “Видимые”, в котором отображаются переменные, используемые вокруг текущей точки останова, и окна «Локальные», где показываются переменные, которые находятся в текущей области видимости.

7. Совместная работа

7.1 Работа с таск-трекером

Для удобства совместной разработки был использован сервис Yougile. Определили задачи проекта, разделили роли.

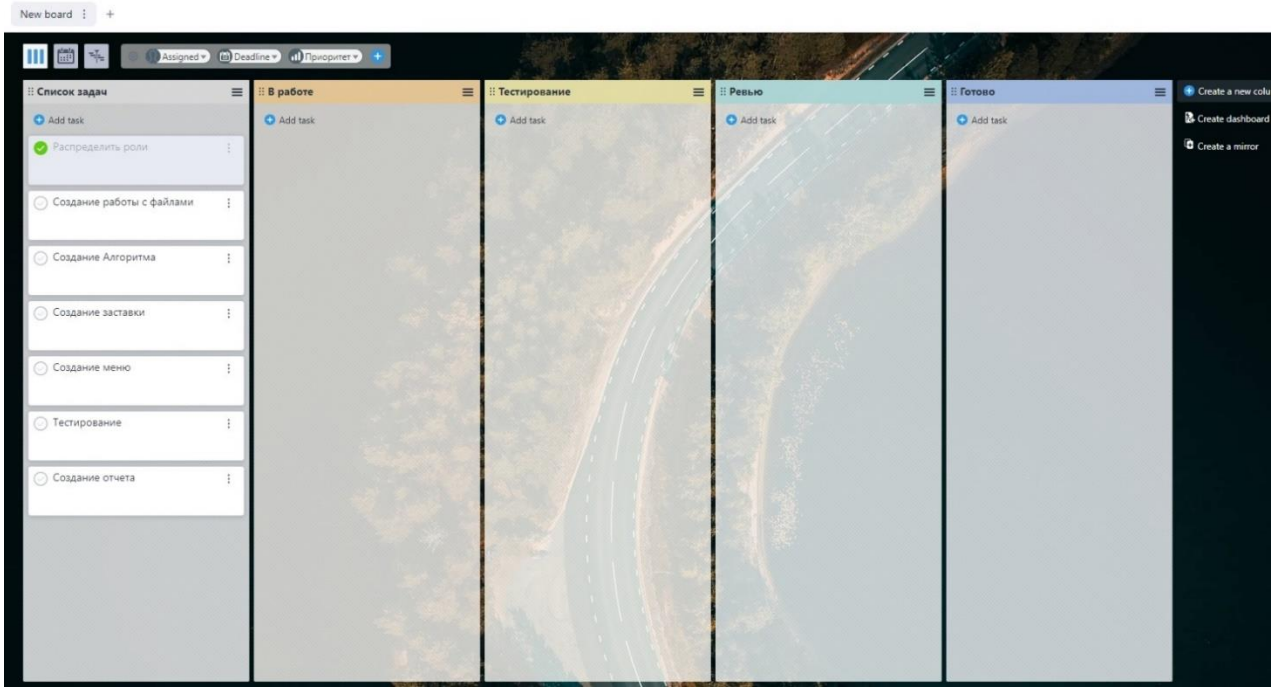


Рисунок 5 – Определение задач

Корректировали статус задач по мере выполнения.

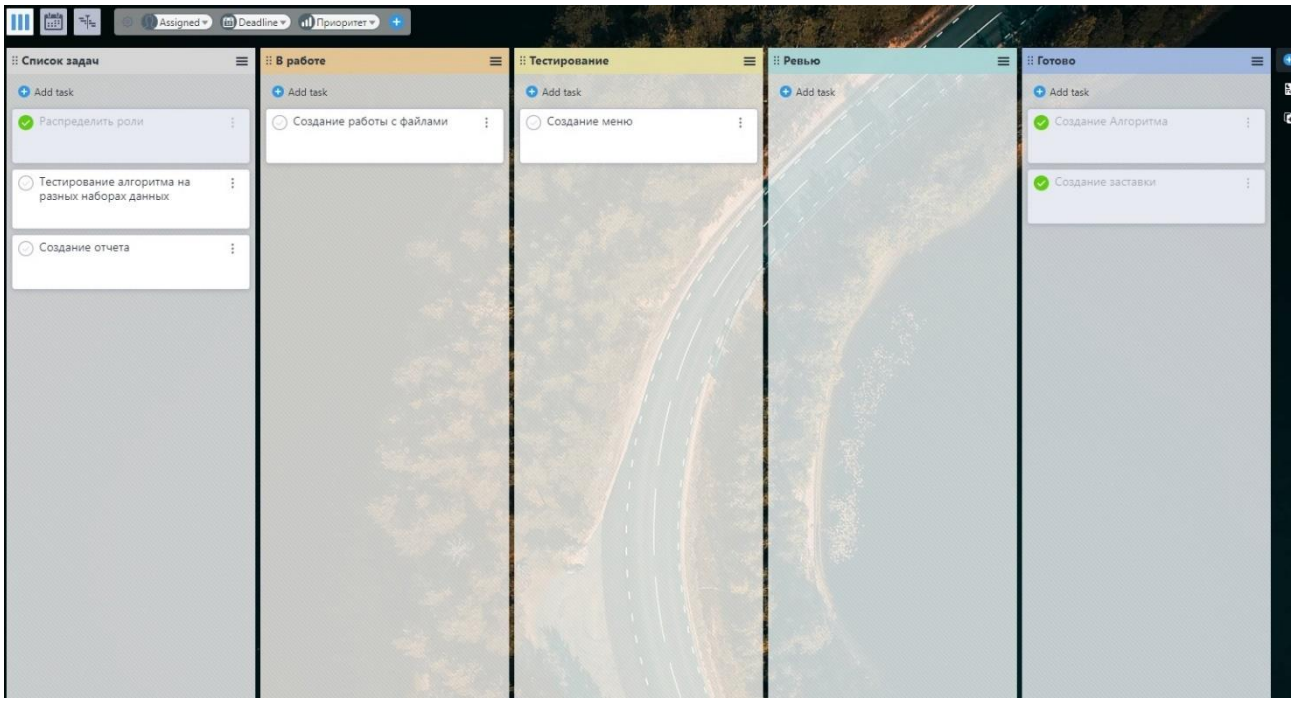


Рисунок 6 – Работа с задачами

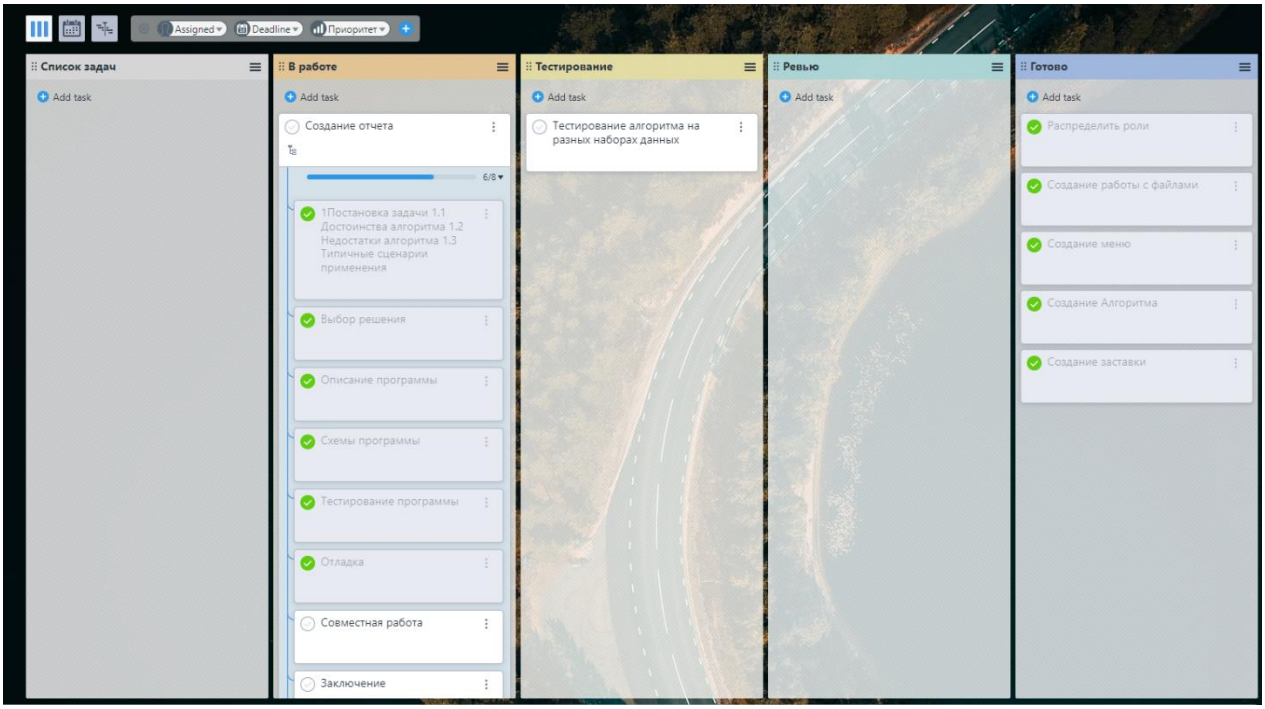


Рисунок 7 – Выполненные задачи

7.2 Работа с системой контроля версий

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

Я склонировала в свой локальный репозиторий удаленный репозиторий с файлами проекта и добавила в программу алгоритм, осуществляющий сортировку массива, создание самого исходного массива и запрос его размера. Это было зафиксировано и загружено на удаленный репозиторий GitHub, на ветку main.

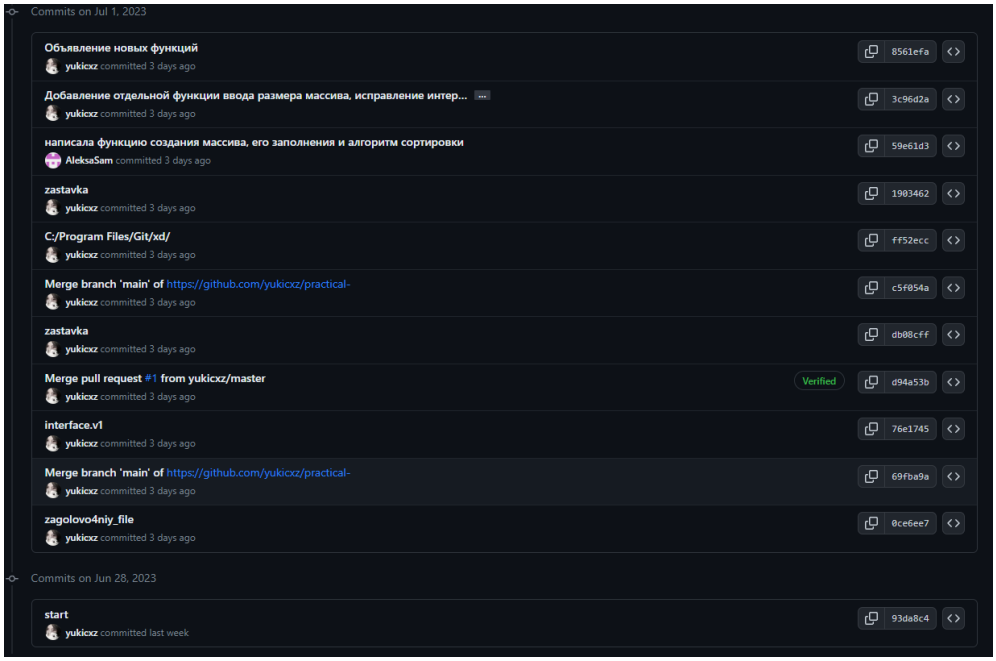


Рисунок 8 – Создание первых версий интерфейса

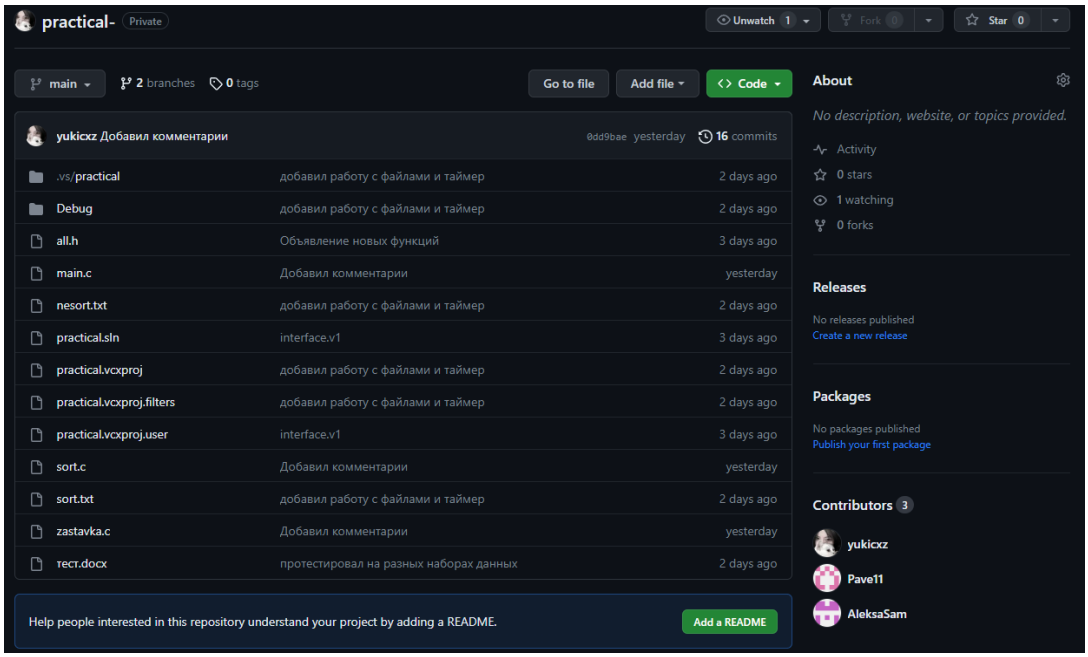


Рисунок 9 – Ветка main

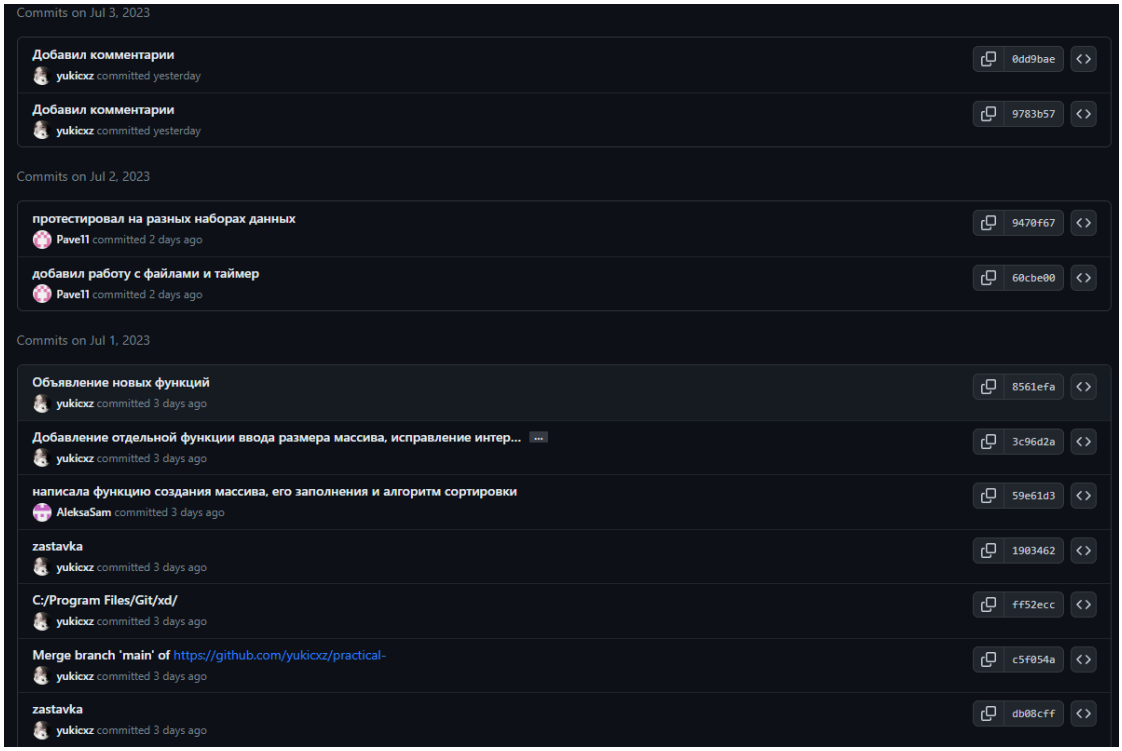


Рисунок 10 – Созданные коммиты

Для загрузки данных на локальный репозиторий, а также отправки данных на удаленный репозиторий были использованы основные команды Git Bash: git push, git clone, git pull и т.д.

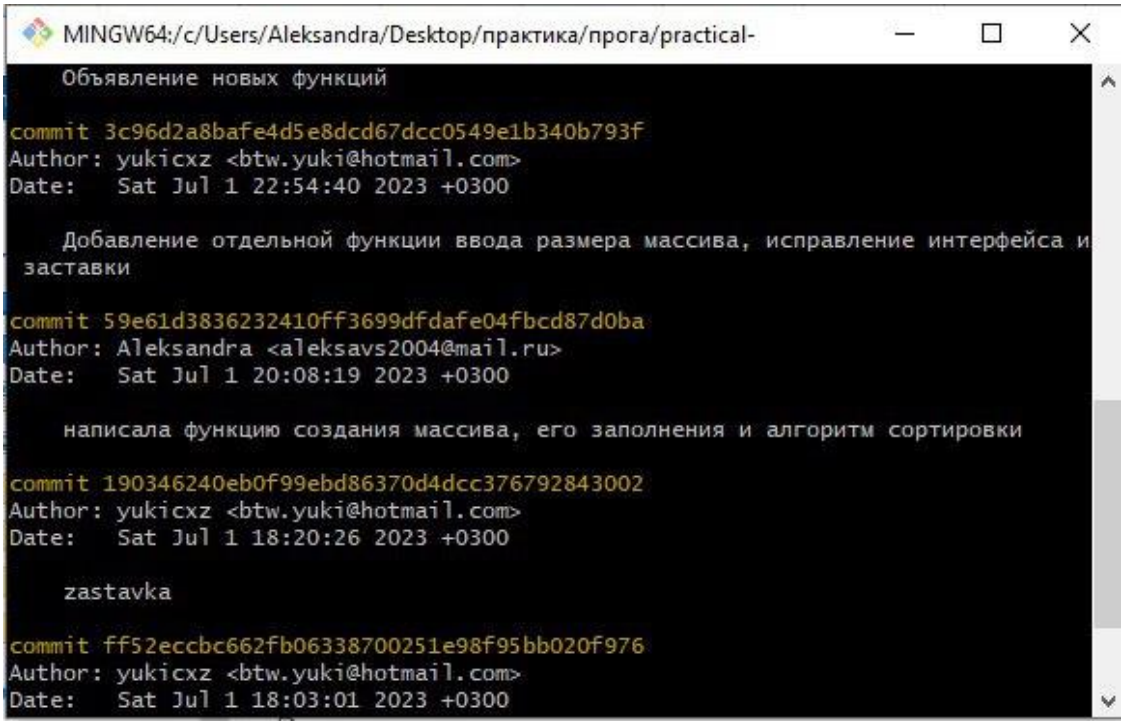


Рисунок 11 – Использование Git Bash

Ссылка на удаленный репозиторий: https://github.com/yukicxz/practical-

Заключение

В ходе выполнения данной практической работы были улучшены базовые знания программирования на языке С. Улучшены навыки отладки, тестирования программ и работы со сложными типами данных.

Данная практическая работа была выполнена в соответствии поставленной задачи, в среде Microsoft Visual Studio 2022. Было проведено исследование компонентов программной среды Microsoft Visual Studio 2022, которые использовались при создании программы, также использовалось множество функций.

В дальнейшем программу можно улучшить путем добавления восходящей сортировки слиянием, а также добавлением графического интерфейса.

Список используемых источников

1. Фундаментальные алгоритмы на С: Части 1-5. Анализ/Структуры данных/Сортировка/Поиск/ Алгоритмы на графах: Пер. с англ./ Роберт Седжвик. – СПб: ООО «ДиаСофтЮП», 2003. – 1136 с.

2. Алгоритмы. Просто как дважды два / И. В. Красиков, И. Е. Красикова. - М. : Эксмо, 2007. - 256 с. - (Просто как два).

3.Томас Габриэль. Программирование на С. 2016г

4. А.А. Тюгашев. Языки программирования. Учебное пособие. 2018 г.

5. Сортировка слиянием [Электронный ресурс] – URL: https://ru.wikipedia.org (дата обращения: 03.06.2023 г.)

Приложение А
Листинги программы

Файл main.c

```
#define _CRT_SECURE_NO_WARNINGS
#include "all.h"
int main()
{
    zastavka(); //запуск функции заставки
    show_menu(); //запуск функции меню
    return 0;
}
```

Файл sort.c

```
#define _CRT_SECURE_NO_WARNINGS
#include "all.h"

int size = 0; // инициализируем размер массива нулем
int sizee() {
    char vibor = ' ';
    system("cls"); //очистка консоли
    printf("\033[93m""Текущая размерность массива: %d""\033[m",
size);
    printf("\033[33m""\nХотите поменять? (Y/N): ""\033[m");
    getchar(); //задержка
    scanf("%c", &vibor); //считывание в переменную
    vibor = toupper(vibor); //определение заглавных и строчных
букв
    if (vibor == 'Y') //если выбор да
    {
        printf("Введите размер массива больше одного: ");
        scanf("%d", &size); //запись размера массива
        printf("Размер массива успешно введен\n");
        return show_menu(); //возврат в меню
    }
    else if (vibor == 'N') //если выбор нет
    {
        system("cls"); //очистка консоли
        return show_menu(); //возврат в меню
    }
}

void mergeSortFunction(int size) {
    setlocale(LC_ALL, "Russian");
    int* a = (int*)malloc(size * sizeof(int));//выделение памяти
под динамический массив
```

```
    FILE* nesort;
    FILE* sort;
    if (a == NULL) {//вывод ошибки в случае, если память не была
выделена
        printf("Ошибка выделения памяти");
        return;
    }
    nesort = fopen("nesort.txt", "w");//открываем файл для
несортированного массива
    // Заполняем элементы массива случайными числами и записываем
в файл
    for (int i = 0; i < size; i++) {
        a[i] = rand() % 10000;
        fprintf(nesort, " %d", a[i]);
    }
    fclose(nesort);//закрываем файл
    time_t start = clock();//время до сортировки
    mergeSort(a, 0, size - 1);// вызываем функцию сортировки
    time_t stop = clock();//время после сортировки
    double time = (stop - start) / 1000.0;//время сортировки
    sort = fopen("sort.txt", "w");//открываем файл для
сортированного массива
    // Записываем отсортированный массив в файл
    for (int i = 0; i < size; i++)
        fprintf( sort, " %d", a[i]);
    free(a);
    fclose(sort);//закрываем файл
    printf("Массив успешно отсортирован\nВремя сортировки:%5.3lf
секунд", time);
    show_menu();
}
void mergeSort(int* a, int l, int r) {
    if (l >= r)//проверяем условие равенства левой и правой
границы последовательности
        return;//управление возвращается вызывающей функции
    int mid = l + (r - l) / 2;// определяем середину массива
    //рекурсивно вызываем функцию сортировки для каждой половины
    mergeSort(a, l, mid);//вызов функции сортировки для первой
половины последовательности
    mergeSort(a, mid + 1, r);//вызов функции сортировки для левой
половины последовательности
    int i = l;// начало первой половины
    int j = mid + 1; // начало второй половины
    int* tmp = (int*)malloc((r - l + 1) * sizeof(int));//
выделение памяти под дополнительный массив, в котором формируется
    //отсортированная последовательность
```

```
    if (tmp == NULL) {//вывод ошибки в случае, если память не была
выделена
        printf("Ошибка выделения памяти");
        return;
    }

    for (int s = 0; s < r - l + 1; s++)//сортировка
последоваельности
    {
        // записываем в формируемую последовательность меньший из
элементов двух путей
        // или остаток первого пути если j > r
        if ((j > r) || ((i <= mid) && (a[i] < a[j]))) //проверка
условия выхода за правую границу второй последовательности
            //или являеся ли элемент первой половины
последовательности меньшим
        {
            tmp[s] = a[i];//записываем в дополнительный массив
элемент первой половины последовательности
            i++;//инкрементируем i и продвигаемся по первой
половине последовательности
        }
        else//если условие не выполняется
        {
            tmp[s] = a[j];//записываем в формируемую
последовательность элемент второй половины последовательности
            j++;// инкрементируем j и продвигаемся по второй
половине последовательности
        }
    }
    // переписываем сформированную последовательность в исходный
массив
    for (int s = 0; s < r - l + 1; s++)
        a[l + s] = tmp[s];

    free(tmp);
}
void centerprintf(const char* str) { //функция определения центра
консоли
    int width = 120; //высота
    int len = strlen(str); //длинна
    int startPos = (width - len) / 2; //определение центра
    for (int i = 0; i < startPos; i++) { //смещение до центра
        putchar(' ');
    }
    printf("%s", str);
}
```

```
int show_menu() { //объявление меню
    setlocale(LC_ALL, "Rus"); // установка русского языка
    char* frm = "%s"; //символьный формат для ассемблерной вставки
    char* text = ("\033[31m""Данного варианта нет. Выберите из
предложенных.\n""\033[m"); //текст ассемблерной вставки
    printf("\n");
    centerprintf("============================\n");
    centerprintf("       =""\033[91m""    Сортировка""\033[m""
Слиянием    =\n");
    centerprintf("============================\n");
    centerprintf("|                           |\n");
    centerprintf("|     1. Сортировка         |\n");
    centerprintf("|                           |\n");
    centerprintf("         |""\033[91m""     2. Размер Массива
""\033[m""|\n");
    centerprintf("|                           |\n");
    centerprintf("        |""\033[31m""     3. Выход
""\033[m""|\n");
    centerprintf("|                           |\n");
    centerprintf("============================\n");


    char choice; //переменная для выбора

    printf("Ваш выбор: ");
    scanf(" %c", &choice); // запись выбора в переменную
    printf("\n");

    switch (choice) { //переключатель для выбора пользователя
    case '1': //если пользователь ввел 1
        system("cls"); //очистка консололи
        if (size <= 0) { //если размер массива не задан
            printf("Размер массива не был введен\n");
            return show_menu(); //возврат в меню
        }
        else {
            mergeSortFunction(size); //функция сортировки
            return show_menu(); //возврат в меню
        }
        break;
    case '2': //если пользователь ввел 2
        system("cls"); //очистка консололи
        sizee(); //функция ввода размера массива
        break;
    case '3': //если пользователь ввел 3
        exit(0); //выход
        break;
```

```
    default: //если не выбран ни один из предложенных вариантов
        system("cls"); //очистка консслоли
        __asm { //начало ассемблерной вставки
            push text //заносим в стек указатель на переменную
текст
            push frm //заносим в стек указатель на формат
            call printf //вызов функции printf
            add esp, 8 //восстановление стека
        }
        return show_menu(); //возврат в меню
    }
}
```

Файл zastavka.c

```
#define _CRT_SECURE_NO_WARNINGS
#include "all.h"
int zastavka()
{
    system("cls"); //очистка консоли
    setlocale(LC_ALL, "Rus"); //установка русского языка
    char title[] = "  Сортировка слиянием\n"; //
    char title2[] = "Выполнили Корнилов В.М Самофалова А.В. Перкин
П.О\n"; //объявление текстовых массивов
    char title3[] = "     Группа 22ВВП2 "; //
    int length = strlen(title); //
    int length2 = strlen(title2); //Определение длинны массивов
    int length3 = strlen(title3); //
    int x, y;

    CONSOLE_SCREEN_BUFFER_INFO csbi; //
    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE),
&csbi); //
    int console_width = csbi.srWindow.Right - csbi.srWindow.Left +
1; // Установка размеров консоли
    int console_height = csbi.srWindow.Bottom - csbi.srWindow.Top
+ 1; //

    x = (console_width - length) / 2; // Определение центра
консоли
    y = console_height / 2; //


    for (int i = 0; i < y; i++) //добавление отступов до центра по
У
        printf("\n"); //
```

```
    for (int i = 0; i < x; i++) //добавление отступов до центра по
x
        printf(" "); //
    for (int j = 0; j < length; j++)
    {
        printf("\033[91m""%c""\033[m", title[j]); //посимвольный
вывод 1 массива
        Sleep(60); //промежуток вывода каждого символа
    }
    Sleep(500);

    for (int i = 0; i < 35; i++) //добавление отступов по x
        printf(" ");
    for (int m = 0; m < length2; m++)
    {
        printf("%c", title2[m]);//посимвольный вывод 2 массива
        Sleep(60); //промежуток вывода каждого символа
    }
    Sleep(500);
    for (int l = 0; l < y; l++) //добавление отступов по y
        printf("\n");
    for (int l = 0; l < x; l++) //добавление отступов по x
        printf(" ");
    for (int n = 0; n < length3; n++)
    {
        printf("%c", title3[n]); //посимвольный вывод 3 массива
        Sleep(60); //промежуток вывода каждого символа
    }
    Sleep(2000);
    system("cls"); //очистка консоли
    return 0;
}
```

Файл all.h

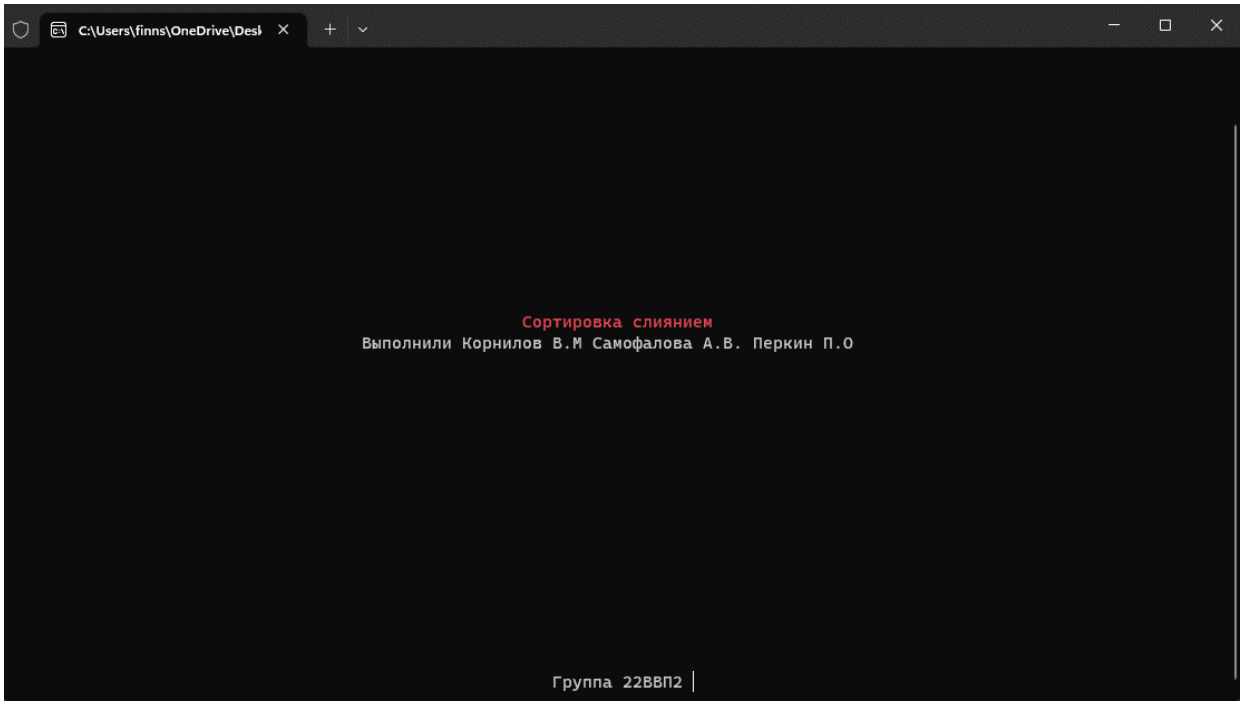
```
#ifndef all_h
#define all_h
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <locale.h>
#include <string.h>
#include <windows.h>

int zastavka();
int show_menu();
int sizee();
```

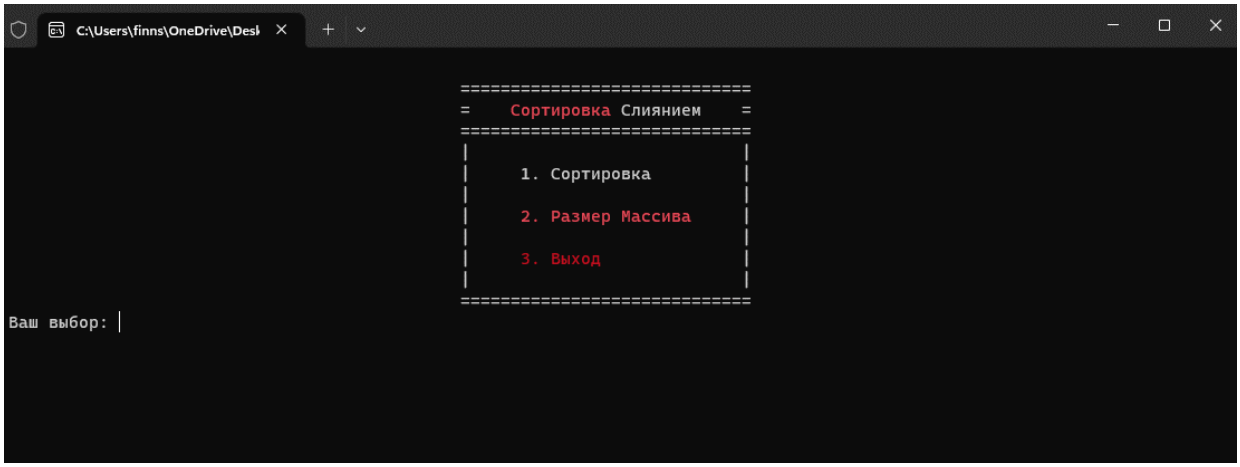
```
#endif // !all_h
```

Приложение В

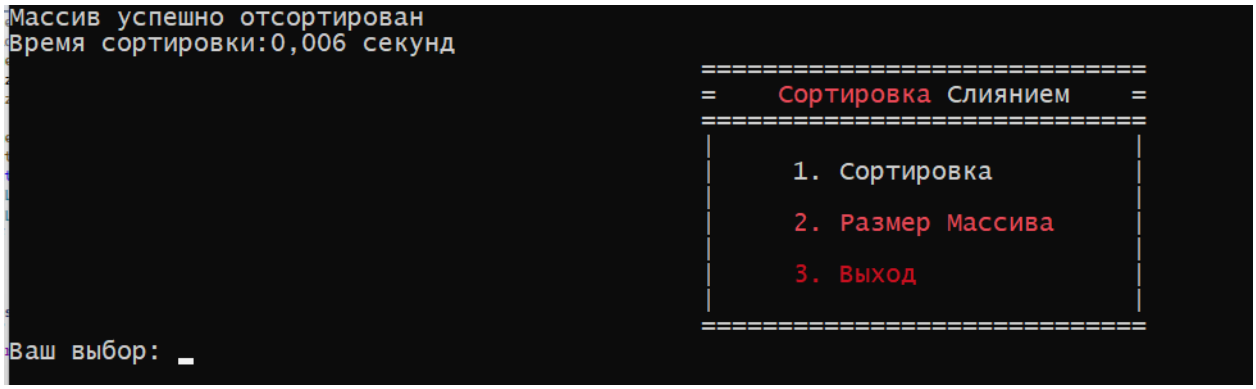
Заставка

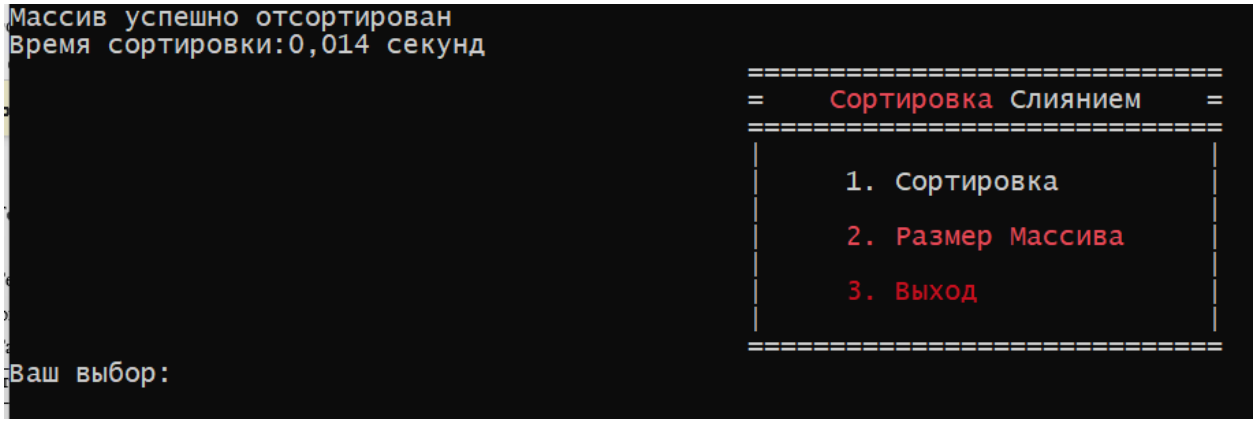


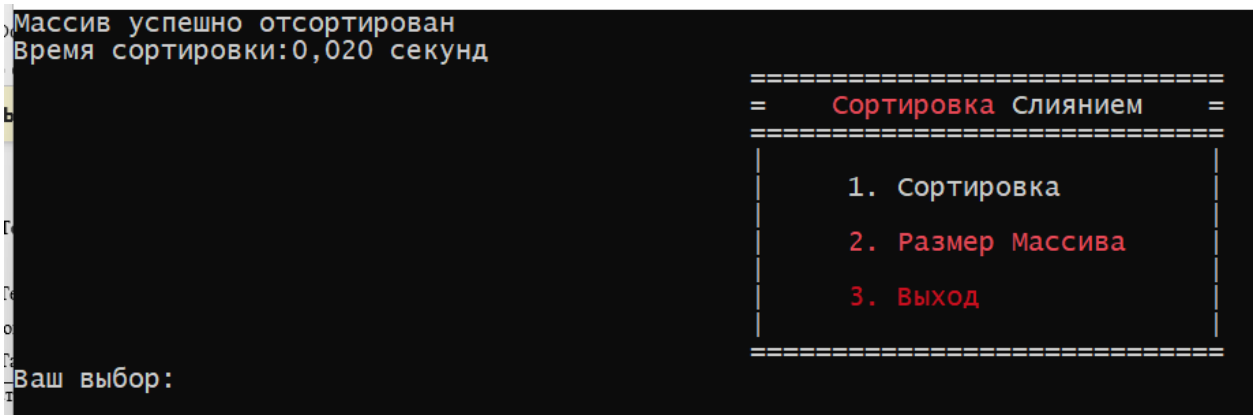
Главное меню

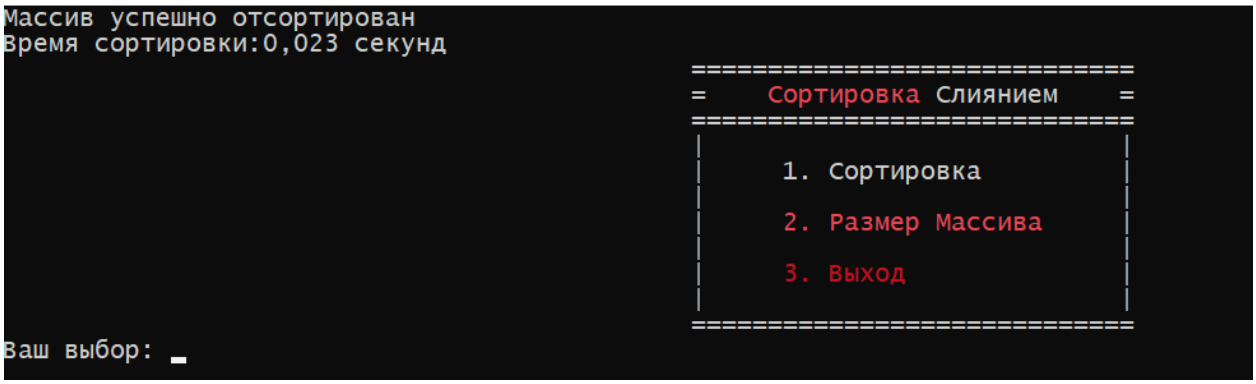


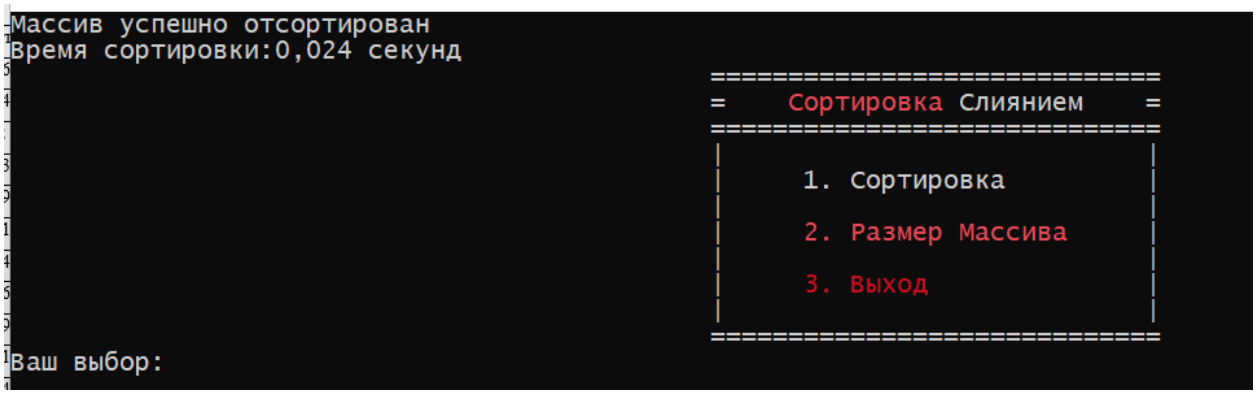
Тестирование

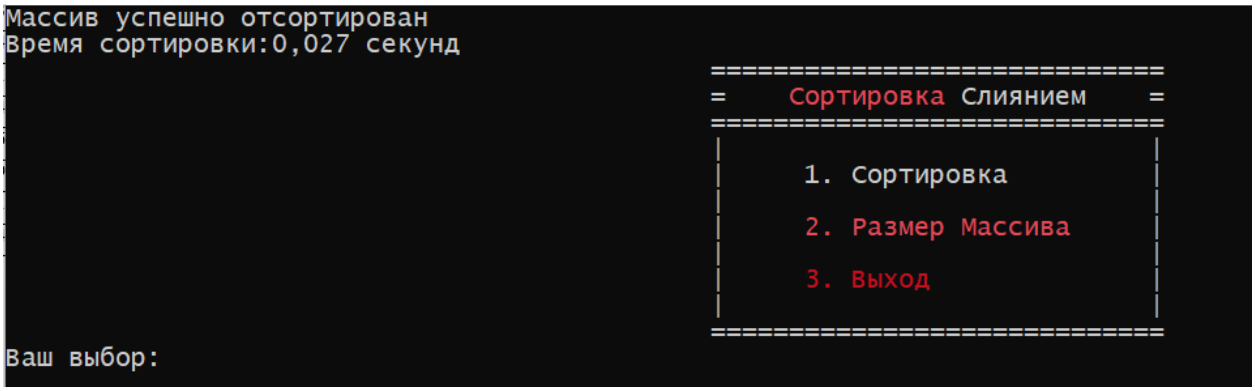


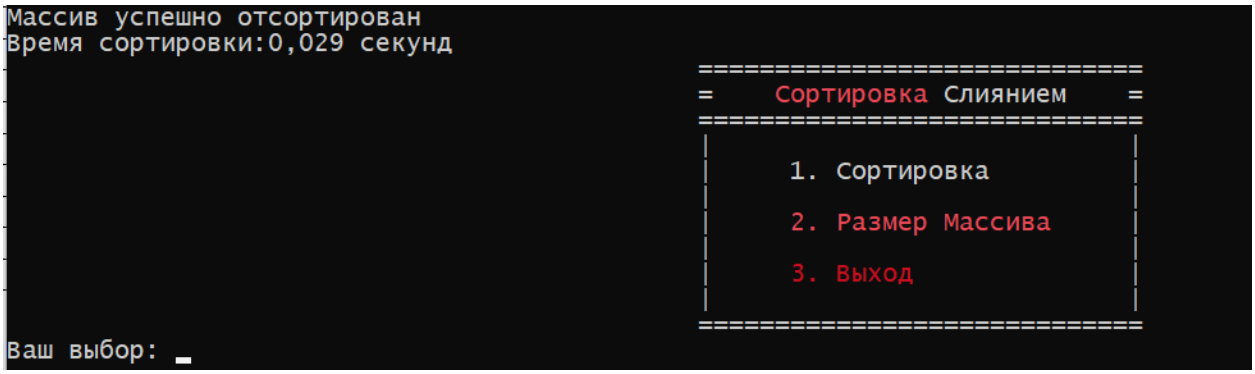

Массив успешно отсортирован
Время сортировки:0,014 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:

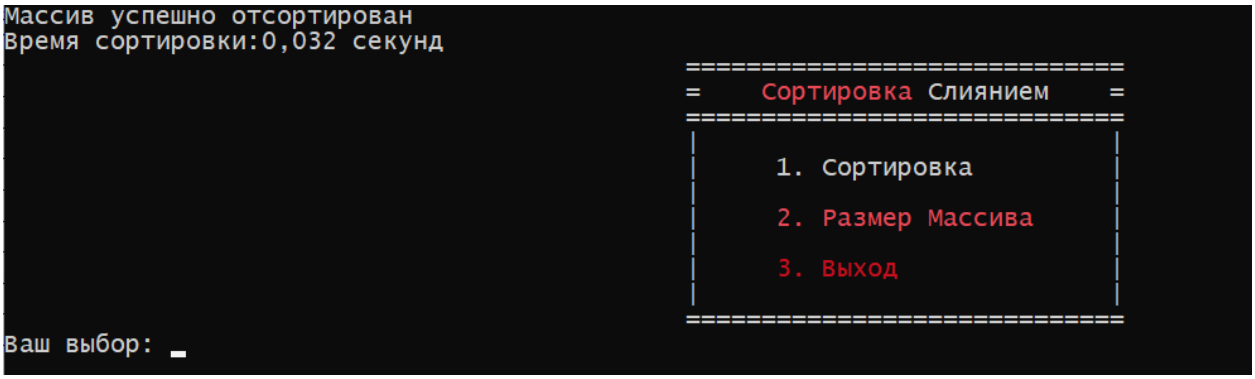

Массив успешно отсортирован
Время сортировки:0,020 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:

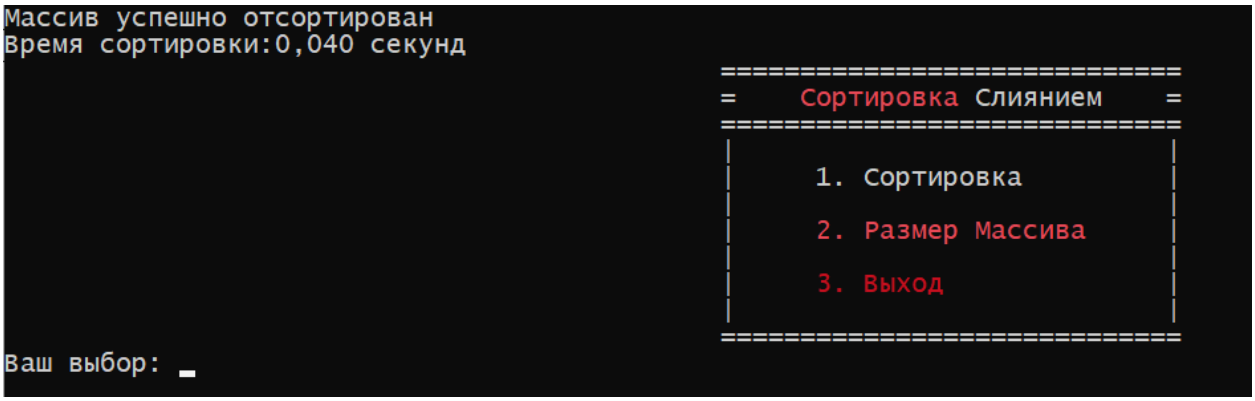

Массив успешно отсортирован
Время сортировки:0,023 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:

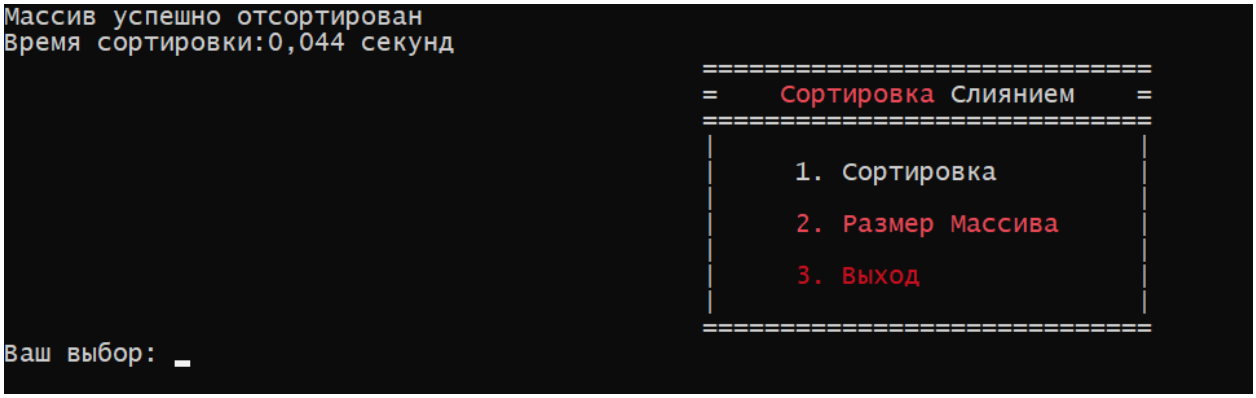

Массив успешно отсортирован
Время сортировки:0,024 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:


Массив успешно отсортирован
Время сортировки:0,027 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:


Массив успешно отсортирован
Время сортировки:0,029 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:


Массив успешно отсортирован
Время сортировки:0,032 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:


Массив успешно отсортирован
Время сортировки:0,040 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор:


Массив успешно отсортирован
Время сортировки:0,044 секунд
Сортировка Слиянием
1. Сортировка
2. Размер Массива
3. Выход
Ваш выбор: