

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)

\_\_\_\_\_ Ганин Иван Романович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

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)

\_\_\_\_\_ Ганин Иван Романович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы  
и сети»

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	
	<b>Общий объём часов</b>	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)

Ганин Иван Романович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

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»

Ганин И.Р. выполнял практическое задание «Сортировка пузырьком». На первоначальном этапе были изучен и проанализирован алгоритм пузырьковой сортировки, был выбран метод решения и язык программирования C, на котором была написана программа сортировки массива методом пузырька. Протестировал и отладил программу. Оформил отчёт.

Бакалавр      Ганин И.Р.      "\_\_\_" \_\_\_\_\_ 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)

Ганин Иван Романович

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

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 Ганин И.Р. решал следующие задачи: создание блок-схем, тестирование программы.

За период выполнения практики были освоены основные понятия и технологии пузырьковой сортировки, реализован метод работы с файлами. Во время выполнения работы Ганин И.Р. показал 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 Ганин И.Р. заслуживает оценки «                    ».

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

## Содержание

Введение .....	7
1 Постановка задачи .....	9
1.1 Требования к программе .....	9
1.2 Достоинства и недостатки алгоритма .....	9
1.3 Типичные сценарии использования .....	9
2 Выбор решения .....	10
3. Схема программы. ....	11
4 Тестирование программы .....	16
4.1 Тестирование интерфейса .....	16
4.2 Тестирование на разных наборах данных .....	17
4.3 Анализ полученных результатов тестирования .....	18
5. Совместная разработка .....	20
Список использованных источников .....	22
Приложение А.....	23
Приложение Б. Листинг .....	26

## **Введение**

В современном мире информационные технологии стали неотъемлемой частью жизни практически любого человека.

Снижение трудоёмкости любых производственных, ускорение социальных процессов и, как следствие, повышение их эффективности сейчас связано с существенно изменившимися за последние десятилетия возможностями сбора, обработки, хранения, передачи и представления информации.

Реализация полученной в результате этих процессов информации совершенно нового качества позволяет обеспечить процесс создания и использования материальных и духовных ценностей на более высоком и принципиально новом уровне.

Таким образом изучение современных информационных технологий обеспечивает возможность человеку без проблем ориентироваться в современных тенденциях и путях развития общества, процессах производства и повышать качество социальной жизни.

Изучение всегда начинается с освоения базовых элементов, таких как методы сортировки, хранения и передачи информации. За период выполнения практики мною были освоены основные понятия и технологии пузырьковой сортировки, реализован метод работы с файлами.

В качестве инструмента для реализации был выбран Microsoft Visual Studio и язык программирования C.

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом. Она является стартовой площадкой для написания, отладки и сборки кода, а также последующей публикации. Помимо стандартных редактора и отладчика, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие элементы и функции, позволяющие упростить процесс разработки, что делает среду Visual Studio одной из самых простых и понятных в освоении и использовании.

Язык С является универсальным языком программирования, который сам по себе не связан с какой-либо операционной системой и с успехом использовался и используется для написания больших вычислительных программ и программ обработки текстов и баз данных. Его также называют языком системного программирования, так как он удобен при написании операционных систем. С является языком относительно низкого уровня, поскольку имеет дело с объектами того же вида, что и большинство ЭВМ, а именно с символами, числами и адресами. Они могут объединяться и пересылаться посредством обычных арифметических и логических операций, осуществляемых реальными ЭВМ. Поэтому язык С крайне удобен для разработки различных приложений и реализации алгоритмов, которые, например, могут использоваться для сортировки данных.



## **1 Постановка задачи**

### **1.1 Требования к программе**

Для выполнения предложенного задания учебной практики необходимы навыки, полученные в ходе изучения курса «Программирование», а также базовые знания по алгоритмизации. Для реализации программы допустимо использовать любые доступные языки программирования, за исключением случаев использования готовых или стандартных библиотек, содержащих в себе готовую реализацию алгоритма сортировки.

### **1.2 Достоинства и недостатки алгоритма**

Достоинства:

Метод сортировки «пузырьком» является наиболее простым и понятным в реализации. Он легко реализуется на всех языках программирования и подходит для начинающих. «Пузырёк» также лежит в основе других более сложных методов сортировки.

Недостатки:

Алгоритм считается учебным и практически не применяется вне учебной литературы, а также не эффективен для больших массивов. Такая ситуация наблюдается из-за большого количества проходов и сравнений на каждом из проходов алгоритма, на что тратится большое количество времени. Время выполнения алгоритма прямо пропорционально квадрату количества элементов ( $N^2$ ), что делает сортировку «пузырьком» самым медленным методом сортировки.

### **1.3 Типичные сценарии использования**

Данный метод сортировки актуален для использования в учебных целях (для реализации начинающим программистом), а также для небольшого массива входных данных. Алгоритм можно применять для сортировки с целью ускорения поиска каких-либо элементов и для упорядочивания элементов.

## **2 Выбор решения**

В качестве среды разработки программы и реализации алгоритма была выбрана программа Microsoft Visual Studio 2022, а в качестве языка программирования был выбран C. C является языком относительно низкого уровня, поскольку имеет дело с объектами того же вида, что и большинство ЭВМ, а именно с символами, числами и адресами. Они могут объединяться и пересылаться посредством обычных арифметических и логических операций, осуществляемых реальными ЭВМ. Поэтому язык C крайне удобен для разработки различных приложений и реализации алгоритмов, которые, например, могут использоваться для сортировки данных.

Для отладки программы использовались несколько возможностей данной среды, а именно трассировка, точка останова, просмотр значений переменных.

Все действия, совершаемые программой, происходят в одном файле.

С помощью пользовательского интерфейса пользователь может выбрать действие, которое он хочет осуществить.

### 3. Схема программы.

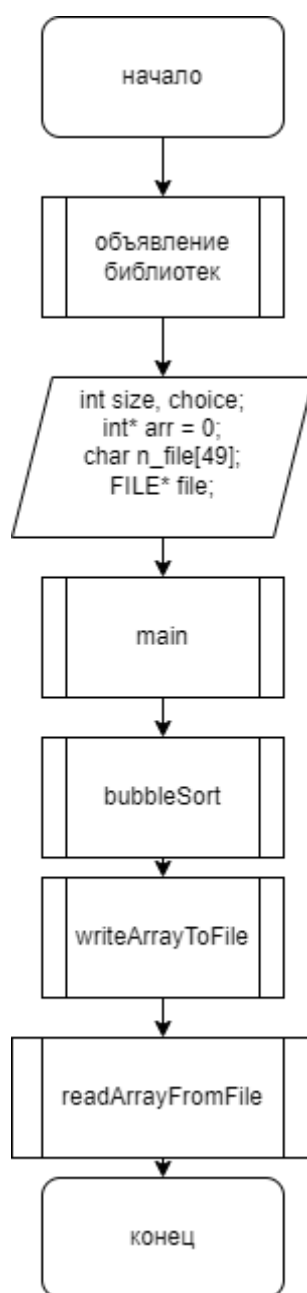


Рисунок 1 – Схема программы

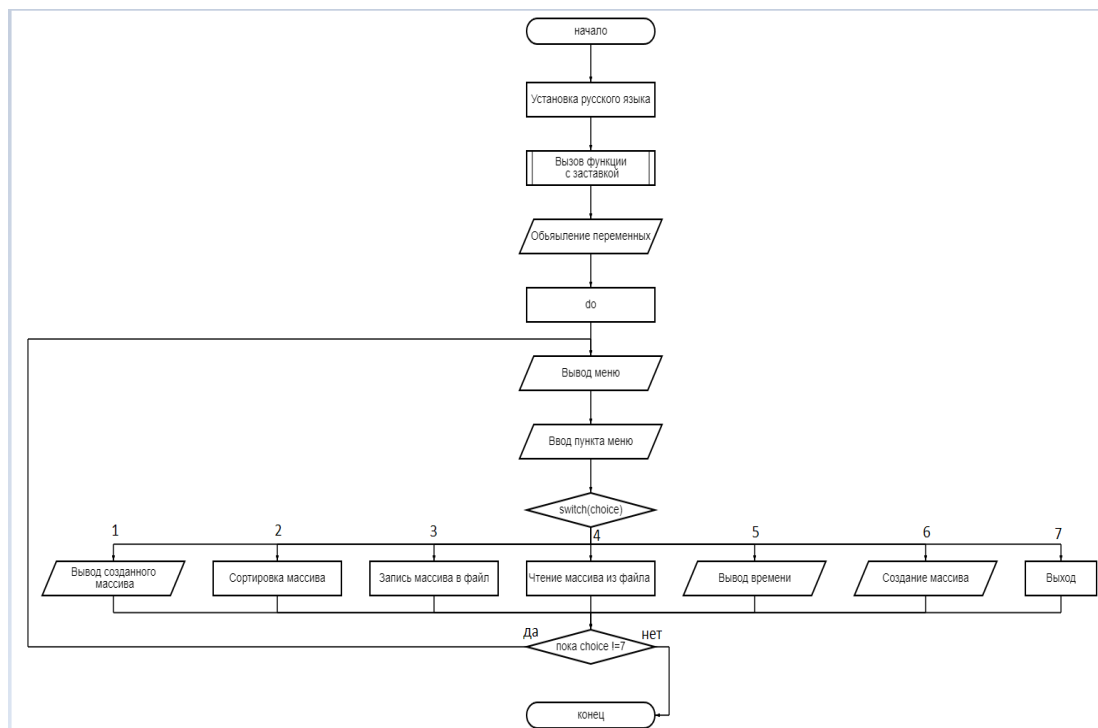


Рисунок 2 – Функция main

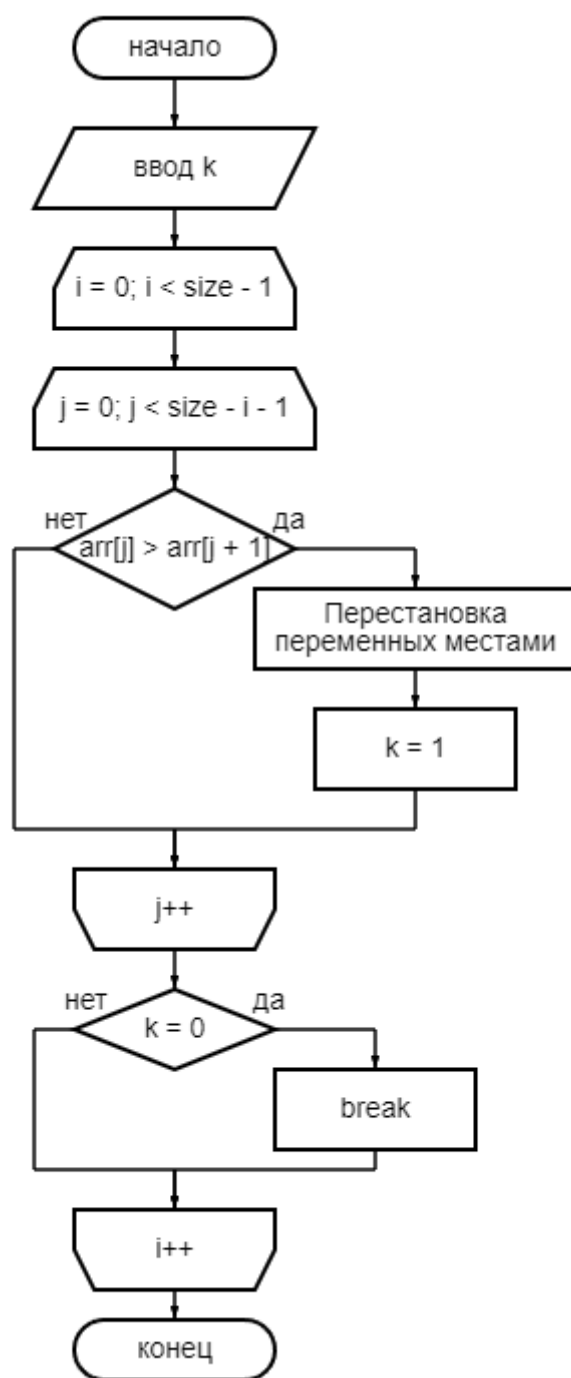


Рисунок 3 – Функция bubbleSort

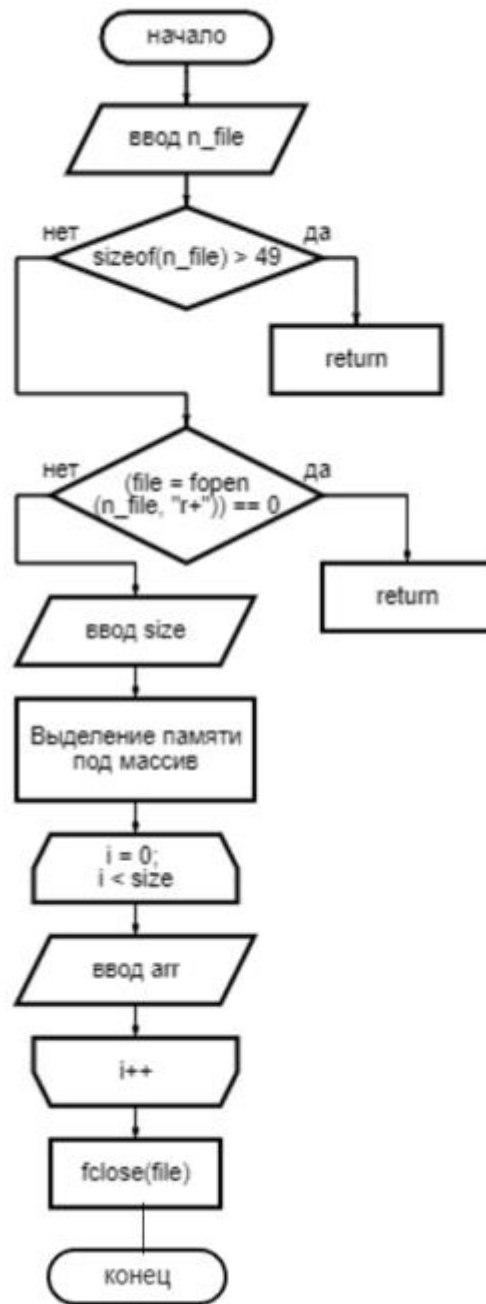


Рисунок 4 – Функция readArrayFromFile

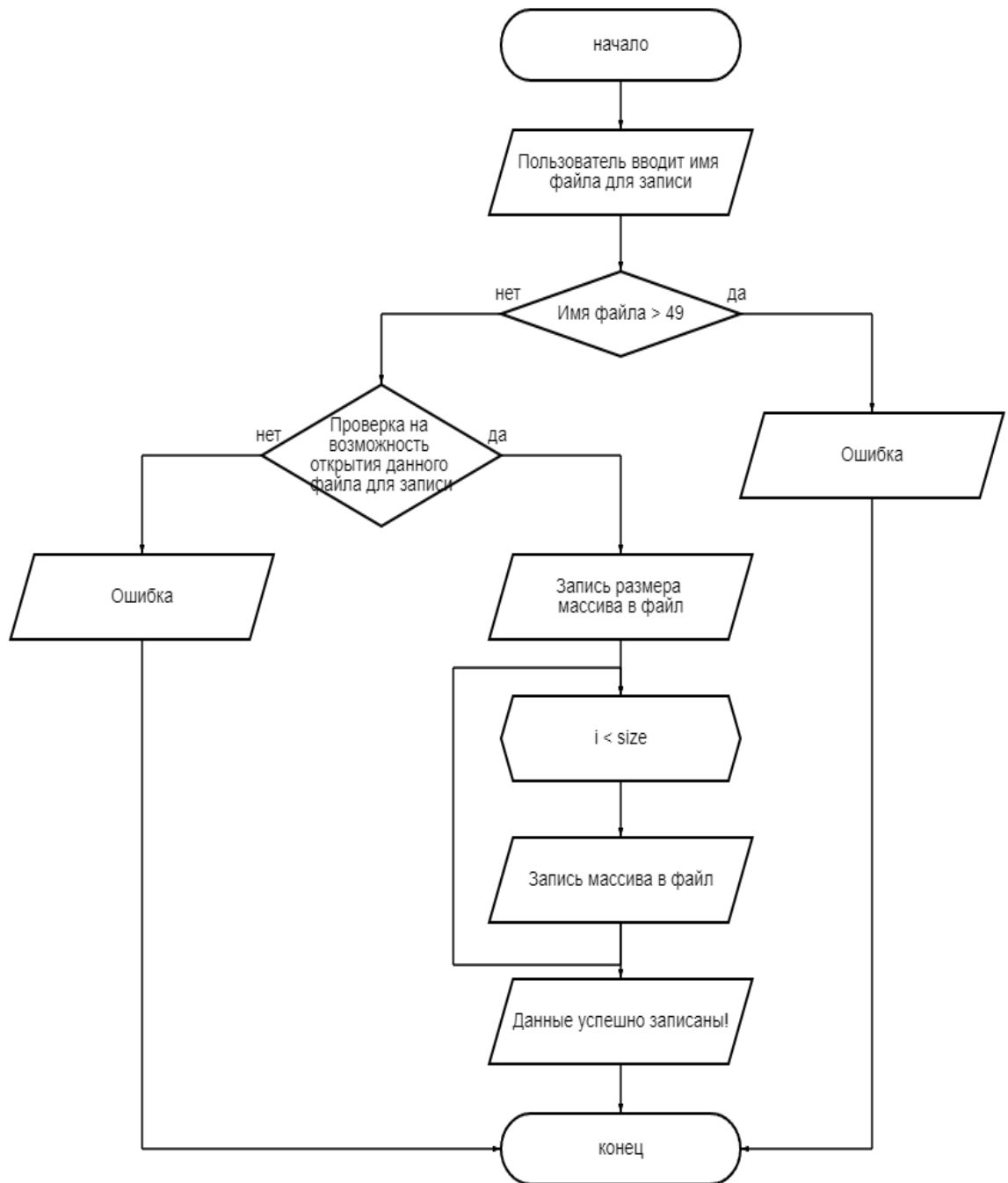
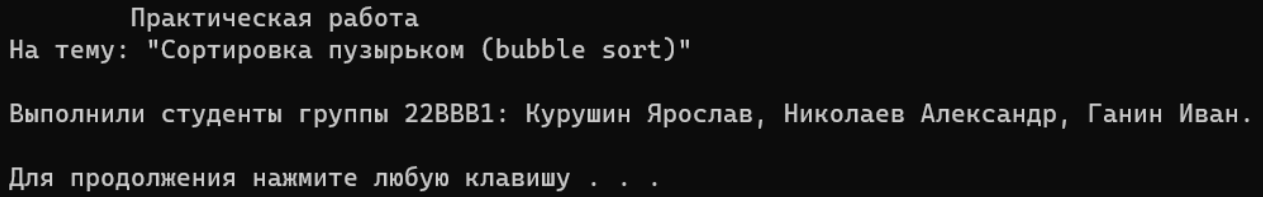


Рисунок 5 – Функция writeArrayToFile

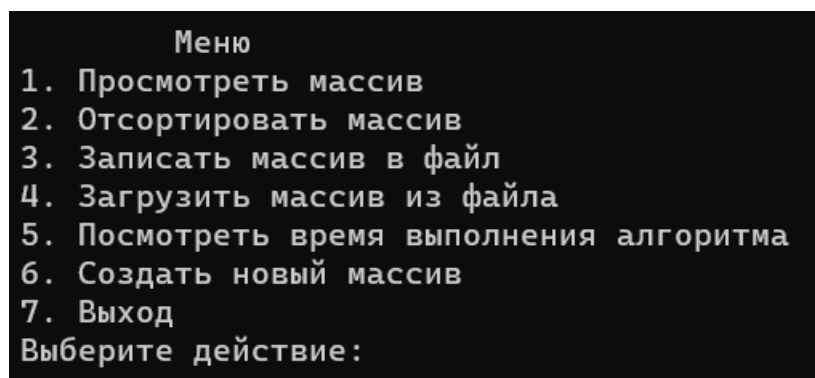
## 4 Тестирование программы

### 4.1 Тестирование интерфейса



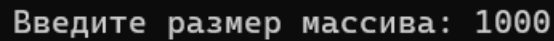
Практическая работа  
На тему: "Сортировка пузырьком (bubble sort)"  
Выполнили студенты группы 22ВВВ1: Курушин Ярослав, Николаев Александр, Ганин Иван.  
Для продолжения нажмите любую клавишу . . .

Рисунок 6 – Главный экран



Меню  
1. Просмотреть массив  
2. Отсортировать массив  
3. Записать массив в файл  
4. Загрузить массив из файла  
5. Посмотреть время выполнения алгоритма  
6. Создать новый массив  
7. Выход  
Выберите действие:

Рисунок 7 - Меню



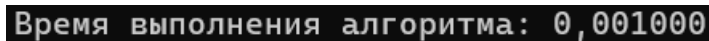
Введите размер массива: 1000

Рисунок 8 – Пункт меню №6



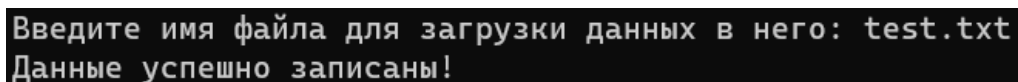
Массив отсортирован!

Рисунок 9 – Пункт меню №2



Время выполнения алгоритма: 0,001000

Рисунок 10 - Пункт меню №5



Введите имя файла для загрузки данных в него: test.txt  
Данные успешно записаны!

Рисунок 11 - Пункт меню №3



```
981) 22522
982) 23674
983) 32417
984) 27240
985) 19663
986) 7960
987) 15280
988) 27585
989) 18803
990) 10335
991) 7811
992) 21130
993) 17809
994) 2011
995) 21707
996) 16579
997) 6646
998) 14008
999) 10319
1000) 31011
```

**Меню**

1. Просмотреть массив
  2. Отсортировать массив
  3. Записать массив в файл
  4. Загрузить массив из файла
  5. Посмотреть время выполнения алгоритма
  6. Создать новый массив
  7. Выход
- Выберите действие:

Рисунок 12 – Пункт меню №1

**До свидания!**

Рисунок 13 – Пункт меню №7

Введите имя файла для загрузки данных из него: test.txt  
Данные успешно прочитаны!

Рисунок 14 - Пункт меню №4

## **4.2 Тестирование на разных наборах данных**

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены на рисунках А.1-А.11.

Таблица 1 - Тестовый набор данных

№ теста	Количество элементов	Время выполнения в секундах
1	10 000	0,242
2	20 000	1,048
3	30 000	2,54
4	40 000	4,51
5	50 000	7,012
6	60 000	10,112
7	70 000	14,756
8	80 000	18,328
9	90 000	23,353
10	100 000	29,099
11	110 000	35,306

### 4.3 Анализ полученных результатов тестирования

Проводя анализ работы алгоритма, мы выяснили, что при увеличении количества элементов последовательности, возрастает также и время выполнения сортировки. Причём увеличение происходит в геометрической прогрессии. Поэтому данная сортировка неэффективна на большом количестве данных.

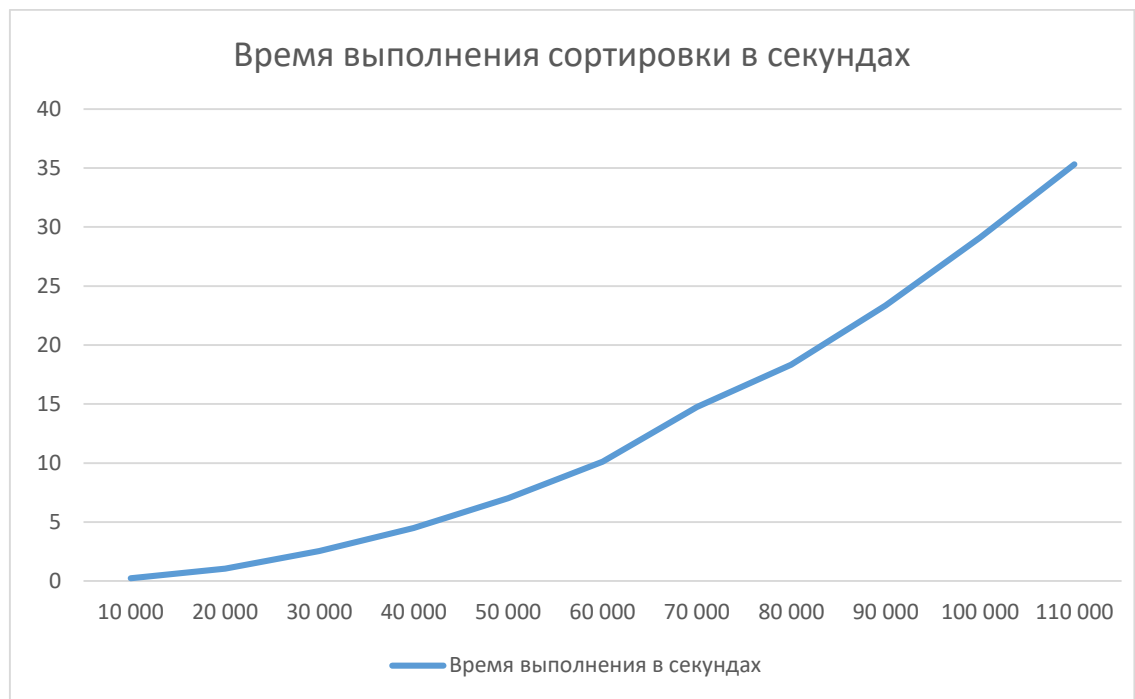


Рисунок 15 - График времени выполнения сортировки

## 5. Совместная разработка

Во время работы над данной практикой, нашей бригадой осуществлялась совместная работа в GitHub.

Первоначальный алгоритм был написан Курушиным Я.С.

После написания программы она была выгружена на удаленный репозиторий GitHub на ветку main.



Рисунок 16 – Загрузка программы на GitHub

После этого, второй участник – Николаев А.А. загрузил данную программу себе на компьютер, с помощью `git clone <ссылка>`, и добавил запись массивов в файл. Также оптимизировал код.



Рисунок 17 – Загрузка доработанного кода на GitHub

После этого Курушин Я.С. исправил мелкие недочеты кода и добавил проверку.

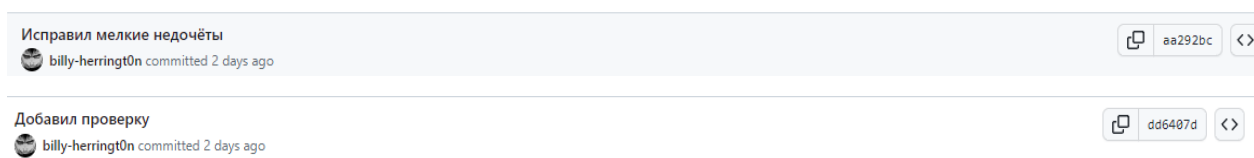


Рисунок 18 – Исправление мелких недочетов

После этого, третий участник – Ганин И.Р. провел тестирование программы и создал блок-схемы. Все скриншоты загрузил в отдельные папки в ветке main



Рисунок 19 – Загрузка блок-схем и тестировка

## **Заключение**

За период выполнения данной учебной практической работы были освоены основные понятия и технологии пузырьковой сортировки, реализован метод работы с файлами. Продемонстрирована работа алгоритма сортировки «пузырьком». Конечная программа реализовывает непосредственно алгоритм, а также выполняет считывание входных данных из файла и запись результатов в файл. В ходе тестирования была произведена отладка, алгоритм выполняется верно, предусмотрены ограничения формата ввода чисел и ситуация ошибки чтения файлов.

В результате тестирования мы пришли к выводу, что данный алгоритм подходит для сортировки небольших массивов, так как при увеличении числа элементов, подвергающихся сортировке, в геометрической прогрессии растёт время выполнения программы. Сортировка «пузырьком» является довольно простым и эффективным алгоритмом в случае, если применять его при работе с небольшими последовательностями чисел, либо с частично отсортированными последовательностями, либо в ознакомительных и учебных целях. Реализация алгоритма проста, однако высокая вычислительная сложность и большие временные затраты на реализацию сортировки сильно ограничивают сферу применения данного метода.

### **Список использованных источников**

1. Дэвид Гриффитс, Дон Гриффитс. Изучаем программирование на С. 2013 г.
2. Введение в си [Электронный ресурс] – Режим доступа:  
<https://habr.com/ru/articles/464075>
3. Язык С [Электронный ресурс] - Режим доступа: <https://prog-cpp.ru/c>
4. Сортировка пузырьком[Электронный ресурс] - Режим доступа:  
<https://foxford.ru/wiki/informatika/sortirovka-metodom-puzyrka>

## Приложение А

```
Время выполнения алгоритма: 0,242000
Меню
```

Рисунок А.1 - Тестирование при 10 000 элементов

```
Время выполнения алгоритма: 1,048000
Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие:
```

Рисунок А.2 - Тестирование при 20 000 элементов

```
Время выполнения алгоритма: 2,540000
Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие:
```

Рисунок А.3 - Тестирование при 30 000 элементов

```
Время выполнения алгоритма: 4,510000
Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие:
```

Рисунок А.4 - Тестирование при 40 000 элементов

```
Время выполнения алгоритма: 7,012000
Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие: _
```

Рисунок А.5 - Тестирование при 50 000 элементов

```
Время выполнения алгоритма: 10,112000
    Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие:
```

Рисунок А.6 - Тестирование при 60 000 элементов

```
Время выполнения алгоритма: 14,756000
    Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие:
```

Рисунок А.7 - Тестирование при 70 000 элементов

```
Время выполнения алгоритма: 18,328000
    Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие:
```

Рисунок А.8 - Тестирование при 80 000 элементов

```
Время выполнения алгоритма: 23,353000
    Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие:
```

Рисунок А.9 - Тестирование при 90 000 элементов



```
Время выполнения алгоритма: 29,099000
    Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие: █
```

Рисунок А.10 - Тестирование при 100 000 элементов

```
Время выполнения алгоритма: 35,306000
    Меню
1. Просмотреть массив
2. Отсортировать массив
3. Записать массив в файл
4. Загрузить массив из файла
5. Посмотреть время выполнения алгоритма
6. Создать новый массив
7. Выход
Выберите действие: █
```

Рисунок А.11 - Тестирование при 110 000 элементов

## Приложение Б. Листинг

### Файл main.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <conio.h>
#include <locale.h>
#include <stdlib.h>
#include <time.h>
int size, choice;
int* arr = 0;
char n_file[49];
FILE* file;
void bubbleSort(int arr[], int size) {
    int k = 0;
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                k++;
            }
            if (k == 0)
                break;
        }
    }
}
void writeArrayToFile(int arr[], int size) {
    printf("Введите имя файла для загрузки данных в него: ");
    scanf("%s", n_file);
    if (sizeof(n_file) > 49) {
        printf("Имя файла превышает доступное значение\n");
        return;
    }
    if ((file = fopen(n_file, "w")) == NULL) {
        printf("\nНевозможно открыть для записи файл: %s\n",
n_file);
        _getch();
        return;
    }
    fprintf(file, "%d\n", size);
    for (int i = 0; i < size; i++)
        fprintf(file, "%d ", arr[i]);

    fclose(file);
    printf("Данные успешно записаны!\n");
}
void readArrayFromFile() {
    printf("Введите имя файла для загрузки данных из него: ");
```

```

scanf("%s", n_file);
if (sizeof(n_file) > 49) {
    printf("Имя файла превышает доступное значение
(49)!");
    return;
}
if ((file = fopen(n_file, "r+")) == 0)
{
    printf("Невозможно открыть файл для чтения %s\n",
n_file);
    _getch();
    return;
}
fscanf(file, "%d", &size);
arr = (int*)malloc(size * sizeof(int));
for (int i = 0; i < size; i++) {
    fscanf(file, "%d", &arr[i]);
}
fclose(file);
printf("Данные успешно прочитаны!\n");
}
void helloScreen() {
    printf("\tПрактическая работа\nНа тему: \"Сортировка
пузырьком (bubble sort)\"\n\nВыполнили студенты группы 22BVB1:
Курушин Ярослав, Николаев Александр, Ганин Иван.\n\n");
    system("PAUSE");
}
void main() {
    setlocale(LC_ALL, "RUS");
    srand(time(NULL));
    helloScreen();
    clock_t start, end;
    double time_spent = 0.0;
    system("cls");
    do {
        printf("\tМеню\n");
        printf("1. Просмотреть массив\n");
        printf("2. Отсортировать массив\n");
        printf("3. Записать массив в файл\n");
        printf("4. Загрузить массив из файла\n");
        printf("5. Посмотреть время выполнения алгоритма\n");
        printf("6. Создать новый массив\n");
        printf("7. Выход\n");
        printf("Выберите действие: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: //Просмотр
                system("cls");
                printf("Массив:\n ");
                for (int i = 0; i < size; i++) {
                    printf("%d) %d\n ", i+1, arr[i]);
                }

```

```

        printf("\n");
        break;
    case 2: // Сортировка
        system("cls");
        start = clock();
        bubbleSort(arr, size);
        end = clock();
        printf("Массив отсортирован!\n");
        time_spent = (double)(end - start) /
CLOCKS_PER_SEC;
        printf("\n");
        break;
    case 3: // Запись в файл
        system("cls");
        writeArrayToFile(arr, size);
        break;
    case 4: // Запись в файл
        system("cls");
        if (arr != NULL)
        {
            free(arr);
        }
        readArrayFromFile();
        break;
    case 5: // Время
        system("cls");
        if (time_spent == 0) {
            printf("Сначала отсортируйте массив!\n");
            break;
        }
        printf("Время выполнения алгоритма: %f\n",
time_spent);
        break;
    case 6: //Ввод данных
        system("cls");
        free(arr);
        printf("Введите размер массива: ");
        scanf("%d", &size);
        arr = (int*)malloc(size * sizeof(int));
        if (arr == NULL) {
            printf("Не удалось выделить память!\n");
            return;
        }
        for (int i = 0; i < size; i++) {
            arr[i] = rand();
        }
        break;
    case 7: // Выход
        system("cls");
        printf("До свидания!\n");
        free(arr);
        break;

```

```
        default:
            system("cls");
            printf("Некорректный выбор!\n");
            break;
    }
} while (choice != 7);
getchar();
}
```