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

(2023/2024 учебный год)

Башвинов Илья Максимович

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 25.06.2024 по 08.07.2024

Кафедра «Вычислительная техника»

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

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

Руководитель практики к.т.н., доцент, Карамышева Н.С.

(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

(2023/2024 учебный год)

Башвинов Илья Максимович

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 25.06.2024 по 08.07.2024

Кафедра «Вычислительная техника»

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

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

Руководитель практики к.т.н., доцент, Карамышева Н.С.

(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	25.06.24 – 25.06.24	
2	Подбор и изучение материала по теме работы	15	26.06.24 – 28.06.24	
3	Разработка алгоритма	43	28.06.24 – 02.07.24	
4	Описание алгоритма и программы	18	02.07.24 – 04.07.24	
5	Тестирование	5	04.07.24 – 04.07.24	
6	Получение и анализ результатов	10	04.07.24 – 06.07.24	
7	Оформление отчёта	15	06.07.24 – 08.07.24	
	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

(2023/2024 учебный год)

Башвинов Илья Максимович

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 25.06.2024 по 08.07.2024

Кафедра «Вычислительная техника»

Башвинов И.М. выполнял практическое задание «Сортировка Шелла». На первоначальном этапе был изучен и проанализирован алгоритм сортировки Шелла, был выбран метод решения и языки программирования C и C++, на которых написана программа сортировки массива выбором. Написал, отладил и протестировал программу. Оформил отчет.

Бакалавр Башвинов И.М. _____ "___" _____ 2024 г.

Руководитель Карамышева Н.С. _____ "___" _____ 2024 г.
Практики

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

ОТЗЫВ

О ПРОХОЖДЕНИИ УЧЕБНОЙ (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКИ

(2023/2024 учебный год)

Башвинов Илья Максимович

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 25.06.2024 по 08.07.2024

Кафедра «Вычислительная техника»

В процессе выполнения практики Башвинов И.М. решал следующие задачи: разработка блок-схем, разработка интерфейса, разработка алгоритма сортировки, разработка api работы с файлами, тестирование алгоритма.

За период выполнения практики было освоено несколько алгоритмов сортировки, различные варианты реализации пользовательских интерфейсов, методы ручного тестирования. Во время выполнения работы Башвинов И.М. показал себя ответственным, добросовестным учеником, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике.

За выполнение работы Башвинов И.М. заслуживает оценки «_____».

Руководитель практики к/н., доцент, Карамышева Н.С. « » 2024 г.

Содержание

Введение

Сортировка - это процесс упорядочивания элементов некоторого множества в соответствии с определенным критерием или порядком.

В настоящее время сортировка данных является одним из ключевых процессов в обработке информации благодаря современному развитию компьютерных технологий. Задачи на сортировку данных встречаются повсеместно в различных профессиональных областях. Алгоритмы сортировки широко применяются в практике обработки информации и составляют отдельный класс алгоритмов. Они используются для обеспечения более эффективного последующего поиска данных.

Существует множество различных алгоритмов сортировки, каждый из которых имеет свои особенности, преимущества и недостатки.

Сортировка Шелла - это один из классических алгоритмов сортировки, который был предложен в 1959 году американским ученым Дональдом Шеллом. Этот алгоритм является усовершенствованным методом сортировки вставками и отличается тем, что он сначала сортирует элементы на некотором расстоянии друг от друга, а затем последовательно уменьшает это расстояние до единицы.

Сортировка Шелла позволяет существенно улучшить скорость сортировки по сравнению с обычной сортировкой вставками, особенно на больших наборах данных. Алгоритм Шелла эффективно работает на различных типах данных и может ускорить сортировку, даже если данные уже частично отсортированы.

1. Разработка и описание алгоритма

Сортировка Шелла — алгоритм сортировки, являющийся усовершенствованным вариантом сортировки вставками.

Фундаментальная идея этого алгоритма сортировки состоит в том, чтобы сгруппировать элементы, находящиеся далеко друг от друга, и соответствующим образом отсортировать их. Затем постепенно уменьшайте промежуток между ними. Иными словами — это сортировка вставками с предварительными «грубыми» проходами.

Алгоритм сортировки Шелла:

1) Инициализируйте значение интервала, h = n/2. (n — размер массива)

2) Поместите все элементы в пределах интервала h в подсписок.

3) Отсортируйте эти подсписки, используя сортировку вставкой.

4) Установите новый интервал, h=h/2.

5) Если h>0, переходим к шагу 2. В противном случае переходим к 6.

6) Результатом будет отсортированный массив.

Достоинства алгоритма сортировки выбором:

- Никакого вызова стека не требуется.
- Легкая реализация.
- Эффективен для менее широко расположенных элементов.

Недостатки алгоритма:

- Неэффективно для массивов огромных размеров.
- Неэффективно для широко распространенных элементов.

2. Выбор решения

Для написания данной программы будут использованы языки программирования Си и Си++. Оба языка обладают рядом преимуществ, которые делают их популярными среди разработчиков программного обеспечения. Например, мощные возможности работы с памятью: в Си присутствуют возможности прямой работы с памятью, что позволяет оптимизировать код и управлять использованием ресурсов. Богатые возможности стандартной библиотеки: Си++ имеет обширную стандартную библиотеку, которая предоставляет разработчикам множество готовых инструментов для работы с файлами, строками, графическими элементами и другими структурами данных. Простота и эффективность: Си - это относительно простой язык программирования, который позволяет разработчикам быстро писать и отлаживать код. Портативность: программы на Си могут быть написаны один раз и запущены на различных платформах без изменений. А также высокая скорость выполнения.

В качестве среды программирования была выбрана программа Microsoft Visual Studio. Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

3. Описание программы

В программе для сортировки Шелла подключены следующие заголовочные файлы: *iostream* – заголовочный файл с классами, функциями и переменными для организации ввода-вывода; *fstream* – заголовочный файл, предоставляющий функционал для считывания данных из файла и для записи в файл; *vector* – загаловочный файл для использования векторов (контейнеров, содержащих коллекцию объектов одного типа); *locale* – заголовочный файл для вывода русского (и не только) языка в консоли; *Window.h* – заголовочный файл, в котором объявляются функции, предоставляющие интерфейс доступа к Windows API (в программе используется для цветного текста); *HANDLE hConsole* – объявление переменной дескриптора операционной системы (далее в программе используется для реализации цветного текста с помощью функции SetConsoleTextAttribute(hConsole, «число, указывающее на цвет»)).

```
#include <iostream>
#include <vector>
#include <fstream>
#include <locale>
#include <Windows.h>

HANDLE hConsole;
```

Далее идет текст функции Sort. В первом цикле переменная interval устанавливается в половину размера вектора. Этот цикл уменьшает интервал между элементами, которые будут сравниваться и перемещаться при сортировке. Внутри этого цикла на каждом шаге происходит вставочная сортировка подмассивов внутри интервала. Т.е. на каждом шаге мы сравниваем элементы через определенные промежутки (interval) и устанавливаем их в нужном порядке, чтобы подготовить данные для окончательной сортировки. Это осуществляется с помощью цикла for с переменной i, которая идет по элементам вектора на заданном интервале. Внутри этого цикла мы сохраняем значение элемента data[i] во временной переменной temp. Затем мы ищем место для вставки этого элемента, сравнивая

его с элементами, находящимися на расстоянии интервала. Мы идем назад от текущего i до начала массива или пока элементы больше temp, сдвигая элементы вправо. После того как найдено правильное место для вставки, элемент на этом места вставляется в массив, итерация цикла заканчивается, и мы идем дальше по массиву. Этот процесс повторяется для всех интервалов, пока интервал не станет равен 1, а затем вектор data завершенно отсортирован. Данная функция вызывается в каждом из видов сортировки Шелла, представленных в программе.

```
void Sort(std::vector<int>& data) {
    int size = data.size();
    for (int interval = size / 2; interval > 0; interval /= 2) {
        for (int i = interval; i < size; i++) {
            int temp = data[i];
            int j;
            for (j = i; j >= interval && data[j - interval] > temp; j -= interval) {
                data[j] = data[j - interval];
            }
            data[j] = temp;
        }
    }
}
```

Далее располагается текст функции выхода в меню, функция которого прописывается позже. Вводится переменная choice, которая при получении значения 1, производит выход из цикла while. Внутри цикла выводится надпись пункта меню “1. Выход в меню”. При вводе цифры 1 с клавиатуры, происходит присвоение переменной choice значения 1, соответственно выход из цикла и выход в меню. При вводе любого другого числа выводится ошибка и предложение ввести правильный вариант. Данная функция вызывается в конце каждого из видов ввода данных для сортировки Шелла.

```
void ExitToMenu() {
    int choice = 0;
    while (choice != 1) {
        SetConsoleTextAttribute(hConsole, 4);
        std::cout << "\n1.Выход в меню\n";
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "\nВведите свой выбор: ";
        std::cin >> choice;
        switch (choice) {
        case 1:
            system("cls");
            break;
        default:
```

```
            system("cls");
            SetConsoleTextAttribute(hConsole, 4);
            std::cout << "Некорректный выбор. Пожалуйста, введите цифру 1 для выхода
в меню.\n\n";
            break;
        }
    }
}
```

Далее идет текст функции реализации ввода элементов массива вручную для сортировки Шелла. В цикле while находится ввод значений. Если введенные значения не являются числовыми, то они игнорируются. При нажатии пользователем клавиши Enter, происходит выход из цикла и заканчивается ввод элементов. Далее исходный массив записывается в файл input.txt. После вызывается функция сортировки, отсортированный массив выводится в консоль и записывается в файл output.txt. После выполнения всех действий происходит очистка вектора data.

```
void ShellSort(std::vector<int>& data) {
    setlocale(LC_ALL, "Rus");

    std::cout << "Введите элементы массива (нажмите Enter когда закончите):\n";
    SetConsoleTextAttribute(hConsole, 7);
    int element;
    while (true) {
        if (std::cin >> element) {
            data.push_back(element);
        }
        else {
            std::cin.clear();
            std::cin.ignore();
        }
        if (std::cin.peek() == '\n') {
            std::cin.ignore();
            break;
        }
    }

    std::ofstream in("input.txt");
    for (int& num : data) {
        in << num << " ";
    }
    in.close();

    SetConsoleTextAttribute(hConsole, 2);
    std::cout << "\nИсходный массив записан в файл 'input.txt'\n";

    Sort(data);

    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "\nОтсортированный массив:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::ofstream out("output.txt");
    for (int& num : data) {
```

```
        std::cout << num << " ";
        out << num << " ";
    }
    std::cout << "\n";
    out.close();
    data.clear();

    SetConsoleTextAttribute(hConsole, 2);
    std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";

    ExitToMenu();
      }
```

Далее идет текст функции реализации ввода случайных элементов массива для сортировки Шелла. Пользователь вводит с клавиатуры размер массива, минимальное и максимальное значения. Исходный массив заполняется случайными элементами с помощью цикла for, где применяется функция rand и учитываются введенные пользователем ограничения. Далее идут условия if и else. Если элементов в массиве больше 100, то они не выводятся на экран, а только записывается в файл. Это применяется как для исходного массива, так и для отсортированного. После выполнения всех действий происходит очистка вектора data.

```
void ShellSortRand(std::vector<int>& data) {
    setlocale(LC_ALL, "Rus");
    int count, minVal, maxVal;

    std::cout << "Введите количество элементов случайного массива:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::cin >> count;
    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "Введите минимальное значение:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::cin >> minVal;
    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "Введите максимальное значение:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::cin >> maxVal;

    for (int i = 0; i < count; i++) {
        int randomNum = rand() % (maxVal - minVal + 1) + minVal;
        data.push_back(randomNum);
    }

    if (count > 100) {
        SetConsoleTextAttribute(hConsole, 2);
        std::cout << "\nИсходный массив записан в файл 'input.txt'";
        std::ofstream in("input.txt");
        for (int& num : data) {
            in << num << " ";
        }
        in.close();
    }
```

```
    else {
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "\nИсходный массив:\n";
        std::ofstream in("input.txt");
        SetConsoleTextAttribute(hConsole, 7);
        for (int& num : data) {
            std::cout << num << " ";
            in << num << " ";
        }
        std::cout << "\n";
        in.close();
        SetConsoleTextAttribute(hConsole, 2);
        std::cout << "\nИсходный массив записан в файл 'input.txt'\n";
    }

    Sort(data);

    if (count > 100) {
        SetConsoleTextAttribute(hConsole, 2);
        std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";
        std::ofstream out("output.txt");
        for (int& num : data) {
            out << num << " ";
        }
        out.close();
    }
    else {
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "\nОтсортированный массив:\n";
        std::ofstream out("output.txt");
        SetConsoleTextAttribute(hConsole, 7);
        for (int& num : data) {
            std::cout << num << " ";
            out << num << " ";
        }
        std::cout << "\n";
        out.close();
        SetConsoleTextAttribute(hConsole, 2);
        std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";
    }

    data.clear();

    ExitToMenu();
      }
```

Далее идет текст функции реализации ввода элементов массива из файла input.txt для сортировки Шелла. Происходит чтение из файла, вывод в консоль исходного и отсортированного массивов, запись отсортированного массива в файл output.txt. После происходит очистка вектора data.

```
void ShellSortFromFile(std::vector<int>& data) {
    setlocale(LC_ALL, "Rus");

    SetConsoleTextAttribute(hConsole, 7);
    std::ifstream in("input.txt");
    int num;
    while (in >> num) {
```

```
        data.push_back(num);
    }
    in.close();
    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "Исходный массив из файла 'input.txt':\n";
    SetConsoleTextAttribute(hConsole, 7);
    for (int& num : data) {
        std::cout << num << " ";
    }
    std::cout << "\n";

    Sort(data);

    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "\nОтсортированный массив:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::ofstream out("output.txt");
    for (int& num : data) {
        std::cout << num << " ";
        out << num << " ";
    }
    std::cout << "\n";
    out.close();
    data.clear();

    SetConsoleTextAttribute(hConsole, 2);
    std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";

    ExitToMenu();
      }
```

Далее идет текст функции реализации меню. Вводится переменная выбора choice. Реализуется цикл while, выход из которого происходит если choice принимает значение 4. Внутри цикла выводится текст пунктов меню, пронумерованных 1-4. При вводе цифры 1 вызывается функция ручного ввода элементов массива. При вводе цифры 2 вызывается функция случайных элементов массива. При вводе цифры 3 вызывается функция чтения элементов из файла. При вводе цифры 4 происходит выход из цикла, соответственно выход из функции и выход из программы. При вводе любого другого числа выходит ошибка и предложение выбрать из существующих вариантов. Также при любом из выборов происходит очистка консоли перед выполнением одного из видов сортировки для удобства пользования программой.

```
void Menu(std::vector<int>& data) {
    setlocale(LC_ALL, "Rus");
    int choice = 0;
    while (choice != 4) {
```

```
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "Меню:\n";
        SetConsoleTextAttribute(hConsole, 7);
        std::cout << "\n1.Ввод элементов массива вручную\n";
        std::cout << "2.Произвольный массив\n";
        std::cout << "3.Элементы массива из файла\n";
        SetConsoleTextAttribute(hConsole, 4);
        std::cout << "4.Выход\n";
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "\nВведите свой выбор: ";
        std::cin >> choice;

        switch (choice) {
        case 1:
            system("cls");
            ShellSort(data);
            break;
        case 2:
            system("cls");
            ShellSortRand(data);
            break;
        case 3:
            system("cls");
            ShellSortFromFile(data);
        case 4:
            system("cls");
            break;
        default:
            system("cls");
            SetConsoleTextAttribute(hConsole, 4);
            std::cout << "Некорректный выбор. Пожалуйста, введите цифру от 1 до 4
для выбора.\n\n";
            break;
        }
    }
      }
```

Затем идет текст главной программы. Сначала выполняется определение вектора data, потом вызов функции GetStdHandle() для получения дескриптора (handle) стандартного вывода консоли в Windows (для использования цветоного текста), а после запускается функция меню.

```
int main() {
    std::vector<int> data;

    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    Menu(data);

    return 0;
      }
```

4. Схема программы

4.1. Блок-схема программы

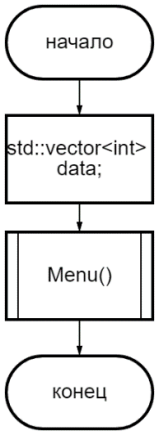

Рисунок 1 – Блок-схема программы

4.2. Блок-схема алгоритма сортировки

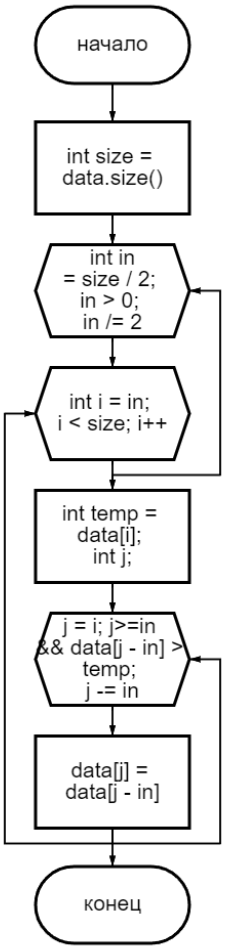

Рисунок 2 – Блок-схема алгоритма сортировки

4.3. Блок-схема функции выхода в меню

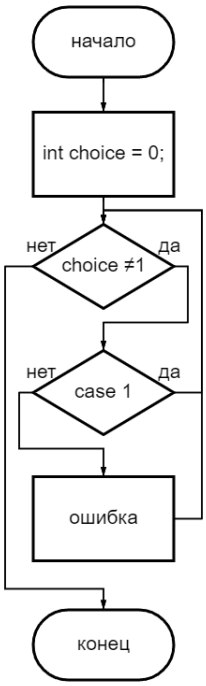

Рисунок 3 – Блок-схема функции выхода в меню

4.4. Блок-схема функции ручного ввода элементов

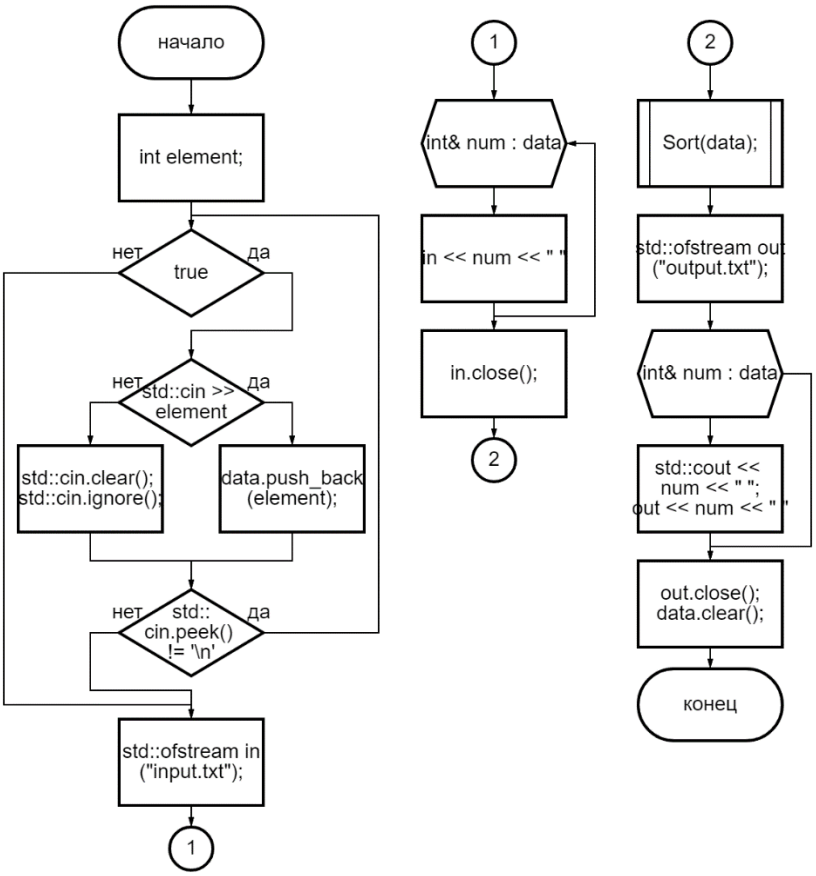

Рисунок 4 – Блок-схема функции ручного ввода элементов

4.5. Блок-схема функции случайных элементов

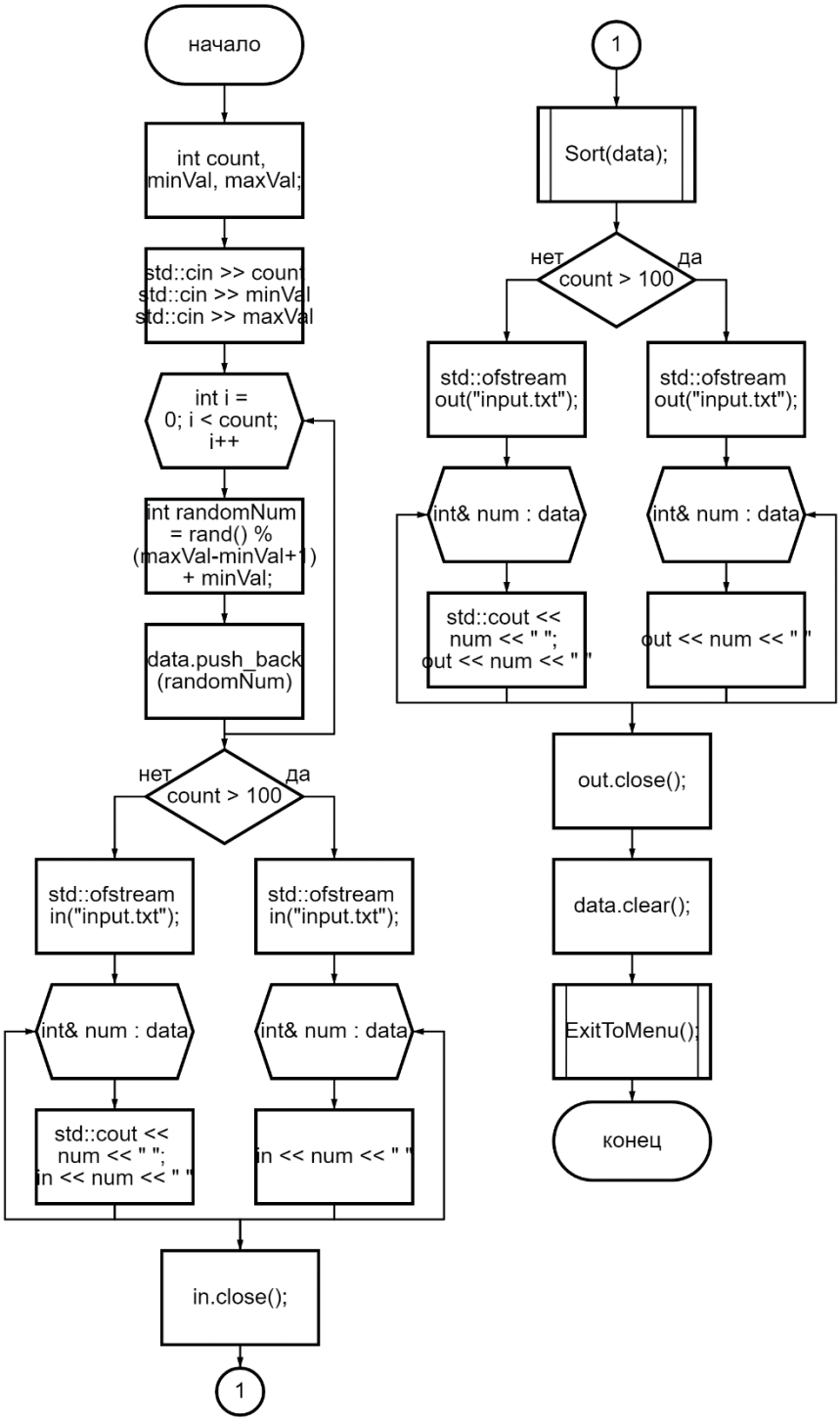

Рисунок 5 – Блок-схема функции случайных элементов

4.6. Блок-схема функции элементов из файла

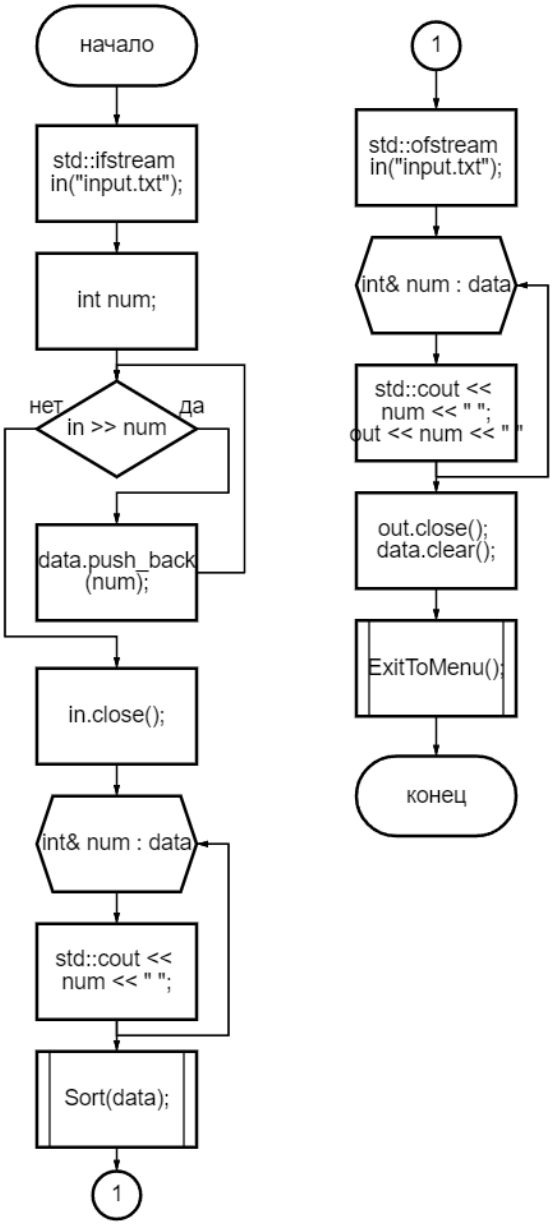

Рисунок 6 – Блок-схема функции элементов из файла

4.7. Блок схема функции меню

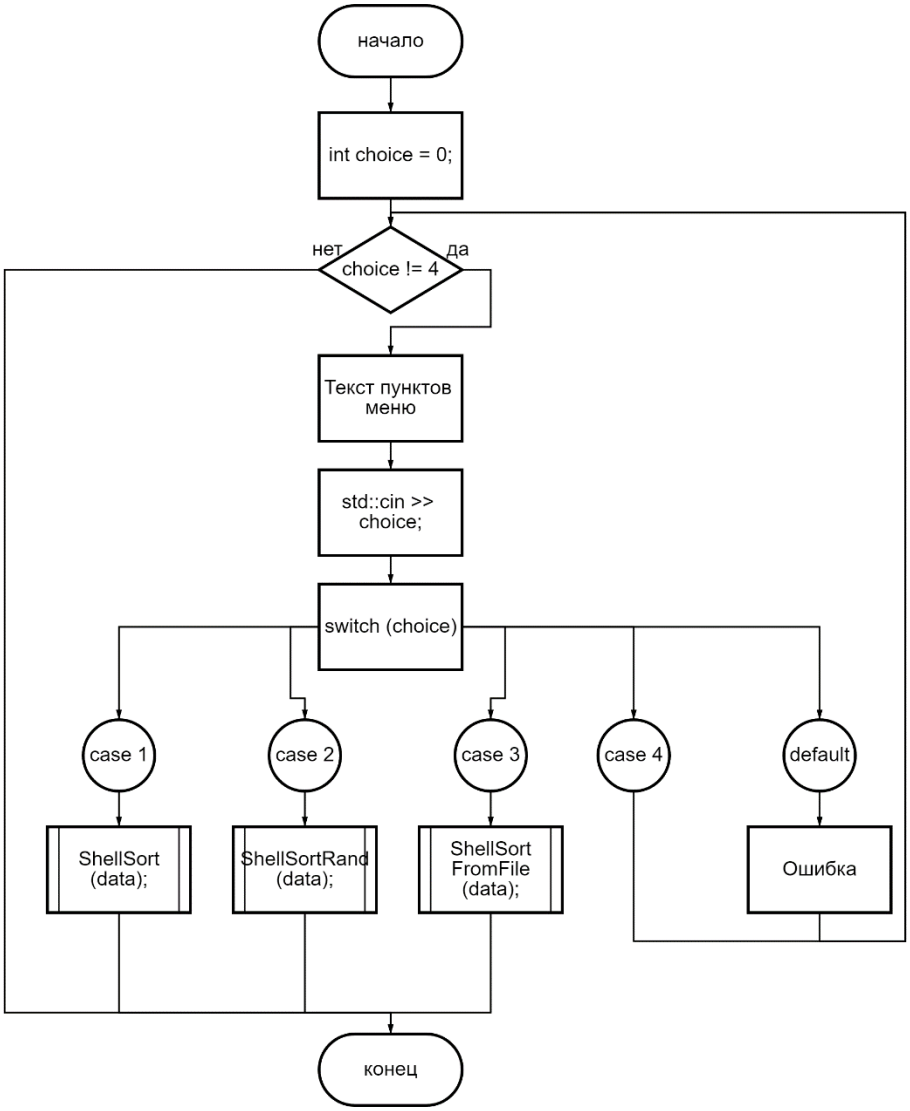

Рисунок 7 – Блок-схема функции меню

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 А.1 – А.11.

Таблица 1.

	Размер массива size	Время выполнения сортировки в миллисекундах
1	10000	9
2	20000	20
3	30000	32
4	40000	45
5	50000	56
6	60000	77
7	70000	91
8	80000	115
9	90000	124
10	100000	144
11	110000	150

5.2. Анализ полученных результатов тестирования

На основании анализа данных, полученных в результате тестирования алгоритма сортировки Шелла, можно сделать вывод, что время, затраченное на работу программы относительно количества элементов увеличивается линейно, то есть с увеличением количества элементов пропорционально увеличивается время работы программы.

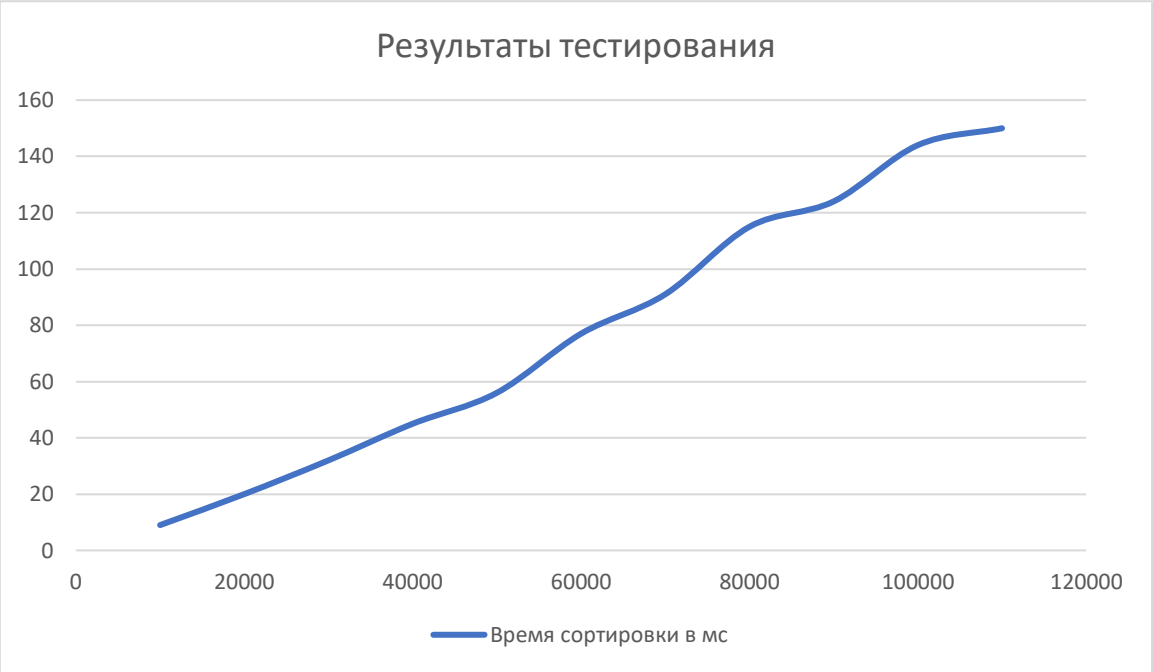

Рисунок 8 – Результаты тестирования

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 локальных переменных.

Точки останова – это прерывание выполнения программы, при котором выполняется вызов отладчика. Отладчик является инструментом для поиска и устранения ошибок в программе, с помощью которого можно исследовать состояние программы.

Команда шаг с заходом (step into) выполняет следующую инструкцию в обычном пути выполнения программы, а затем приостанавливает выполнение программы, чтобы мы могли проверить состояние программы с помощью отладчика. Если выполняемый оператор содержит вызов функции, шаг с заходом заставляет программу перескакивать в начало вызываемой функции, где она приостанавливается.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. После завершения написания программы, мною были выявлены и исправлены ошибки.

7. Совместная разработка

Для удобства был создан репозиторий. Ссылка на него: https://github.com/xex123/ShellSort

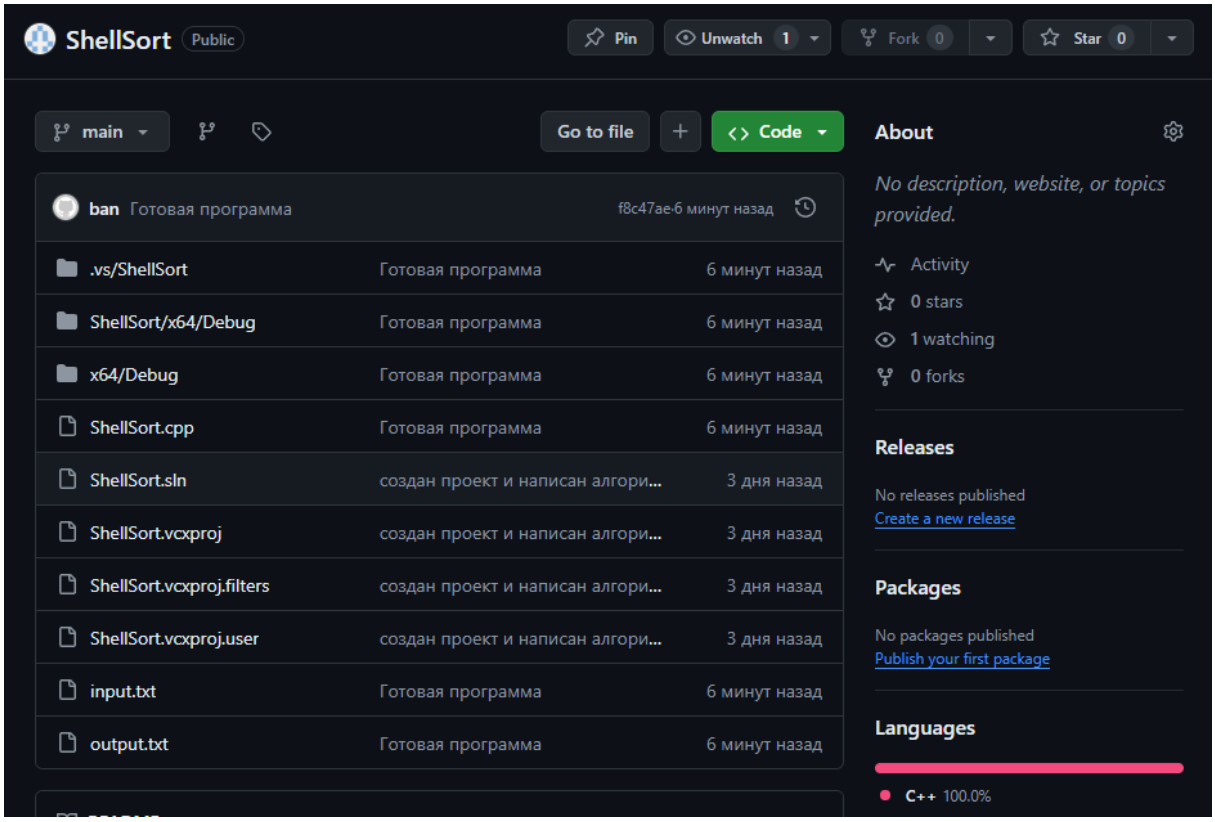

Рисунок 9 – Репозиторий и ветка main

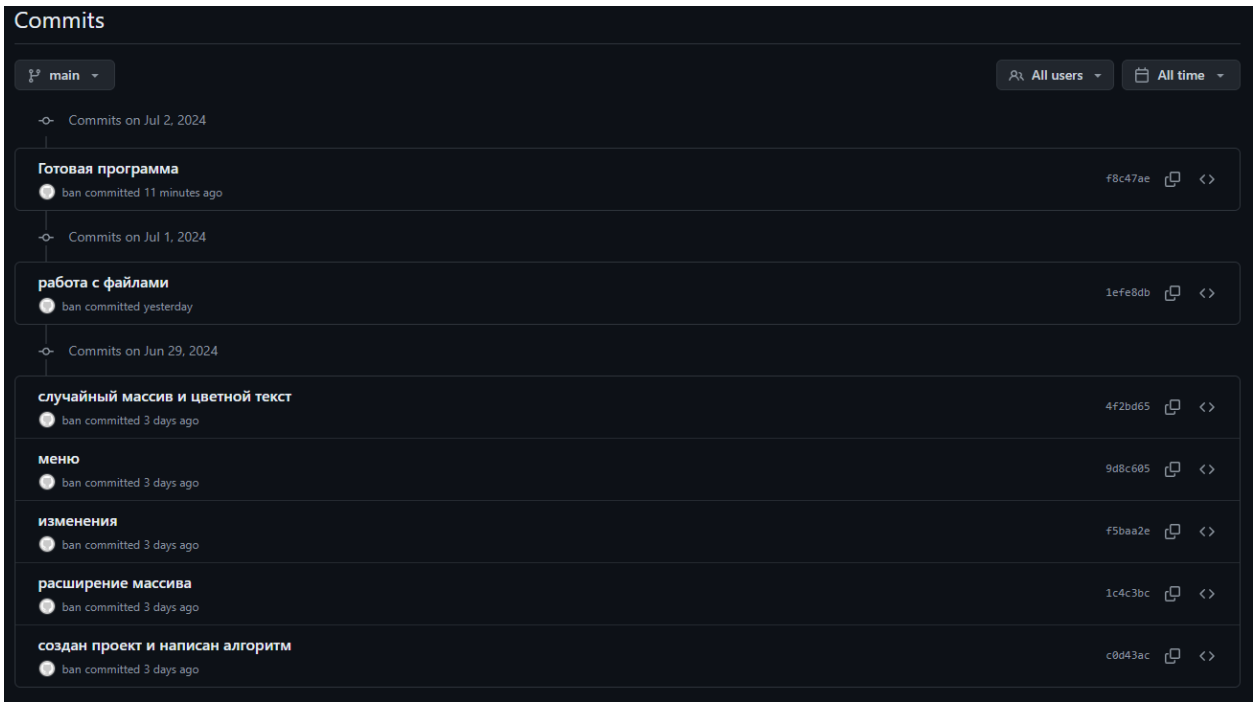

Рисунок 10 – Созданные коммиты

Для загрузки на локальный и глобальный репозиторий были использованы основные команды git bash.

```
Илья@DESKTOP-PTOQLRI MINGW64 /d/ShellSort (main)
$ git add .

Илья@DESKTOP-PTOQLRI MINGW64 /d/ShellSort (main)
$ git commit -m "Готовая программа"
[main f8c47ae] Готовая программа
 17 files changed, 10 insertions(+), 4 deletions(-)
 delete mode 100644 .vs/ShellSort/FileContentIndex/60d72553-3224-4c6e-a4f6-73242
6434a91.vsidx
 create mode 100644 .vs/ShellSort/FileContentIndex/f5a434b2-de01-4c4e-9a4d-530b4
29c3916.vsidx
 create mode 100644 input.txt
 create mode 100644 output.txt

Илья@DESKTOP-PTOQLRI MINGW64 /d/ShellSort (main)
$ git push -u origin main
Enumerating objects: 52, done.
Counting objects: 100% (52/52), done.
Delta compression using up to 12 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (28/28), 708.02 KiB | 519.00 KiB/s, done.
Total 28 (delta 14), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (14/14), completed with 14 local objects.
To https://github.com/xex123/ShellSort.git
   1efe8db..f8c47ae  main -> main
branch 'main' set up to track 'origin/main'.

Илья@DESKTOP-PTOQLRI MINGW64 /d/ShellSort (main)
$
```

Рисунок 11 – Команды add, commit и push

Заключение

При выполнении данной работы были получены навыки совместной работы с помощью сервисов GitHub, навыки использования программы Git Bash. Был изучен алгоритм сортировки Шелла.

Мною был написан алгоритм сортировки Шелла, алгоритм создания и заполнения файла, а также было написано меню выбора метода заполнения исходного массива.

При выполнении практической работы были улучшены базовые навыки программирования на языках C и 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.

Список используемой литературы

1. ГОСТ 19.701 – 90 Схемы алгоритмов, программ, данных и систем.
2. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2-е издание.: Пер. с англ. – М.,2009.
3. Роберт Седжвик. Фундаментальные алгоритмы на C++. Части 1-4. Анализ. Структуры данных.
4. И.В.Красиков, И.Е.Красикова. Алгоритмы. Просто как дважды два, 2007.
5. Сортировка Шелла [электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Сортировка_Шелла

Приложение А. Результаты тестирование программы

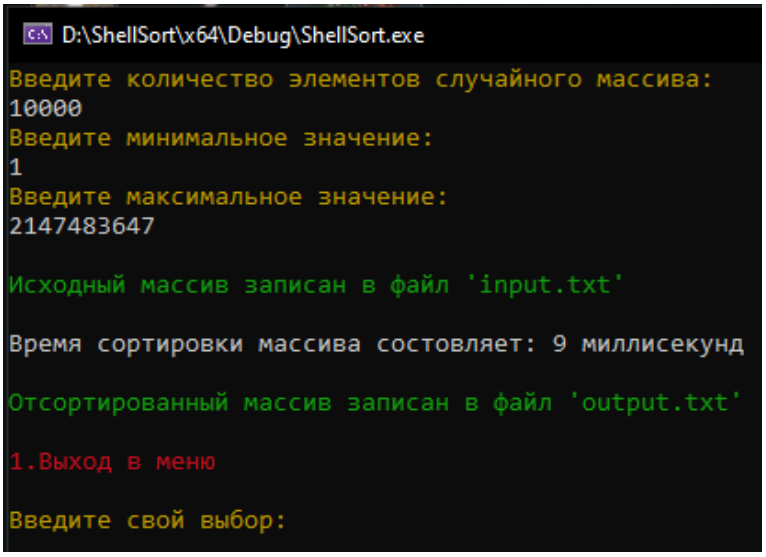

Рисунок А.1

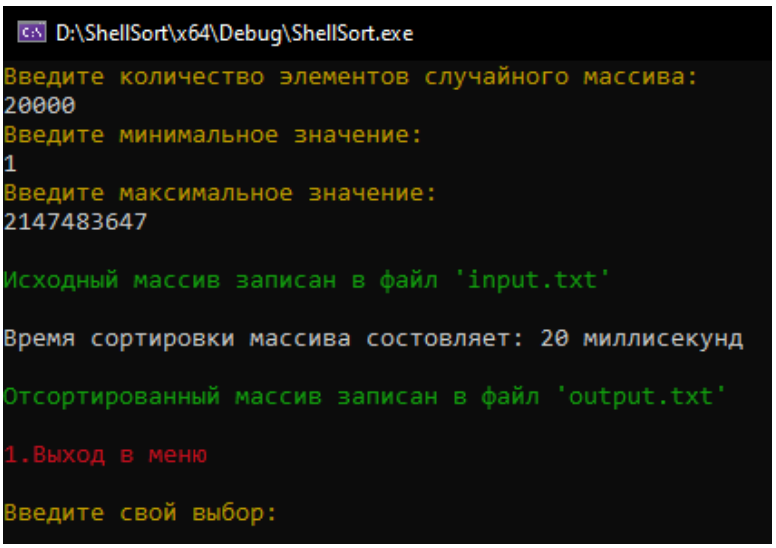

Рисунок А.2

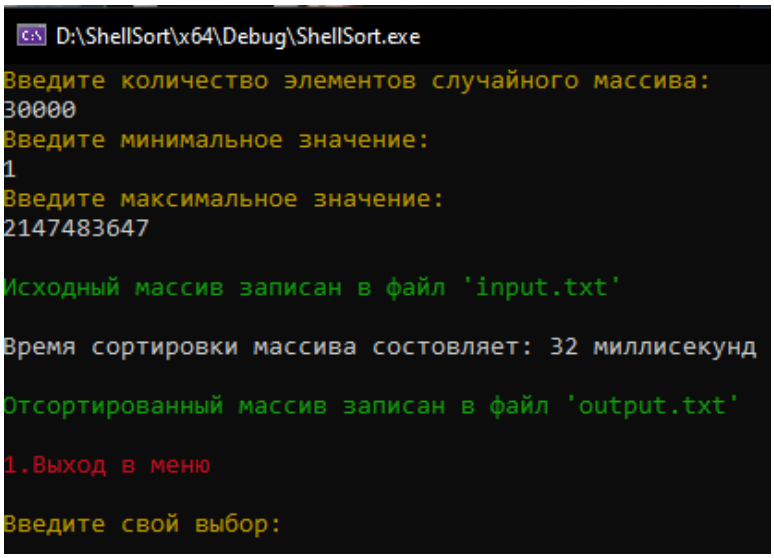

Рисунок А.3

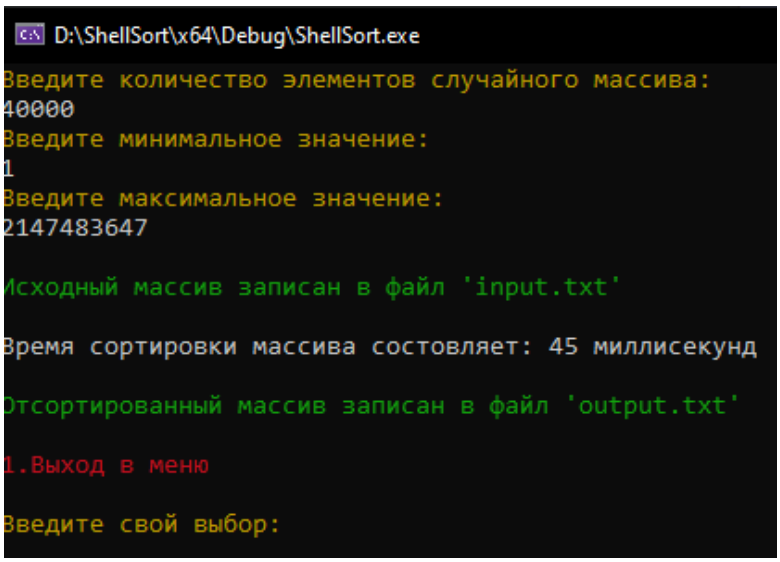

Рисунок А.4

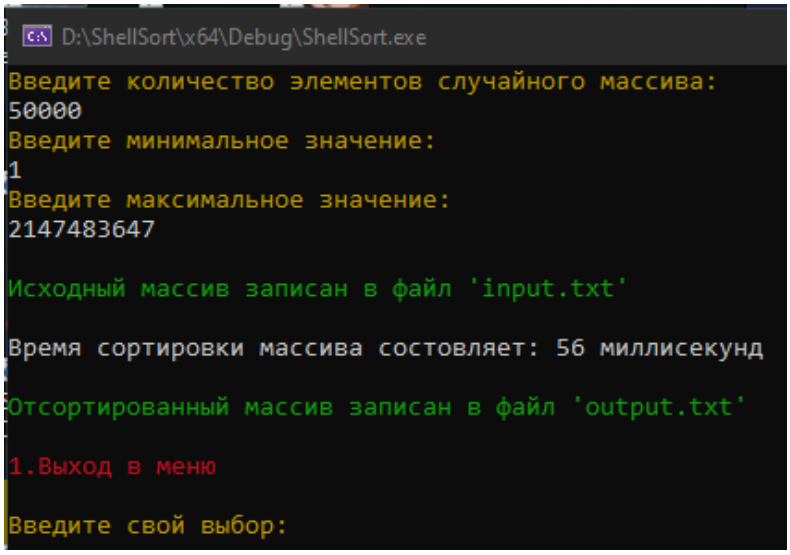

Рисунок А.5

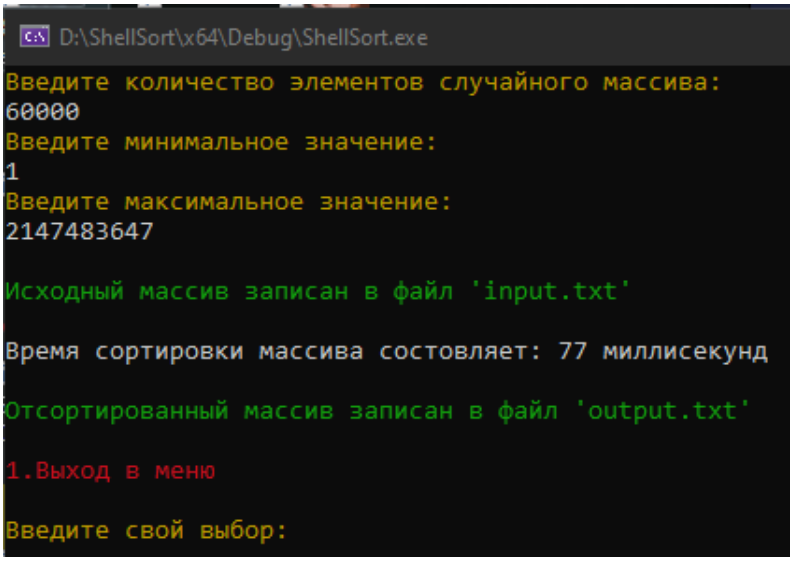

Рисунок А.6

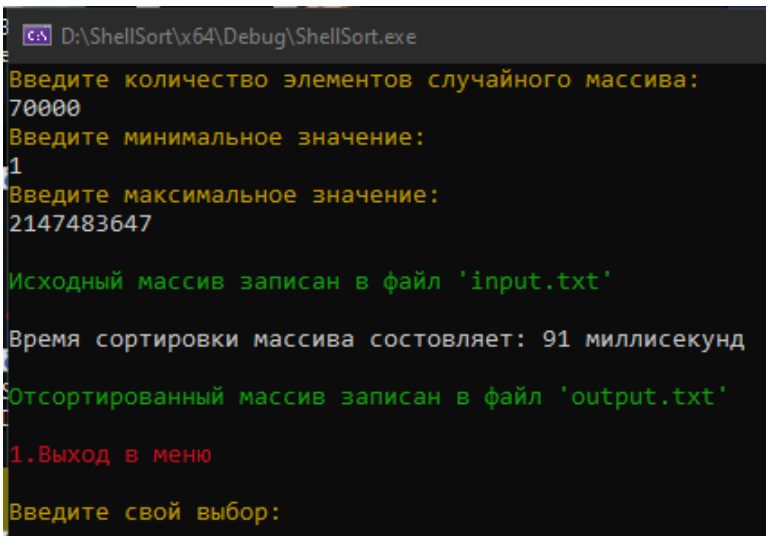

Рисунок А.7

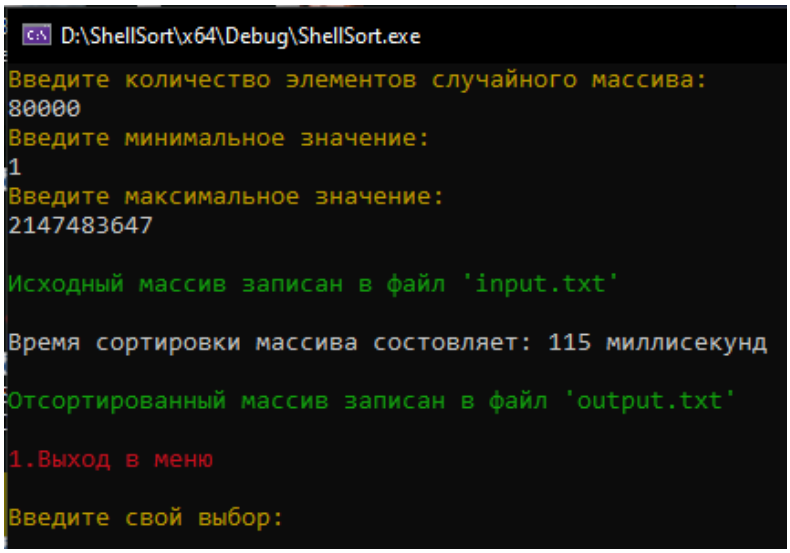

Рисунок А.8

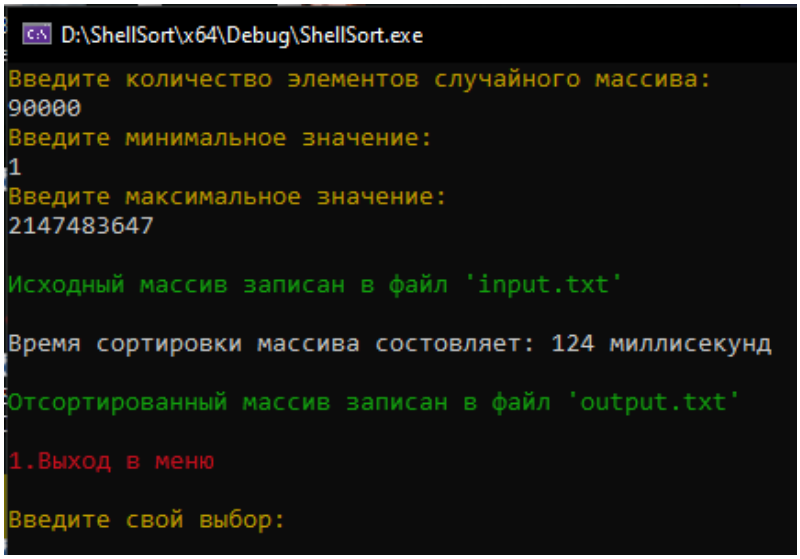

Рисунок А.9

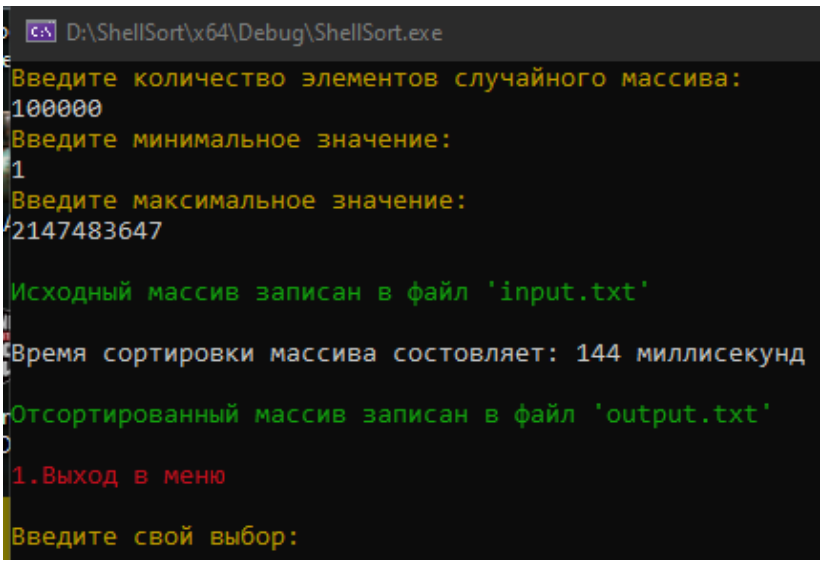

Рисунок А.10

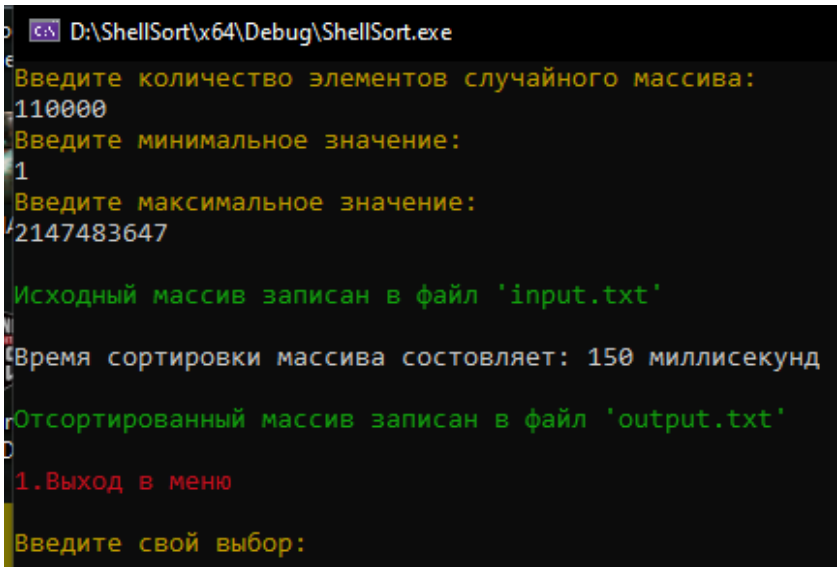

Рисунок А.11

Приложение Б. Листинг программы

```
#include <iostream>
#include <vector>
#include <fstream>
#include <locale>
#include <Windows.h>

HANDLE hConsole;

void Sort(std::vector<int>& data) {
    int size = data.size();
    clock_t start = clock();
    for (int interval = size / 2; interval > 0; interval /= 2) {
        for (int i = interval; i < size; i++) {
            int temp = data[i];
            int j;
            for (j = i; j >= interval && data[j - interval] > temp; j -= interval) {
                data[j] = data[j - interval];
            }
            data[j] = temp;
        }
    }
    clock_t stop = clock();
    double duration = double(stop - start) / CLOCKS_PER_SEC;
    SetConsoleTextAttribute(hConsole, 7);
    std::cout << "\n\nВремя сортировки массива состовляет: " << duration * 1000 << "
миллисекунд\n";
}

void ExitToMenu() {
    int choice = 0;
    while (choice != 1) {
        SetConsoleTextAttribute(hConsole, 4);
        std::cout << "\n1.Выход в меню\n";
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "\nВведите свой выбор: ";
        std::cin >> choice;
        switch (choice) {
        case 1:
            system("cls");
            break;
        default:
            system("cls");
            SetConsoleTextAttribute(hConsole, 4);
            std::cout << "Некорректный выбор. Пожалуйста, введите цифру 1 для выхода
в меню.\n\n";
            break;
        }
    }
}

void ShellSort(std::vector<int>& data) {
    setlocale(LC_ALL, "Rus");

    std::cout << "Введите элементы массива (нажмите Enter когда закончите):\n";
    SetConsoleTextAttribute(hConsole, 7);
    int element;
    while (true) {
        if (std::cin >> element) {
            data.push_back(element);
        }
        else {
            std::cin.clear();
```

```
            std::cin.ignore();
        }
        if (std::cin.peek() == '\n') {
            std::cin.ignore();
            break;
        }
    }

    std::ofstream in("input.txt");
    for (int& num : data) {
        in << num << " ";
    }
    in.close();

    SetConsoleTextAttribute(hConsole, 2);
    std::cout << "\nИсходный массив записан в файл 'input.txt'\n";

    Sort(data);

    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "\nОтсортированный массив:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::ofstream out("output.txt");
    for (int& num : data) {
        std::cout << num << " ";
        out << num << " ";
    }
    std::cout << "\n";
    out.close();
    data.clear();

    SetConsoleTextAttribute(hConsole, 2);
    std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";

    ExitToMenu();
}

void ShellSortRand(std::vector<int>& data) {
    setlocale(LC_ALL, "Rus");
    int count, minVal, maxVal;

    std::cout << "Введите количество элементов случайного массива:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::cin >> count;
    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "Введите минимальное значение:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::cin >> minVal;
    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "Введите максимальное значение:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::cin >> maxVal;

    for (int i = 0; i < count; i++) {
        int randomNum = rand() % (maxVal - minVal + 1) + minVal;
        data.push_back(randomNum);
    }

    if (count > 100) {
        SetConsoleTextAttribute(hConsole, 2);
        std::cout << "\nИсходный массив записан в файл 'input.txt'";
        std::ofstream in("input.txt");
        for (int& num : data) {
            in << num << " ";
        }
```

```
            in.close();
        }
        else {
            SetConsoleTextAttribute(hConsole, 6);
            std::cout << "\nИсходный массив:\n";
            std::ofstream in("input.txt");
            SetConsoleTextAttribute(hConsole, 7);
            for (int& num : data) {
                std::cout << num << " ";
                in << num << " ";
            }
            std::cout << "\n";
            in.close();
            SetConsoleTextAttribute(hConsole, 2);
            std::cout << "\nИсходный массив записан в файл 'input.txt'\n";
        }

        Sort(data);

        if (count > 100) {
            SetConsoleTextAttribute(hConsole, 2);
            std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";
            std::ofstream out("output.txt");
            for (int& num : data) {
                out << num << " ";
            }
            out.close();
        }
        else {
            SetConsoleTextAttribute(hConsole, 6);
            std::cout << "\nОтсортированный массив:\n";
            std::ofstream out("output.txt");
            SetConsoleTextAttribute(hConsole, 7);
            for (int& num : data) {
                std::cout << num << " ";
                out << num << " ";
            }
            std::cout << "\n";
            out.close();
            SetConsoleTextAttribute(hConsole, 2);
            std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";
        }

        data.clear();

        ExitToMenu();
    }

    void ShellSortFromFile(std::vector<int>& data) {
        setlocale(LC_ALL, "Rus");

        SetConsoleTextAttribute(hConsole, 7);
        std::ifstream in("input.txt");
        int num;
        while (in >> num) {
            data.push_back(num);
        }
        in.close();
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "Исходный массив из файла 'input.txt':\n";
        SetConsoleTextAttribute(hConsole, 7);
        for (int& num : data) {
            std::cout << num << " ";
        }
        std::cout << "\n";
```

```
    Sort(data);

    SetConsoleTextAttribute(hConsole, 6);
    std::cout << "\nОтсортированный массив:\n";
    SetConsoleTextAttribute(hConsole, 7);
    std::ofstream out("output.txt");
    for (int& num : data) {
        std::cout << num << " ";
        out << num << " ";
    }
    std::cout << "\n";
    out.close();
    data.clear();

    SetConsoleTextAttribute(hConsole, 2);
    std::cout << "\nОтсортированный массив записан в файл 'output.txt'\n";

    ExitToMenu();
}

void Menu(std::vector<int>& data) {
    setlocale(LC_ALL, "Rus");
    int choice = 0;
    while (choice != 4) {
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "Меню:\n";
        SetConsoleTextAttribute(hConsole, 7);
        std::cout << "\n1.Ввод элементов массива вручную\n";
        std::cout << "2.Произвольный массив\n";
        std::cout << "3.Элементы массива из файла\n";
        SetConsoleTextAttribute(hConsole, 4);
        std::cout << "4.Выход\n";
        SetConsoleTextAttribute(hConsole, 6);
        std::cout << "\nВведите свой выбор: ";
        std::cin >> choice;

        switch (choice) {
        case 1:
            system("cls");
            ShellSort(data);
            break;
        case 2:
            system("cls");
            ShellSortRand(data);
            break;
        case 3:
            system("cls");
            ShellSortFromFile(data);
        case 4:
            system("cls");
            break;
        default:
            system("cls");
            SetConsoleTextAttribute(hConsole, 4);
            std::cout << "Некорректный выбор. Пожалуйста, введите цифру от 1 до 4
для выбора.\n\n";
            break;
        }
    }
}

int main() {
    std::vector<int> data;
```

```
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    Menu(data);

    return 0;
}
```