

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)

_____ Кучин Алексей Викторович

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/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)

_____ Кучин Алексей Викторович

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/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	2.07.24 – 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)

Кучин Алексей Викторович

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»

Кучин А.В. выполнял практическое задание «Быстрая сортировка». На первоначальном этапе была изучена реализация алгоритма сортировки, сделанные другим участником бригады. Была реализована функция подсчёта метрик алгоритма сортировки, проведено тестирование программы на наличие ошибок, багов, а так-же проверена производительность. Информация о найденных ошибках была передана коллегам для исправления. Оформил отчёт.

Бакалавр Кучин А.В. " ____ " _____ 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 (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ

(2023/2024 учебный год)

Кучин Алексей Викторович

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 Кучин А.В. решал следующие задачи: тестирование и проверка написанных участниками бригады функций, написание кода подсчёта статистики алгоритма сортировки.

За период выполнения практики был изучен алгоритм быстрой сортировки, а также основные случаи его применения. Во время выполнения работы Кучин А.В. показал себя крайне ответственным, добросовестным студентом, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и сортировке.

За выполнение работы Кучин А.В. заслуживает оценки «_____».

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

2024 г.

СОДЕРЖАНИЕ

Введение.....	2
1 Постановка задачи.....	3
1.1 Достоинства алгоритма сортировки вставками	3
1.2 Недостатки алгоритма сортировки вставками	3
1.3 Типичные сценарии применения данного алгоритма	4
2 Выбор решения.....	5
3 Описание программы.....	6
4 Схемы программы	8
5 Тестирование программы	10
6 Отладка.....	11
7 Совместная разработка	12
Заключение	14
Список используемой литературы.....	15
Приложение А. Листинг программы.....	16
Приложение Б. Результаты тестирования программы	21

Введение

Сортировка данных — это фундаментальная операция в информатике, которая позволяет оптимизировать поиск, визуализацию, обработку и сравнение данных. Она широко используется в IT, включая базы данных, поисковые системы, операционные системы, алгоритмы машинного обучения, компьютерную графику и обработку изображений. Сортировка данных позволяет упорядочить информацию для более эффективного поиска, обработки и анализа, что делает её ключевым инструментом во многих областях. Благодаря сортировке возможно находить нужные данные быстрее, визуализировать их более наглядно и ускорить выполнение многих алгоритмов.

QuickSort - это алгоритм сортировки принципа "разделяй и властвуй", который рекурсивно разбивает исходный список на два подсписка и сортирует их независимо, а затем объединяет полученные отсортированные под списки. Этот метод сортировки является одним из самых быстрых и эффективных.

Данный алгоритм был изобретен британским ученым Чарльзом Э. Хоаром в 1959 году, когда он работал в компании Elliott Brothers в Лондоне. Хоар занимался разработкой программного обеспечения для компьютера Elliott 803, и быстрая сортировка стала одним из его ключевых достижений в области компьютерных наук. Его изобретение легло в основу многих современных алгоритмов сортировки, а сам алгоритм QuickSort до сих пор активно применяется в различных сферах.

QuickSort широко используется в различных областях, включая обработку данных, базы данных, алгоритмы поиска и сортировки, а также в различных приложениях, где требуется эффективная сортировка данных. Он используется в операционных системах, системах управления базами данных, текстовых редакторах, компиляторах и других программных системах.

1 Постановка задачи

Поставленная задача: необходимо заполнить массив из n -ого количества элементов случайными числами, записать данные элементы в отдельный файл. После этого выполнить быструю сортировку над данными, находящимися в массиве, записать отсортированные данные в другой файл, посчитать время выполнения и количество перестановок значений массива при сортировке.

Использовать сервис GitHub для совместной работы. Создать и выложить коммиты, характеризующие действия, выполненные каждым участником бригады.

Оформить отчет по проведенной практике.

1.1 Достоинства алгоритма сортировки вставками

- Быстрая сортировка удобная для реализации программными средствами, независимо от выбранного языка программирования;
- операция легко разделяется на несколько отдельных процессов;
- быстрая сортировка является наиболее оптимальным решением в случае выполнения операций над массивом с последовательным доступом, когда отсутствует возможность перейти в начало в произвольный момент;
- считается самым быстрым алгоритмом сортировки.

1.2 Недостатки алгоритма сортировки вставками

- Алгоритм малоэффективен для небольших наборов данных;
- при наличии двух элементов с одинаковым ключом их порядок не будет соблюден;
- временная сложность составляет $O(n^2)$. Это происходит если выбран неудачный опорный элемент.

1.3 Типичные сценарии применения данного алгоритма

- В операционных системах применяется для сортировки очередей процессов, директорий файловой системы и различных внутренних структур данных;
- в базах данных применяется во время создания индексов, обработки запросов и различных внутренних операций;
- является алгоритмом сортировки по умолчанию в стандартных библиотеках языков Си++, Java, Python и Ruby;
- в компьютерной графике применяется для сортировки полигонов, пикселей и различных графических элементов во время процесса рендера и обработки;
- в сетевых маршрутизаторах для сортировки самих сетевых маршрутов и различных сетевых данных для их оптимизации.

2 Выбор решения

Нашей бригадой было выбрано вести разработку в среде Microsoft Visual Studio на языке C++.

Для написания данной программы будет использован язык программирования Си ++. Этот язык является распространённым языком программирования. C++ был создан в качестве расширения языка программирования С. Его первоначальная цель заключалась в том, чтобы сделать С более гибким и мощным языком, добавив возможности объектно-ориентированного программирования. Современный Си++ обладает объектно-ориентированными, универсальными и функциональными функциями в дополнение к средствам низкоуровневого манипулирования памятью. Указанные преимущества Си++ обеспечивают хорошее качество разработки почти любого вида программного продукта.

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Для удобства совместной разработки был выбран сервис GitHub. Это веб-сервис для хостинга IT-проектов и их совместной разработки. Сервис бесплатен для проектов с открытым исходным кодом и небольших частных проектов. Для крупных корпоративных проектов предлагаются различные платные тарифные планы.

3 Описание программы

При запуске программы появляется главное окно, обладающее следующим функционалом:

- генерация массива случайных чисел;
- чтение массива из файла;
- выбор размера массива для случайных чисел;
- сохранение результата в файл;
- выбор файла, из которого будет производиться чтение массива.
- запуск алгоритма сортировки.

На рисунке 1 представлено главное окно программы.

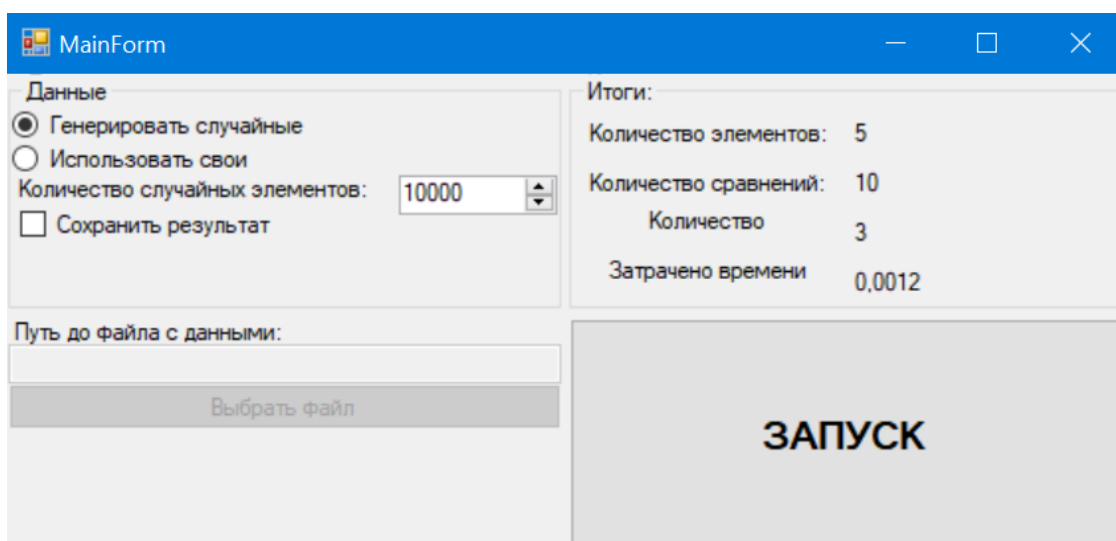


Рисунок 1 - Главное окно программы

Пользователь выбирает необходимый ему пункт, в случае выбора генерации случайного массива пользователю необходимо указать количество случайных элементов, как показано на рисунке 2.

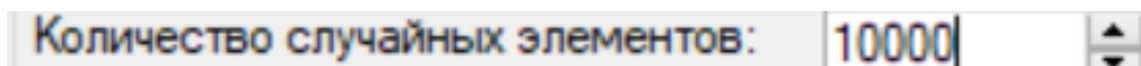


Рисунок 2 - Выбор размера массива случайных чисел

После того как данные были введены, генерируется массив указанной ранее длины из случайных чисел.

Далее над этими данными выполняется быстрая сортировка, при которой последний элемент массива сравнивается с остальными, постепенно «выстраивая стену», т.е. элементы младше перемещаются влево, а элементы старше остаются на своем месте.

После сортировки, в случае если был выбран пункт сохранение результата отсортированный массив записывается в указанный пользователем файл

Программа так же осуществляет подсчет количества перестановок элементов массива и времени, которое заняла сортировка.

Подробный алгоритм работы программы представлен в пункте 4 на рисунках 3 - 4.

Листинг программы приведен в приложении А.

4 Схемы программы

Подробный алгоритм работы программы представлен в на рисунке 3.

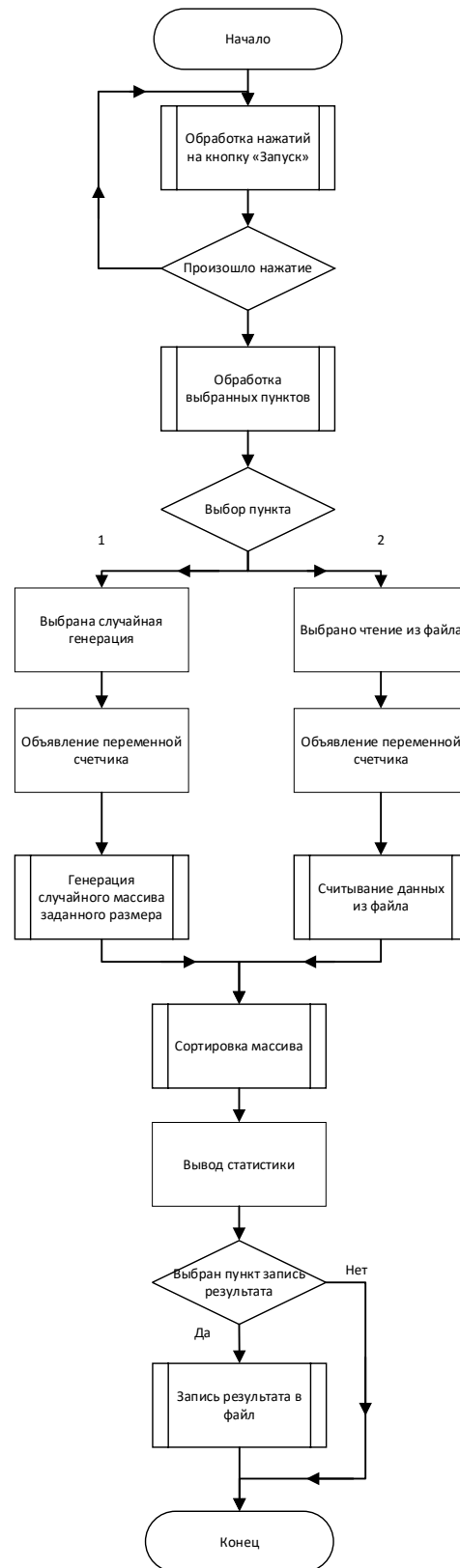


Рисунок 3 - блок-схема программы с учётом вывода метрик

Подробный алгоритм сортировки представлен в на рисунке 4.

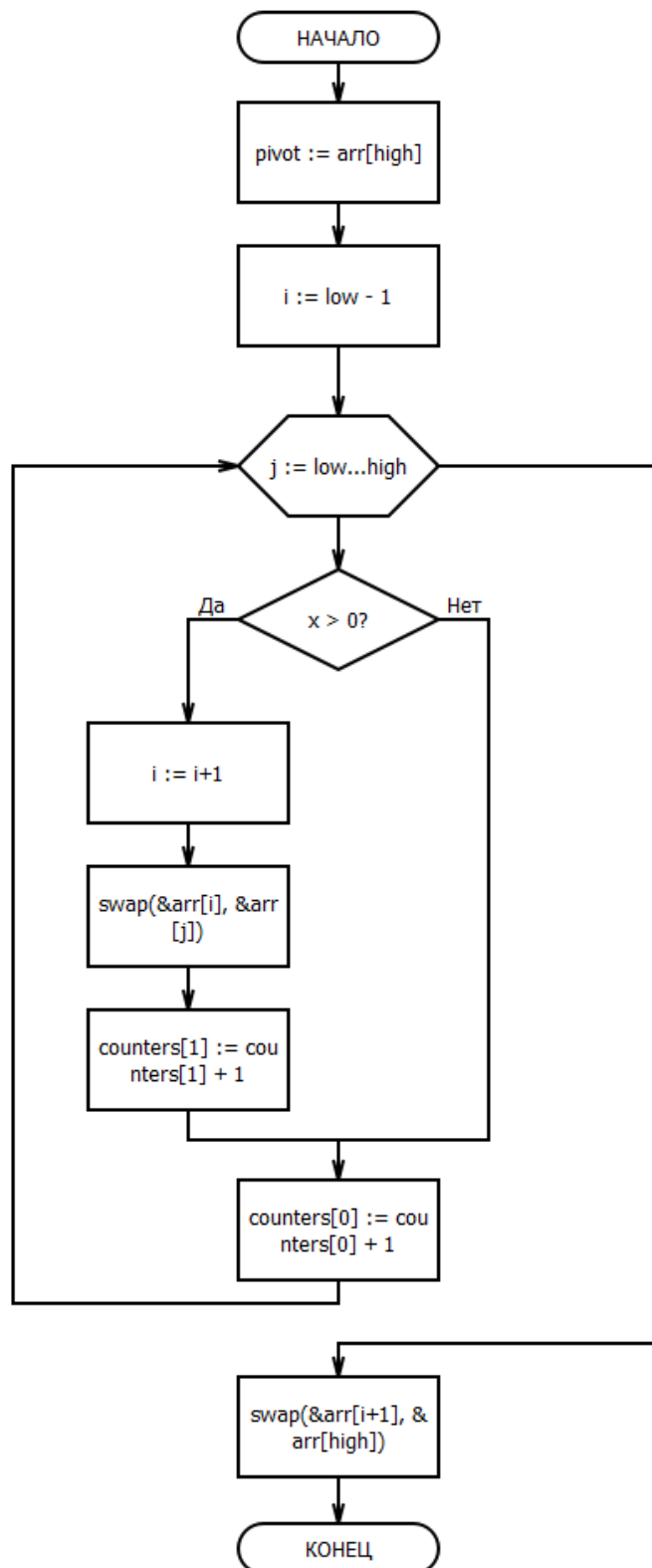
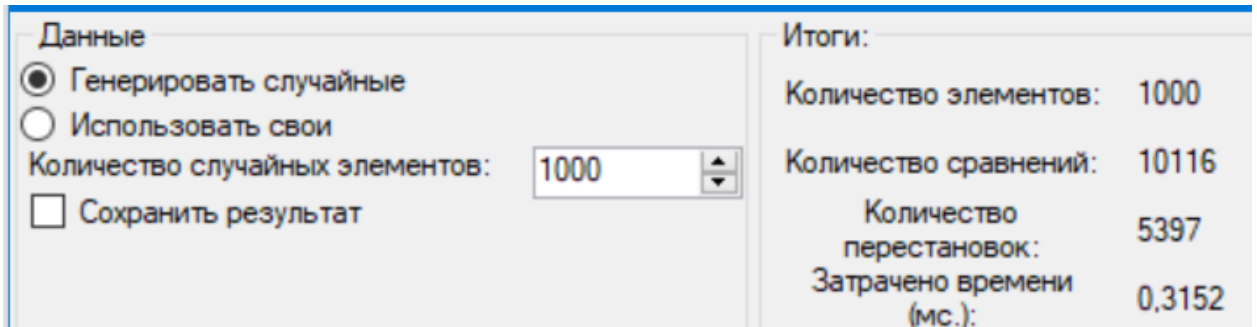


Рисунок 4 - Блок-схема алгоритма, выполняющего разбитие массива на подмассивы

5 Тестирование программы

Тестирование показало, что подсчёт перестановок, сравнений, а так-же кол-во элементов выводятся на экран пользователя верны.



Данные		Итоги:	
<input checked="" type="radio"/> Генерировать случайные		Количество элементов:	1000
<input type="radio"/> Использовать свои		Количество сравнений:	10116
Количество случайных элементов:	1000	Количество перестановок:	5397
<input type="checkbox"/> Сохранить результат		Затрачено времени (мс.):	0,3152

Рисунок 5 – Вывод метрик после работы программы

6 Отладка

В качестве среды разработки была выбрана программа Microsoft Visual Studio, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ.

Для отладки программы использовались средства, доступные в программе Microsoft Visual Studio, которая содержит всё необходимое для разработки и отладки программ.

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.

Также при отладке алгоритма сортировки использовалось средство Visual Studio для просмотра памяти, чтобы оценить правильность работы.

На рисунке 6 представлен процесс выполнения отладки.

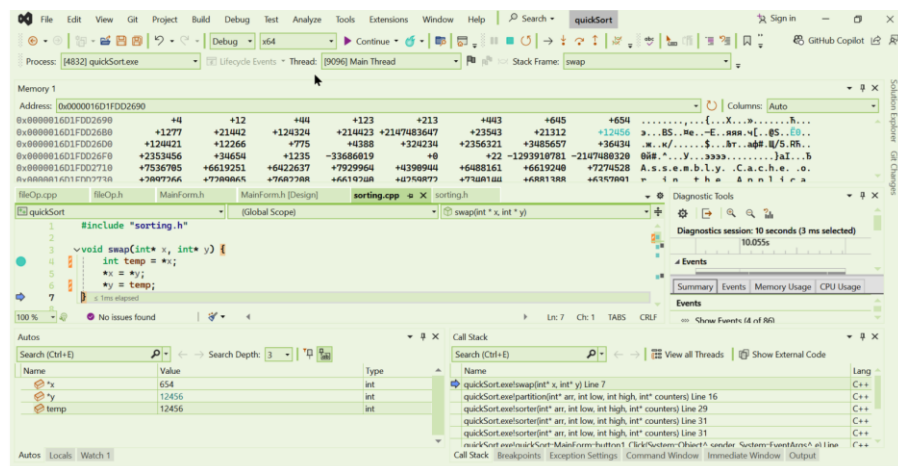


Рисунок 6 - Процесс отладки программы

7 Совместная разработка

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

Мною был создан модуль реализующий алгоритм быстрой сортировки, а также создан репозиторий в среде Visual Studio.

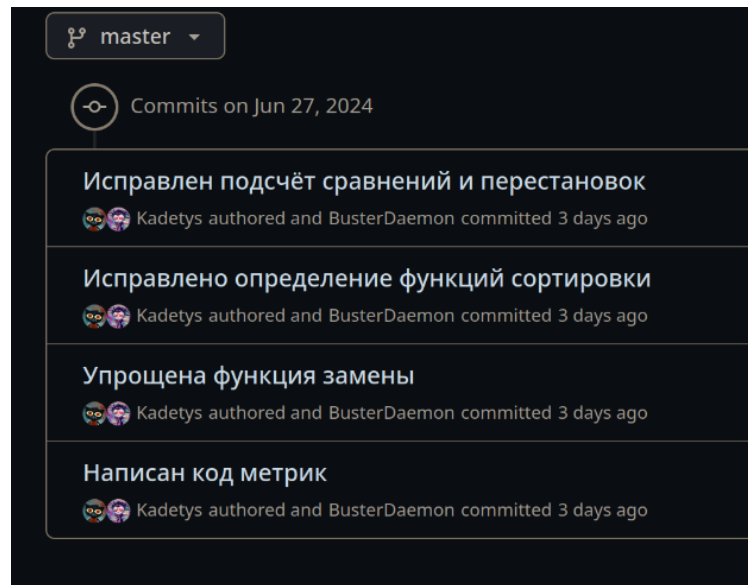


Рисунок 7 - Созданные коммиты

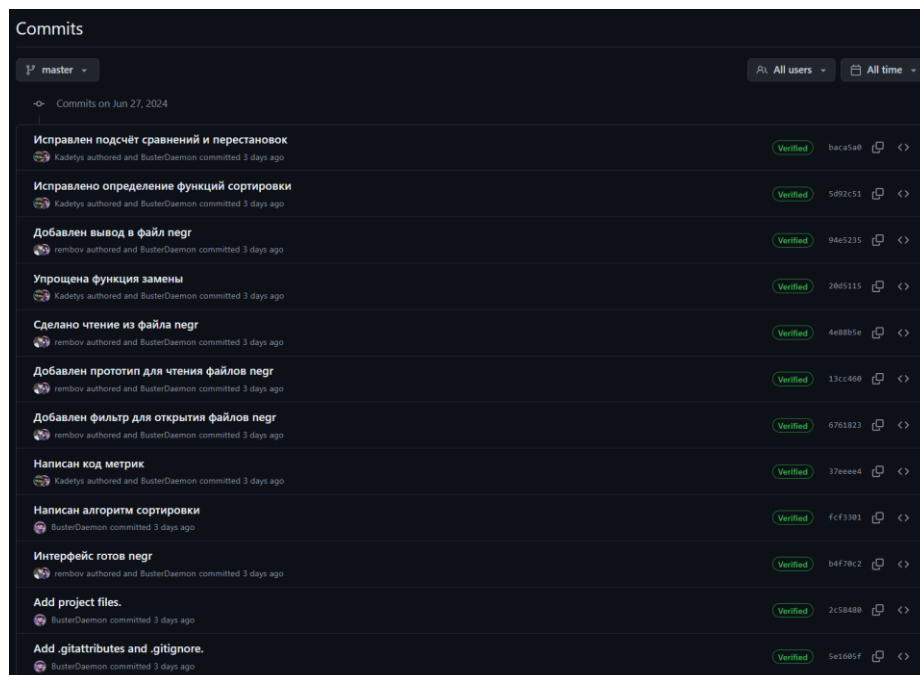


Рисунок 8 - Ветка master

Для управления репозиторием и последующей загрузкой данных на GitHub использовалась система контроля версий Git.

```
Admin@Best-Komp MINGW64 ~/Desktop/Git
$ git clone https://github.com/BusterDaemon/pguQuickSort.git
Cloning into 'pguQuickSort'...
remote: Enumerating objects: 66, done.
remote: Counting objects: 100% (66/66), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 66 (delta 35), reused 66 (delta 35), pack-reused 0
Receiving objects: 100% (66/66), 17.63 KiB | 376.00 KiB/s, done.
Resolving deltas: 100% (35/35), done.

Admin@Best-Komp MINGW64 ~/Desktop/Git
$ cd pguQuickSort

Admin@Best-Komp MINGW64 ~/Desktop/Git/pguQuickSort (master)
$ git commit
[detached HEAD a649867] Упрощена функция замены
Date: Wed Jun 26 10:02:30 2024 -0700
1 file changed, 3 insertions(+), 9 deletions(-)

Admin@Best-Komp MINGW64 ~/Desktop/Git/pguQuickSort (master)
$ git commit
[detached HEAD 213fccd] Исправлено определение функций сортировки
Date: Wed Jun 26 11:06:31 2024 -0700
1 file changed, 2 insertions(+), 2 deletions(-)

Admin@Best-Komp MINGW64 ~/Desktop/Git/pguQuickSort (master)
$ git push origin master
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 16 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (15/15), 2.12 KiB | 2.13 MiB/s, done.
Total 15 (delta 10), reused 8 (delta 6), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (10/10), completed with 5 local objects.
To https://github.com/BusterDaemon/pguQuickSort.git
+ 9329544...213fccd master -> master (forced update)

Admin@Best-Komp MINGW64 ~/Desktop/Git/pguQuickSort (master)
$
```

Рисунок 9 – Использование команд git commit и git push

Ссылка на удаленный репозиторий:
<https://github.com/BusterDaemon/pguQuickSort>.

Заключение

В ходе прохождения учебной практики были освоены навыки совместной разработки с использованием сервиса Github и системы контроля версий Git. Был изучен алгоритм быстрой сортировки.

Мною был создан репозиторий проекта с использованием среды Visual Studio и впоследствии выложен на Github. Также я реализовал алгоритм быстрой сортировки массива и подключил его к основной программе.

Также в ходе были улучшены навыки программирования на языке Си++, улучшены навыки отладки программ с использованием встроенных средств Visual Studio.

В дальнейшем программу можно улучшить путем оптимизации алгоритма с использованием многопоточного программирования встроенными библиотеками языка Си++.

Список используемой литературы

1. ГОСТ 19.701 – 90 Схемы алгоритмов, программ, данных и систем.
2. Гуриков, С. Р. Основы алгоритмизации и программирования на Visual C++ : учебное пособие / С.Р. Гуриков. — Москва : ИНФРА-М, 2022. — 515 с. — (Высшее образование: Бакалавриат). — DOI 10.12737/1039154. - ISBN 978-5-16-015500-5. - Текст : электронный. - URL: <https://znanium.com/catalog/product/1039154> (дата обращения: 27.06.2024). — Режим доступа: по подписке.
3. Царев, Р. Ю. Алгоритмы и структуры данных (CDIO): Учебник / Царев Р.Ю., Прокопенко А.В. - Краснояр.:СФУ, 2016. - 204 с.: ISBN 978-5-7638-3388-1. - Текст : электронный. - URL: <https://znanium.com/catalog/product/967108> (дата обращения: 29.06.2024). — Режим доступа: по подписке.
4. DSA Quicksort [Электронный ресурс] — Режим доступа: https://www.w3schools.com/dsa/dsa_algo_quicksort.php, свободный — Загл. с экрана (дата обращения 30.06.2024).

Приложение А. Листинг программы

Файл MainForm.h

```
private:                                                                    System::Void
radioButton1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
    if (radioButton1->Checked) {
        filePath->Enabled = false;
        selectFile->Enabled = false;
        dataQua->Enabled = true;
    }
}

private:                                                                    System::Void
radioButton2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
    if (radioButton2->Checked) {
        filePath->Enabled = true;
        selectFile->Enabled = true;
        dataQua->Enabled = false;
    }
}

private:    System::Void    selectFile_Click(System::Object^
sender, System::EventArgs^ e) {
    if (openDataFile->ShowDialog() ==
System::Windows::Forms::DialogResult::OK) {
        filePath->Text = openDataFile->FileName;
    }
}

private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
    if (radioButton2->Checked && filePath->Text == "") {
        MessageBox::Show("Укажите файл с данными!", "ОШИБКА");
        return;
    }
    if (radioButton1->Checked && dataQua->Value > 1) {
        // counter[0] - Счётчик сравнений
        // counter[1] - Счётчик замен
        unsigned long long counters[2] = { 0, 0 };
        int size = System::Decimal::ToInt32(dataQua->Value);
        int* arr = new int[size];
        std::srand(std::time(nullptr));

        for (int i = 0; i < size; i++) {
            arr[i] = std::rand();
        }

        sizeLbl->Text = size.ToString();
        auto a_time = std::chrono::system_clock::now();

        sorter(arr, 0, size-1, counters);
    }
}
```

```

        auto b_time = std::chrono::system_clock::now();

        std::chrono::duration<double,      std::ratio<1,      1000>>
durat = b_time - a_time;

        timeLbl->Text = durat.count().ToString();
        cmpLbl->Text = counters[0].ToString();
        swpLbl->Text = counters[1].ToString();

        if (saveData->Checked) {
            if (saveSortedData->ShowDialog() ==
System::Windows::Forms::DialogResult::OK) {
                char* fPath =
(char*)(void*)System::Runtime::InteropServices::Marshal::StringTo
oHGlobalAnsi(saveSortedData->FileName);
                writeFile(fPath, arr, &size);
            }
        }

        delete arr;
    }
    if (radioButton2->Checked) {
        int size = 0;
        char* fPath =
(char*)(void*)System::Runtime::InteropServices::Marshal::StringTo
oHGlobalAnsi(filePath->Text);
        int* arrRes = readFile(fPath, &size);
        if (arrRes == nullptr) {
            MessageBox::Show("Не могу обработать файл!",
"ОШИБКА");
            return;
        }
        unsigned long long counters[2] = { 0, 0 };
        sizeLbl->Text = size.ToString();

        auto a_time = std::chrono::system_clock::now();
        sorter(arrRes, 0, size - 1, counters);
        auto b_time = std::chrono::system_clock::now();
        std::chrono::duration<double,      std::ratio<1,      1000>>
durat = b_time - a_time;

        timeLbl->Text = durat.count().ToString();
        cmpLbl->Text = counters[0].ToString();
        swpLbl->Text = counters[1].ToString();

        if (saveData->Checked) {
            if (saveSortedData->ShowDialog() ==
System::Windows::Forms::DialogResult::OK) {
                char* fPath =

```

```

(char*) (void*) System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(saveSortedData->FileName);
        writeFile(fPath, arrRes, &size);
    }
}

delete arrRes;
}
}

```

Файл sorting.h

```

int partition(int[], int, int, unsigned long long*);
void sorter(int[], int, int, unsigned long long*);
void swap(int*, int*);

```

Файл fileOp.h

```

int* readFile(char*, int*);
void writeFile(char*, int*, int*);

```

Файл sorting.cpp

```

#include "sorting.h"

void swap(int* x, int* y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

int partition(int arr[], int low, int high, unsigned long long* counters) {
    int pivot = *(arr + high);
    int i = (low - 1);

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
            counters[1]++;
        }
        counters[0]++;
    }

    swap(&arr[i+1], &arr[high]);

    return (i + 1);
}

```

```

void sorter(int arr[], int low, int high, unsigned long long*
counters) {
    if (low < high) {
        int pi = partition(arr, low, high, counters);
        sorter(arr, low, pi - 1, counters);
        sorter(arr, pi + 1, high, counters);
    }
}

```

Файл fileOp.cpp

```

#include "fileOp.h"
#include <fstream>
#include <iostream>
#include <vector>

int* readFile(char* filePath, int* outSize) {
    std::ifstream file;
    file.open(filePath);
    if (!file.is_open()) {
        return nullptr;
    }

    char line[UINT16_MAX];
    file.getline(line, UINT16_MAX);
    auto arr = std::string(line);
    size_t pos = 0;
    std::vector<int> vec;

    while ((pos = arr.find(",")) != std::string::npos) {
        auto tok = arr.substr(0, pos);
        vec.insert(vec.end(), std::atoi(tok.c_str()));
        arr.erase(0, pos + 1);
    }
    file.close();
    int* result = new int[vec.size()];
    *outSize = vec.size();
    std::copy(vec.begin(), vec.end(), result);

    return result;
}

void writeFile(char* fPath, int* arr, int* arrSize) {
    std::ofstream file;
    file.open(fPath);
    if (!file.is_open()) {
        return;
    }
    for (int i = 0; i < *arrSize; i++) {

```



```
        file << arr[i] << " ";  
    }  
    file.close();  
    return;  
}
```

Приложение Б. Результаты тестирования программы

The screenshot shows the 'MainForm' application window. On the left, under the 'Данные' (Data) section, the 'Генерировать случайные' (Generate random) radio button is selected. The 'Количество случайных элементов' (Number of random elements) is set to 16384. Below this is a text field for the file path and a 'Выбрать файл' (Select file) button. On the right, the 'Итоги:' (Results) section displays the following statistics:

Итоги:	
Количество элементов:	16384
Количество сравнений:	296968
Количество перестановок:	124392
Затрачено времени (мс.):	1.93

A large 'ЗАПУСК' (Start) button is located at the bottom right of the window.

Рисунок Б.1

The screenshot shows the 'MainForm' application window. The 'Использовать свои' (Use my own) radio button is selected. The 'Количество случайных элементов' is set to 1000. The file path is 'C:\Users\busterd\Desktop\test_data.txt'. The 'Итоги:' section displays the following statistics:

Итоги:	
Количество элементов:	27
Количество сравнений:	117
Количество перестановок:	29
Затрачено времени (мс.):	0.8182

A large 'ЗАПУСК' (Start) button is located at the bottom right of the window.

Рисунок Б.2

The screenshot shows the 'MainForm' application window. The 'Генерировать случайные' radio button is selected. The 'Количество случайных элементов' is set to 20000000. The 'Сохранить результат' (Save result) checkbox is unchecked. The file path is empty. The 'Итоги:' section displays the following statistics:

Итоги:	
Количество элементов:	20000000
Количество сравнений:	6500321349
Количество перестановок:	196288577
Затрачено времени (мс.):	9571.6153

A large 'ЗАПУСК' (Start) button is located at the bottom right of the window.

Рисунок Б.3