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 (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКЕ**

(2023/2024 учебный год)

                                          Пичаев Иван Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 20.06.2024 по 17.07.2024

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 (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                   Пичаев Иван Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения                  1                 семестр                 2

Период прохождения практики с 20.06.2024 по 17.07.2024

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 | 26 | 20.06.24 –  24.06.24 |  |
| 2 | Подбор и изучение материала по теме работы | 26 | 24.06.24 –  26.06.24 |  |
| 3 | Установка виртуальной машины | 26 | 26.06.24 –  28.06.24 |  |
| 4 | Установка операционной системы | 30 | 01.07.24 –  04.07.24 |  |
| 5 | Разработка программы на языке Си | 30 | 04.07.24 –  09.07.24 |  |
| 6 | Тестирование и отладка | 38 | 09.07.24 –  12.07.24 |  |
| 7 | Оформление отчёта | 40 | 12.07.24 –  17.07.24 |  |
|  | **Общий объём часов** | 216 |  |  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЧЁТ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                            Пичаев Иван Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 20.06.2024 по 17.07.2024

Кафедра «Вычислительная техника»

Пичаев И.А. выполнял практическое задание «Шейкерная сортировка». На первоначальном этапе были изучен и проанализирован алгоритм шейкерной сортировки, был выбран метод решения и язык программирования С, на котором была написана программа шейкерной сортировки массива. Отладил программу. Оформил отчёт.

Бакалавр Пичаев И.А. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Руководитель Зинкин С.А.\_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

                                            Пичаев Иван Андреевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Прикладной искусственный интеллект»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 20.06.2024 по 17.07.2024

Кафедра «Вычислительная техника»

В процессе выполнения практики Пичаев И.А решал следующие задачи: анализ работы алгоритма шейкерной сортировки, создание программы выполняющая шейкерную сортировку, отладка программы.

За период выполнения практики были освоены основные понятия и технологии шейкерной сортировки. Во время выполнения работы Пичаев И. А. 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 Пичаев И. А. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики д.т.н., профессор, Зинкин С.А. « » 2024 г.

Содержание

Введение

1. Постановка задачи
2. Выбор решения
3. Описание программы
4. Схема программы
5. Тестирование программы
6. Отладка
7. Совместная разработка  
   Заключение

Список используемой литературы

Приложения

**Введение**

В настоящее время сортировка данных является одним из самых необходимых процессов обработки данных. Задачи по сортировке данных часто встречаются в различных профессиональных областях. Алгоритмы сортировки составляют отдельный класс алгоритмов, которые используются почти во всех задачах обработки информации. Они тесно связаны друг с другом и применяются для обеспечения более быстрого поиска.

Важность алгоритмов сортировки заключается в их способности организовывать и структурировать данные для более эффективного доступа и обработки. Правильный выбор алгоритма сортировки может значительно повысить производительность системы и сократить время выполнения операций.

Кроме того, алгоритмы сортировки играют ключевую роль в решении многих задач, таких как поиск, фильтрация, группировка, аналитика данных и т.д. Они помогают упорядочивать данные таким образом, чтобы было удобно работать с ними и извлекать из них нужную информацию.

Таким образом, понимание и применение алгоритмов сортировки является необходимым навыком для специалистов, занимающихся обработкой и анализом данных, и позволяет им эффективно управлять информационными процессами.

Шейкерная сортировка, также известная как коктейльная сортировка, является усовершенствованным методом пузырьковой сортировки. Она представляет собой эффективный алгоритм сортировки, который осуществляет сравнения и обмен элементов списка в обоих направлениях, уменьшая количество проходов по массиву. Этот метод становится особенно полезным для сортировки элементов в случае, когда массив частично отсортирован или количество перестановок на каждом шаге минимально.

**1 Постановка задачи**

Поставленная задача: необходимо заполнить массив из n-ого количества элементов числами, введенными с клавиатуры или случайными числами, записать данные элементы в отдельный файл. Либо заполнить массив с числами и заранее загруженного файла. После этого выполнить шейкерную 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. **Достоинства алгоритма сортировки вставками**

– алгоритм удобен для работы с массивами небольшого размера или на почти отсортированном наборе данных;

– Улучшение производительности по сравнению с пузырьковой сортировкой.

– Сокращение диапазона возможных замен за один проход, что уменьшает общее время работы.

**1.2 Недостатки алгоритма сортировки вставками**

– высокая алгоритмическая сложность О(n²);

–сортировка больших массивов занимает много времени.

**1.3 Типичные сценарии применения данного алгоритма**

– оптимизация других алгоритмов для повышения производительности;

– товары в магазине (сортировка по цене, весу);

– студенты в вузе (сортировка по среднему балу, кол-ву прогулов)

**2 Выбор решения**

Для написания данной программы будет использован язык программирования Си. Этот язык является распространённым языком программирования. При разработке языка Си был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. Си – это язык программирования общего назначения, хорошо известный своей эффективностью, экономичностью и переносимостью. Указанные преимущества Си обеспечивают хорошее качество разработки почти любого вида программного продукта.

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.

Для удобства совместной работы использовался мессенджер Discord. Он позволяет общаться как в голосовом, так и в текстовом чате. Это помогает коордировать действия всей бригады и эффективнее работать над проектом.

1. **Описание программы**

При запуске программы выводиться меню из 3 пунктов

1. Ввод значений массива с клавиатуры и сортировка их по возрастанию
2. Заполнение массива значениями из заранее созданного файла и его сортировка по возрастанию
3. Создание массива со случайными значениями и сортировка его по возрастанию

Пользователю требуется выбрать тот пункт, который ему требуется. При выборе пункта 1 пользователь вводит размер массива, а затем значения элементов. Значения записываются в файл list.txt.

printf("Введите размер массива: ");

if (scanf("%d", &n) != 1 || n <= 0) {

printf("Ошибка: введено некорректное значение. Пожалуйста, введите целое число.\n");

return 1;

}

mass = (int\*)malloc(n \* sizeof(int));

FILE\* createdFile = fopen("list.txt", "w");

printf("Введите элементы массива:\n");

for (int i = 0; i < n; i++) {

if (scanf("%d", &mass[i]) != 1) {

printf("Ошибка: введено некорректное значение. Пожалуйста, введите целое число.\n");

free(mass);

return 1;

}

fprintf(createdFile, "%d\n", mass[i]);

}

fclose(createdFile);

printf("\nМассив создан, сохранен в файле 'list.txt'");

При выборе пункта 2 пользователь вводит название файла со значениями, которые затем записываются в массив.

printf("Введите название файла: ");

scanf("%s", filename);

FILE\* file = fopen(filename, "r");

if (file == NULL) {

printf("Ошибка при открытии файла.\n");

return 1;

}

int number;

while (fscanf(file, "%d", &number) == 1) {

n++;

mass = (int\*)realloc(mass, n \* sizeof(int));

mass[n - 1] = number;

}

fclose(file);

printf("\nМассив создан");

При выборе пункта 3 пользователь вводит размер массива, который затем заполняется случайными значениями от -1000 до 1000. Значения записываются в файл list.txt.

printf("Введите размер массива: ");

if (scanf("%d", &n) != 1 || n <= 0) {

printf("Ошибка: введено некорректное значение. Пожалуйста, введите положительное целое число.\n");

return 1;

}

mass = (int\*)malloc(n \* sizeof(int));

srand(time(0));

FILE\* createdFile = fopen("list.txt", "w");

for (int i = 0; i < n; i++) {

mass[i] = (rand() % 2001) - 1000;

fprintf(createdFile, "%d\n", mass[i]);

}

fclose(createdFile);

printf("\nМассив создан, сохранен в файле 'list.txt'");

Далее выполняется шейкерная сортировка, при котором наибольшие элементы массива перемещаются вправо, а наименьшие влево. Сначала элемент начиная с левого края сравнивается со следующим и если он больше, то значения меняются позициями, а если меньше остается на своей позиции, а следующий элемент начинает сдвигаться. Так наибольший элемент массива попадает в конец массива, и правая граница сдвигается на один, чтобы не учитывать уже отсортированный элемент. Аналогично в левый край сдвигается наименьший элемент массива, и левая граница также сдвигается на один элемент. Так происходит до тех пор, пока границы не встретятся или пока в цикле есть перемещения. Если перемещений не было алгоритм заканчивается досрочно.

int swapped = 1;

int start = 0;

int end = n - 1;

while (swapped) {

swapped = 0;

for (int i = start; i < end; i++) {

if (mass[i] > mass[i + 1]) {

swap(&mass[i], &mass[i + 1]);

swapped = 1;

(\*count)++;

}

}

if (!swapped)

break;

swapped = 0;

end--;

for (int i = end - 1; i >= start; i--) {

if (mass[i] < mass[i - 1]) {

swap(&mass[i], &mass[i - 1]);

swapped = 1;

(\*count)++;

}

}

start++;

}

После этого отсортированный массив записывается в файл sorted\_list.txt.

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

Листинг программы приведен в приложении A.

**4 Схемы программы**

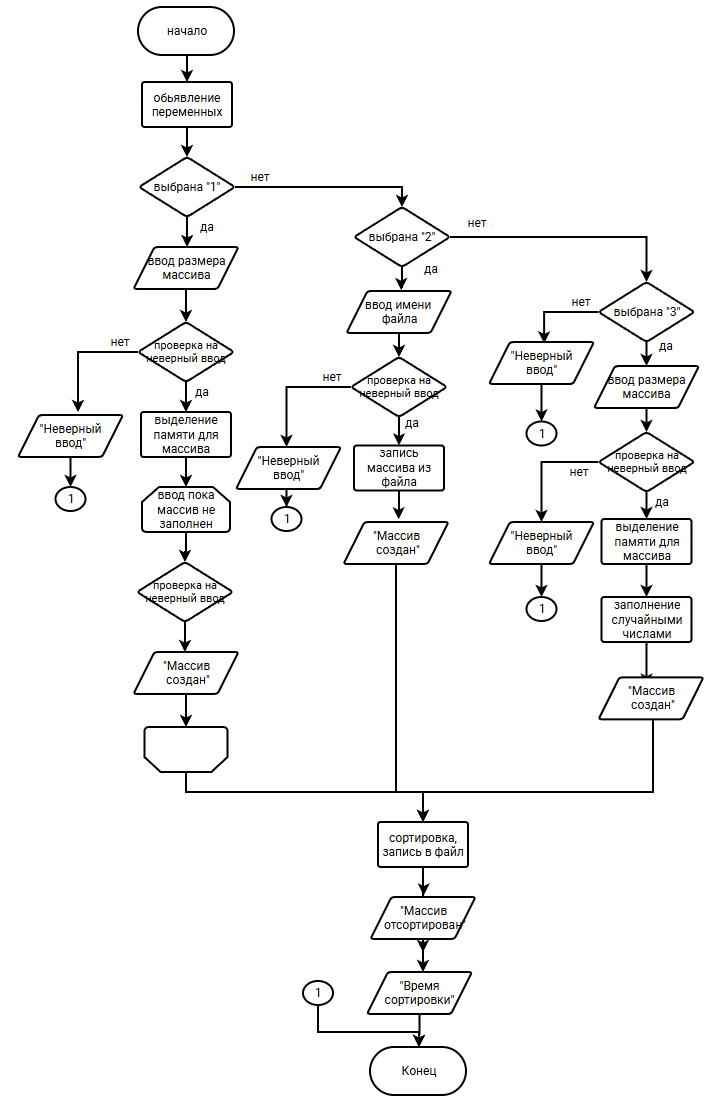


Рисунок 1 – Блок-схема программы

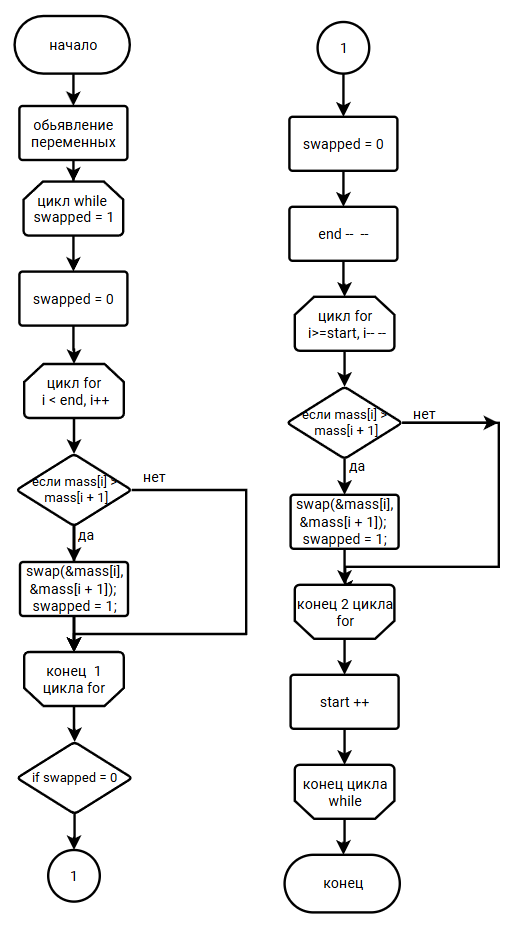


Рисунок 2 – Блок-схема алгоритма

**5 Тестирование программы**

На основании анализа данных, полученных в результате тестирования алгоритма шейкерной сортировки, можно сделать вывод, что продолжительность выполнения программы прямо пропорциональна числу обрабатываемых элементов. Другими словами, при росте объема данных время, необходимое для сортировки, возрастает в линейной зависимости.

Рисунок 3 – 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.

**7 Совместная работа**

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

Мною была написана программа сортировки, она была загружена на удаленный репозиторий Github, на ветку main.

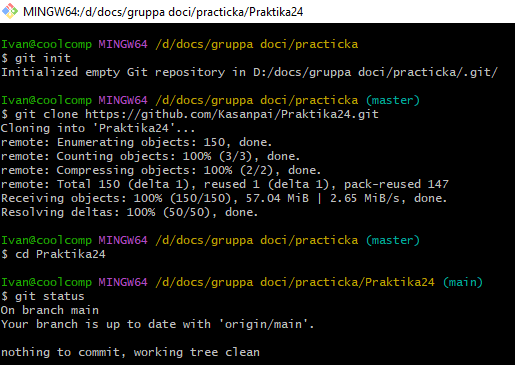


Рисунок 4 – Инициализация репозитория и загрузка файлов с удаленного репозитория

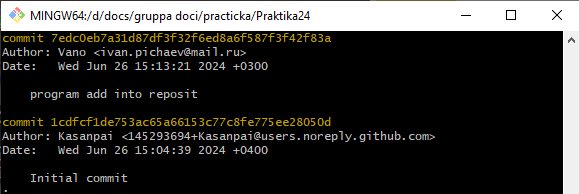


Рисунок 5 – Загрузка программы на удаленный репозитория

Для удобства совместной работы использовался мессенджер Discord. Он позволяет общаться как в голосовом, так и в текстовом чате, а также пересылать файлы. Это помогает коордировать действия всей бригады и эффективнее работать над проектом.

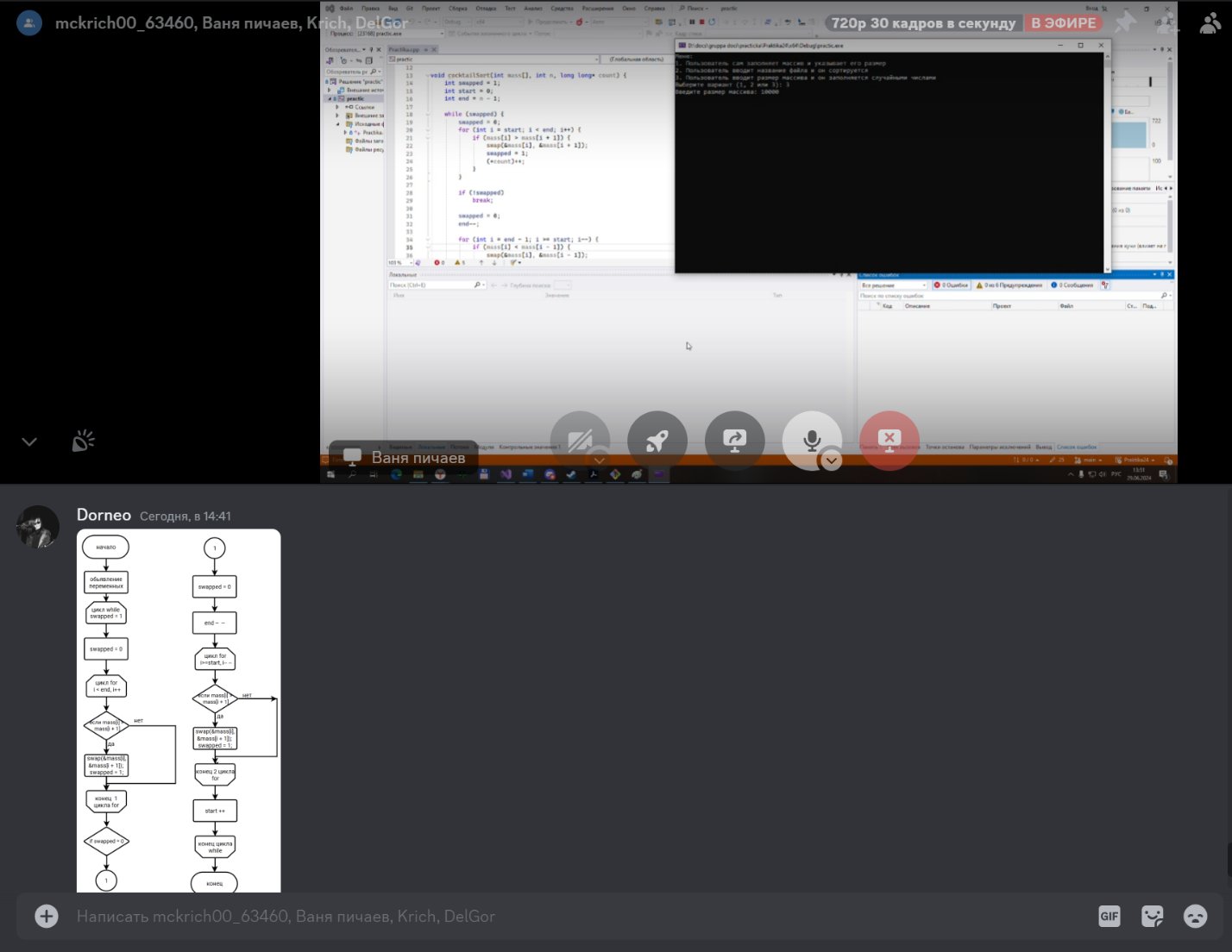


Рисунок 6 – Совместная работа в Discord

**Заключение**

**Список 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. Сортировка вставками [Электронный ресурс] – URL: https://ru.wikipedia.org (дата обращения: 02.07.2023 г)

**Приложение A. Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <Windows.h>

void swap(int\* start, int\* end) {

int elem = \*start;

\*start = \*end;

\*end = elem;

}

void cocktailSort(int mass[], int n, long long\* count) {

int swapped = 1;

int start = 0;

int end = n - 1;

while (swapped) {

swapped = 0;

for (int i = start; i < end; i++) {

if (mass[i] > mass[i + 1]) {

swap(&mass[i], &mass[i + 1]);

swapped = 1;

(\*count)++;

}

}

if (!swapped)

break;

swapped = 0;

end--;

for (int i = end - 1; i >= start; i--) {

if (mass[i] < mass[i - 1]) {

swap(&mass[i], &mass[i - 1]);

swapped = 1;

(\*count)++;

}

}

start++;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

printf("Меню:\n");

printf("1. Пользователь сам заполняет массив и указывает его размер\n");

printf("2. Пользователь вводит название файла и он сортируется\n");

printf("3. Пользователь вводит размер массива и он заполняется случайными числами\n");

printf("Выберите вариант (1, 2 или 3): ");

scanf("%d", &choice);

int n = 0;

int\* mass = NULL;

long long count = 0;

if (choice == 1) {

printf("Введите размер массива: ");

if (scanf("%d", &n) != 1 || n <= 0) {

printf("Ошибка: введено некорректное значение. Пожалуйста, введите целое число.\n");

return 1;

}

mass = (int\*)malloc(n \* sizeof(int));

FILE\* createdFile = fopen("list.txt", "w");

printf("Введите элементы массива:\n");

for (int i = 0; i < n; i++) {

if (scanf("%d", &mass[i]) != 1) {

printf("Ошибка: введено некорректное значение. Пожалуйста, введите целое число.\n");

free(mass);

return 1;

}

fprintf(createdFile, "%d\n", mass[i]);

}

fclose(createdFile);

printf("\nМассив создан, сохранен в файле 'list.txt'");

}

else if (choice == 2) {

char filename[100];

printf("Введите название файла: ");

scanf("%s", filename);

FILE\* file = fopen(filename, "r");

if (file == NULL) {

printf("Ошибка при открытии файла.\n");

return 1;

}

int number;

while (fscanf(file, "%d", &number) == 1) {

n++;

mass = (int\*)realloc(mass, n \* sizeof(int));

mass[n - 1] = number;

}

fclose(file);

printf("\nМассив создан");

}

else if (choice == 3) {

printf("Введите размер массива: ");

if (scanf("%d", &n) != 1 || n <= 0) {

printf("Ошибка: введено некорректное значение. Пожалуйста, введите положительное целое число.\n");

return 1;

}

mass = (int\*)malloc(n \* sizeof(int));

srand(time(0));

FILE\* createdFile = fopen("list.txt", "w");

for (int i = 0; i < n; i++) {

mass[i] = (rand() % 2001) - 1000;

fprintf(createdFile, "%d\n", mass[i]);

}

fclose(createdFile);

printf("\nМассив создан, сохранен в файле 'list.txt'");

}

else {

printf("Неверный выбор. Попробуйте еще раз.\n");

return 1;

}

time\_t start = clock();

cocktailSort(mass, n, &count);

time\_t end = clock();

double time = ((double)(end - start)) / 1000;

FILE\* sortedFile = fopen("sorted\_list.txt", "w");

for (int i = 0; i < n; i++) {

fprintf(sortedFile, "%d\n", mass[i]);

}

fclose(sortedFile);

printf("\nМассив отсортирован, сохранен в файле 'sorted\_list.txt'");

printf("\n\nВремя сортировки: %f секунд\n", time);

printf("Количество перестановок: %lld\n", count);

free(mass);

return 0;

}