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**

(2024/2025 учебный год)

                                            Свинухов Николай Дмитриевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.25 по 08.07.25

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**

(2024/2025 учебный год)

                                              Свинухов Николай Дмитриевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения                  1                 семестр                 2

Период прохождения практики с 25.06.25 по 08.07.25

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 | 25.06.2025 -  25.06.2025 |  |
| 2 | Подбор и изучение материала по теме работы | 15 | 26.06.2025 –  28.06.25 |  |
| 3 | Разработка алгоритма | 43 | 01.07.25 –  03.07.25 |  |
| 4 | Описание алгоритма и  программы | 18 | 03.07.25 –  04.07.25 |  |
| 5 | Тестирование | 5 | 04.07.25 –  05.07.25 |  |
| 6 | Получение и анализ результатов | 10 | 05.07.25 –  08.07.25 |  |
| 7 | Оформление отчёта | 15 | 05.07.25 –  08.07.2025 |  |
|  | **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**

(2024/2025 учебный год)

                                               Свинухов Николай Дмитриевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.25 по 08.07.25

Кафедра «Вычислительная техника»

В процессе выполнения практики Свинухов Н.Д. решал следующие задачи: создание алгоритма сортировки Шелла, анализ работы алгоритма, рандомный и ручной ввод чисел.

За период выполнения практики были освоены основные понятия и технологии сортировки Шелла, реализован метод работы с файлами. Во время выполнения работы Свинухов Н.Д.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 Агафонов Е. А. заслуживает оценки «\_\_\_\_\_\_».

Бакалавр           Свинухов Н.Д.   \_\_\_\_\_\_\_\_\_\_\_\_     "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_  2025 г.

Руководитель   Зинкин С.А. \_ \_\_\_\_\_\_\_\_\_\_\_     "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_  2025 г.

практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ  УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2024/2025 учебный год)

                                             Свинухов Николай Дмитриевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

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

Год обучения1семестр 2

Период прохождения практики с 25.06.25 по 08.07.25

Кафедра «Вычислительная техника»

В процессе выполнения практики Свинухов Н.Д. решал следующие задачи: создание алгоритма сортировки Шелла, анализ работы алгоритма, рандомный и ручной ввод чисел.

За период выполнения практики были освоены основные понятия и технологии сортировки Шелла, реализован метод работы с файлами. Во время выполнения работы Свинухов Н.Д. 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 Агафонов Е. А. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики д.т.н., профессор, Зинкин С.А. «    »                        2025 г.

**Содержание**

[Введение 7](#_Toc202752118)

[1 Постановка задачи 8](#_Toc202752119)

[1.1 Достоинства алгоритма сортировки Шелла: 8](#_Toc202752120)

[1.2 Недостатки алгоритма сортировки Шелла: 9](#_Toc202752121)

[2 Выбор решения 10](#_Toc202752122)

[3 Описание программы 11](#_Toc202752123)

[4 Схема программы 14](#_Toc202752124)

[4.1 Блок-схема программы 14](#_Toc202752125)

[4.2 Блок-схема алгоритма 15](#_Toc202752126)

[5 Тестирование программы 16](#_Toc202752127)

[5.1 Тестирование на разных наборах данных 16](#_Toc202752129)

[5.2 Анализ полученных результатов тестирования 16](#_Toc202752130)

[5.3 Тестирование отрицательных чисел. 17](#_Toc202752131)

[Заключение 18](#_Toc202752135)

[Список литературы 19](#_Toc202752136)

[Приложение А Листинг 20](#_Toc202752137)

# Введение

Сортировка данных является важной частью современного мира. Сейчас, когда огромные потоки различной информации повсеместно встречаются в обычной жизни, не говоря уже о работе в самых разных сферах или учёбе, очень важно владеть способностью управлять разными объёмами данных для упрощения их считывания и усваивания.

Есть большое множество различных алгоритмов сортировки данных, их более детальное изучение и осваивание может помочь развить логику, мышление человека, а так же подбор разных видов сортировок может помочь программистам в будущем не только понимать различные принципы сортировки, но и позволит находить более быстрые и эффективные для конкретных задач и создание программы, которая будет выполнять сортировку в качестве практики является важным элементов изучения и совершенствования навыков программирования. Как правило, термин сортировка понимают, как процесс перестановки объектов некоторого множества в определенном порядке.

Сортировка Шелла – хороший алгоритм, который подходит для больших объёмов данных. Этот вид сортировки является усложненной версией сортировок вставками, однако решающий одну важную проблему этого вида сортировок. Сортировка вставками является относительно простым видом сортировки, её алгоритм достаточно понятен, но для больших объёмов данных этот метод может оказаться достаточно долгим. Сортировка Шелла сложнее вставок, делая сортировку по группам и шагам, но является более быстрой и не менее эффективной альтернативой сортировки вставками для больших объёмов данных.

1. **Постановка задачи**

Необходимо реализовать и продемонстрировать работу алгоритмов сортировки. Конечная программа должна реализовывать непосредственно алгоритм, а также выполнять считывание входных данных из файла и запись результатов в файл и отображать все данные для пользователя.

Форматы входных и выходных файлов определяются произвольно (в основном используется .txt). Программа должна обрабатывать данные, предоставленные в корректно сформированных входных файлах. Если данные в файле некорректны, то программа должна выводить соответствующее сообщение.

Конечные программы не должны использовать реализации алгоритмов сортировки, предоставляемые стандартными или сторонними библиотеками, включая системные.

В случае наличия реализации требуемого алгоритма в стандартных библиотеках языка программирования выполнить сравнение производительности реализованного алгоритма со стандартным. Сделать выводы.

В качестве результатов выполнения работы предоставляются: отчет с описанием реализованного алгоритма, исполняемая программа, включая ее исходный код, наборы тестовых данных (в зависимости от варианта).

Анализ работы алгоритма должен выполняться на нескольких различных наборах данных.

Отчет должен содержать следующие элементы: титульный лист, индивидуальный план, лист отчета с краткими итогами, отзыв руководителя, краткое описание алгоритма, блок-схема алгоритма, блок-схема программы, преимущества и недостатки алгоритма, непосредственная реализация алгоритма без использования стандартных или сторонних библиотек.

* 1. Достоинства алгоритма сортировки Шелла:
* эффективность для массивов, в которых элементы изначально сильно перемешаны;
* относительная простота реализации (сложнее сортировки вставками, но всё ещё достаточно простой);
* скорость сортировки.
  1. Недостатки алгоритма сортировки Шелла:
* зависимость от шагов (из-за них сортировка может оказаться менее быстрой или эффективной);
* проигрыш по скорости сложным алгоритмам на больших данных;
* использование постоянного пространства для вспомогательных переменных (данные, сравниваемые через шаги).

1. **Выбор решения**

В качестве языка программирования для выполнения данной работы будет использоваться C. Это достаточно простой язык программирования, что при этом не отменяет его распространённости и эффективности. Язык C является достаточно изученным из курса программирования, а также оптимальным вариантом для создания кода программы, которая сможет выполнить алгоритм сортировки, в частности нужный конкретно для это работы сортировки Шелла. Достаточно хорошее понимание данного языка, а также его простота и эффективность позволяют хорошо создавать различные программные продукты и сделает C незаменимым при выполнении данной практической работы.

В качестве среды программирования была выбрана программа Microsoft Visual Studio. Microsoft Visual Studio — это программная среда по разработке приложений, она достаточно популярна и проста в использовании и позволяет использовать вышеупомянутый язык программирования C для создания программы в благоприятных условиях разработки.

Для удобства совместной работы будет использован GitHub, где каждый участник сможет выложить свою часть работы и заняться своей частью задачи. Так же в GitHub каждый участник работы может сохранить отчёт с его работой и ролью. Популярность, простота и эффективность данной платформы делают её хорошим выбором для командной работы, распределению обязанностей и эффективному выполнению поставленных задач в бригаде.

1. **Описание программы**

При запуске программы выходят пункты меню.

1. Создание массива ручным вводом
2. Создание массива случайных чисел (в заданном диапазоне)
3. Сама сортировка Шелла
4. Сохранение массива в файл
5. Загрузка массива из файла
6. Выход

Пользователю нужно выбрать пункт, который ему требуется. При выборе пункта 1 или 2 будет выведен запрос на количество чисел в массиве.

printf("Размер массива: ");

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

При этом программа не только запрашивает количество чисел в массиве, но и проверяет на корректность ввода. Если число некорректно (буква или отрицательное или 0), то пользователю будет выведена строка о некорректности ввода и цикл (выбор в меню) очиститься и позволит заново выбрать пункт меню.

while (getchar() != '\n');

continue;

В случае 1 значение size, которое содержит количество чисел массива, перейдёт в функцию create\_array\_manual, в которой можно будет набрать нужные числа.

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

while (scanf\_s("%d", &arr[i]) != 1) {

printf("Ошибка! Введите целое число: ");

while (getchar() != '\n'); // Очистка буфера.

}

}

В этой функции так же происходит проверка на корректность ввода. В случае, если пользователь выберет 2, то ему так же нужно будет указать минимум и максимум (диапазон чисел).

srand((unsigned int)time(NULL));

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

arr[i] = rand() % (max - min + 1) + min;

}

return arr;

}

Программа рандомно сгенерирует в нужном диапазоне нужное количество чисел. Если пользователь выберет 3, то последний созданный массив будет отсортирован по алгоритму Шелла (если массив не создан или не загружен, то это вызовет ошибку).

for (int gap = size / 2; gap > 0; gap /= 2) {

for (int i = gap; i < size; i++) {

int temp = arr[i];

int j;

// Сдвигаем элементы.

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {

arr[j] = arr[j - gap];

res.swaps++; // Считаем перестановки.

}

arr[j] = temp;

}

}

Где так же идёт подсчёт количества перестановок. В этой функции и идёт подсчёт времени.

res.time\_mcs = ((double)(clock() - start)) / CLOCKS\_PER\_SEC \* 1000000;

if (res.time\_mcs < 0.000001) res.time\_mcs = 0.000001; // Гарантируем не ноль.

return res;

Для времени и количества перестановок создана структура в заголовочном файле.

4 позволяет сохранить массив в файл (используются 2 функции: save\_array\_to\_file и save\_results), а 5 позволяет загрузить массив из файла и работать с ним (большая функция load\_array\_from\_file). 6 – позволяет завершить программу и выйти из цикла (break).

1. **Схема программы**
   1. Блок-схема программы

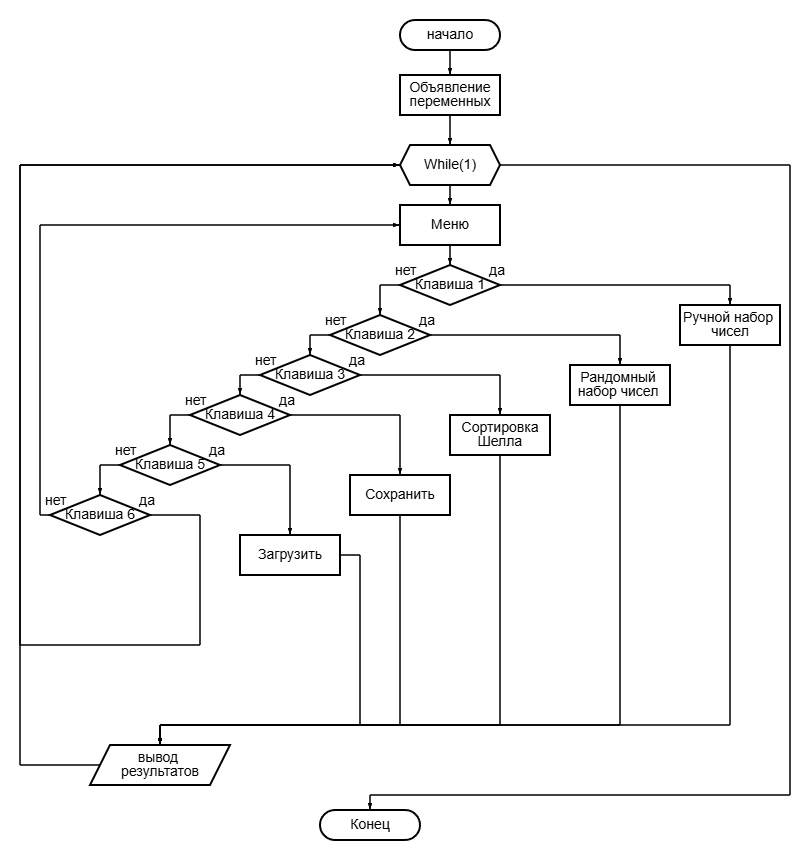


Рисунок 1 – Блок-схема программы

* 1. Блок-схема алгоритма

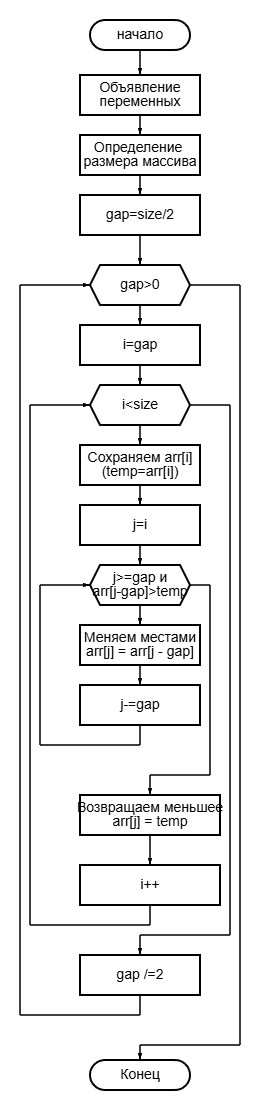


Рисунок 2 – Блок-схема алгоритма

1. **Тестирование программы**

Подробным проведением тестирования программы занимался Свинухов Н.Д.

Программа позволяет пользователю создать массив вручную или случайным образом, сохранить его в файл, загрузить из файла и отсортировать по алгоритму Шелла. Также выводится информация о количестве перестановок и времени сортировки.

5.1 Тестирование на разных наборах данных

Тестирование проводилось на массивах размером от 10 000 до 100 000 элементов. Изменялось только количество элементов в массиве. Массивы заполнялись случайными числами в диапазоне от -10000 до 10000.

Пример запуска автоматической генерации:

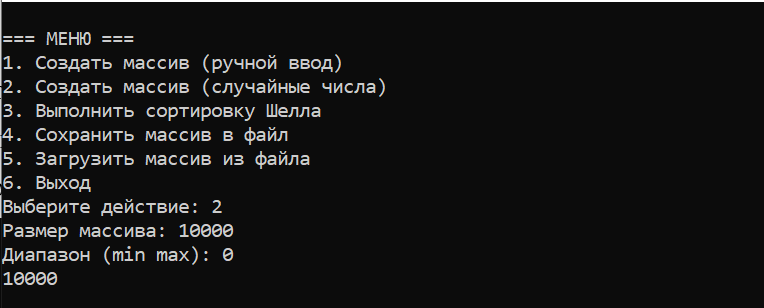


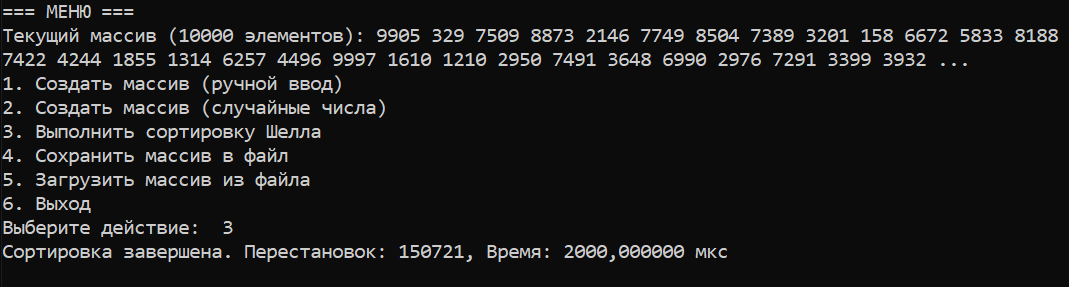
Рисунок 3 – Создание массива из 10000 элементов  


Рисунок 4 – Время сортировки массива из 10000 элементов.

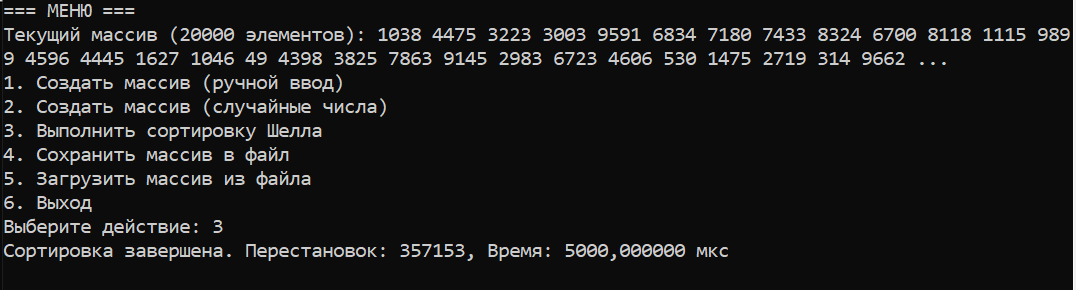


Рисунок 5 – Время сортировки массива из 20000 элементов.

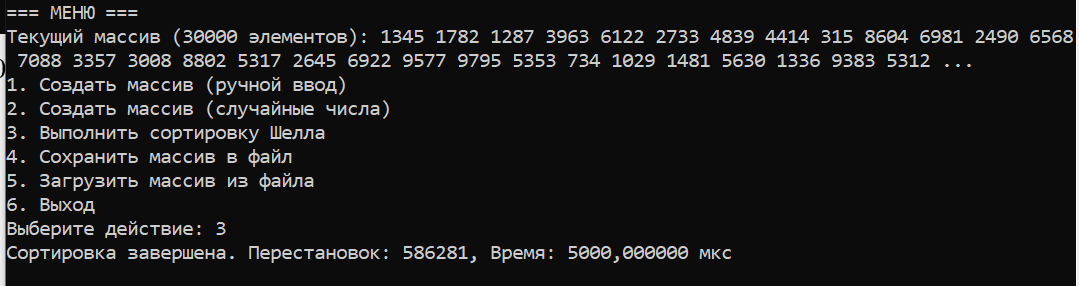


Рисунок 6 – Время сортировки массива из 30000 элементов.

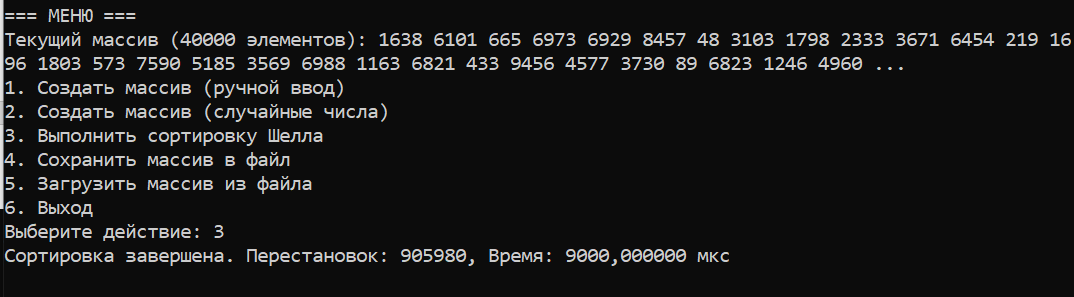


Рисунок 7 – Время сортировки массива из 40000 элементов.

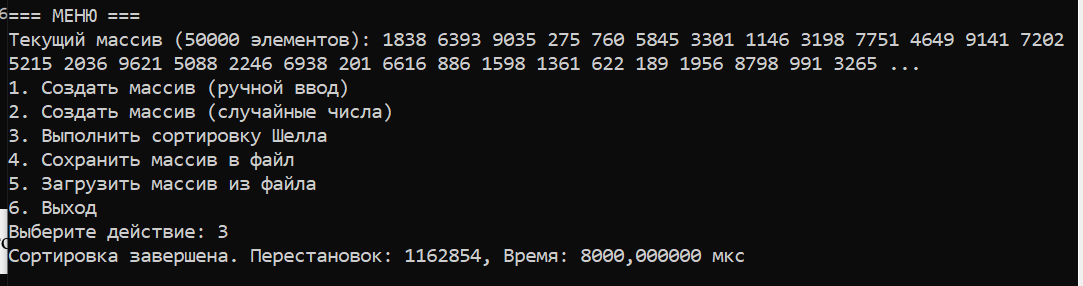


Рисунок 8 – Время сортировки массива из 50000 элементов.

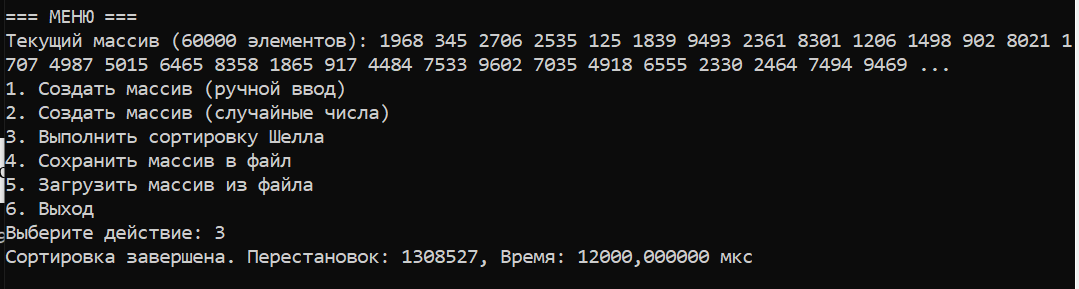


Рисунок 9 – Время сортировки массива из 60000 элементов.

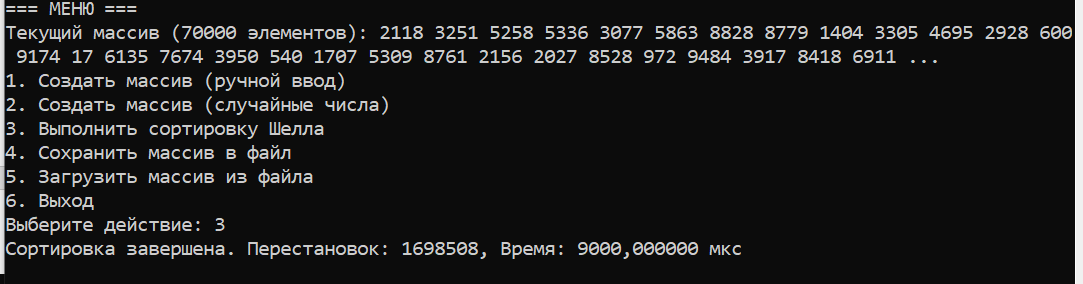


Рисунок 10 – Время сортировки массива из 70000 элементов.

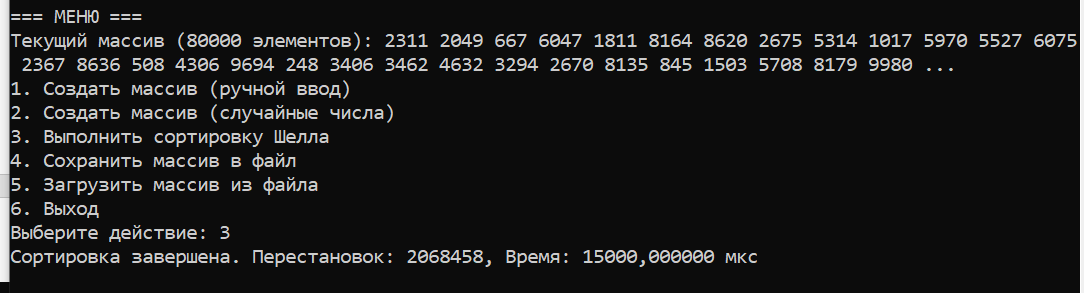


Рисунок 11 – Время сортировки массива из 80000 элементов.

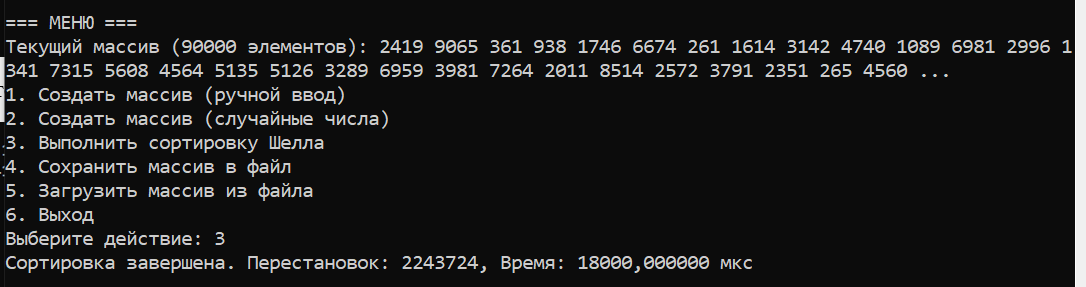


Рисунок 12 – Время сортировки массива из 90000 элементов.

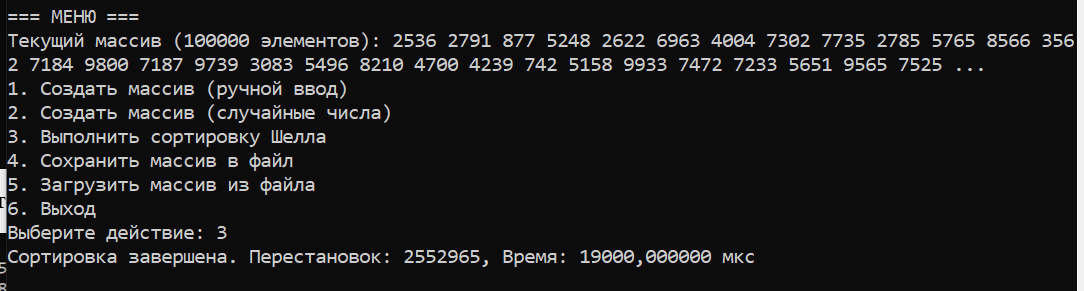


Рисунок 13 – Время сортировки массива из 100000 элементов.

В таблице 1 показаны результаты тестирования.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Размерность массива | Перестановки | Время сортировки (мкс) |
| 1 | 10000 | 150721 | 2000.0 |
| 2 | 20000 | 357153 | 5000.0 |
| 3 | 30000 | 586281 | 5000.0 |
| 4 | 40000 | 905980 | 9000.0 |
| 5 | 50000 | 1162854 | 8000.0 |
| 6 | 60000 | 1308527 | 12000.0 |
| 7 | 70000 | 1698508 | 9000.0 |
| 8 | 80000 | 2068458 | 15000.0 |
| 9 | 90000 | 2243724 | 18000.0 |
| 10 | 100000 | 2552965 | 19000.0 |

**5.2 Анализ полученных результатов тестирования**

Из результатов видно, что время выполнения сортировки и количество перестановок увеличиваются с ростом количества элементов. Однако скорость увеличения остается ниже, чем у простых алгоритмов вроде сортировки вставками. Это подтверждает эффективность алгоритма Шелла для массивов среднего размера. При этом он всё же уступает по скорости более современным алгоритмам на больших объемах данных.

Рисунок 14 – График зависимости времени сортировки от количества элементов

Рисунок 15 – График зависимости количества перестановок от количества элементов

5.3 Тестирование отрицательных чисел.

Тестирование проводилось на минимальном значении «-10000» и максимальным значением «0».

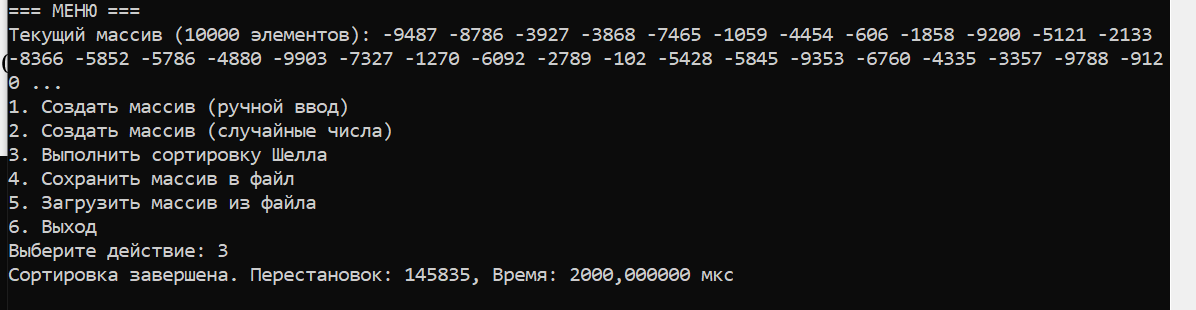


Рисунок 16 – Сортировка отрицательных чисел.

Итак, мы можем видеть, что при сортировке не важно является число положительным или отрицательным.

# Заключение

При выполнении данной работы были получены навыки работы в команде, координации и реализации проекта путём использования GitHub, закреплены знания использования Git.

Был изучен сам алгоритм Шелла, выявлены его преимущества и недостатки, принцип работы и теперь имеется навык повторение алгоритма в будущих работах.

Так же была создана многомодульная программа, которая позволяет создать массив чисел, работать с файлами и выполнять саму сортировку Шелла, что так же улучшило понимание языка C, закрепила навыки работы с программой.

В дальнейшем опыт данной работы можно будет использовать при написании других сложных программ, где нужно будет организовать сортировку каких-либо объёмов данных.

Ссылка на репозиторий: https://github.com/westnikst/Practice.git

# Список литературы

1. Алгоритмы. Просто как дважды два / И.В. Красиков, И.Е. Красикова
2. Языки программирования / А.А. Тюгашев / Учебное пособие, 2018 г.
3. Язык программирования С++ / Стивен Прата / 6-е издание, 2015г
4. Искусство программирования / Дональд Кнут / Сортировка и поиск, 3 том, 1998г
5. Wikipedia.org [Электронный ресурс] // Wikipedia URL: https://ru.wikipedia.org/wiki/Сортировка\_пузырьком (дата посещения 08.07.2021)

**Приложение А. Результат работы программы**

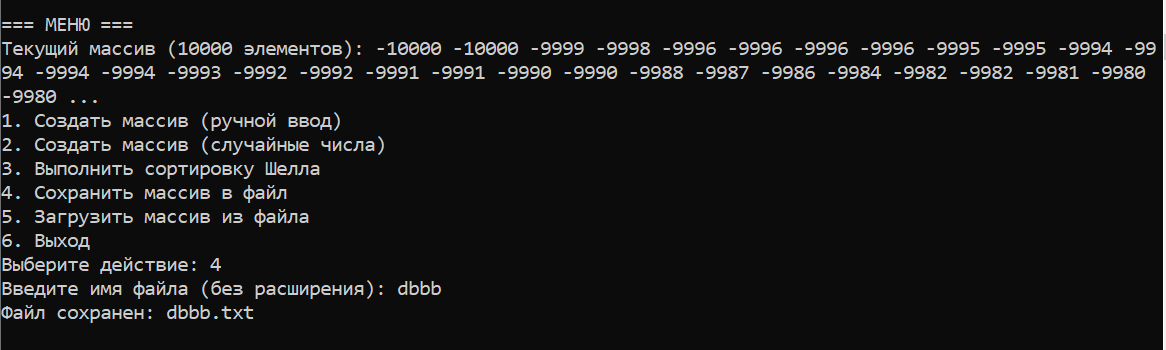


Рисунок 14(а) – Результат работы программы

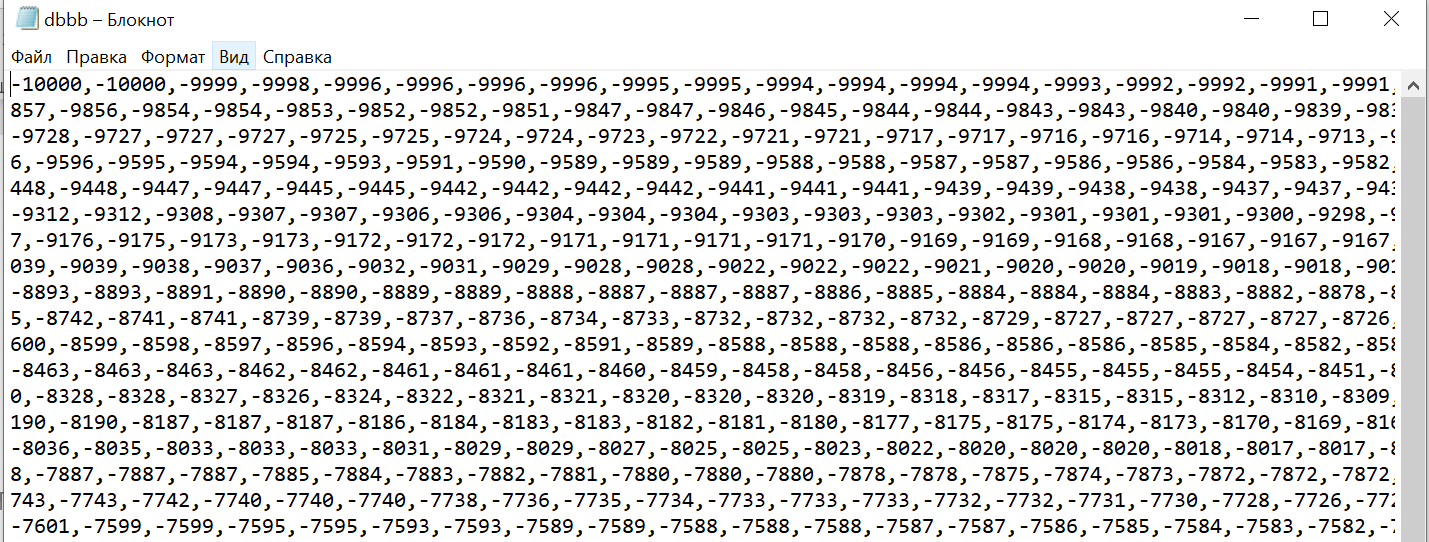


Рисунок 14(б) – Результат работы программы

# Приложение B Листинг

**Файл Prak.cpp**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include "array.h"

#include "sort.h"

#include "file.h"

int main() {

system("chcp 1251 > nul");

setlocale(LC\_ALL, "RUS");

int choice, size = 0, min, max;

int\* original = NULL;

SortResult result;

char filename[100];

while (1) {

printf("\n=== МЕНЮ ===\n");

// Отображение текущего массива

if (original && size > 0) {

printf("Текущий массив (%d элементов): ", size);

int display\_count;

if (size > 30) {

display\_count = 30;

}

else {

display\_count = size;

}

for (int i = 0; i < display\_count; i++) {

printf("%d ", original[i]);

}

if (size > 30)

printf("...");

printf("\n");

}

printf("1. Создать массив (ручной ввод)\n");

printf("2. Создать массив (случайные числа)\n");

printf("3. Выполнить сортировку Шелла\n");

printf("4. Сохранить массив в файл\n");

printf("5. Загрузить массив из файла\n");

printf("6. Выход\n");

printf("Выберите действие: ");

if (scanf\_s("%d", &choice) != 1) {

printf("Ошибка ввода! Пожалуйста, введите число.\n");

while (getchar() != '\n');

continue;

}

if (choice == 1) {

printf("Размер массива: ");

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

printf("Некорректный размер!\n");

while (getchar() != '\n');

continue;

}

if (original) free(original);

original = create\_array\_manual(size);

}

else if (choice == 2) {

printf("Размер массива: ");

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

printf("Некорректный размер!\n");

while (getchar() != '\n');

continue;

}

printf("Диапазон (min max): ");

if (scanf\_s("%d %d", &min, &max) != 2 || min > max) {

printf("Некорректный диапазон! Минимум должен быть меньше максимума.\n");

while (getchar() != '\n');

continue;

}

original = create\_array\_random(size, min, max);

}

else if (choice == 3) {

if (!original || size <= 0) {

printf("Ошибка: массив не создан!\n");

continue;

}

// Сортируем массив напрямую

result = shell\_sort(original, size);

printf("Сортировка завершена. Перестановок: %d, Время: %.6f мкс\n",

result.swaps, result.time\_mcs);

}

else if (choice == 4) {

if (!original) {

printf("Ошибка: массив не создан!\n");

continue;

}

save\_results(original, size);

}

else if (choice == 5) {

printf("Имя файла (без расширения): ");

if (scanf\_s("%99s", filename, (unsigned)\_countof(filename)) != 1) {

printf("Ошибка ввода имени файла!\n");

while (getchar() != '\n');

continue;

}

if (original) free(original);

original = load\_array\_from\_file(filename, &size);

}

else if (choice == 6) {

free(original);

printf("Программа завершена\n");

break;

}

else {

printf("Неверный выбор!\n");

}

}

return 0;

}

**file.h**

#ifndef FILE\_H

#define FILE\_H

int\* load\_array\_from\_file(const char\* filename, int\* size);

void save\_array\_to\_file(const char\* filename, int arr[], int size);

void save\_results(int array[], int size); // Упрощённая функция сохранения

#endif

**array.h**

#ifndef ARRAY\_H

#define ARRAY\_H

int\* create\_array\_manual(int size); // ручной ввод

int\* create\_array\_random(int size, int min, int max); // случайные числа

#endif

**sort.h**

#ifndef SORT\_H

#define SORT\_H

// структура для хранения результатов сортировки (время, количество итерраций)

typedef struct { // typedef для имени структуры

int swaps; // количество перестановок

double time\_mcs; // время выполнения

} SortResult; //имя структуры

SortResult shell\_sort(int arr[], int size); // сортировка Шелла

#endif

**file.cpp**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "file.h"

// Загрузка массива из файла

int\* load\_array\_from\_file(const char\* input\_filename, int\* size) {

char filename[260]; // Буфер для имени файла

// Копируем введённое имя

strcpy\_s(filename, sizeof(filename), input\_filename);

// Добавляем .txt если нет расширения

if (strchr(filename, '.') == NULL) {

strcat\_s(filename, sizeof(filename), ".txt");

}

FILE\* file;

if (fopen\_s(&file, filename, "r") != 0) {

printf("Ошибка открытия файла %s!\n", filename);

return NULL;

}

// Определяем размер файла

fseek(file, 0, SEEK\_END);

long file\_size = ftell(file);

fseek(file, 0, SEEK\_SET);

// Чтение файла

char\* buffer = (char\*)malloc(file\_size + 1);

fread(buffer, 1, file\_size, file);

buffer[file\_size] = '\0';

fclose(file);

// Убираем символ новой строки

if (file\_size > 0 && buffer[file\_size - 1] == '\n') {

buffer[file\_size - 1] = '\0';

}

// Считаем количество чисел

\*size = 1;

for (char\* p = buffer; \*p; p++) {

if (\*p == ',') (\*size)++;

}

// Создаём массив чисел

int\* arr = (int\*)malloc(\*size \* sizeof(int));

if (!arr) {

free(buffer);

return NULL;

}

// Разбиение строки

char\* next\_token = NULL;

char\* token = strtok\_s(buffer, ",", &next\_token);

for (int i = 0; token && i < \*size; i++) {

arr[i] = atoi(token);

token = strtok\_s(NULL, ",", &next\_token);

}

free(buffer);

return arr;

}

// Сохранение массива в файл

void save\_array\_to\_file(const char\* filename, int arr[], int size) {

FILE\* file;

if (fopen\_s(&file, filename, "w") != 0) {

printf("Ошибка создания файла!\n");

return;

}

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

fprintf(file, "%d", arr[i]);

if (i < size - 1) fprintf(file, ",");

}

fclose(file);

}

// Сохранение результатов (упрощённая версия)

void save\_results(int array[], int size) {

char filename[100];

printf("Введите имя файла (без расширения): ");

scanf\_s("%99s", filename, (unsigned)sizeof(filename));

// Очистка буфера после scanf\_s

int c;

while ((c = getchar()) != '\n');

// Добавляем расширение .txt

char fullname[120];

sprintf\_s(fullname, sizeof(fullname), "%s.txt", filename);

save\_array\_to\_file(fullname, array, size);

printf("Файл сохранен: %s\n", fullname);

}

**array.cpp**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "array.h"

// Ручной ввод чисел.

int\* create\_array\_manual(int size) {

int\* arr = (int\*)malloc(size \* sizeof(int)); // Динамическое выделение памяти.

if (!arr) return NULL;

printf("Введите %d целых чисел:\n", size);

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

while (scanf\_s("%d", &arr[i]) != 1) {

printf("Ошибка! Введите целое число: ");

while (getchar() != '\n'); // Очистка буфера.

}

}

return arr;

}

// Генерация случайных чисел.

int\* create\_array\_random(int size, int min, int max) {

if (min > max) { // Проверка корректности диапазона

printf("Ошибка: минимум должен быть меньше максимума!\n");

return NULL;

}

int\* arr = (int\*)malloc(size \* sizeof(int));

srand((unsigned int)time(NULL));

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

arr[i] = rand() % (max - min + 1) + min;

}

return arr;

}

**sort.cpp**

#include <time.h>

#include "sort.h"

// Сама сортировка Шелла.

SortResult shell\_sort(int arr[], int size) {

SortResult res = { 0, 0.0 }; // Инициализация результатов.

clock\_t start = clock(); // Замер времени.

// Алгоритм сортировки.

for (int gap = size / 2; gap > 0; gap /= 2) {

for (int i = gap; i < size; i++) {

int temp = arr[i];

int j;

// Сдвигаем элементы.

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {

arr[j] = arr[j - gap];

res.swaps++; // Считаем перестановки.

}

arr[j] = temp;

}

}

// Вычисляем время.

res.time\_mcs = ((double)(clock() - start)) / CLOCKS\_PER\_SEC \* 1000000;

if (res.time\_mcs < 0.000001) res.time\_mcs = 0.000001; // Гарантируем не ноль.

return res;

}