Оглавление

[1. Введение 2](#_Toc138949249)

[2. Постановка задачи 5](#_Toc138949250)

[3. Выбор решения 6](#_Toc138949251)

[4. Описание программы 7](#_Toc138949252)

[5. Схемы программы 8](#_Toc138949253)

[6. Тестирование программы 10](#_Toc138949254)

[7. Отладка 12](#_Toc138949255)

[8. Совместная разработка 13](#_Toc138949256)

[9. Заключение 15](#_Toc138949257)

[Список используемой литературы 16](#_Toc138949258)

[Приложение А 17](#_Toc138949259)

[Приложение Б “Листинг” 20](#_Toc138949260)

# 1. Введение

Сортировка выбором — это алгоритм сортировки массивов, в котором на каждой итерации во всей последовательности неотсортированных данных выбирается минимальный элемент (при сортировке по возрастанию) и помещается в первую позицию неотсортированной последовательности. Тем самым готовая (отсортированная) последовательность увеличивается на один элемент, а исходная (неотсортированная) последовательность на один элемент уменьшается.

Выборочная сортировка - это обобщение [быстрой сортировки](https://translated.turbopages.org/proxy_u/en-ru.ru.36705299-649c72ad-6830c1a0-74722d776562/https/en.wikipedia.org/wiki/Quicksort). Там, где quicksort разделяет свои входные данные на две части на каждом шаге на основе одного значения, называемого сводной, samplesort вместо этого берет из своих входных данных большую [выборку](https://translated.turbopages.org/proxy_u/en-ru.ru.36705299-649c72ad-6830c1a0-74722d776562/https/en.wikipedia.org/wiki/Sample_(statistics)) и соответственно делит свои данные на сегменты. Подобно quicksort, он затем рекурсивно сортирует сегменты.

В экспериментах, проведенных Фрейзером и Маккелларом, алгоритму выборочной сортировки требовалось на 15% меньше сравнений, чем quicksort.

Среди алгоритмов квадратичной сортировки (алгоритмов сортировки с простым средним значением [Θ(n2)](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Big_O_notation#Family_of_Bachmann%E2%80%93Landau_notations)) сортировка по выбору почти всегда превосходит [пузырьковую сортировку](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Bubble_sort) и [сортировку gnome](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Gnome_sort). [Сортировка по вставке](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Insertion_sort) очень похожа в том смысле, что после k-й итерации первые k элементы в массиве располагаются в отсортированном порядке. Преимущество сортировки по вставке в том, что она сканирует только столько элементов, сколько необходимо для размещения k+1st элемента, в то время как сортировка по выбору должна сканировать все оставшиеся элементы, чтобы найти k+1st элемент.

Простой расчет показывает, что сортировка по вставке обычно выполняет примерно вдвое меньше сравнений, чем сортировка по выбору, хотя она может выполнять столько же или намного меньше в зависимости от порядка, в котором массив находился до сортировки. Можно рассматривать как преимущество для некоторых приложений [реального](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Real-time_computing) времени то, что сортировка по выбору будет выполняться одинаково независимо от порядка расположения массива, в то время как время выполнения сортировки по вставке может значительно различаться. Однако чаще всего это преимущество сортировки по вставке в том смысле, что она выполняется намного эффективнее, если массив уже отсортирован или "близок к сортировке".

Хотя сортировка по выбору предпочтительнее сортировки по вставке с точки зрения количества операций записи (n−1 обмены против до n(n−1)/2 обменов, причем каждый обмен состоит из двух операций записи), это примерно в два раза превышает теоретический минимум, достигаемый при [циклической сортировке](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Cycle_sort), которая выполняет не более n операций записи. Это может быть важно, если операции записи значительно дороже операций чтения, например, при использовании [EEPROM](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/EEPROM) или [флэш-памяти](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Flash_memory), где каждая запись сокращает срок службы памяти.

Сортировка по выбору может быть реализована без непредсказуемых ответвлений в интересах [предикторов](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Branch_predictor) ответвлений процессора, путем нахождения местоположения минимума с помощью кода без ответвлений и последующего безусловного выполнения замены.

Наконец, сортировка по выбору значительно превосходит большие массивы благодаря алгоритмам Θ(n\*log(n)) [разделяй и властвуй](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Divide_and_conquer_algorithm), таким как [сортировка слиянием](https://translated.turbopages.org/proxy_u/en-ru.ru.dec033d7-649c733b-a6bacd6d-74722d776562/https/en.wikipedia.org/wiki/Mergesort). Однако сортировка по вставке или сортировка по выбору обычно выполняются быстрее для небольших массивов (т. Е. менее 10-20 элементов). Полезной оптимизацией на практике для рекурсивных алгоритмов является переключение на сортировку по вставке или сортировку по выбору для "достаточно маленьких" подсписков.

# 2. Постановка задачи

По программе: Необходимо заполнить массив из n-ого количества элементов случайными числами, записать данные элементы в отдельный файл. После этого выполнить выборочную сортировку над данными, находящимися в массиве, записать отсортированные данные в другой файл.

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

Оформить отчет по проведенной практике.

Достоинства данного алгоритма сортировки:

- Простой для реализации.

- Требуется мало времени для сортировки. (Быстрее пузырьковой сортировки)

- Не требует дополнительной памяти.

Недостатки данного алгоритма сортировки:

- Не рекомендуется для сортировки больших массивов.

- Временная сложность является квадратичной.

- Отсутствие адаптивности.

- Отсутствие стабильности: Это означает, что элементы с равными значениями могут менять свой относительный порядок после сортировки.

Типичные сценарии применения:

- Сортировка списков интернет – магазинов.

- Сортировка результатов спортивного соревнования.

- Сортировка списка задач по срокам выполнения.

- Сортировка массива студентов по их оценкам.

# 3. Выбор решения

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

Все действия, совершаемые программой, происходят в одном файле.

Массив данных заполняется случайными элементами с использованием цикла for.

После заполнения массива, данные переписываются в файл “Data.txt”, а массив сортируется с помощью функции customselectionSort.

После того, как массив будет отсортирован, данные из массива переписываются в другой файл “Sorted.txt”.

Программа завершает свою работу.

Если файл не был найден или не может быть создан – программа выдает ошибку.

# 4. Описание программы

При запуске программы происходит вывод сообщения “Введите нужное количество данных (n>0)”, после чего Пользователю необходимо вписать нужное количество сортируемых данных.

Затем пользователь выбирает минимальное число и максимальное. Данные, записанные в файл, в последствие, будут сгенерированы в диапазоне MIN\_VALUE-MAX\_VALUE. Если разность между MIN\_VALUE-MAX\_VALUE будет больше, чем 32500, то максимальное число, полученное в ходе программы, будет равно MIN\_VALUE + 32500.

После того, как данные были введены, генерируется массив из случайных чисел, эти числа записываются в файл “Data.txt”.

Далее вызывается функция customselectionSort, которая сортирует данные, находящиеся в массиве, выполняя выборочную сортировку.

В конце программы отсортированные данные записываются в другой файл “Sorted.txt”.

В процессе тестировки в программу вводится таймер, позволяющий совершать отладку и тестировку.

Подробный алгоритм работы программы и функции сортировки представлен в пункте 5 на рисунках 1 “Блок-схема алгоритма” и рисунке 2 “Блок-схема программы”.

# 5. Схемы программы

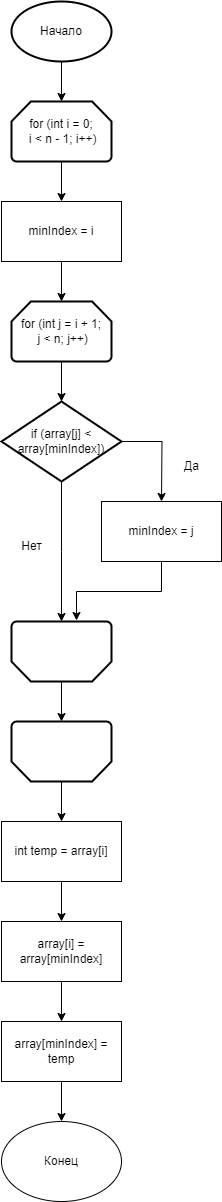


Рисунок 1 “Блок-схема алгоритма”

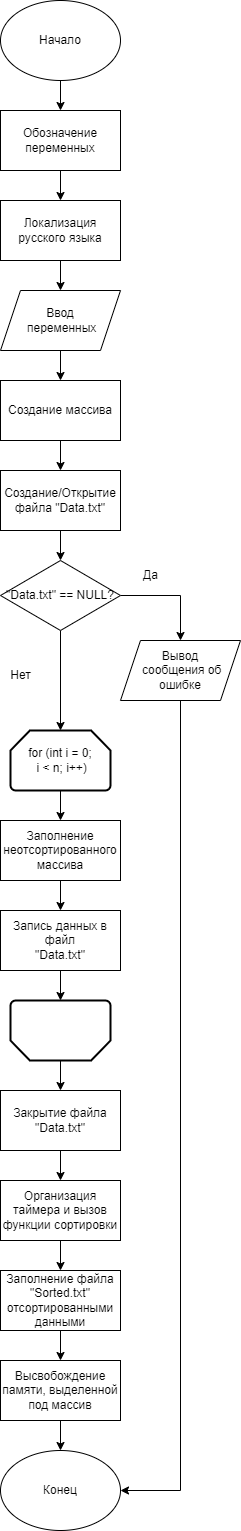


Рисунок 2 “Блок-схема программы”

# 6. Тестирование программы

Тестовый набор данных представлен в таблице 1и 2.

Снизу представлены диаграммы. (Диаграмма 1 и Диаграмма 2, в которых представлена линейная зависимость количества данных от времени, затраченного на выполнение сортировки)

Результаты тестирования приведены в Приложении А на рисунках

А.6 - А.15.

Таблица 1 – Тестовый набор данных на диапазоне от 1 до 1000

|  |  |  |
| --- | --- | --- |
|  | Кол-во данных | Время выполнения сортировки (сек) |
| 1 | 10000 | 0,12 |
| 2 | 30000 | 1,08 |
| 3 | 50000 | 3,00 |
| 4 | 70000 | 6,01 |
| 5 | 90000 | 9,76 |

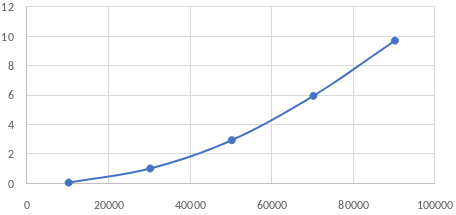


Диаграмма 1 “Тестирование программы на диапазоне от 1 до 1000”

Таблица 2 – Тестовый набор данных на диапазоне от 1 до 15000

|  |  |  |
| --- | --- | --- |
|  | Кол-во данных | Время выполнения сортировки (сек) |
| 1 | 10000 | 0,12 |
| 2 | 30000 | 1,09 |
| 3 | 50000 | 2,96 |
| 4 | 70000 | 5,86 |
| 5 | 90000 | 9,69 |

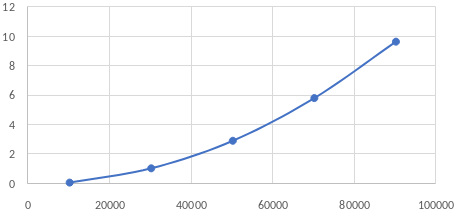


Диаграмма 2 “Тестирование на диапазоне от 1 до 15000”

В результате выполнения тестирования было выявлено время работы данного алгоритма и получен конечный результат, описанный в таблицах.

# 7. 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 глобальных и локальных переменных.

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

# 8. Совместная разработка

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

Данная программа была написана владельцем репозитория - Трундовым Н.А.

После написания программы, она была выгружена на удаленный репозиторий Github, на отдельную ветку под названием Trundov.

(См. Рисунок 3)

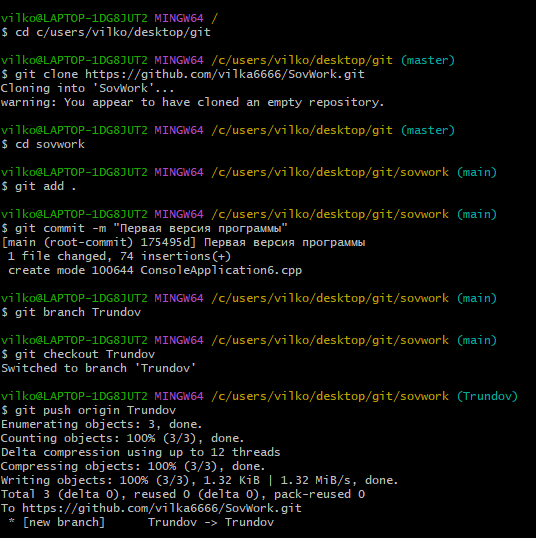


Рисунок 3

После этого, второй участник – Евдокимов Р.Е. загрузил данную программу себе на компьютер, с помощью git clone <ссылка>, и добавил в нее алгоритм, считающий время работы программы в секундах и фиксирующий значение по окончании. После чего создал собственную ветку, и выгрузил обновленный код программы на удаленный репозиторий GitHub.

(См. Рисунок 4)

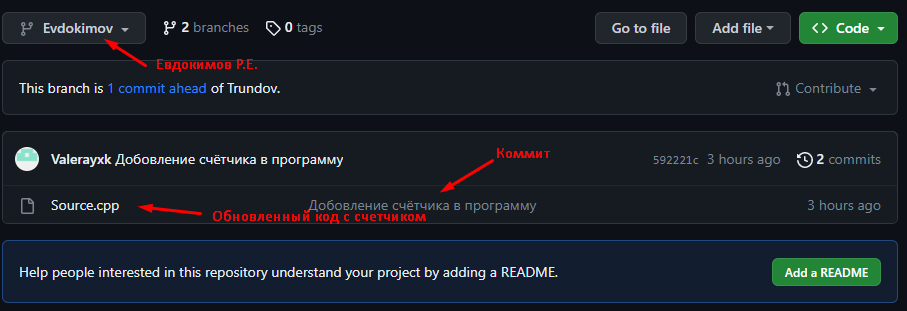


Рисунок 4

После выполнения данных изменений, Евдокимов Р.Е. обнаружил ошибку и исправил. Выгрузил на удаленный репозиторий новую версию программы. (См. Рисунок 5)

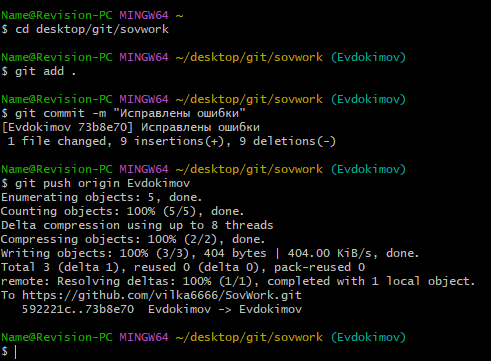


Рисунок 5

Ссылка на удаленный репозиторий:

https://github.com/vilka6666/SovWork.git

# 9. Заключение

Нашей бригадой были получены навыки совместной работы с помощью сервиса GitHub, навыки использования программы Git Bash. Нами так же был изучен алгоритм выборочной сортировки. Трундов Н.А. написал программу, выполняющую данную сортировку над массивом случайно сгенерированных чисел, и оформил отчет по данной практике. Евдокимов Р.Е. выполнил тестирование и отладку данной программы.

Так же при выполнении практической работы были улучшены наши базовые навыки программирования на языках С/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. Можно повысить максимальную разность чисел.

# Список используемой литературы

1. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.

2. Прата С. Язык программирования С++. 2019 г.

3. А.А. Тюгашев. Языки программирования. Учебное пособие. 2018 г.

4. [Деннис Ритчи](https://yandex.ru/search/?clid=2456108&text=%D0%94%D0%B5%D0%BD%D0%BD%D0%B8%D1%81%20%D0%A0%D0%B8%D1%82%D1%87%D0%B8&lr=49&noreask=1&ento=0oCglydXcxOTM5MjUYAioKcnV3MTU5ODU4MWou0K_Qt9GL0Log0L_RgNC-0LPRgNCw0LzQvNC40YDQvtCy0LDQvdC40Y8g0KHQuHIM0JDQstGC0L7RgNGLlfhdsQ), [Брайан Керниган](https://yandex.ru/search/?clid=2456108&text=%D0%91%D1%80%D0%B0%D0%B9%D0%B0%D0%BD%20%D0%9A%D0%B5%D1%80%D0%BD%D0%B8%D0%B3%D0%B0%D0%BD&lr=49&noreask=1&ento=0oCglydXcyNDY3MjAYAioKcnV3MTU5ODU4MWou0K_Qt9GL0Log0L_RgNC-0LPRgNCw0LzQvNC40YDQvtCy0LDQvdC40Y8g0KHQuHIM0JDQstGC0L7RgNGLGjVc_A). Язык программирования Си. 1978 г.

5. Солдатенко, И.С. Основы программирования на языке Си: учеб. Пособие. 2017г.

6. [Левитин А. В.](https://www.wikidata.org/wiki/Q21694518) Глава 3. Метод грубой силы: Сортировка выбором.2006.

7. Роберт Седжвик. Часть III. Глава 6. Элементарные методы сортировки: 6.2 Сортировка выбором // Алгоритмы на C++ = Algorithms in C++. 2011.

8. [Кормен, Т.](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%BC%D0%B5%D0%BD,_%D0%A2%D0%BE%D0%BC%D0%B0%D1%81), [Лейзерсон, Ч.](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%B9%D0%B7%D0%B5%D1%80%D1%81%D0%BE%D0%BD,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7_%D0%AD%D1%80%D0%B8%D0%BA" \o "), [Ривест, Р.](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D0%B2%D0%B5%D1%81%D1%82,_%D0%A0%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%9B%D0%B8%D0%BD%D0%BD" \o "Ривест, Рональд Линн), [Штайн, К.](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D0%B0%D0%B9%D0%BD,_%D0%9A%D0%BB%D0%B8%D1%84%D1%84%D0%BE%D1%80%D0%B4" \o "Штайн, Клиффорд) Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. — 2-е изд.  2005.

# Приложение А

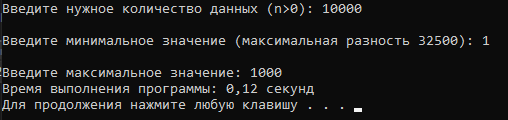


Рисунок А.6

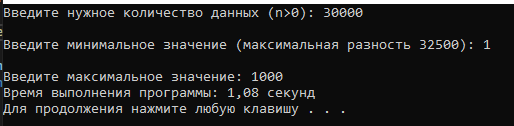


Рисунок А.7

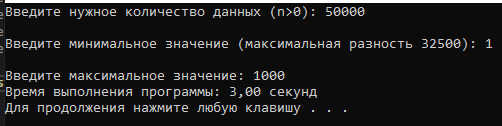


Рисунок А.8

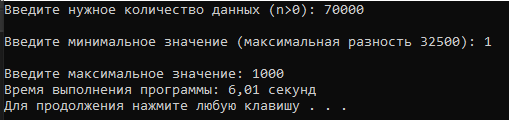


Рисунок А.9

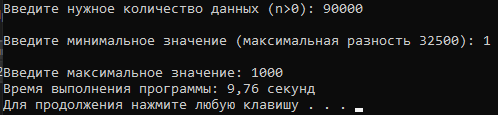


Рисунок А.10

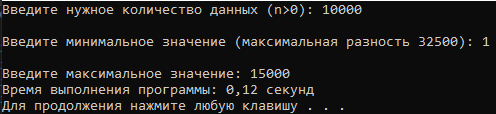


Рисунок А.11

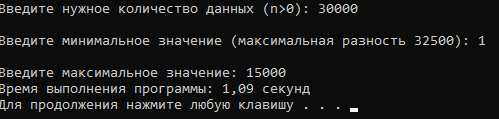


Рисунок А.12

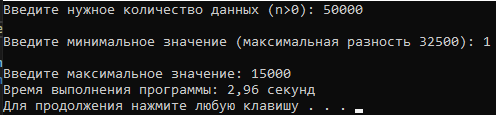


Рисунок А.13

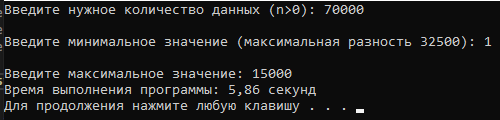


Рисунок А.14

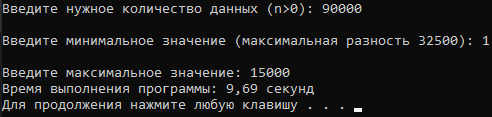


Рисунок А.15

# Приложение Б “Листинг”

#define \_CRT\_SECURE\_NO\_WARNINGS // Исправление ошибок fopen

#include <stdio.h>

#include <locale.h>

#include <windows.h> // Подключение библиотек

#include <time.h>

void customselectionSort(int array[], int n) { // Функция сортировки

for (int i = 0; i < n - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < n; j++) {

if (array[j] < array[minIndex]) {

minIndex = j;

}

}

int temp = array[i];

array[i] = array[minIndex];

array[minIndex] = temp;

}

}

int main()

{

int n; // Обозначение переменных

int MIN\_VALUE;

int MAX\_VALUE;

setlocale(LC\_ALL, "RUS"); // Локализация Русского языка

printf("Введите нужное количество данных (n>0): ");

scanf("%ld", &n); // Ввод кол-ва данных

printf("\nВведите минимальное значение (максимальная разность 32500): ");

scanf("%ld", &MIN\_VALUE); // Ввод минимального значения данных

printf("\nВведите максимальное значение: ");

scanf("%ld", &MAX\_VALUE); // Ввод максимального значения данных

int\* array = NULL;

array = (int\*)malloc(n \* sizeof(int)); // Создание неотсортированного массива

FILE\* record = fopen("Data.txt", "w"); // Открытие файла

if (record == NULL) {

printf("Ошибка открытия файла.\n"); // Проверка на открытие файла

}

srand(time(NULL));

for (int i = 0; i < n; i++) { // Цикл заполнения файла

int number = rand() % (MAX\_VALUE - MIN\_VALUE + 1) + MIN\_VALUE;

array[i] = number; // Заполнение неотсортированного массива

fprintf(record, "%ld\n", number); // Запись данных в файл

}

fclose(record); // Закрытие файла

clock\_t start = clock(); // Запуск таймера

customselectionSort(array, n); // Вызов функции сортировки

clock\_t end = clock(); // Остановка таймера

double time\_taken = ((double)(end - start)) / CLOCKS\_PER\_SEC; // Вычисление времени выполнения

printf("Время выполнения программы: %.2f секунд\n", time\_taken);

FILE\* sorted = fopen("Sorted.txt", "w"); // Открытие файла для записи отсортированных данных

for (int i = 0; i < n; i++) { // Цикл заполнения отсортированного файла

fprintf(sorted, "%ld\n", array[i]);

}

fclose(sorted); // Закрытие файла для записи отсортированных данных

system("pause"); // Остановка программы

free(array); // Высвобождение памяти

return 0;

}