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**

(2021/2022 учебный год)

Алёшина Анна Васильевна

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 обучения1семестр 2

Период прохождения практики с 28.06.2022 по 11.07.2022

Кафедра «Вычислительная техника»

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

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

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

*(должность, ученая степень, ученое звание)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

|  |  |
| --- | --- |
|  | Утвержден на заседании кафедры  «Вычислительная техника»  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.  Заведующий кафедрой                                              М.А. Митрохин |
|  |  |

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2021/2022 учебный год)

                                            Алёшина Анна Васильевна

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 28.06.2022 по 11.07.2022

Кафедра «Вычислительная техника»

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

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

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

*(должность, ученая степень, ученое звание)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Планируемая форма работы во время практики | Количество часов | Календарные сроки проведения работы | Подпись  руководителя  практики от вуза |
| 1 | Выбор темы и разработка индивидуального плана проведения работ | 2 | 28.06.2022 -  28.06.2022 |  |
| 2 | Подбор и изучение материала по теме работы | 15 | 30.06.2022 –  02.07.22 |  |
| 3 | Разработка алгоритма | 43 | 02.07.22 –  06.07.22 |  |
| 4 | Описание алгоритма и программы | 18 | 6.07.22 –  08.07.22 |  |
| 5 | Тестирование | 5 | 08.07.22 –  08.07.22 |  |
| 6 | Получение и анализ результатов | 10 | 08.07.22 –  10.07.22 |  |
| 7 | Оформление отчёта | 15 | 10.07.22 –  11.07.2022 |  |
|  | **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**

(2021/2022 учебный год)

Алёшина Анна Васильевна

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 обучения1семестр 2

Период прохождения практики с 28.06.2022 по 11.07.2022

Кафедра «Вычислительная техника»

Алёшина А.В. выполняла практическое задание «Сортировка Шелла». На первоначальном этапе были изучен и проанализирован алгоритм сортировки Шелла, был выбран метод решения и язык программирования С++, на котором была написана программа сортировки массива методом Шелла. Также, осуществила работу с файлами. Протестировала и отладила программу. Оформила отчёт.

Бакалавр Алёшина А.В. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Руководитель Карамышева Н.С. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2021/2022 учебный год)

Алёшина Анна Васильевна

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 обучения1семестр 2

Период прохождения практики с 28.06.2022 по 11.07.2022

Кафедра «Вычислительная техника»

В процессе выполнения практики Алёшина А.В. решал следующие задачи: тестирование на разных наборах данных, анализ полученных результатов тестирования.

За период выполнения практики были освоены основные понятия и технологии сортировки Шелла, реализован метод работы с файлами. Во время выполнения работы Алёшина А.В. показала себя добросовестным, ответственным 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 Алёшина А.В. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики к.т.н., доцент, Карамышева Н.С. « » 2022 г.

**Оглавление**

[**Ведение** 7](#_Toc107494922)

[**1. Постановка задачи** 9](#_Toc107494923)

[1.1 Достоинства алгоритма 9](#_Toc107494924)

[1.2 Недостатки алгоритма 9](#_Toc107494925)

[1.3 Типичные сценарии применения 9](#_Toc107494926)

[**2. Выбор решения** 10](#_Toc107494927)

[**3. Описание программы** 17](#_Toc107494928)

[**4. Схемы программы** 17](#_Toc107494929)

[4.1 Блок-схема программы 17](#_Toc107494930)

[4.2 Блок-схема алгоритма 17](#_Toc107494932)

[**5. Тестирование программы** 18](#_Toc107494934)

[5.1 Тестирование на разных наборах данных 18](#_Toc107494935)

[5.2 Анализ полученных результатов тестирования (анализ работы алгоритма) 18](#_Toc107494935)9

[**6. Отладка** 20](#_Toc107494941)

[**Заключение** 22](#_Toc107494942)

[**Список используемой литературы** 23](#_Toc107494945)

[**Приложение А** 24](#_Toc107494946)

[**Приложение Б** 28](#_Toc107494967)

**Ведение**

Сортировка Шелла, являющийся усовершенствованным вариантом [сортировки вставками](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0%D0%BC%D0%B8). Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга. Иными словами - это сортировка вставками с предварительными «грубыми» проходами.

При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии{\displaystyle d} {\displaystyle d}. После этого процедура повторяется для некоторых меньших значений{\displaystyle d}, а завершается сортировка Шелла упорядочиванием элементов{\displaystyle d=1}. Эффективность сортировки Шелла в определённых случаях обеспечивается тем, что элементы «быстрее» встают на свои места в простых методах сортировки, например, [пузырьковой](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BC_%D0%BF%D1%83%D0%B7%D1%8B%D1%80%D1%8C%D0%BA%D0%B0), каждая перестановка двух элементов уменьшает количество [инверсий](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F_(%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0)) в списке максимум на 1, а при сортировке Шелла это число может быть больше. Невзирая на то, что сортировка Шелла во многих случаях медленнее, чем [быстрая сортировка](https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0), она имеет преимущество - отсутствие потребности в памяти под стек.

7

**1. Постановка задачи**

Необходимо разработать алгоритм сортировки Шелла. Конечная программа должна реализовывать непосредственно алгоритм, а также выполнять считывание входных данных из файла и запись результатов в файл и отображать все данные в интерфейсе.

# 1.1 Достоинства алгоритма

* Отсутствие потребности в памяти под стек;
* Отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до O(n²), что хуже, чем худшее гарантированное время для сортировки Шелла.

# Недостатки алгоритма

* Значительно уступает другим алгоритмам на малых наборах данных
* Имеет очень узкую область применения
* Необходимость подбирать эффективную последовательность distance

# 1.3 Типичные сценарии применения

* Замена сортировки по вставке, если для выполнения заданной задачи требуется много времени.
* Для вызова накладных расходов стека мы используем shell sort.
* Для наборов данных среднего и большого размера.
* В сортировке вставки, чтобы уменьшить количество операци.

8

**2. Выбор решения**

В качестве языка программирования выбран Си++. Си++ - это высокоуровневый язык программирования общего назначения. Широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), разнообразных прикладных программ, [драйверов](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ.

В качестве программной среды по разработке приложений для ОС Windows была использована Microsoft Visual studio, так как в ней есть весь необходимый функционал.

Программа написана так, что пользователь сам может задать количество элементов в последовательности. Последовательность элементов в программе представлена массивом. Для эффекетивности массив задается не вручную, а рандомно или же пользователь может выбрать уже готовый массив из файла.

В результате работы программы на экране видна вся необходимая информация: количество элементов последовательности, исходная последовательность, результат сортировки, затрачиваемое время выполнения алгоритма.

9

**3. Описание программы**

В программе подключен следующий заголовочный файл: iostream - это [заголовочный файл](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) с классами, функциями и переменными для организации ввода-вывода в языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B). fstream - заголовочный файл, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтени/записи данных из/в [файл](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB). windows.h необходим для использования в программе функционала, предоставляемого операционной системой windows.

#include <iostream>

#include <fstream>

#include <windows.h>

Функция main осуществляет управление всеми модулями программы.

create\_and\_save() создаёт динамический массив и запрашивает у пользователя способ инициализации массива.

int\* create\_and\_save()

{

std::cout << "Выбирете способ инициализации массива:\n1)Ввести вручную размерность и инициализировать случайно\n2)Из файла\n > ";

int answer = 0;

std::cin >> answer;

int\* array = nullptr;

Первый способ - ввод вручную массива. При этом функция create\_and\_save() вызывает функцию init\_array() и передаёт ей размерность массива который ввёл пользователь.

if (answer == 1)

{

max\_size = set\_max\_size();

if (max\_size < 1)

{

return 0;

}

array = new int[max\_size];

init\_array(array);

}

После чего функция init\_array() заполняет весь массив рандомными числами и возвращают управление функции create\_and\_save().

bool init\_array(int\* array)

{

for (int counter = 0; counter < max\_size; ++counter)

{

array[counter] = rand() % 201 - 101;

}

return 0;

}

Второй способ - чтение массива из файла. В этом случае функция create\_and\_save() вызывает функцию init\_array\_from\_file().

else if (answer == 2)

{

array = init\_array\_from\_file(max\_size);

}

Она открывает файл, название которого вводит пользователь, и читает данные, тем самым заполняя массив. Далее возвращает управление обратно функции create\_and\_save().

int\* init\_array\_from\_file(int& max\_size)

{

std::cout << "Введите название файла:\n > ";

std::string answer = "";

std::cin >> answer;

std::ifstream file(answer);

file >> max\_size;

if (max\_size < 1)

{

return 0;

}

int\* array = new int[max\_size];

for (int counter = 0; counter < max\_size; ++counter)

{

file >> array[counter];

}

file.close();

return array;

}

После инициализации массива функция create\_and\_save() запрашивает у пользователя если необходимость сохранения исходного массива.

Если надо сохранить массив вызывается функция print\_to\_file(), если же не надо сохранить массив то функция возвращает управление функции main().

std::cout << "Сохранить исходный массив в файл?(1 - да/0 - нет)\n > ";

std::cin >> answer;

if (answer)

{

print\_to\_file(array);

}

return array;

}

bool print\_to\_file(int\* array)

{

std::cout << "Введите название файла для записи:\n > ";

std::string name = "";

std::cin >> name;

std::ofstream file(name);

if (!file)

{

std::cout << "Ошибка открытия файла.\n";

return 1;

}

file << max\_size << std::endl;

for (int counter = 0; counter < max\_size; ++counter)

{

file << array[counter] << ' ';

}

file.close();

return 0;

}

**4. Схемы программы**

4.1 Блок-схема программы

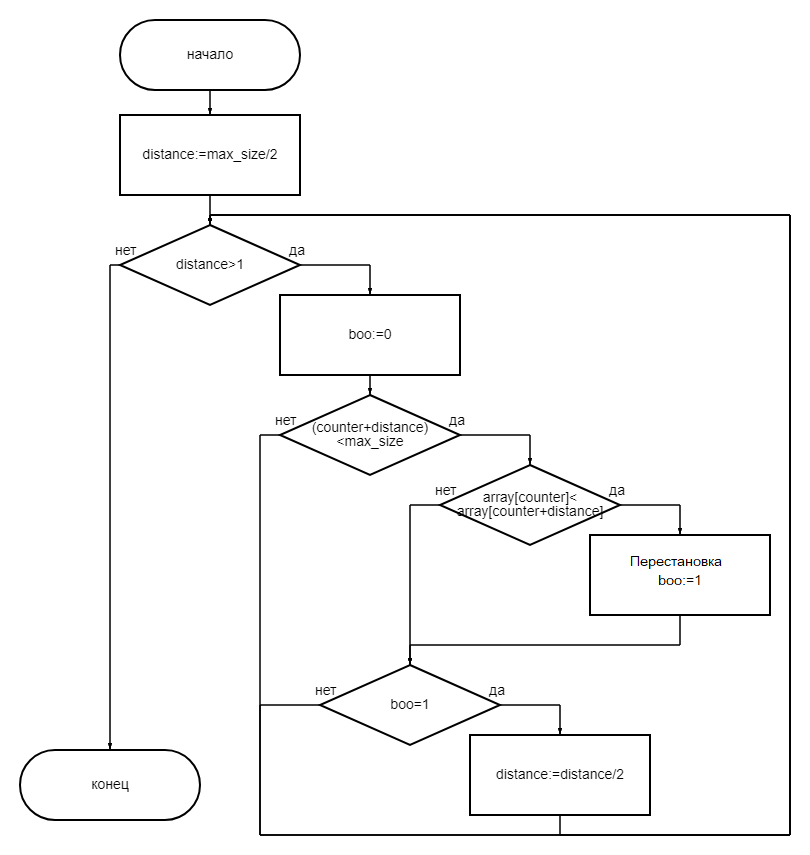


Рисунок 1- Схема алгоритма программы

4.2 Блок-схема алгоритма

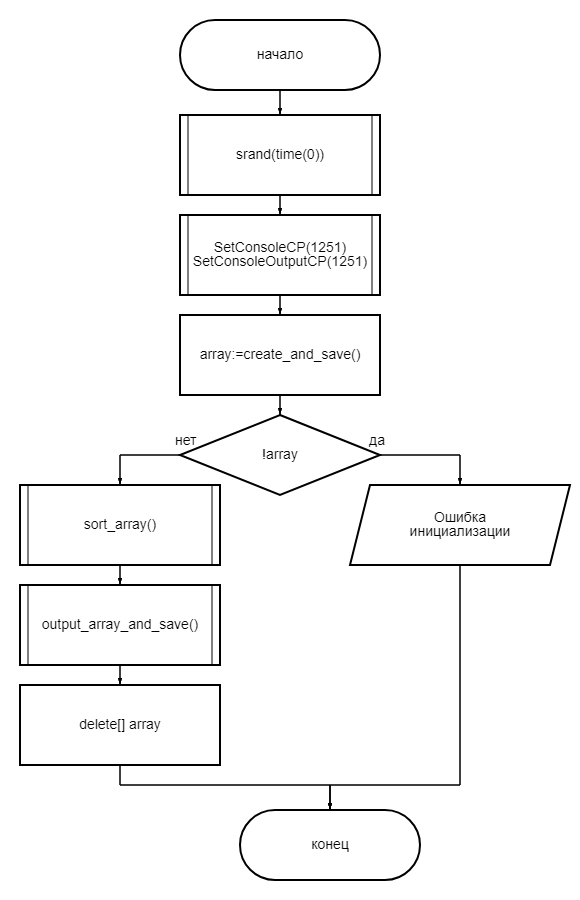


Рисунок 2 - Схема функции main

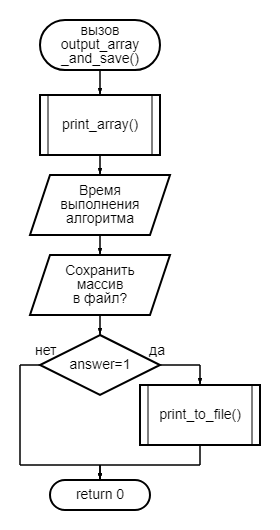
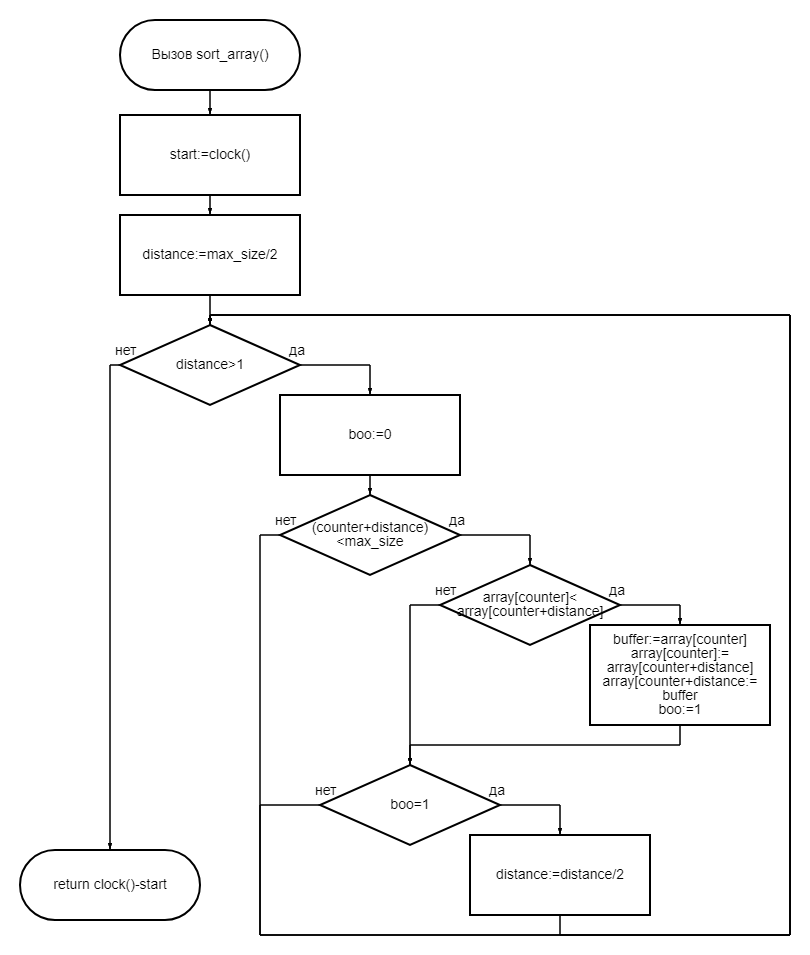


Рисунок 3 - Схема функции sort\_array и output\_array\_and\_save

**5. Тестирование программы**

5.1 Тестирование на разных наборах данных

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены в приложении А на рисунках 9-18

Таблица 1- График времени работы алгоритма

|  |  |  |
| --- | --- | --- |
| № теста. | Количество элементов. | Время выполнения сортировки в миллисекундах. |
| 1 | 1 000 | 110 |
| 2 | 2 000 | 229 |
| 3 | 3 000 | 389 |
| 4 | 4 000 | 529 |
| 5 | 5 000 | 667 |
| 6 | 6 000 | 848 |
| 7 | 7 000 | 974 |
| 8 | 8 000 | 1055 |
| 9 | 9 000 | 1294 |
| 10 | 10 000 | 1526 |

Рисунок 4 - Результаты тестирования

5.2 Анализ полученных результатов тестирования (анализ работы алгоритма)

Проводя анализ работы алгоритма, мы выяснили, что при увеличении количества элементов последовательности, возрастает также и время выполнения сортировки. Поэтому данная сортировка неэффективна на большом количестве данных.

**6. Отладка**

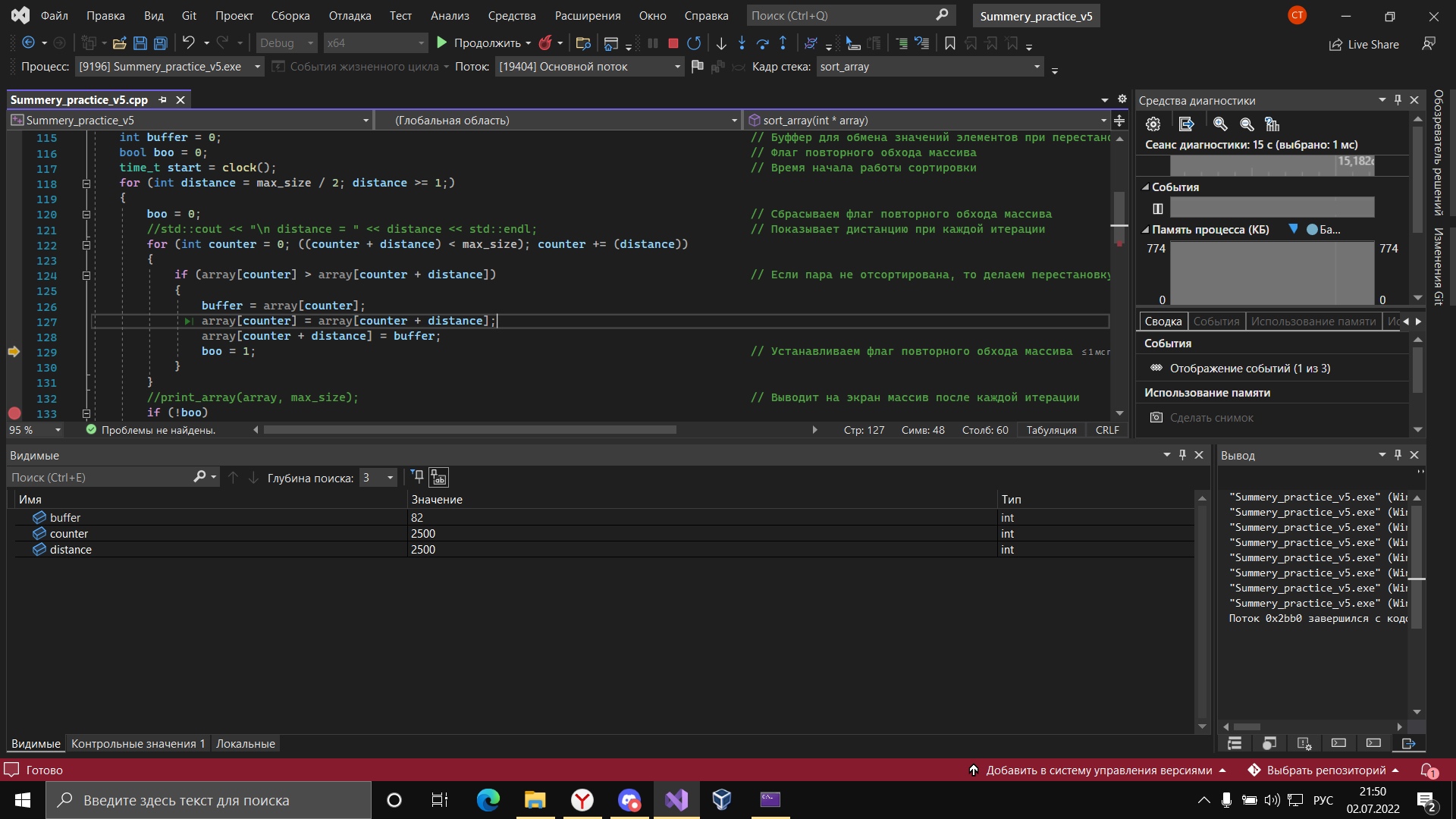


Рисунок 5 – Отладка программы

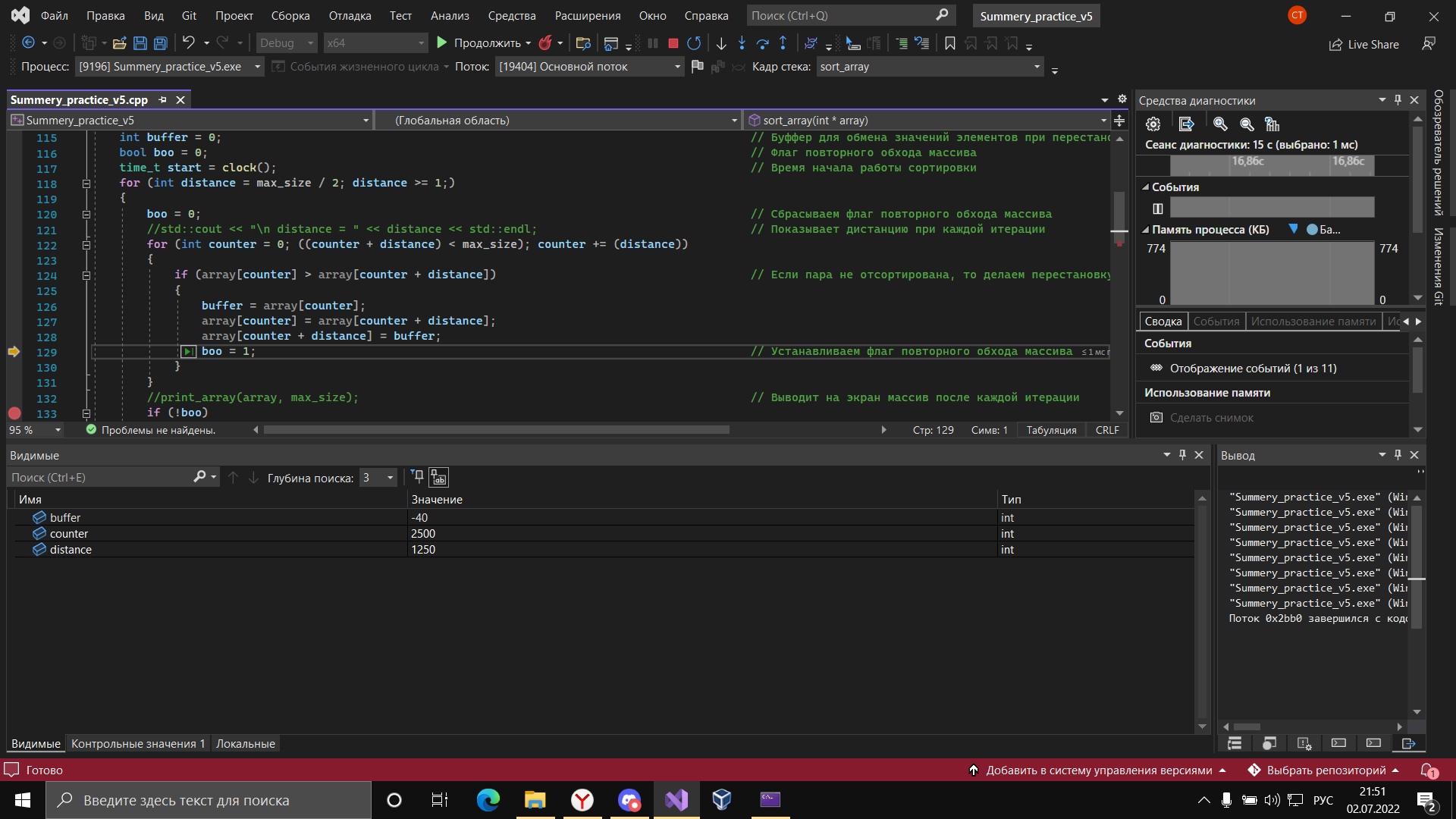


Рисунок 6 - Отладка программы

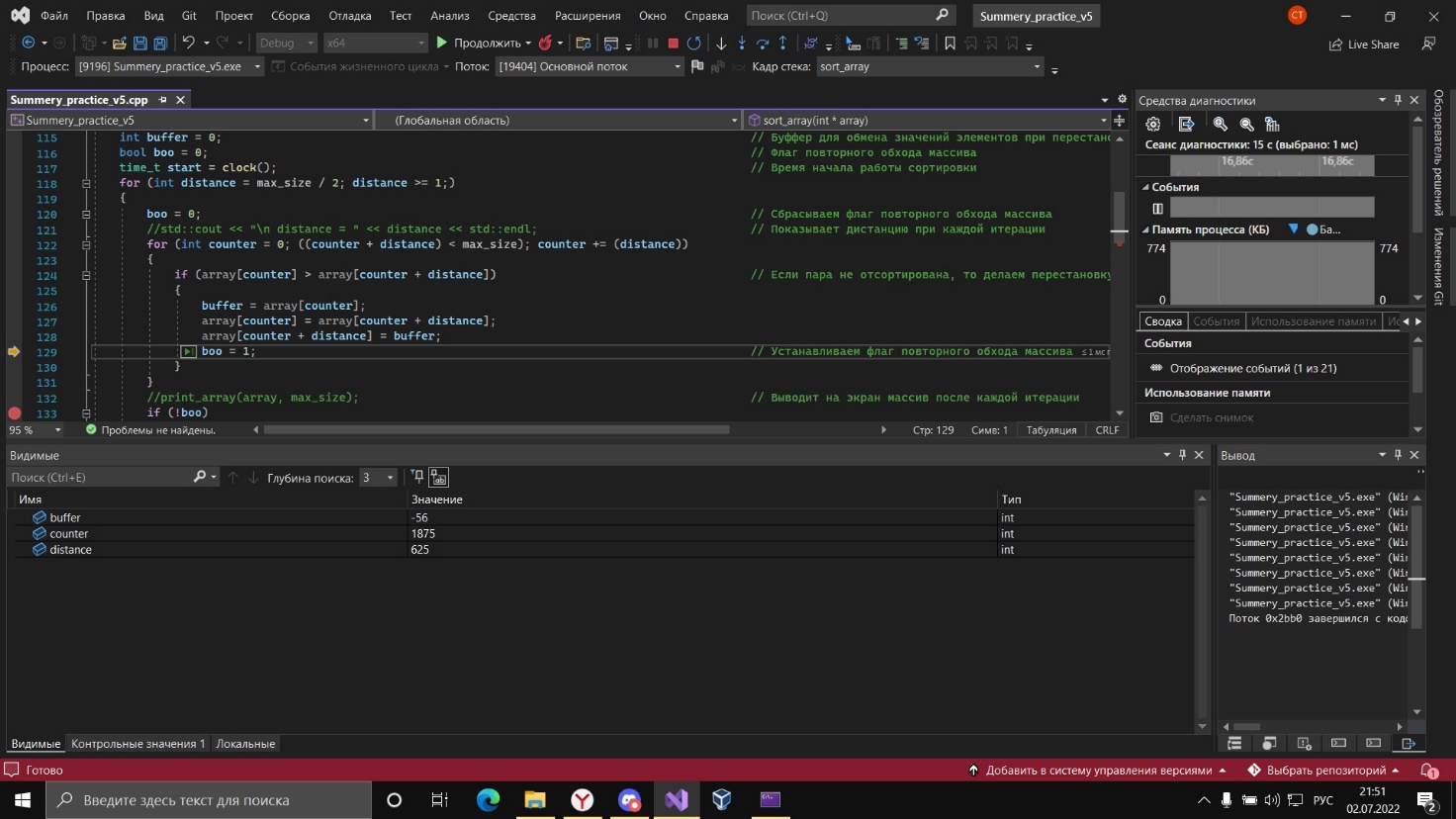


Рисунок 7 - Отладка программы

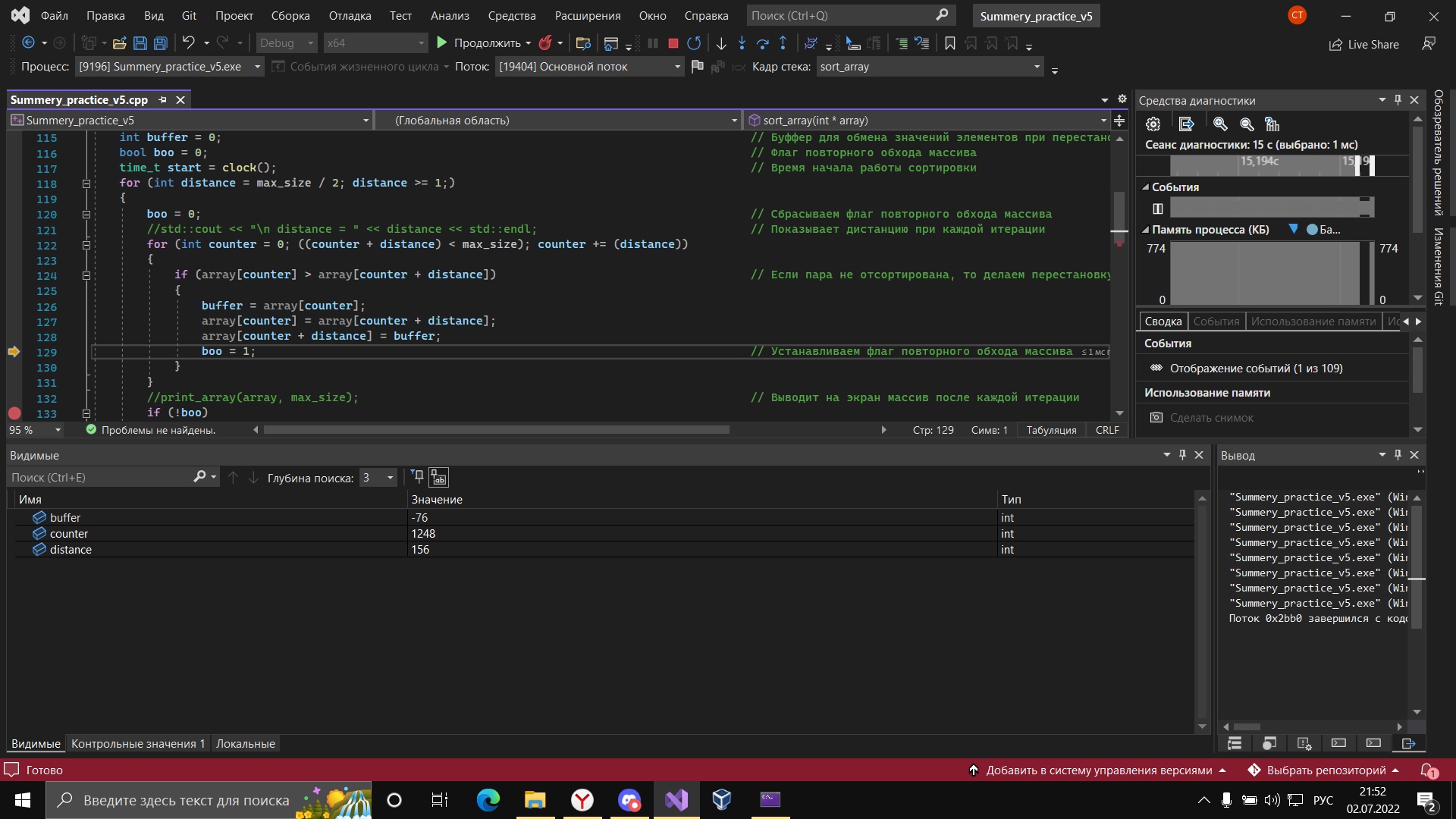


Рисунок 8 - Отладка программы

**Заключение**

В ходе работы мы написали программу, использующую алгоритм сортировки Шелла. Протестировали его на разных наборах данных. В результате тестирования мы пришли к выводу, что данный алгоритм подходит для сортировки любых массивов данных, но его потенциал раскрывается на массивах с большой размерностью, так как на малых массивах он проигрывает алгоритмам быстрой сортировки.

Таким образом сортировка Шелла является эффективной на больших последовательностях. Реализация данного алгоритма проста, но высокая вычислительная сложность не позволяет применять данный алгоритм в повсеместном использовании.

**Список используемой литературы**

1. Учебное пособие. Д.В. Шагбазян, А.А. Штанюк, Е.В. Малкина «Алгоритмы сортировки. Анализ, реализация, применение.»
2. wiki/Сортировка\_Шелла [Электронный ресурс] // /Wikipedia

Режим доступа – URL. <https://ru.wikipedia.org/wiki/Сортировка_Шелла>

1. ru/post/335920/ [Электронный ресурс ] //Habr

Режим доступа – URL. <https://habr.com/ru/post/335920/>

1. /shellsort/ [Электронный ресурс] // Geeksgorgeeks

Режим доступа – URL. <https://www.geeksforgeeks.org/shellsort/>

1. ru.stackoverflow.com [Электронный ресурс] // StackOwerflow

Режим доступа – URL. <https://ru.stackoverflow.com>

**Приложение А**

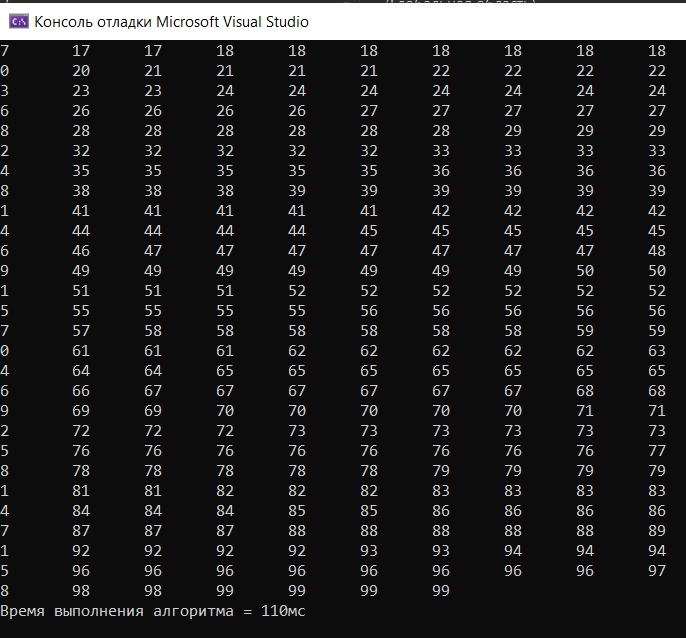


Рисунок 9 – Результат тестирования массива разрядностью 1000

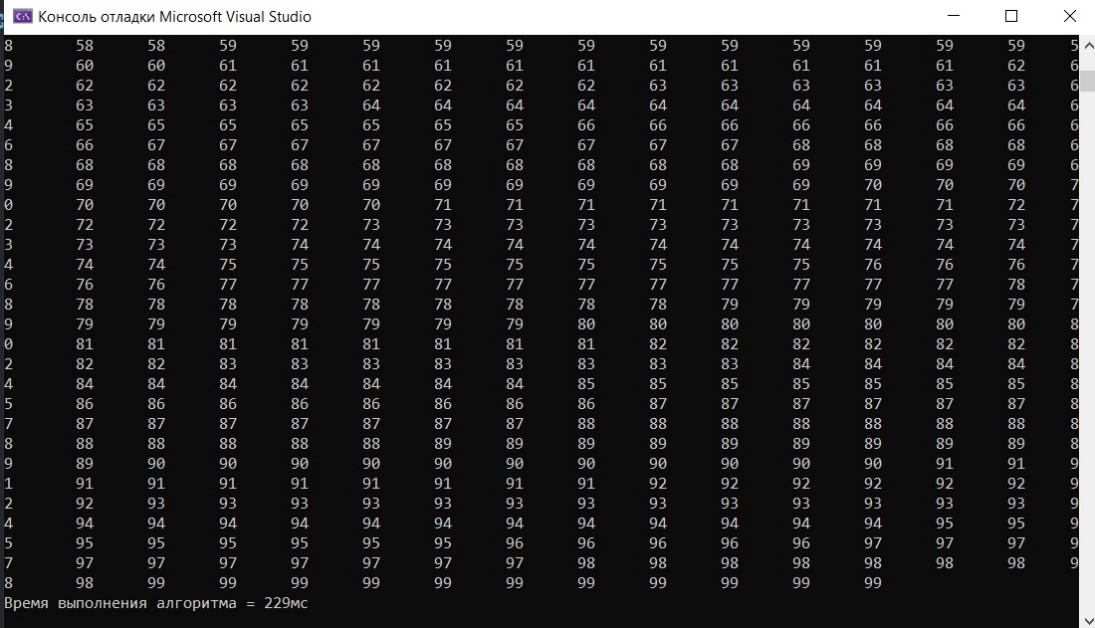


Рисунок 10 – Результат тестирования массива разрядностью 2000

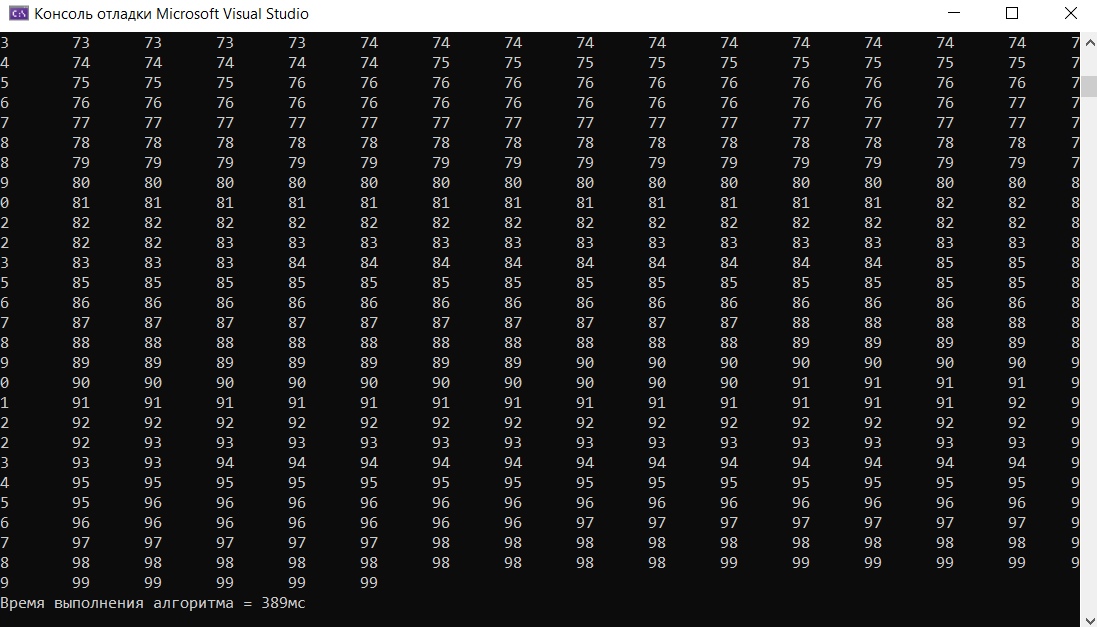


Рисунок 11 – Результат тестирования массива разрядностью 3000

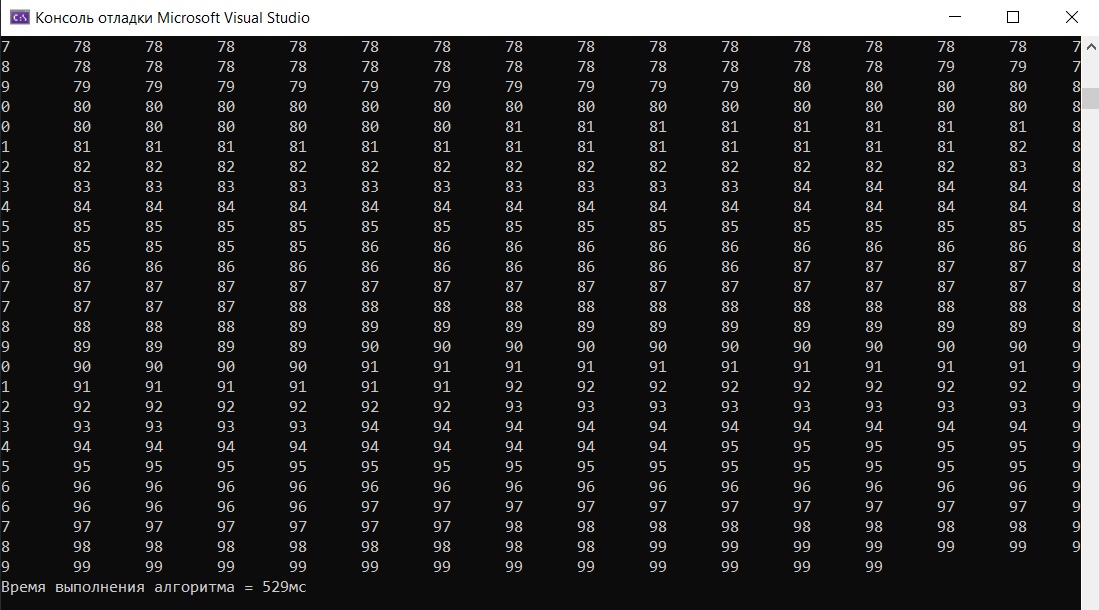


Рисунок 12 – Результат тестирования массива разрядностью 4000

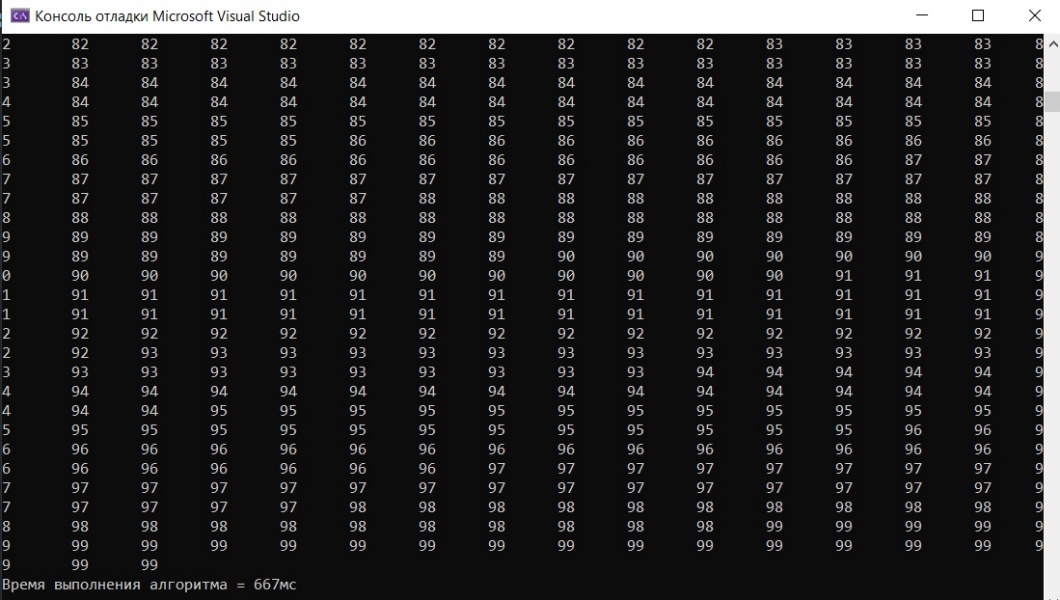


Рисунок 13 – Результат тестирования массива разрядностью 5000

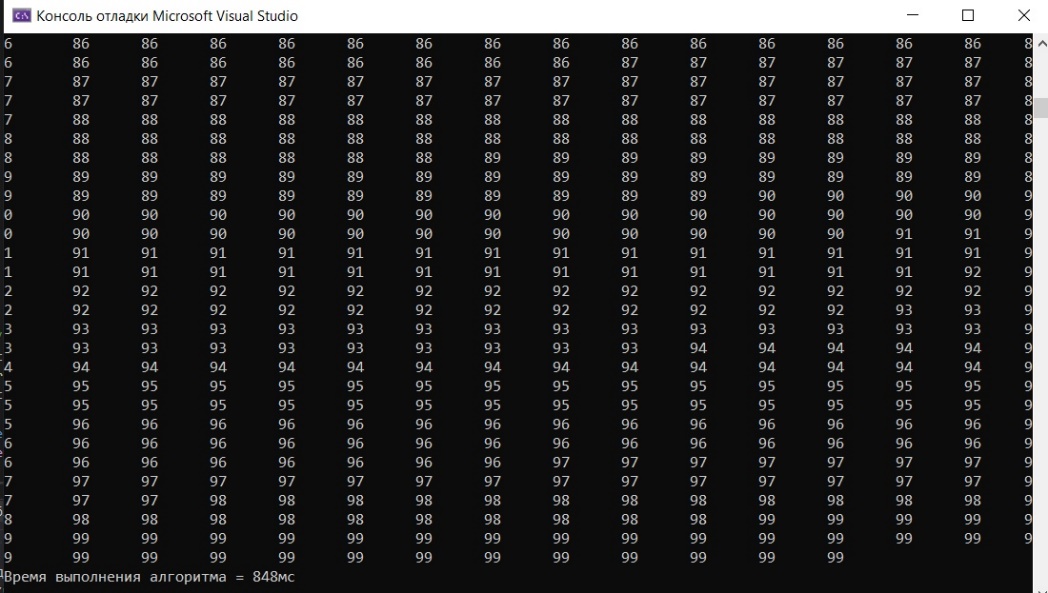


Рисунок 14 – Результат тестирования массива разрядностью 6000

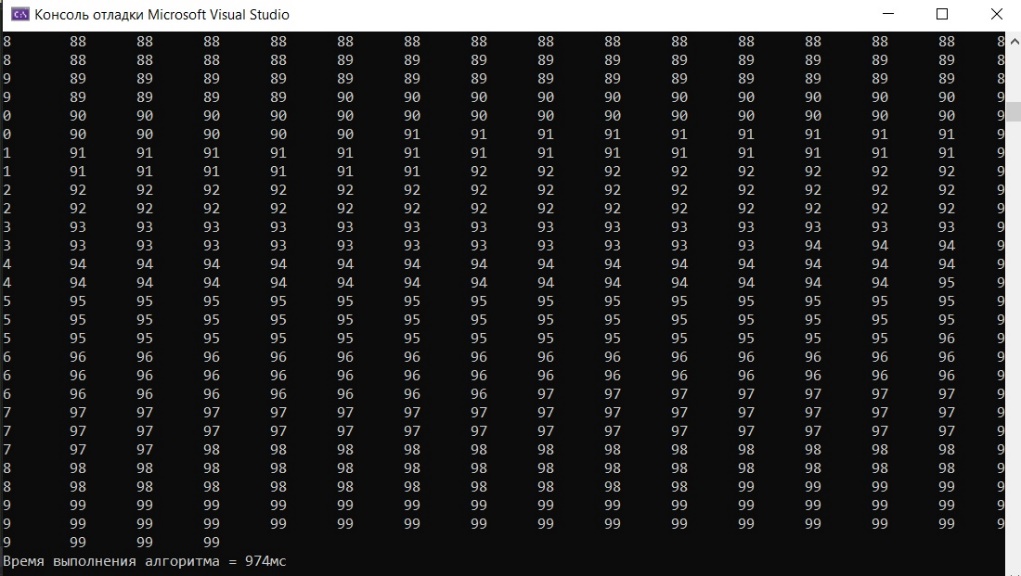


Рисунок 15 – Результат тестирования массива разрядностью 7000

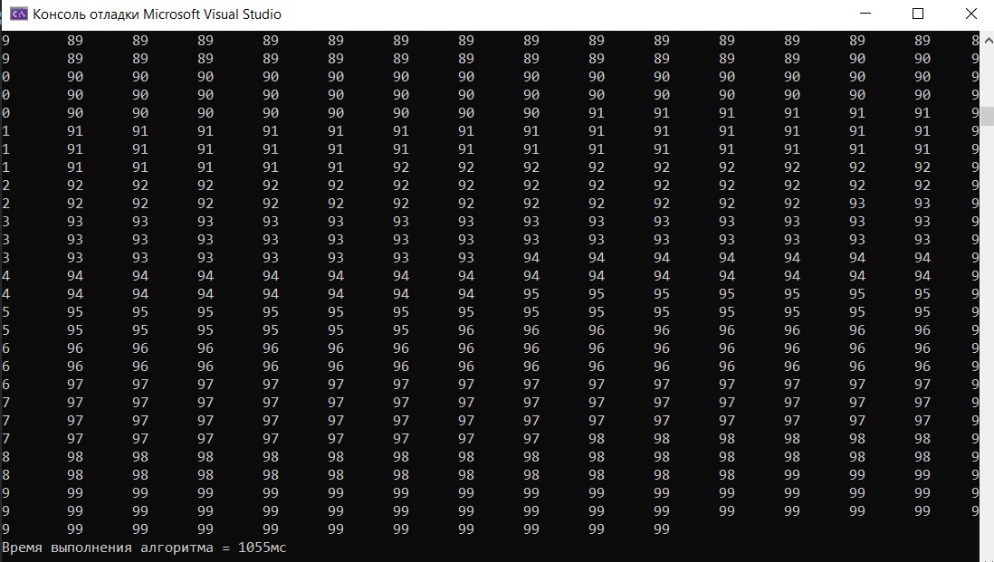


Рисунок 16 – Результат тестирования массива разрядностью 8000

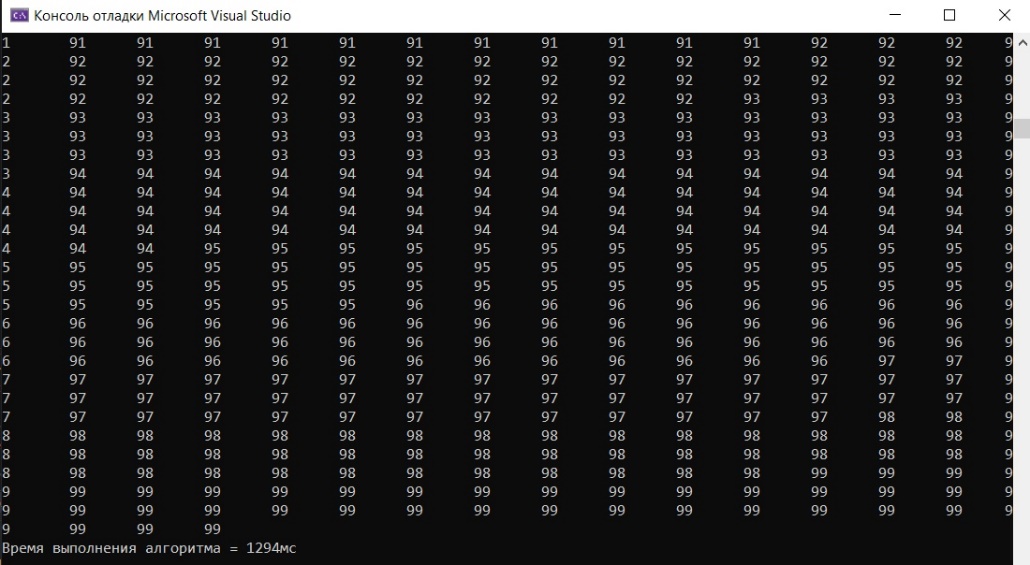


Рисунок 17 – Результат тестирования массива разрядностью 9000

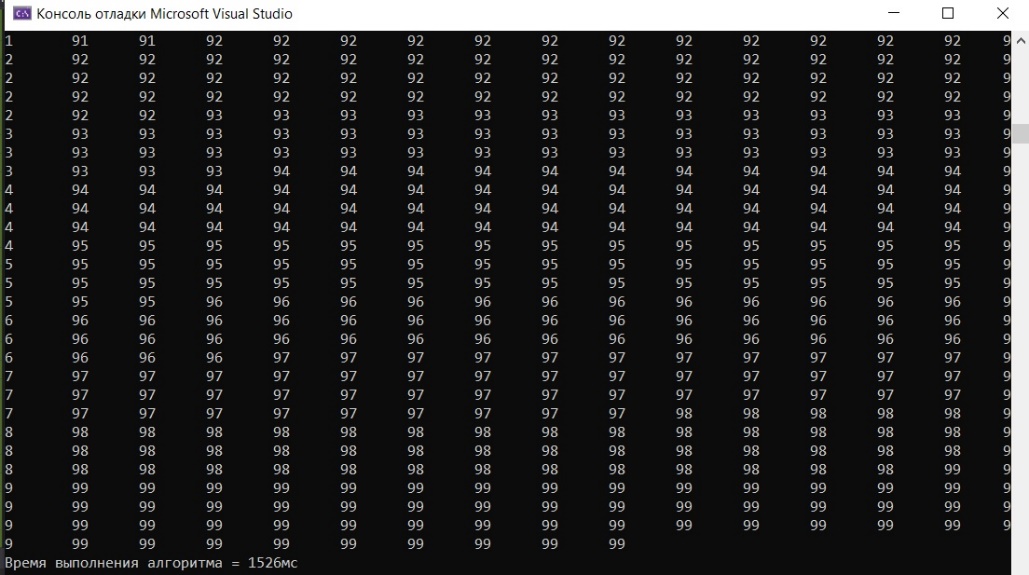


Рисунок 18 – Результат тестирования массива разрядностью 10000

**Приложение Б**

**Листинг**

//This programm sorting array to increasing with method Shell

#include <iostream>

#include <fstream>

#include <windows.h>

int max\_size = 0;

int set\_max\_size()

{

std::cout << "Введите количество элементов в массиве:\n > ";

max\_size = 0;;

std::cin >> max\_size;

return max\_size;

}

void print\_array(int\* array)

{

for (int counter = 0; counter < max\_size; ++counter)

{

std::cout << counter << '\t';

}

std::cout << std::endl;

for (int counter = 0; counter < max\_size; ++counter)

{

std::cout << array[counter] << '\t';

}

std::cout << '\n';

}

bool init\_array(int\* array)

{

for (int counter = 0; counter < max\_size; ++counter)

{

array[counter] = rand() % 201 - 101;

}

return 0;

}

int\* init\_array\_from\_file(int& max\_size)

{

std::cout << "Введите название файла:\n > ";

std::string answer = "";

std::cin >> answer;

std::ifstream file(answer);

file >> max\_size;

if (max\_size < 1)

{

return 0;

}

int\* array = new int[max\_size];

for (int counter = 0; counter < max\_size; ++counter)

{

file >> array[counter];

}

file.close();

return array;

}

bool print\_to\_file(int\* array)

{

std::cout << "Введите название файла для записи:\n > ";

std::string name = "";

std::cin >> name;

std::ofstream file(name);

if (!file)

{

std::cout << "Ошибка открытия файла.\n";

return 1;

}

file << max\_size << std::endl;

for (int counter = 0; counter < max\_size; ++counter)

{

file << array[counter] << ' ';

}

file.close();

return 0;

}

int\* create\_and\_save()

{

std::cout << "Выбирете способ инициализации массива:\n1)Ввести вручную размерность и инициализировать случайно\n2)Из файла\n > ";

int answer = 0;

std::cin >> answer;

int\* array = nullptr;

if (answer == 1)

{

max\_size = set\_max\_size();

if (max\_size < 1)

{

return 0;

}

array = new int[max\_size];

init\_array(array);

}

else if (answer == 2)

{

array = init\_array\_from\_file(max\_size);

}

else

{

std::cout << "Неверный ввод.\n";

system("pause");

return 0;

}

std::cout << "Исходный массив:\n";

print\_array(array);

std::cout << "Сохранить исходный массив в файл?(1 - да/0 - нет)\n > ";

std::cin >> answer;

if (answer)

{

print\_to\_file(array);

}

return array;

}

time\_t sort\_array(int\* array)

{

int buffer = 0;

bool boo = 0;

time\_t start = clock();

for (int distance = max\_size / 2; distance >= 1;)

{

boo = 0;

//std::cout << "\n distance = " << distance << std::endl;

for (int counter = 0; ((counter + distance) < max\_size); counter += (distance))

{

if (array[counter] > array[counter + distance])

{

buffer = array[counter];

array[counter] = array[counter + distance];

array[counter + distance] = buffer;

boo = 1;

}

}

//print\_array(array, max\_size);

if (!boo)

{

distance /= 2;

}

}

return (clock() - start);

}

bool output\_array\_and\_save(int\* array, time\_t time)

{

std::cout << "Конечный массив:\n";

print\_array(array);

std::cout << "Время выполнения алгоритма = " << time << "мс" << std::endl;

bool answer = 0;

std::cout << "Сохранить конечный массив?(1 - да/0 - нет)\n > ";

std::cin >> answer;

if (answer)

{

return print\_to\_file(array);

}

return 0;

}

int main()

{

srand(time(0));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int\* array = create\_and\_save();

if (!array)

{

std::cout << "Ошибка инициализации массива. Программа завершена." << std::endl;

system("pause");

return 0;

}

output\_array\_and\_save(array, sort\_array(array));

delete[] array;

return 0;

}