

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**  
(2022/2023 учебный год)

\_\_\_\_\_  
Китаев Ярослав Евгеньевич

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 29.06.2023 по 12.07.2023

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**

(2022/2023 учебный год)

\_\_\_\_\_  
Китаев Ярослав Евгеньевич

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 29.06.2023 по 12.07.2023

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)

| №<br>п/п | Планируемая<br>форма работы во<br>время практики                           | Количество<br>часов | Календарные сроки<br>проведения работы | Подпись<br>руководителя<br>практики от вуза |
|----------|--|---------------------|--|---|
| 1        | Выбор темы и<br>разработка<br>индивидуального<br>плана проведения<br>работ | 2                   | 29.06.2023 -<br>29.06.2023             |   |
| 2        | Подбор и изучение<br>материала по теме<br>работы                           | 15                  | 30.06.2023 –<br>02.07.23               |   |
| 3        | Разработка<br>алгоритма  | 43                  | 02.07.23 –<br>06.07.23                 |   |
| 4        | Описание<br>алгоритма и<br>программы                                       | 18                  | 6.07.23 –<br>08.07.23                  |   |
| 5        | Тестирование   | 5                   | 08.07.23 –<br>08.07.23                 |   |
| 6        | Получение и<br>анализ результатов  | 10                  | 08.07.23 –<br>10.07.23                 |   |
| 7        | Оформление<br>отчёта   | 15                  | 10.07.23 –<br>12.07.2023               |   |
|          | <b>Общий объём<br/>часов</b>   | 108                 |  |   |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ  
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ОТЧЁТ

О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ

(2022/2023 учебный год)

Китаев Ярослав Евгеньевич

---

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 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

---

Китаев Я.Е. выполнял практическое задание «Сортировка пузырьком». На первоначальном этапе были изучен и проанализирован алгоритм пузырьковой сортировки, был выбран метод решения и язык программирования C\C++, на котором была написана программа сортировки массива методом пузырька. Протестировал и отладил программу. Оформил отчёт.

Бакалавр      Китаев Я.Е.      \_\_\_\_\_      " \_\_\_\_ " \_\_\_\_\_ 2023 г.

Руководитель      Зинкин С.А.      \_\_\_\_\_      " \_\_\_\_ " \_\_\_\_\_ 2023 г.  
практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ  
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

Китаев Ярослав Евгеньевич

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 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

В процессе выполнения практики Китаев Я.Е. решал следующие задачи: создание алгоритма пузырьковой сортировки, анализ работы алгоритма, сравнение существующих методов сортировки.

За период выполнения практики были освоены основные понятия и технологии пузырьковой сортировки. Во время выполнения работы Китаев Я.Е. 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 Китаев Я.Е. заслуживает оценки «      ».

Руководитель практики д.т.н., профессор, Зинкин С.А. «      » 2023

## Содержание

|  |                                     |
|--|-------------------------------------|
| <b>Введение .....</b>                      | <b>7</b>                            |
| <b>1. Постановка задачи.....</b>           | <b>8</b>                            |
| <b>1.1. Достоинства алгоритма .....</b>    | <b>8</b>                            |
| <b>1.2. Недостатки алгоритма .....</b>     | <b>9</b>                            |
| <b>2. Выбор решения.....</b>               | <b>9</b>                            |
| <b>3. Описание программы .....</b>         | <b>11</b>                           |
| <b>4. Схемы программы.....</b>             | <b>13</b>                           |
| <b>4.1. Блок-схема алгоритма .....</b>     | <b>13</b>                           |
| <b>5. Тестирование программы .....</b>     | <b>14</b>                           |
| <b>6. Отладка .....</b>                    | <b>14</b>                           |
| <b>7. Совместная разработка .....</b>      | <b>16</b>                           |
| <b>Заключения .....</b>                    | <b>18</b>                           |
| <b>Список используемых источников.....</b> | <b>19</b>                           |
| <b>Приложение А .....</b>                  | <b>Error! Bookmark not defined.</b> |
| <b>Приложение Б .....</b>                  | <b>Error! Bookmark not defined.</b> |

## **Введение**

Информация является основным элементом передачи сведений в нашей жизни. С давних времён люди поняли, что данные удобнее хранить и передавать в структурированном виде. Примером такого хранения информации является библиотека. Книги сортируют по жанру, месту создания и многим другим критериям.

Раньше люди занимались сортировкой большого объёма данных вручную, и зачастую это был очень долгий и трудоёмкий процесс. За тысячи лет способы передачи, хранения и обработки информации эволюционировали. В наше время обработка данных происходит автоматически, при помощи специализированных программ.

Для этого они используют различные алгоритмы сортировки данных. Сортировка это последовательное расположение или разбиение на группы чего-либо в зависимости от выбранного критерия. За всю историю человечества было придумано огромное множество различных алгоритмов сортировок. От самых простых, например, «сортировка пузырьком» до сложных «timsort».

## **1. Постановка задачи**

Постановка задачи — важный этап разработки любой программы, без которого нельзя начать ни один проект. Важно правильно поставить задачу, следуя которой, получится реализовать программу.

Для постановки задачи необходимо разработать алгоритм сортировки пузырьком. Для этого должны быть изучены следующие вопросы:

- способы работы с консолью
- реализация функций для работы с файлами
- алгоритм сортировки пузырьком

После необходимо разделить разработку программы на части, которыми займутся разные участники группы. Разработка программы подразумевает собой работу с интерфейсом, обработку данных и вывод данных. Первый участник займется созданием интерфейса, конечная программа должна позволить пользователю выбирать, какие действия он хочет выполнить. Второй участник займется обработкой информации, он должен обеспечить правильную работу генератора чисел, который будет заполнять данные. Также этот человек должен написать алгоритм сортировки чисел, согласно заданию. Третий участник группы должен использовать данные, полученные 2 участником, и организовать систему их вывода, чтобы пользователь мог проверить правильность работы алгоритма.

### **1.1. Достоинства алгоритма**

Алгоритм сортировки пузырьком имеет множество преимуществ. Он отлично подходит для учебных целей, он прост в освоении и на его основе можно сделать более совершенные виды сортировки: сортировка расческой и шейкером.

- Простота реализации. Создаётся цикл, который поочерёдно сравнивает 2 соседних элемента и, если левый элемент больше правого, меняете их местами.



- Возможность оптимизации процесса. Можно уменьшать каждое последующее количество проходов по массиву на 1 элемент
- Если массив уже отсортирован, программа быстро завершит работу
- Допускает эффективный проход как от начала к концу, так и от конца к началу.

## **1.2. Недостатки алгоритма**

Не смотря на все плюсы, алгоритм сортировки пузырьком также имеет множество недостатков, которые обусловлены его простотой. Пузырьковая сортировка имеет временную сложность  $O(N^2)$ , низкая скорость выполнения алгоритма сортировки не позволяет быстро и эффективно сортировать большое количество чисел. Алгоритм является учебным, он плохо подходит для использования в программе или для тестирования. При большом количестве элементов, цикл может повторяться миллионы раз.

Владимир Добосиевич объяснил, почему метод сортировки пузырьком является таким неэффективным. Причиной тому являются черепахи - небольшие по значению элементы, которые находятся в конце списка. Элементы, имеющие маленькое значение довольно быстро сортируются, а большой элемент в начале списка будет очень долго двигаться к концу.

Метод сортировки пузырьком основан на сравнении, что требует использование оператора сравнения, это значительно увеличивает время выполнения операций.

## **2. Выбор решения**

Язык программирования C имеет множество инструментов для работы с разными данными, в том числе и числами. Часто программе нужно уметь обрабатывать большое количество информации, для этого существуют различные алгоритмы сортировки, позволяющие структурировать информацию.

Сортировка пузырьком — один из самых известных алгоритмов сортировки. Здесь нужно последовательно сравнивать значения соседних элементов и менять числа местами, если предыдущее оказывается больше последующего. Таким образом элементы с большими значениями оказываются в конце списка, а с меньшими остаются в начале (Рисунок 1).

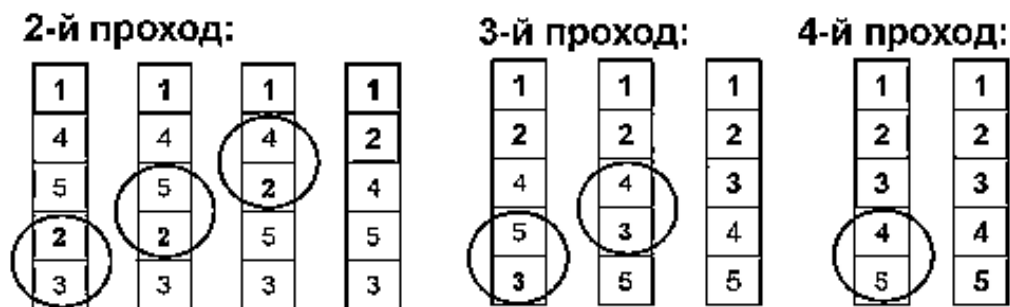


Рисунок 1 – алгоритм работы сортировки методом пузырька.

Общая идея алгоритма состоит в следующем:

- Создать цикл, в который будет выполняться, пока все элементы не встанут на свое место. Элементы по итогу должны встать в порядке возрастания.
- Сравнивать 2 рядом стоящих элемента массива чисел, проверяя какой элемент больше. Если левый элемент больше, то их меняют местами. И переходит к следующим элементам.
- Повторять алгоритм до тех пор, пока самые большие числа не будут находиться у правого края массива, а маленькие — у левого.

Для вывода удобного для пользователя интерфейса было решено использовать оператор `switch()`. Он удобен для создания меню программы, так как позволяет реализовать выбор тех или иных действий с помощью ввода символов.

Важно сделать системы ввода-вывода информации. Для ввода размера массива пользователь будет вводить информацию в консоль. Для вывода неотсортированного и отсортированного массива, программа будет

записывать данные в текстовый документ. Оно использует стандартные библиотеки ввода-вывода, которые позволяют открывать, закрывать и работать с файлами на диске. Функциональность ввода-вывода языка Си по текущим стандартам реализуется на низком уровне. Язык Си превращает все операции с файлами в операции с потоками байтов, которые могут быть как «потоками ввода», так и «потоками вывода». Однако в отличие от некоторых ранних языков программирования, язык Си не имеет прямой поддержки произвольного доступа к файлам данных; Из-за этого приходится создавать поток, ищущий ту или иную часть файла, прежде чем работать с ней.

### 3. Описание программы

Для сортировки пузырьком были подключены следующие заголовочные файлы: *iostream* – заголовочный файл с классами, функциями и переменными для организации ввода-вывода; ***Header*** – заголовочный файл, предоставляющий функционал для считывания данных из файла и для записи в файл; *stdio.h* - заголовочный файл стандартной библиотеки языка Си; *stdlib.h* - заголовочный файл стандартной библиотеки языка Си, который содержит в себе функции, занимающиеся выделением памяти, контролем процесса выполнения программы, преобразованием типов и другие; *time.h* - заголовочный файл стандартной библиотеки языка программирования С, содержащий типы и функции для работы с датой и временем;

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <iostream>
```

Алгоритм сортировки пузырьковым методом.

```
#include "Header.h"

int sortid(int size, int* mas)
{
    long long int trans = 0;
    int save;
```

```

    for (int exit = 1; exit < size; exit++) {
        for (int r = 0; r < size - exit; r++) {
            if (mas[r] > mas[r + 1]) {
                save = mas[r];
                mas[r] = mas[r + 1];
                mas[r + 1] = save;
                trans++;
            }
        }
    }
    return 0;
}

```

Наш алгоритм состоит из двух циклов `for` и содержится в функции `sortid.cpp`.

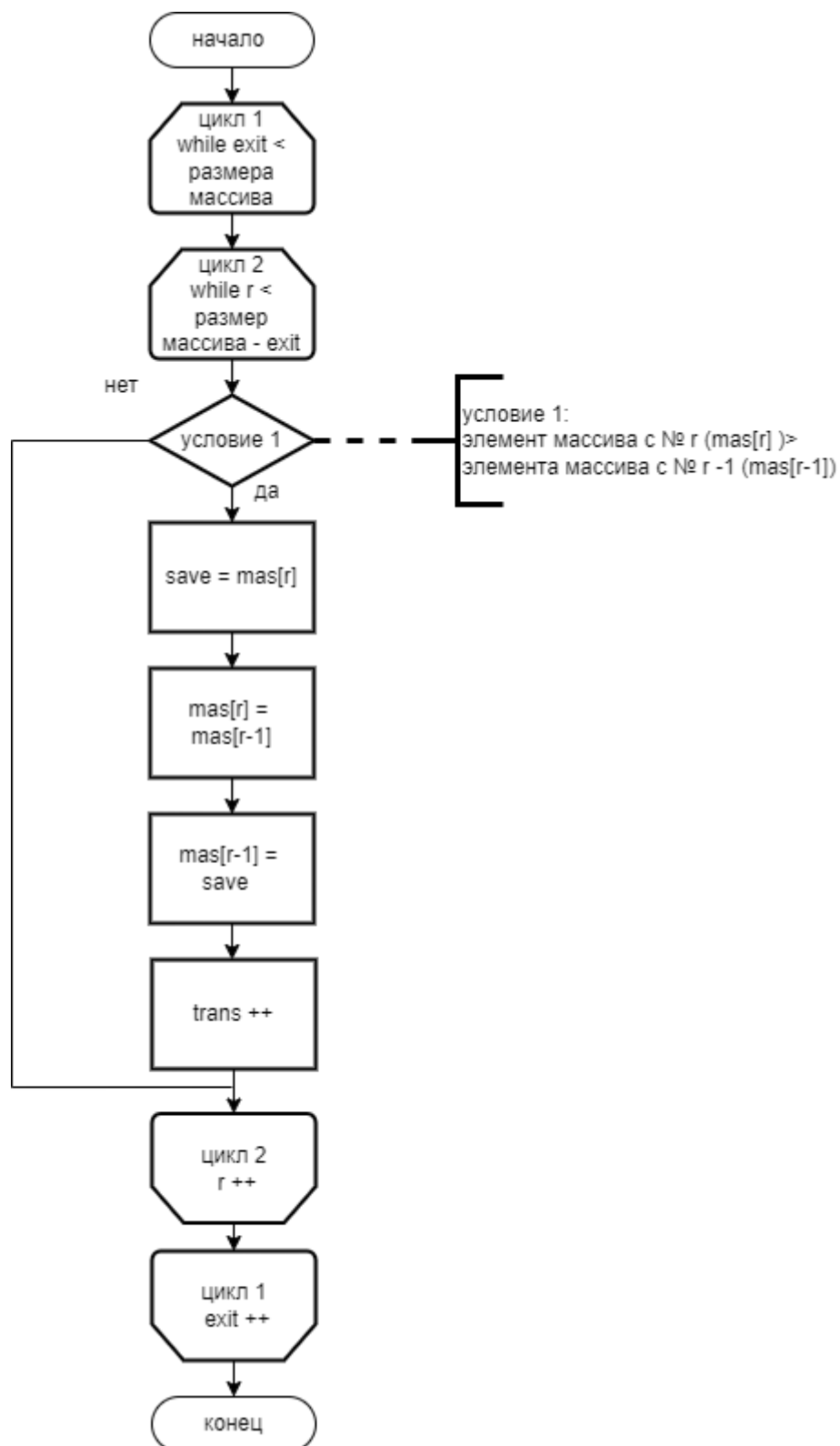
Первый цикл изменяет переменную `exit`, которая показывает сколько элементов с конца не должен проверять второй цикл. После первого прохода на последнее место встанет максимальный элемент, поэтому первый цикл добавит к переменной `exit` единицу. Цикл будет продолжаться, пока переменная `exit` не станет равна размеру массива.

Второй цикл проверяет массив от начала до (размер массива – `exit`), сравнивая два соседних элемента, если число содержащееся в массиве под номером `r` больше числа содержащегося в массиве под номером `r+1`, то программа меняет эти элементы местами.

## 4. Схемы программы

### 4.1. Блок-схема алгоритма

Схема программы. Функция Sortid.cpp



## 5. Тестирование программы

Тестирование выявило, что время, затраченное на работу программы относительно количества элементов, увеличивается не линейно, то есть количество элементов и затраченное время находятся в квадратичной зависимости.

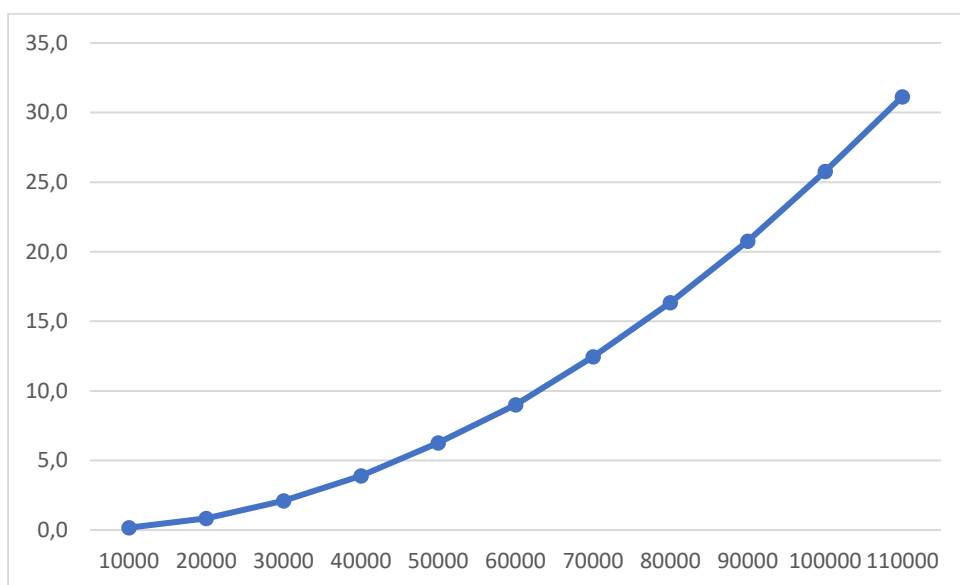


Рисунок 2 – Результаты тестирования

## 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 останова и пошаговое выполнение кода программы, анализ содержимого локальных переменных.

Точки останова – это места в коде, которые позволяют остановить программу и включить отладку. Они позволяют изучить поведение кода и его функции, чтобы попытаться определить ошибку ( рисунок 3 – 5)

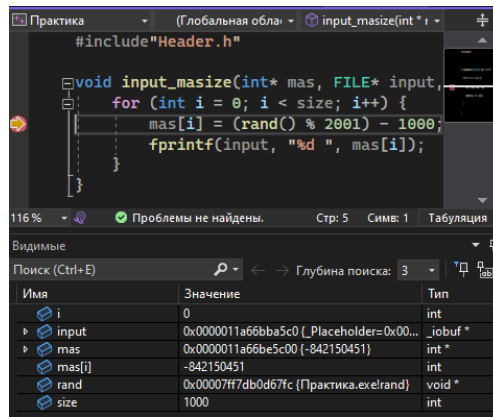


Рисунок 3 – протокол трассировки программы

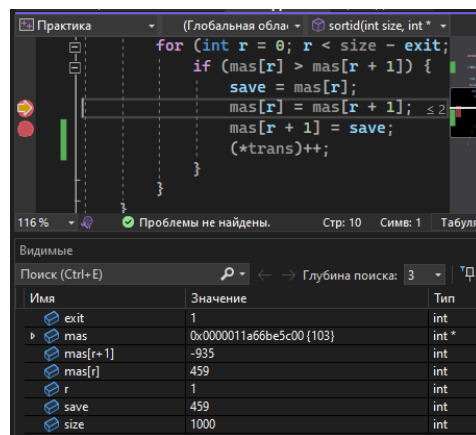


Рисунок 4 – протокол трассировки программы

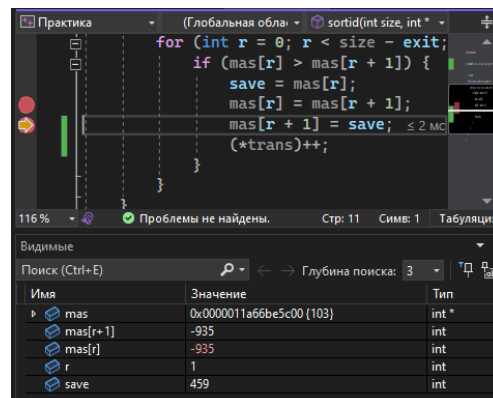


Рисунок 5 – протокол трассировки программы

## 7. Совместная разработка

Для совместной разработки мы использовали крупнейший веб-сервис для хостинга IT-проектов - github.

Помимо возможности передачи проекта от одного участника команды к другому, github содержит полезные функции сохранения промежуточных результатов работы и, в случае ошибки, возвращения к ним.

Для удобства мы решили разбить проект на несколько этапов, которые должны выполняться друг за другом, то есть выбрали каскадный метод разработки программы. Каждый участник выбрал себе по задаче, которую должен был реализовать.

Для начала был создан модуль для работа с файлами и заполнение массива случайными числами от -1000 до 1001.

Потом была разработана программа, которая выполняет сортировку массива.

И в конце для готового алгоритма было разработано «Меню».

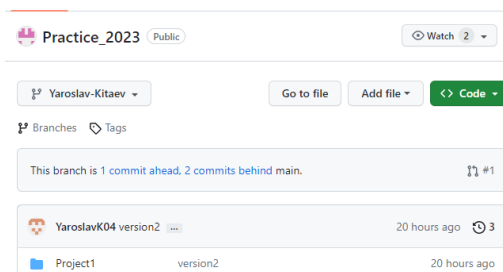


Рисунок 6 — личная ветка студента, ответственного за разработку алгоритма.

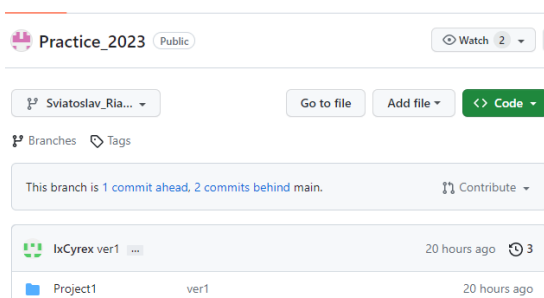


Рисунок 7 — личная ветка студента, ответственного за разработку системы ввода/вывода.



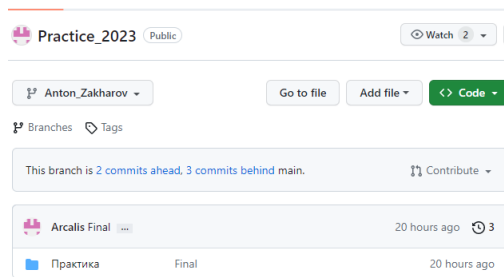


Рисунок 8 — личная ветка студента, ответственного за разработку интерфейса.

Ссылка на репозиторий: [https://github.com/Arcalis/Practice\\_2023.git](https://github.com/Arcalis/Practice_2023.git)

## **Заключения**

При выполнении данной практической работы были получены навыки совместной разработки программ. Был освоен алгоритм пузырьковой сортировки. Также были получены навыки отладки, работы с файлами и тестирования программ в среде Visual Studio 2022 на языках Си/C++.

В рамках практической работы была разработана программа сортировки массива пузырьковым методом.

### **Список используемых источников**

1. Клеменс Б. Язык С в XXI веке / Клеменс Б. – ДМК-Пресс, 2018 г. – 376 с.
2. Керниган Б. Ритчи Д. Язык программирования Си / Керниган, Б. Ритчи Д. – Вильямс, 2019 г. – 288 с.
3. Березин Б.И. Березин С.Б.. Начальный курс С и С++ 1996

## Приложение А.

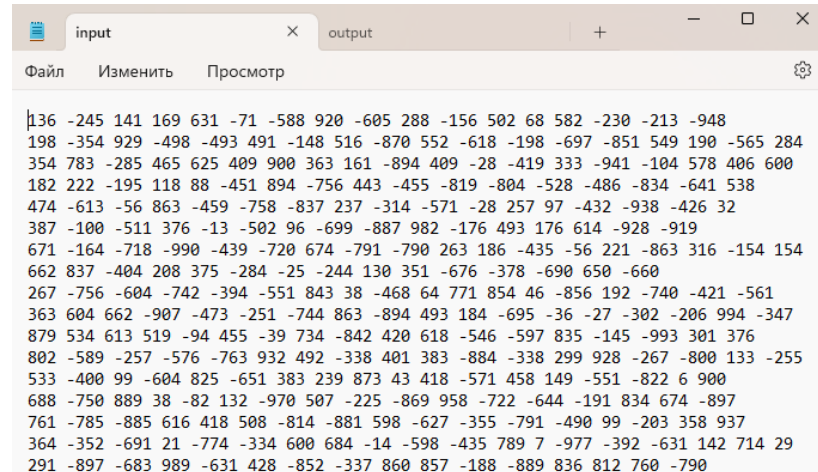
### Результаты работы программы

Сортировка выполнена успешно! Результат находится в файле output.txt

Количество перестановок – 99725874.

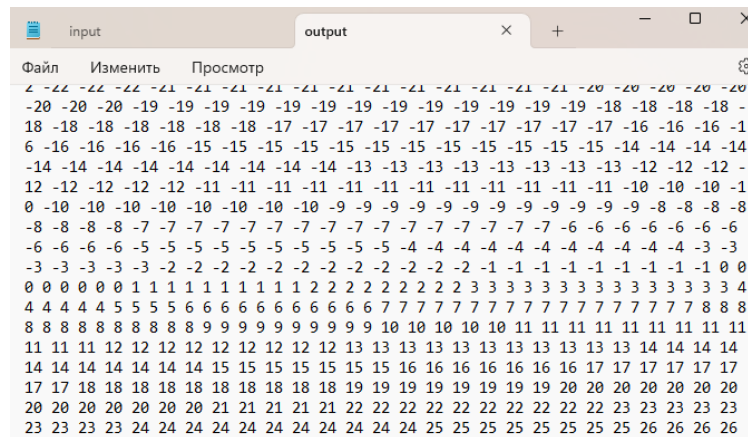
Время выполнения сортировки – 0,828 с.

Рисунок А.1



```
136 -245 141 169 631 -71 -588 920 -605 288 -156 502 68 582 -230 -213 -948
198 -354 929 -498 -493 491 -148 516 -870 552 -618 -198 -697 -851 549 190 -565 284
354 783 -285 465 625 409 900 363 161 -894 409 -28 -419 333 -941 -104 578 406 600
182 222 -195 118 88 -451 894 -756 443 -455 -819 -804 -528 -486 -834 -641 538
474 -613 -56 863 -459 -758 -837 237 -314 -571 -28 257 97 -432 -938 -426 32
387 -100 -511 376 -13 -502 96 -699 -887 982 -176 493 176 614 -928 -919
671 -164 -718 -990 -439 -720 674 -791 -790 263 186 -435 -56 221 -863 316 -154 154
662 837 -404 208 375 -284 -25 -244 130 351 -676 -378 -690 650 -660
267 -756 -604 -742 -394 -551 843 38 -468 64 771 854 46 -856 192 -740 -421 -561
363 604 662 -907 -473 -251 -744 863 -894 493 184 -695 -36 -27 -302 -206 994 -347
879 534 613 519 -94 455 -39 734 -842 420 618 -546 -597 835 -145 -993 301 376
802 -589 -257 -576 -763 932 492 -338 401 383 -884 -338 299 928 -267 -800 133 -255
533 -400 99 -604 825 -651 383 239 873 43 418 -571 458 149 -551 -822 6 900
688 -750 889 38 -82 132 -970 507 -225 -869 958 -722 -644 -191 834 674 -897
761 -785 -885 616 418 508 -814 -881 598 -627 -355 -791 -490 99 -203 358 937
364 -352 -691 21 -774 -334 600 684 -14 -598 -435 789 7 -977 -392 -631 142 714 29
291 -897 -683 989 -631 428 -852 -337 860 857 -188 -889 836 812 760 -790
```

Рисунок А.2 – файл input



```
4 -44 -44 -44 -41 -41 -41 -41 -41 -41 -41 -41 -41 -41 -41 -41 -40 -40 -40 -40
-20 -20 -20 -19 -19 -19 -19 -19 -19 -19 -19 -19 -19 -19 -19 -18 -18 -18 -18 -
18 -18 -18 -18 -18 -18 -18 -17 -17 -17 -17 -17 -17 -17 -17 -17 -16 -16 -16 -1
6 -16 -16 -16 -16 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -14 -14 -14
-14 -14 -14 -14 -14 -14 -14 -14 -14 -13 -13 -13 -13 -13 -13 -13 -13 -12 -12 -12
-12 -12 -12 -12 -12 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -11 -10 -10 -10
0 -10 -10 -10 -10 -10 -10 -10 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -8 -8 -8
-8 -8 -8 -8 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -6 -6 -6 -6 -6
-6 -6 -6 -6 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -4 -4 -4 -4 -4
-3 -3 -3 -3 -3 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -1 -1 -1 -1 -1 -1
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 10 10 10 10 10 11 11 11 11 11 11 11
11 11 11 12 12 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13 13 13 13 13 13 14 14 14 14
14 14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 16 16 16 16 16 16 16 16 16 17 17 17 17 17 17
17 17 18 18 18 18 18 18 18 18 18 18 18 18 18 19 19 19 19 19 19 19 19 19 19 20 20 20 20 20 20
20 20 20 20 20 20 20 21 21 21 21 21 21 21 22 22 22 22 22 22 22 22 22 22 22 22 22 23 23 23 23
23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 26 26 26 26
```

Рисунок А.3 – файл output

## Приложение Б

### Листинги программы

#### Файл main.cpp

```
#include "Header.h"

int main()
{
    srand(time(NULL));
    FILE* input, * output;
    int size;
    bool work = true;
    int* mas;
    int x = 0;

    input = fopen("input", "w+");
    output = fopen("output", "w+");

    setlocale(LC_ALL, "Rus");
    printf("Добро пожаловать! Данная программа выполняет сортировку массива случайно  
сгенерированных чисел.\n\nСначала введите размер массива:\n");
    scanf_s("%d", &size);

    while (size <= 1)
    {
        printf("Попробуйте еще раз.");
        scanf_s("%d", &size);
    }
    mas = new int[size];

    while (work == true)
    {
        system("cls");
        printf("Выберете следующее действие:\n1. Изменить размер массива.\n2.  
Заполнить массив случайными числами.\n3. Отсортировать массив.\n");

        getchar();
        char comand[] = { getchar() };
        switch (comand[0])
        {
            case '1':
                printf(" Введите количество чисел: ");
                scanf_s("%d", &size);

                while (size <= 1)
                {
                    printf("Попробуйте еще раз.");
                    scanf_s("%d", &size);
                }
                mas = new int[size];

                break;

            case '2':
                input_masize(mas, input, size);
                break;
            case '3':
                sortid(size, mas);
                output_mas(size, mas, output);
                work = false;
                break;
        }
    }
}
```

```

    }
}
return 0;
}

```

### Файл sortid.cpp

```

#include"Header.h"

int sortid(int size, int* mas)
{
    double start = GetTickCount();
    long long int trans = 0;
    int save;
    for (int exit = 1; exit < size; exit++) {
        for (int r = 0; r < size - exit; r++) {
            if (mas[r] > mas[r + 1]) {
                save = mas[r];
                mas[r] = mas[r + 1];
                mas[r + 1] = save;
                trans++;
            }
        }
    }
    double final = GetTickCount() - start;
    system("cls");
    printf("Сортировка выполнена успешно! Результат находится в файле
output.txt\nКоличество перестановок - %lli.\nВремя выполнения сортировки - %g с.",
trans, final / 1000);
    return 0;
}

```

### Файл input.cpp

```

#include"Header.h"

void input_masize(int* mas, FILE* input, int size) {
    for (int i = 0; i < size; i++) {
        mas[i] = (rand() % 2001) - 1000;
        fprintf(input, "%d ", mas[i]);
    }
}

```

### Файл output.cpp

```

#include"Header.h"

void output_mas(int size, int *mas, FILE* output) {
    for (int i = 0; i < size; i++) {
        fprintf(output, "%d ", mas[i]);
    }
}

```