|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное автономное образовательное учреждение высшего образования  «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» |

Кафедра информационно-сетевых технологий (№53)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ОТЧЁТ ПО ПРАКТИКЕ  ЗАЩИЩЁН С ОЦЕНКОЙ  Руководитель |  | | | |
| ассистент |  |  |  | Гуков С.Ю. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | учебная | |
| тип практики |  | |
| на тему индивидуального задания | | Реализация и практическое сравнение |
| производительности различных алгоритмов сортировок | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнен | Примеровым Примером Примеровичем |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.02 |  | Информационные системы и технологии |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 09.03.02 |  | Информационные технологии в дизайне |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 5131 |  | 04.06.2022 |  | Примеров П.П. |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2022

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

09.03.02 Информационные системы и технологии

на прохождение *учебной* практики обучающегося направления подготовки/ специальности

Примеров Пример Примерович

1. Фамилия, имя, отчество обучающегося:

5131

1. Группа:
2. Тема индивидуального задания:

Реализация алгоритмов сортировки и анализа текстовых данных

Visual Studio или Xcode; вариант задания (кириллица, по алфавиту, по возрастанию,

1. Исходные данные:

игнорировать числа, быстрая сортировка); десять текстов (от 3 тыс. до 15 тыс. символов)

1. Содержание отчетной документации:
   1. индивидуальное задание;
   2. отчёт, включающий в себя:
   * титульный лист;
   * материалы о выполнении индивидуального задания (содержание определяется кафедрой);
   * выводы по результатам практики;
   * список использованных источников.
   1. отзыв руководителя от профильной организации (при прохождении практики в профильной организации).
2. Срок представления отчета на кафедру: «04» июня 2022 г.

04.06.2022

ассистент

Гуков С.Ю.

Руководитель практики

должность, уч. степень, звание подпись, дата инициалы, фамилия

СОГЛАСОВАНО

Руководитель практики от профильной организации

должность подпись, дата инициалы, фамилия

Задание принял к исполнению:

Примеров П.П.

07.02.2022

Обучающийся

дата подпись инициалы, фамилия

Санкт–Петербург 2022

# ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 1](#_bookmark0)

1. [Индивидуальный вариант 2](#_bookmark1)
2. [Блок-схема алгоритма программы 3](#_bookmark2)
3. [Описание алгоритма программы 4](#_bookmark3)
   1. [Описание основной функции main 4](#_bookmark4)
   2. [Описание функции for\_sort\_create 4](#_bookmark5)
   3. [Описание функций word\_len и word\_len\_amount 5](#_bookmark6)
   4. [Описание функции shell\_sort 5](#_bookmark7)
4. [Тестирование программы и графики 6](#_bookmark8)
5. [Работа с Github 10](#_bookmark9)
   1. [Загрузка репозитория 10](#_bookmark10)
   2. [Файл README.md 11](#_bookmark11)

[ЗАКЛЮЧЕНИЕ 13](#_bookmark12)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_bookmark13)

[ПРИЛОЖЕНИЕ А 15](#_bookmark14)

# ВВЕДЕНИЕ

Данный отчёт является приложением к выполненному заданию по учебной практике и демонстрацией результатов для оценки по работе. В процессе выполнения учебной практики различных трудностей в связи с недостатком прикладных знаний по дисциплине не возникало. Работа выполнена на основе учебной программы весеннего семестра с использованием источников, представленных в списке литературы.

Необходимо реализовать программу, которая будет считывать текст из файла и с минимальными погрешностями разбивать его на массив отдельных слов. Далее, согласно параметрам выбранного варианта, необходимо отсортировать этот массив слов. Путь либо название считываемого файла должно вводиться через консоль. Отсортированный массив необходимо вывести в выходной файл. В консоль результат выводить не нужно. Также необходимо вывести информацию с простейшим анализом введенного текста и в консоль, и в отдельный файл с данными:

* исходный текст
* параметры выбранного варианта
* количество слов в исходном тексте
* время выполнения сортировки
* количество слов на каждую букву алфавита

Необходимо провести 10 тестов, то есть повторить такие действия с разными по длине (от 3 тыс. до 15 тыс. символов) и структуре текстами. Соответственно, должно получиться 30 файлов – 10 файлов с входными текстами, 10 с отсортированными массивами и 10 с анализированием текстов.

# Индивидуальный вариант

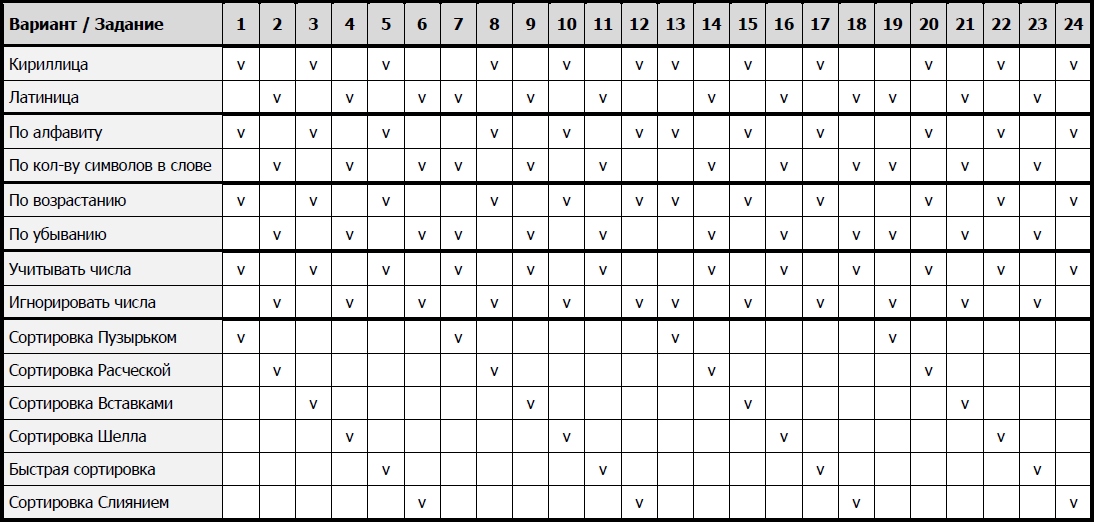
Индивидуальные параметры для каждого варианта представлены на рисунке (рисунок 1)

Рисунок 1 – Параметры вариантов

Выполненный вариант работы – 22. Следовательно необходимо было обработать текст кириллицы, разбив его на слова с учетом чисел, отсортировать эти слова по возрастанию сортировкой Шелла и вывести в выходной файл.

# Блок-схема алгоритма программы

Блок-схема алгоритма программы представлена на рисунке 2

Рисунок 2 – Блок-схема алгоритма

# Описание алгоритма программы

# Описание основной функции main

# Подключение поддержки кириллицы в консоли

# Создание переменной name\_file типа string, в которую с консоли запишется название файла, введенное пользователем

# Создается переменная text, в которую записывается вывод функции getText

# Создается вектор строк array\_of\_words, в него записывается вывод функции separateWords

# Вызывается функция sort

# Вызывается функция writeToResult

# Создается вектор чисел, в него записывается вывод функции getArrayCountWords

# Создается переменная words\_count, в нее записывается размер вектора строк array\_of\_words

# Вызывается функция writeToAnalysis

# Описание функции getText

# Функция принимает переменную имени файла name\_file, возвращает строку

# Создается пустая строка text

# Создается переменная file\_original типа ifstream, для работы с файлом

# Открывается файл с именем original + name\_file

# Если файл открылся, то считываем символ из файла, если получилось, то прибавляем символ к исходной строке

# Закрываем файл

# Если файл не открылся, выводим в консоль, что исходный файл не открылся и досрочно завершаем программу

# Описание функции separateWords

# Функция принимает исходный текст, возвращает массив слов

# Создается пустой вектор строк array\_of\_words, и пустая строка s

# К строке два раза добавляем кавычки по обе стороны

# Создаем строки с русскими алфавитами и числами

# Запускается цикл, который идет по всем символам строки от 2 до длины строки – 2, счетчик i

# В нем есть условие: если итый символ есть в нижнем или верхнем регистре кириллицы, то прибавляем к строке s этот символ. Если cимвол после итого символа не буква и не тире, то мы добавляем в массив слов строку s, после снова делаем ее пустой

# Иначе: если символ цифра или символ запятая, а после идет цифра, то добавляем к строке s этот символ. Если следующий символ не равен цифре и запятой, или послеследующий символ не равен цифре, а следующий символ равен запятой, то добавляем в массив слов строку s, после снова делаем ее пустой

# Иначе: если символ '-', а следующий символ цифра или буква и s не пустая строка, то прибавляем к строке s этот символ

# Описание функции writeToResult

# Функция принимает строку имени файла и вектор слов

# Создается переменная файла, и открывается файл

# Проходимся по всему вектору слов и записываем каждое слово в файл

# Закрываем файл

# Описание функции getArrayCountWords

# Функция принимает вектор слов, возвращает вектор количества слов на каждую букву

# Создается вектор длиной 33, каждый элемент которого, количество слов, первая буква имеет в алфавите номер равный индексу элемента

# Проходимся по всему вектору слов и если слово не равно числу, и если нашли в верхнем регистре, то увеличиваем элемент с индексом равным номеру большой буквы в верхнем регистре, иначе увеличиваем элемент с индексом равным номеру маленькой буквы в нижнем регистре

# Описание функции sort

# Функция принимает вектор слов со ссылкой, возвращает время сортировки

# Создается строк с цифрами и минусом

# Создаем вектор чисел и вектор слов

# Проходимся по всему вектору слов и если первая буква слов – это число или -, то добавляем слово в массив чисел, иначе добавляем слово в субмассив слов

# Создается переменная start\_time типа int, в нее записывается количество временных тактов, прошедших с начала запуска программы

# Запускаем цикл for, со счетчиком step равным длине субмассива слов, деленной на два, до момент пока step > 0, каждую итерацию делим step на два

# Запускаем второй цикл for со счетчиком i равным step, пока i < длины субмассива слов, каждую итерацию увеличиваем i на один

# Переменной tmp присваиваем значение элемента субмассива слов с индексом i. Переменной number\_tmp присваиваем вывод функции findInAlphabet (место в алфавите первой буквы данного слова)

# Запускаем третий цикл for со счетчиком j равным i, пока j >= step, каждую итерацию уменьшаем j на step и если место в алфавите первой буквы слов из субмассива с индексом j – step > number\_tmp присваиваем слову из субмассива с индексом j слово из субмассива с индексом j – step

# После завершения третьего цикла присваиваем слову из субмассива с индексом j значение переменной tmp

# Аналогично сортируем массив чисел, только в этом случае мы сравниваем не строки, а строки, преобразованные в дробные числа, если такое преобразование возможно

# Создается переменная end\_time типа int, в нее записывается количество временных тактов, прошедших с начала запуска программы

# Потом мы сливаем два получившихся массив в один

# Описание функции isNumber

# Функция принимает строку s, возвращает булево значение

# Пытаемся преобразовать строку в число, если получается выводим один, если случается ошибка, ловим исключение и выводим ноль

# Описание функции findInAlphabet

# Функция принимает символ a, возвращает номер этого символа в алфавите

# Создаем строки с нижним и верхним регистрами

# Если символ из верхнего регистра, возвращаем номер из верхнего регистра, иначе возвращаем номер из нижнего регистра.

# Описание функции writeToAnalysis

# Функция принимает строку имени файла, строку с тектом, количество слов, вектор количества слов на каждую букву и время

# Создается переменная файла, и открывается файл

# Записывается в файл информация

# Выводится в консоль информация

# Создается строка с нижним регистром

# Проходимся по всему вектору количества слов на каждую букву и выводим соответсвующую букву из нижнего регистра и элемент соответствующий элемент этого вектора

# Закрываем файл

# Тестирование программы и графики

# Для проверки кол-ва слов счётчиком является Microsoft Word. На

# рисунках 3-11 представлены тесты работы программы на 2 текстах

Рисунок 3. Исходный текст №3

# 

Рисунок 4. Отсортированный текст №3

# 

Рисунок 5. Анализ текста №3

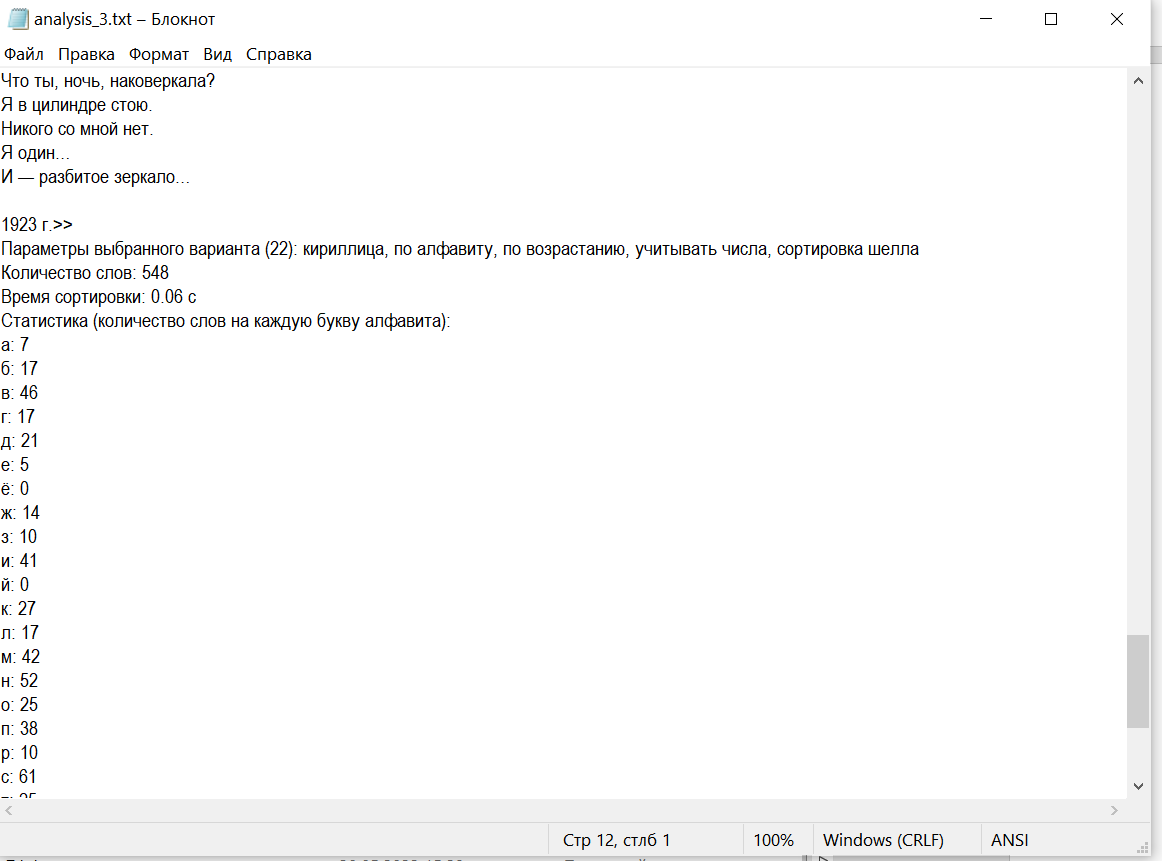


Рисунок 6. Анализ текста №3

Для текста №3 Microsoft Word насчитал 548 слов, программа насчитала 548.

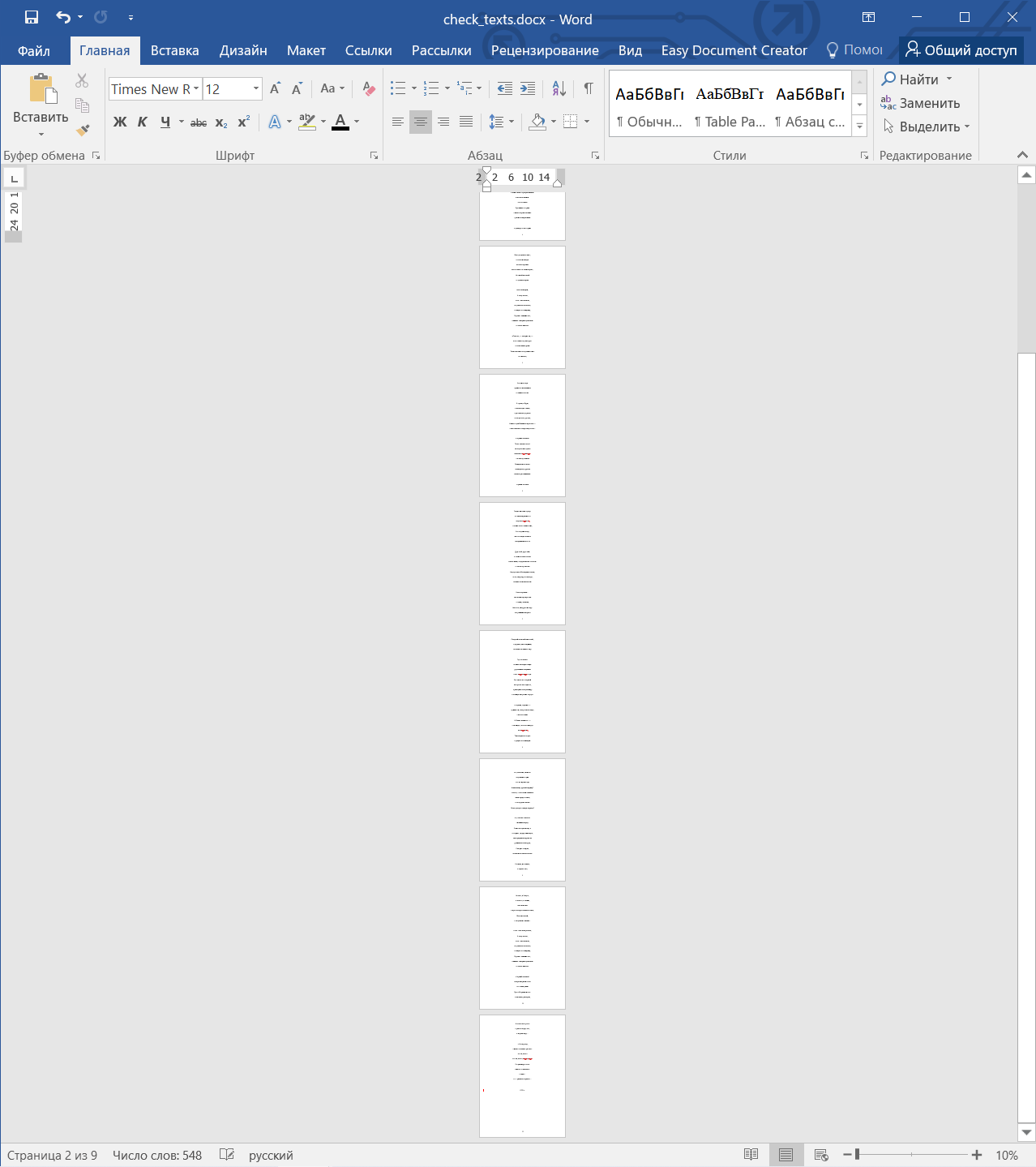


Рисунок 7. Проверка текста №3

# 

Рисунок 8. Исходный текст №5

# 

Рисунок 9. Отсортированный текст №5

# 

Рисунок 10. Анализ текста №5

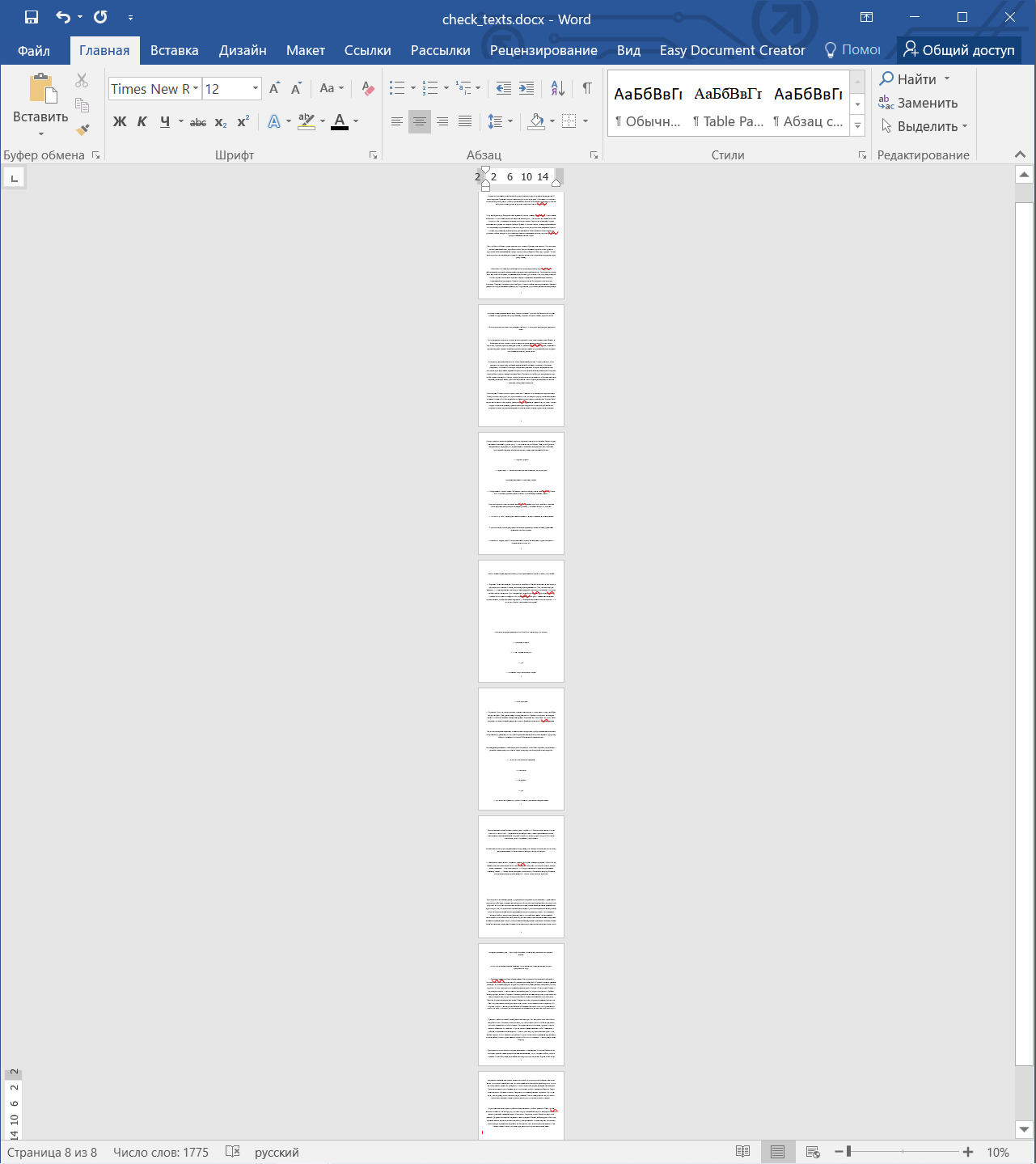


Рисунок 11. Проверка текста №5

Для текста №3 Microsoft Word насчитал 1775 слов, программа насчитала 1776.

График зависимости времени выполнения сортировки от количества слов представлен на графике 1

График 1 - Зависимость времени сортировки от кол-ва символов

# Работа с Github

# Загрузка репозитория

# В процессе написания кода, также была осуществлена практика в выгрузке программы на github. Было загружено 6 коммита в репозиторий для учебной практики. Также был создан файл README.md, описывающий принцип работы программы. Файл содержит в себе область применения программы, скриншоты с исходными данными, результатами работы, описания блоков кода на языке C++; Загрузка коммитов показана на рисунке 12

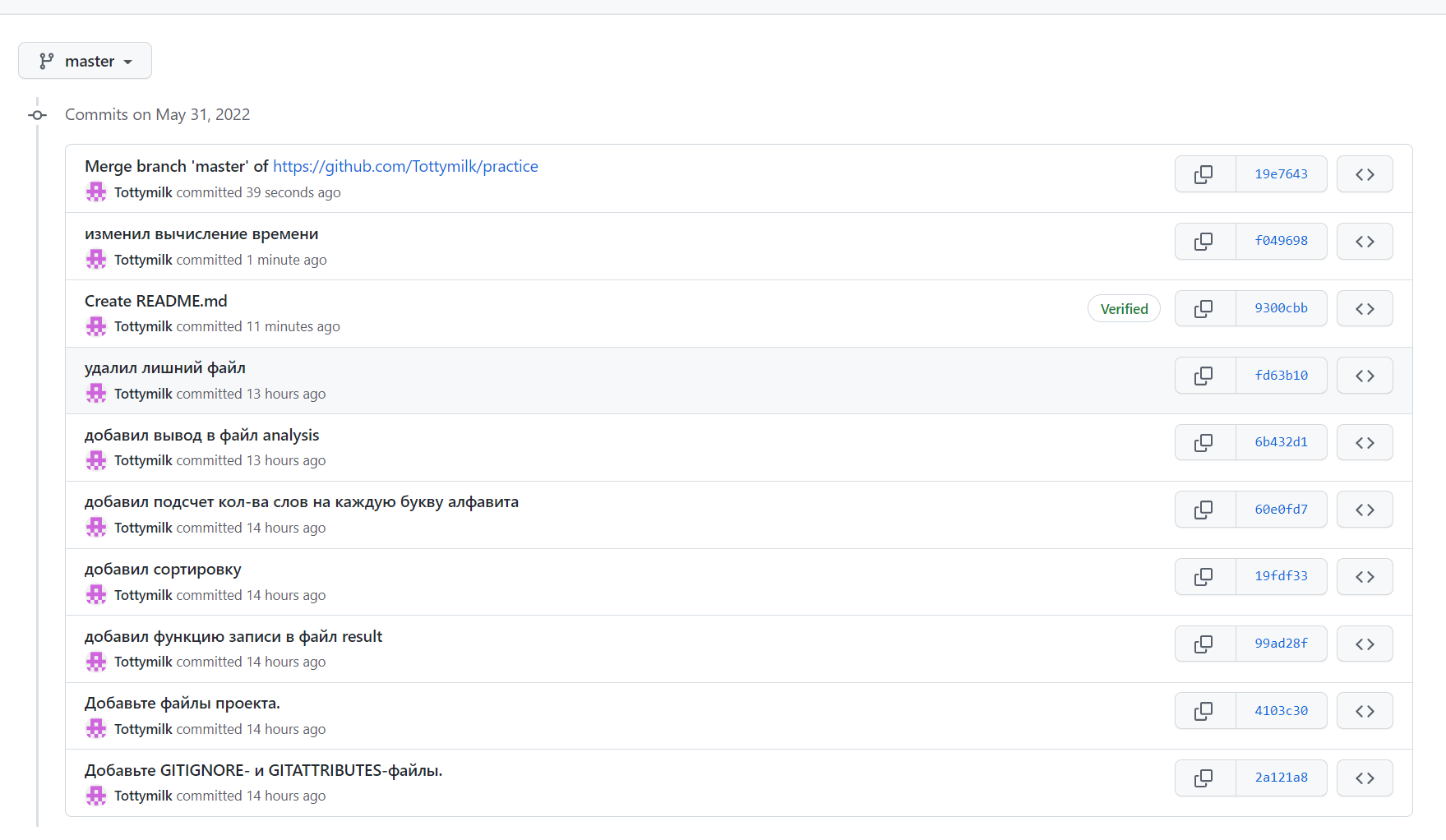


Рисунок 12 – Коммиты на Github

Процесс загрузки в репозиторий достаточно прост: По мере написания кода периодически необходимо обновлять его в репозитории после каждого «идейного» изменения. Сначала необходимо выбрать вкладку «Git» на верхней панели в visual studio, что представлено на рисунке 18

# 