|  |
| --- |
| 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](#_Toc104910836)

[1 Индивидуальный вариант 2](#_Toc104910837)

[2 Блок-схема алгоритма программы 3](#_Toc104910838)

[3 Описание алгоритма программы 4](#_Toc104910839)

[3.2 Описание функции getText 4](#_Toc104910850)

[3.3 Описание функции separateWords 5](#_Toc104910858)

[3.4 Описание функции writeToResult 6](#_Toc104910867)

[3.5 Описание функции getArrayCountWords 6](#_Toc104910872)

[3.6 Описание функции sort 7](#_Toc104910876)

[3.7 Описание функции isNumber 8](#_Toc104910890)

[3.8 Описание функции findInAlphabet 8](#_Toc104910893)

[3.9 Описание функции writeToAnalysis 8](#_Toc104910897)

[4 Тестирование программы и графики 10](#_Toc104910905)

[4.1 Проверка кол-ва слов 10](#_Toc104910906)

[4.2 Анализ сложности 19](#_Toc104910913)

[5 Работа с Github 20](#_Toc104910915)

[5.1 Загрузка репозитория 20](#_Toc104910916)

[5.2 Файл README.md 23](#_Toc104910918)

[ЗАКЛЮЧЕНИЕ 24](#_Toc104910919)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 26](#_Toc104910920)

[ПРИЛОЖЕНИЕ А 27](#_Toc104910921)

# 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

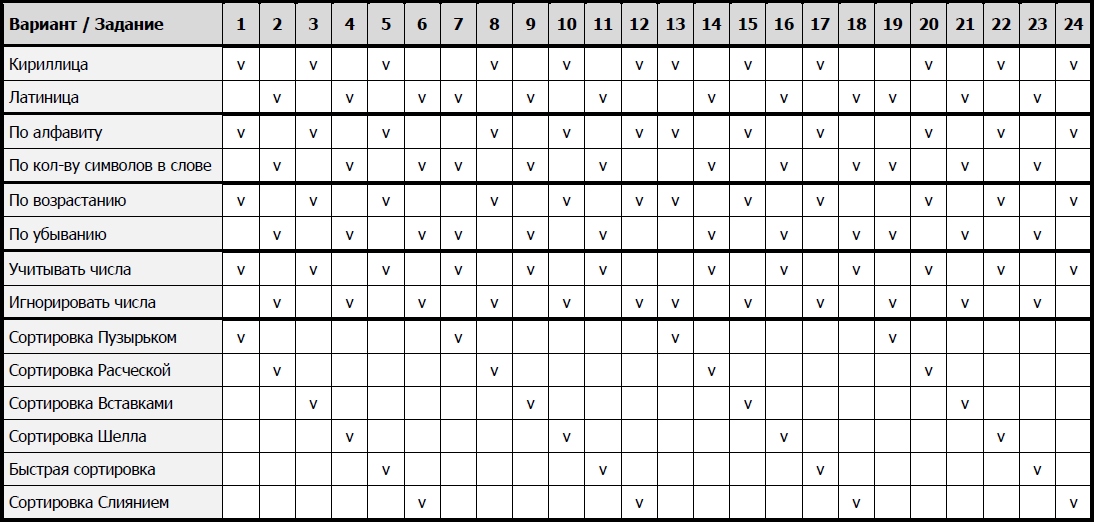
Индивидуальные параметры для каждого варианта представлены на рисунке (рисунок 1)

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

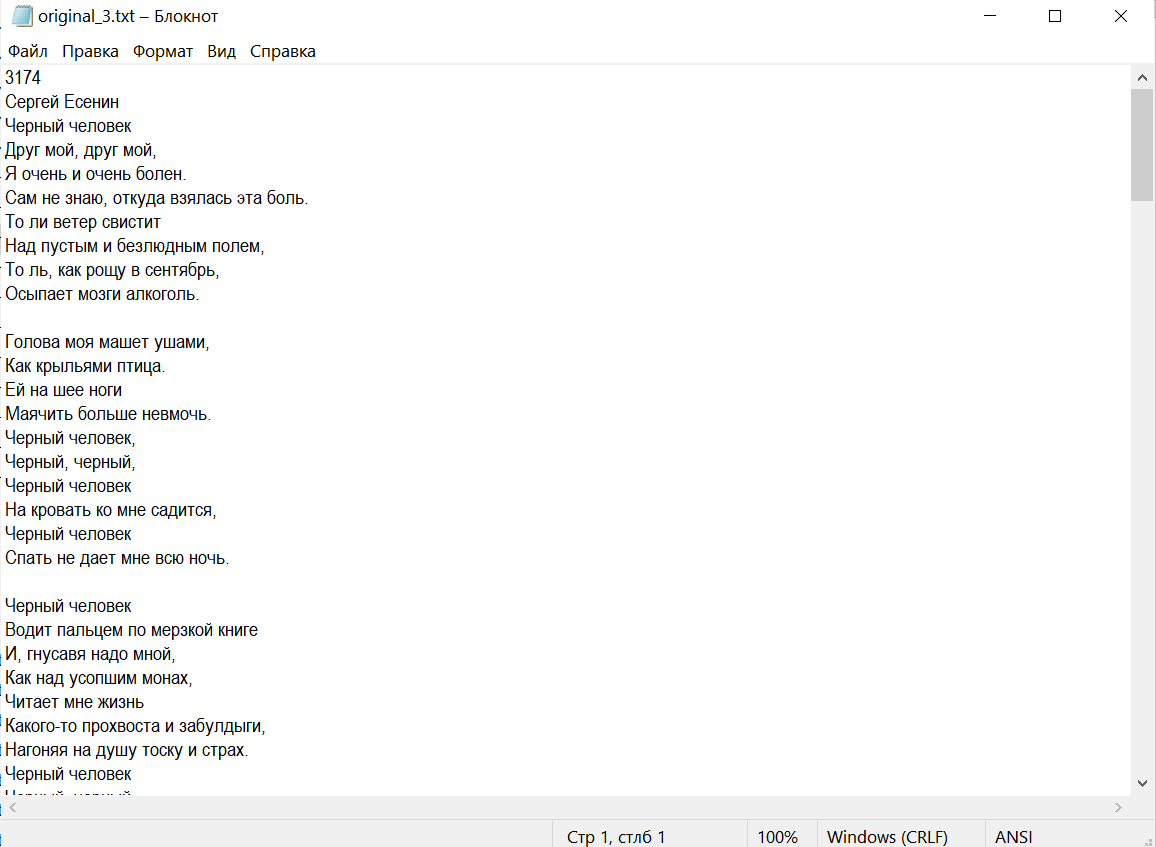
# Проходимся по всему вектору количества слов на каждую букву и выводим соответсвующую букву из нижнего регистра и элемент соответствующий элемент этого вектора

# Закрываем файл

# Тестирование программы и графики

# Проверка кол-ва слов

# 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

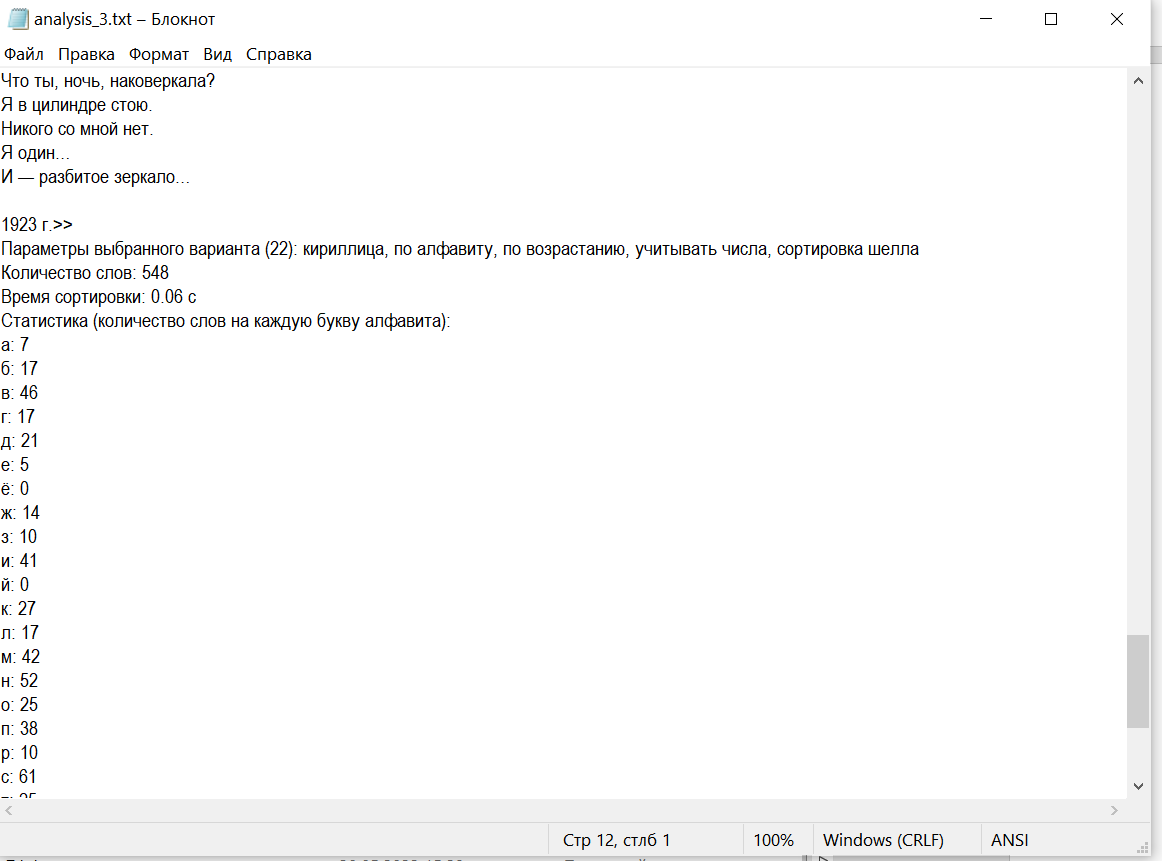


Рисунок 6 - Анализ текста №3

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

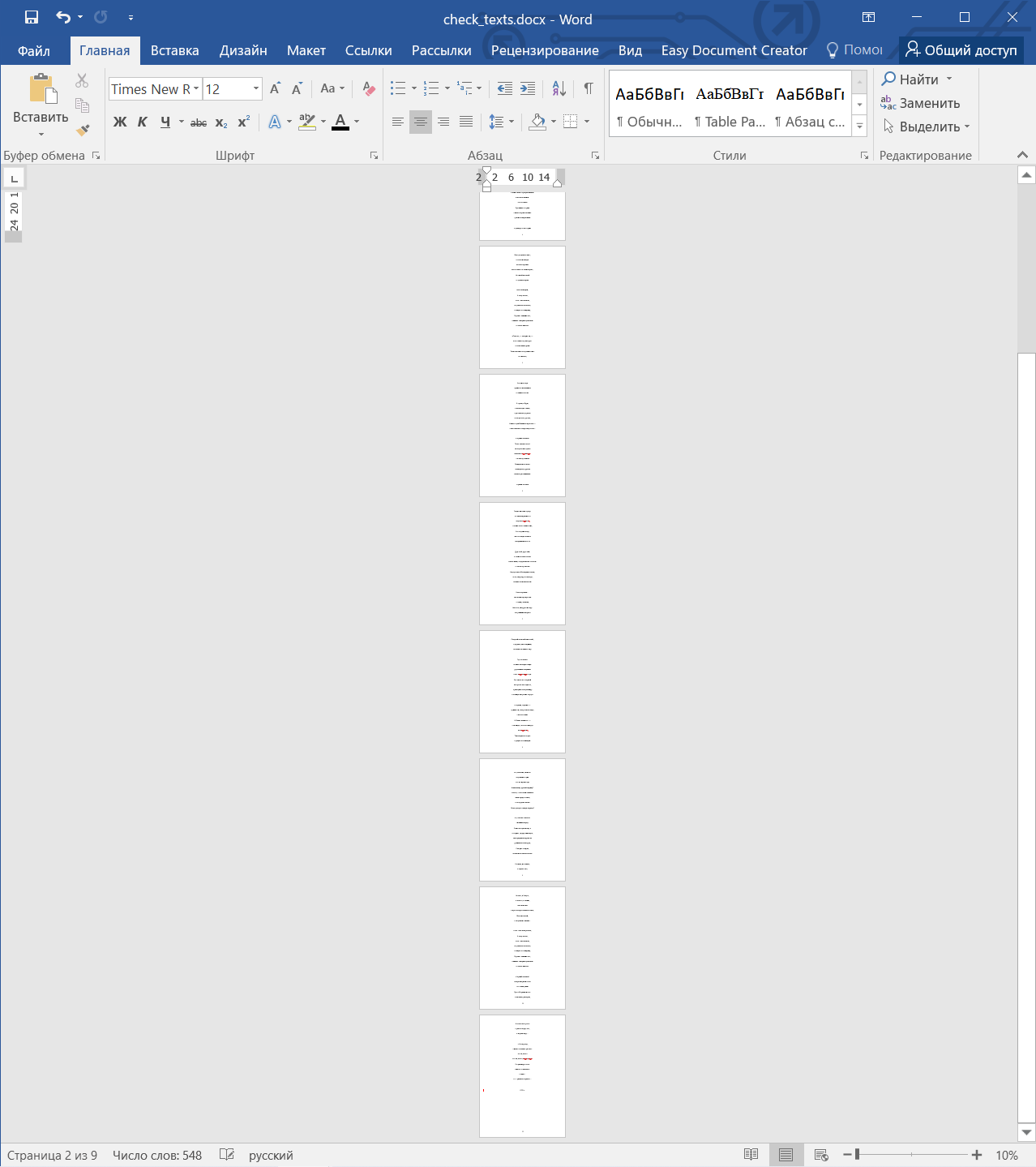


Рисунок 7 - Проверка текста №3

# 

Рисунок 8 - Исходный текст №5

# 

Рисунок 9 - Отсортированный текст №5

# 

Рисунок 10 - Анализ текста №5

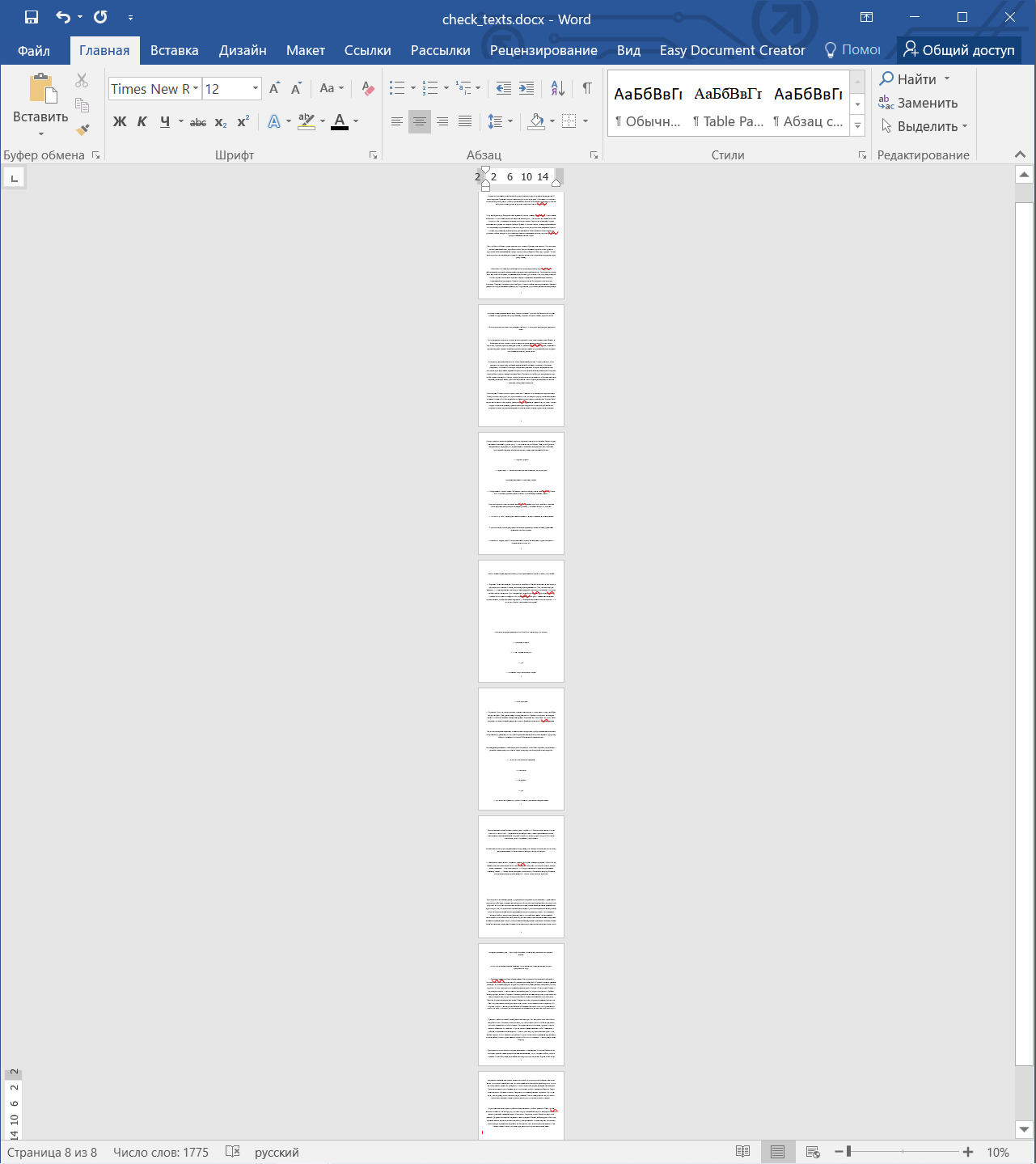


Рисунок 11 - Проверка текста №5

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

# Анализ сложности

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

График 1 - Зависимость времени сортировки от кол-ва символов

Можно сделать вывод, что при увеличении кол-ва символов в тексте, время выполнения сортировки возрастает прямо пропорционально.

Следует рассмотреть сложность сортировки Шелла. Сложность данного метода варьируется от O(𝑁^2) до O(N ∗ log𝑁) в зависимости от выбора интервалов. Сложность O(N ∗ log𝑁) при малых значениях с прямой зависимостью O(N).

# 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

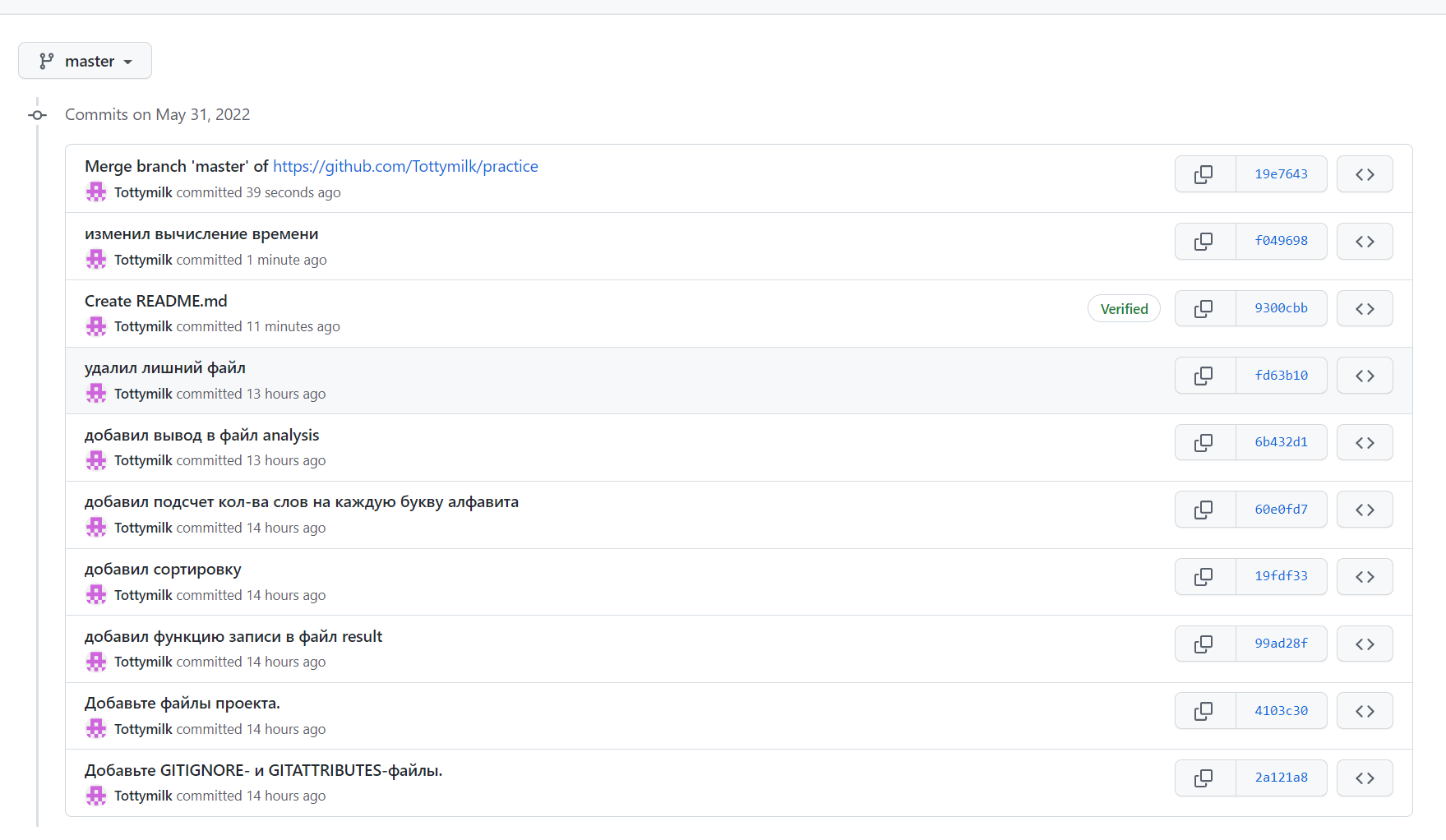


Рисунок 12 – Коммиты на Github

Процесс загрузки в репозиторий достаточно прост: по мере написания кода, после каждого «идейного» изменения необходимо обновлять репозиторий, отправляя коммиты. Сначала необходимо нажать на вкладку «Git» на верхней панели Visual Studio (рисунок 13), после нужно на вкладку «Фиксация или скрытие…» (рисунок 14), и ввести в поле какие изменения были сделаны (рисунок 15).

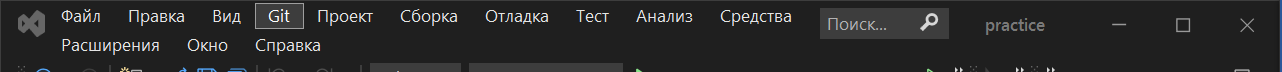


Рисунок 13 – Открытие вкладки «Git»

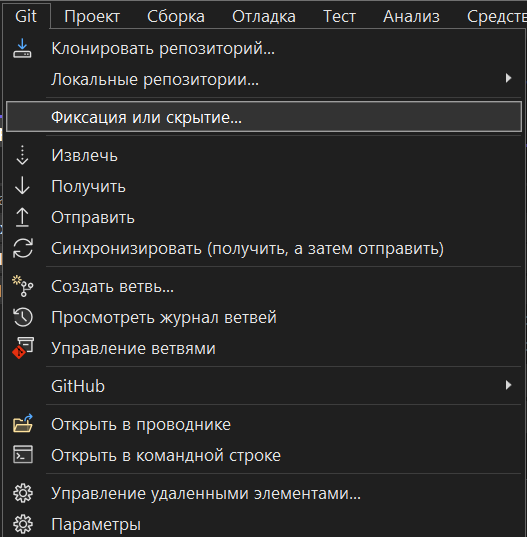


Рисунок 14 – Открытие вкладки «Фиксация или скрытие…»

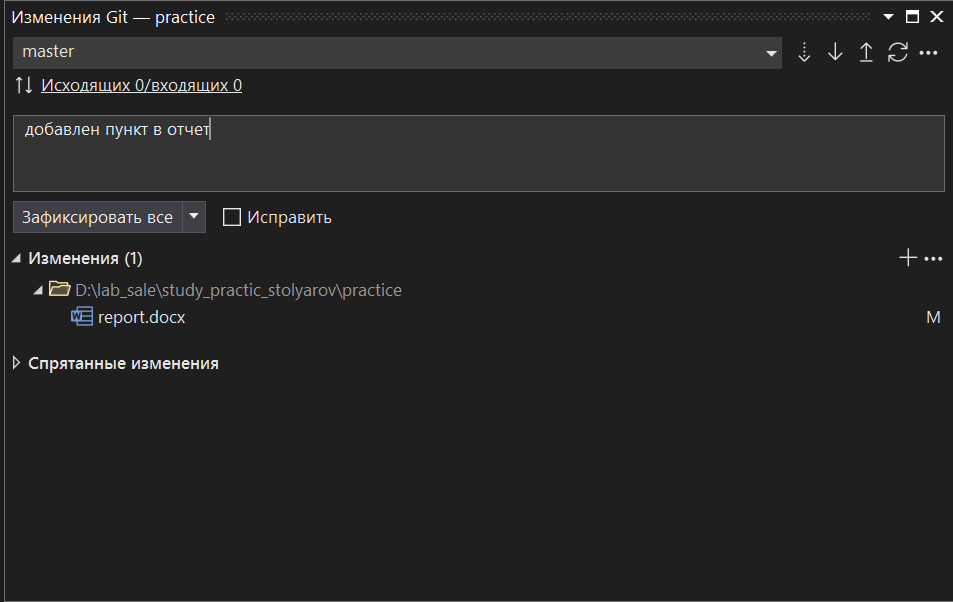


Рисунок 15 – Добавление описания изменений

# Файл README.md

После завершения работы с проектом, был оформлен файл README.md. Текст файла README.md содержит полное пояснение к программе, подробное описание всех частей кода, их функций в отдельности. Также в файле имеются скриншоты для более наглядного понимания о чём идёт речь в тексте. Во фрагментах описания кода присутствует блок непосредственного встроенного кода на C++. Скриншоты файла README представлены на рисунках 20, 21

Рисунок 16 – Файл README.md

Рисунок 17 – Файл README.md

Ссылка на репозиторий:

https://github.com/Tottymilk/practice

# ЗАКЛЮЧЕНИЕ

В ходе работы мною была реализована программная функция на языке C++.

При корректно введенных входных данных программа сортирует слова текста, считываемого из файла, и выводит результат в выходной файл. Кроме того, по завершении работы алгоритма на консоли и в отдельном файле можно увидеть анализ исходного текста: количество слов на каждую букву алфавита. Также рядом с анализом текста можно увидеть исходный текст, количество слов в нем, параметры варианта и время выполнения программной функции.

Таким образом, для успешного выполнения поставленной задачи, был изучен принцип работы алгоритма сортировки методом Шелла и его возможная реализация в коде на языке C++. Для выполнения задач помимо сортировки, были реализованы остальные функции исходного кода, имеющие вспомогательный характер (разделить текст на слова и числа, записать результаты в файл).

После реализации программного кода, были проведены 10 тестов на файлах с различным количеством символов. По результатам тестов был построен график зависимости времени от количества входных данных. После построения был сделан вывод. При небольших размерах входных данных, время сортировки практически не отличается, а периодический рост и падения во времени можно объяснить особенностью системы и неточностью в измерениях. Однако, при значительном увеличении входных данных можно однозначно сказать, что время сортировки напрямую зависит от входного количества символов.

Для GitHub были изучены базовые принципы работы на платформе: создание публичного репозитория, выгрузка файла проекта в репозиторий, создание README файла проекта.

Программа выполняет поставленную задачу согласно варианту задания, в зависимости от исходных данных.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Документация Microsoft Docs Справочник по языку C++ - URL: <https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170>
2. Справочник по языку C++ cplusplus.com – URL:

<https://m.cplusplus.com/>

1. Справочник по языку C++ cppreference.com – URL:

<https://en.cppreference.com/w/>

1. Курс лекций по дисциплине «Основы программирования» за весенний семестр 2022 г.
2. Справочник по работе с Github – URL:

<https://docs.github.com/en>

# ПРИЛОЖЕНИЕ А

Исходный код программы

#include <iostream>

#include <string>

#include <iostream>

#include <Windows.h>

#include <vector>

#include <fstream>

using namespace std;

vector<string> separateWords(string text);

string getText(string name\_file);

void writeToResult(string name\_file, vector<string> array\_of\_words);

bool isNumber(string s);

int sort(vector <string>& array\_of\_words);

int findInAlphabet(char a);

vector<int> getArrayCountWords(vector<string> array\_of\_words);

void writeToAnalysis(string name\_file, string text, int word\_count, vector<int> number\_words\_array, int time);

int main()

{

setlocale(0, ""); // поддержка кириллицы в консоли (вывод)

string name\_file; //название исходного файла

cout << "Введите номер файла: ";

cin >> name\_file;

//функция получения текста

string text = getText(name\_file);

//функция разбивает текст на слова

vector <string> array\_of\_words = separateWords(text);

//сортировка Шелла

int time = sort(array\_of\_words);

//запись в файл result

writeToResult(name\_file, array\_of\_words);

//подсчет количества слов на каждую букву

vector<int> number\_words\_array = getArrayCountWords(array\_of\_words);

int words\_count = array\_of\_words.size(); //количество слов

//запись в файл analysis

writeToAnalysis(name\_file, text, words\_count, number\_words\_array, time);

return 0;

}

string getText(string name\_file)

{

string text = ""; //исходная строка

ifstream file\_original; //создаем переменную файла

file\_original.open("original\_" + name\_file + ".txt"); // открываем файл

if (file\_original.is\_open()) //если файл открылся

{

char ch;

while (file\_original.get(ch)) //пока можем считать символ из файла

{

text += ch;; //прибавляем символ к строке

}

file\_original.close(); // Закрытие файла

}

else

{

cout << "Файл original не открылся";

exit(0);

}

return text;

}

vector<string> separateWords(string text)

{

vector<string> array\_of\_words; //массив слов

string s = "";

text = '"' + text + '"'; //добавляем кавычки

text = '"' + text + '"'; //добавляем кавычки

string high\_letters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"; //алфавит русский

string low\_letters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

string numbers = "0123456789";

for (int i = 2; i < text.length() - 2; i++)

{

if (high\_letters.find(text[i]) != -1 || low\_letters.find(text[i]) != -1) //если символ строки буква из кириллицы

{

s += text[i]; //собираем слово

if (high\_letters.find(text[i + 1]) == -1 && low\_letters.find(text[i + 1]) == -1 && text[i + 1] != '-')

{

array\_of\_words.push\_back(s); //добавляем в массив строку

s = ""; //обнуляем строку

}

}

else

{

if (numbers.find(text[i]) != -1 || text[i] == ',' && numbers.find(text[i + 1]) != -1) //если символ строки цифра или запятая

{

s += text[i]; //собираем слово

if (numbers.find(text[i + 1]) == -1 && text[i + 1] != ',' || numbers.find(text[i + 2]) == -1 && text[i + 1] == ',') //следующий символ не цифра и не запятая или след символ запятая а после не цифра

{

array\_of\_words.push\_back(s); //добавляем в массив строку

s = ""; //обнуляем строку

}

}

else

{

if (text[i] == '-' && (numbers.find(text[i + 1]) != -1 || ((high\_letters.find(text[i + 1]) != -1 || low\_letters.find(text[i + 1]) != -1) && s != ""))) //если символ строки минус и следующий символ цифра

{

s += text[i]; //собираем слово

}

}

}

}

return array\_of\_words;

}

void writeToResult(string name\_file, vector <string> array\_of\_words)

{

ofstream file\_result;

file\_result.open("result\_" + name\_file + ".txt"); // открываем файл на запись в него

//вывод слов

for (int i = 0; i < array\_of\_words.size(); i++)

{

file\_result << array\_of\_words[i] << endl;

}

file\_result.close();

}

int sort(vector <string>& array\_of\_words)

{

string numbers = "0123456789-"; //цифры и минус, то с чего может начинаться число

int i, j, step, number\_tmp, start\_time, end\_time;

string tmp;

vector <string> array\_of\_numbers;

vector <string> subarray\_of\_words;

//получаем массив чисел и массив слов

for (i = 0; i < array\_of\_words.size(); i++)

{

if (numbers.find(array\_of\_words[i][0]) != -1)

array\_of\_numbers.push\_back(array\_of\_words[i]);

else

subarray\_of\_words.push\_back(array\_of\_words[i]);

}

start\_time = clock();

//сортируем массив слов

for (step = subarray\_of\_words.size() / 2; step > 0; step /= 2)

{

for (i = step; i < subarray\_of\_words.size(); i++)

{

tmp = subarray\_of\_words[i];

number\_tmp = findInAlphabet(subarray\_of\_words[i][0]); //номер в алфавите

for (j = i; j >= step; j -= step)

{

if (number\_tmp < findInAlphabet(subarray\_of\_words[j - step][0]))

subarray\_of\_words[j] = subarray\_of\_words[j - step];

else

break;

}

subarray\_of\_words[j] = tmp;

}

}

//сортируем массив чисел

double double\_number\_tmp, double\_number;

for (step = array\_of\_numbers.size() / 2; step > 0; step /= 2)

{

for (i = step; i < array\_of\_numbers.size(); i++)

{

tmp = array\_of\_numbers[i];

if (isNumber(array\_of\_numbers[i])) //если строка - дробное число

{

double\_number\_tmp = stod(array\_of\_numbers[i]); //временной переменной присваиваем число

for (j = i; j >= step; j -= step)

{

if (isNumber(array\_of\_numbers[j])) //если строка - дробное число

{

if (double\_number\_tmp < stod(array\_of\_numbers[j - step])) //сравниваем число временной переменной и число array\_of\_numbers[j - step]

array\_of\_numbers[j] = array\_of\_numbers[j - step];

else

break;

}

}

array\_of\_numbers[j] = tmp;

}

}

}

end\_time = clock();

array\_of\_words = {};

//сливаем массивы обратно в один массив

for (i = 0; i < subarray\_of\_words.size(); i++)

{

array\_of\_words.push\_back(subarray\_of\_words[i]);

}

for (i = 0; i < array\_of\_numbers.size(); i++)

{

array\_of\_words.push\_back(array\_of\_numbers[i]);

}

return end\_time - start\_time;

}

bool isNumber(string s)

{

try

{

double a = stod(s);

return 1;

}

catch (invalid\_argument e)

{

return 0;

}

}

int findInAlphabet(char a)

{

//функция возвращает индекс вхождения символа в алфавит

string low\_letters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

string high\_letters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

if (high\_letters.find(a) != -1)

return high\_letters.find(a);

else

return low\_letters.find(a);

}

vector<int> getArrayCountWords(vector<string> array\_of\_words)

{

vector<int> count\_words\_array(33);

//Каждый элемент массив - это количество слов на букву

string high\_letters = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"; //алфавит русский

string low\_letters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

for (int i = 0; i < array\_of\_words.size(); i++) //идем по всему массиву строк

{

if (!isNumber(array\_of\_words[i]))

{

count\_words\_array[findInAlphabet(array\_of\_words[i][0])]++;// увеличиваем элемент с индексом равным номеру маленькой буквы в нижнем регистре

}

}

return count\_words\_array;

}

void writeToAnalysis(string name\_file, string text, int word\_count, vector<int> number\_words\_array, int time)

{

ofstream file\_analysis;

file\_analysis.open("analysis\_" + name\_file + ".txt"); // открываем файл на запись в него

file\_analysis

<< "Исходный текст: " << endl

<< "<<" << text << ">>" << endl

<< "Параметры выбранного варианта (22): кириллица, по алфавиту, по возрастанию, учитывать числа, сортировка шелла" << endl

<< "Количество слов: " << word\_count << endl

<< "Время сортировки: " << static\_cast<double>(time) / 1000 << " с" << endl

<< "Статистика (количество слов на каждую букву алфавита): " << endl;

cout

<< "Исходный текст: " << endl

<< "<<" << text << ">>" << endl

<< "Параметры выбранного варианта (22): кириллица, по алфавиту, по возрастанию, учитывать числа, сортировка шелла" << endl

<< "Количество слов: " << word\_count << endl

<< "Время сортировки: " << static\_cast<double>(time) / 1000 << " с" << endl

<< "Статистика (количество слов на каждую букву алфавита): " << endl;

string lo\_reg = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

//вывод количества слов на каждую букву

for (int i = 0; i < number\_words\_array.size(); i++)

{

file\_analysis << lo\_reg[i] << ": " << number\_words\_array[i] << endl;

cout << lo\_reg[i] << ": " << number\_words\_array[i] << endl;

}

file\_analysis.close();

}