|  |
| --- |
| 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 | Мягкова Даниил Львович |
| 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 бизнесе |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 5136 |  | 04.06.2022 |  | Мягков Д. Л. |
|  | 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

Мягков Даниил Львович

1. Фамилия, имя, отчество обучающегося:

5136

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, сортировка вставками); 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:

Мягков Д. Л

07.02.2022

Обучающийся

дата подпись инициалы, фамилия

Санкт–Петербург 2022

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc105164291)

[1 Индивидуальный вариант 5](#_Toc105164292)

[2 Блок-схема алгоритма программы 6](#_Toc105164293)

[3 Описание алгоритма программы 7](#_Toc105164294)

[3.2 Описание функции get\_string 7](#_Toc105164303)

[3.3 Описание функции split\_text 7](#_Toc105164305)

[3.4 Описание функции counting\_words 7](#_Toc105164307)

[3.5 Описание функции sort 8](#_Toc105164309)

[3.6 Описание функции writing\_to\_result 8](#_Toc105164313)

[3.7 Описание функции get\_number\_file 8](#_Toc105164315)

[3.8 Описание функции alphabet 8](#_Toc105164317)

[3.9 Описание функции writing\_to\_analysis 8](#_Toc105164319)

[4 Тестирование программы и графики 9](#_Toc105164321)

[4.1 Проверка кол-ва слов 9](#_Toc105164322)

[4.2 Анализ сложности 17](#_Toc105164329)

[5 Работа с Github 18](#_Toc105164331)

[5.1 Загрузка репозитория 18](#_Toc105164332)

[5.2 Файл README.md 21](#_Toc105164334)

[ЗАКЛЮЧЕНИЕ 23](#_Toc105164335)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 24](#_Toc105164336)

[ПРИЛОЖЕНИЕ А 25](#_Toc105164337)

# **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. 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**

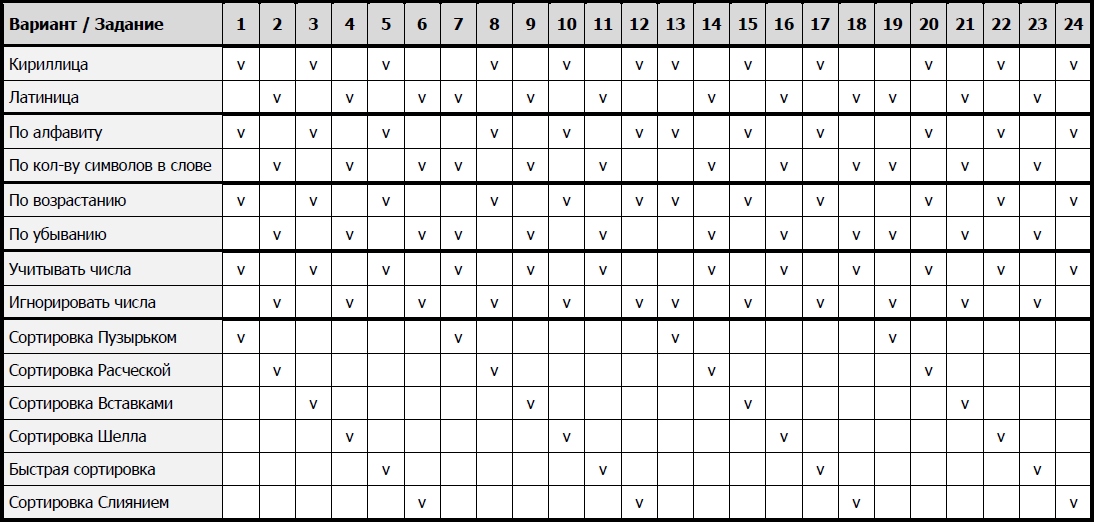
Индивидуальные параметры для каждого варианта представлены на рисунке (рисунок 1)

Рисунок 1 – Параметры вариантов

Выполненный вариант работы – 15. Следовательно было необходимо обработать текст кириллицы, разбив его на слова без учета чисел, отсортировать эти слова по возрастанию сортировкой вставками.

# **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 number\_file типа string, в которую с консоли запишется название файла, введенное пользователем, и переменной source\_string типа string, в которой будет находиться исходный текст

# Создается вектор строк words, это массив слов

# Вызывается функция sort, она отсортирует массив слов и вернет время сортировки

# Создается вектор count\_words типа int, в него записывается вывод функции counting\_words

# Вызывается функция writing\_to\_result, записывает в файл отсортированный массив

# Вызывается функция writing\_to\_analysis, записывает в файл анализ

# Описание функции get\_string

# Функция открывает файл, посимвольно считывает из него символы и формирует исходную строку, которую потом и возвращает

# Описание функции split\_text

# Получает строку на вход, проходится по всем символам и формирует слова, которые при определенных условиях добавляет в массив слов, который потом возвращает

# Описание функции counting\_words

# Создается массив чисел, проходим по всем словам, увеличиваем в массиве чисел элемент с индексом равным номеру буквы в алфавите

# Описание функции sort

# Замеряем время до сортировки

# Проходимся по всему массиву, запоминаем каждый элемент, и двигаем все элементы до него пока он не встанет на свое место

# Замеряем время после сортировки

# Описание функции writing\_to\_result

# Открываем файл и записываем в него слова в цикле

# Описание функции get\_number\_file

# Считываем с консоли название файла

# Описание функции alphabet

# Возвращает алфавит

# Описание функции writing\_to\_analysis

# Открываем файл и записываем в него анализ текста

# 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

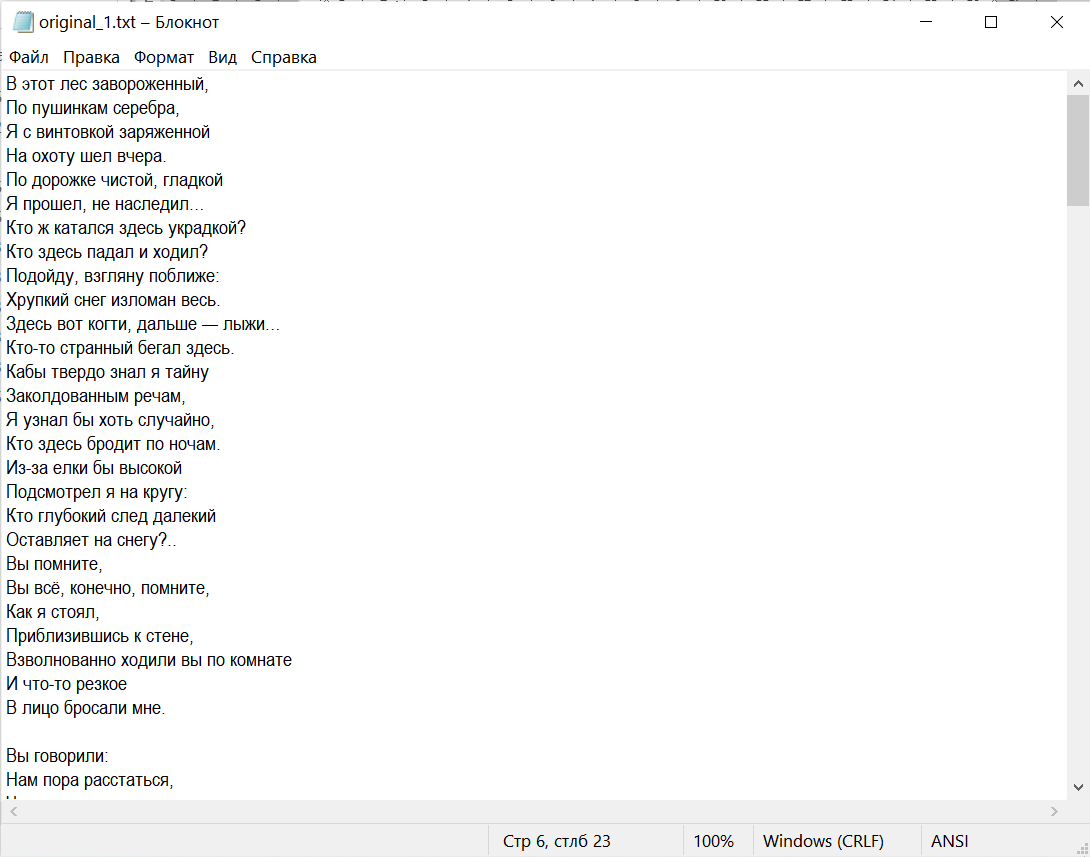


Рисунок 3 - Исходный текст №1

# 

Рисунок 4 - Отсортированный текст №1

# 

Рисунок 5 - Анализ текста №1

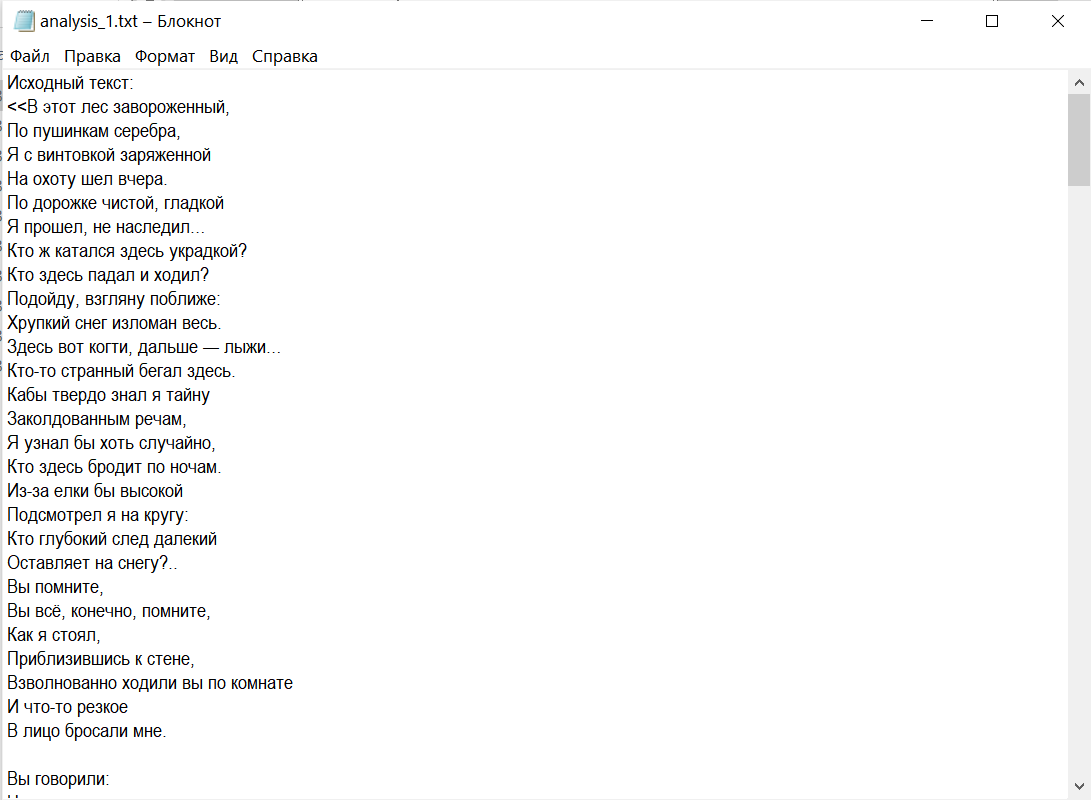


Рисунок 6 - Анализ текста №1

Для текста №3 Microsoft Word насчитал 622 слов, программа насчитала 622.

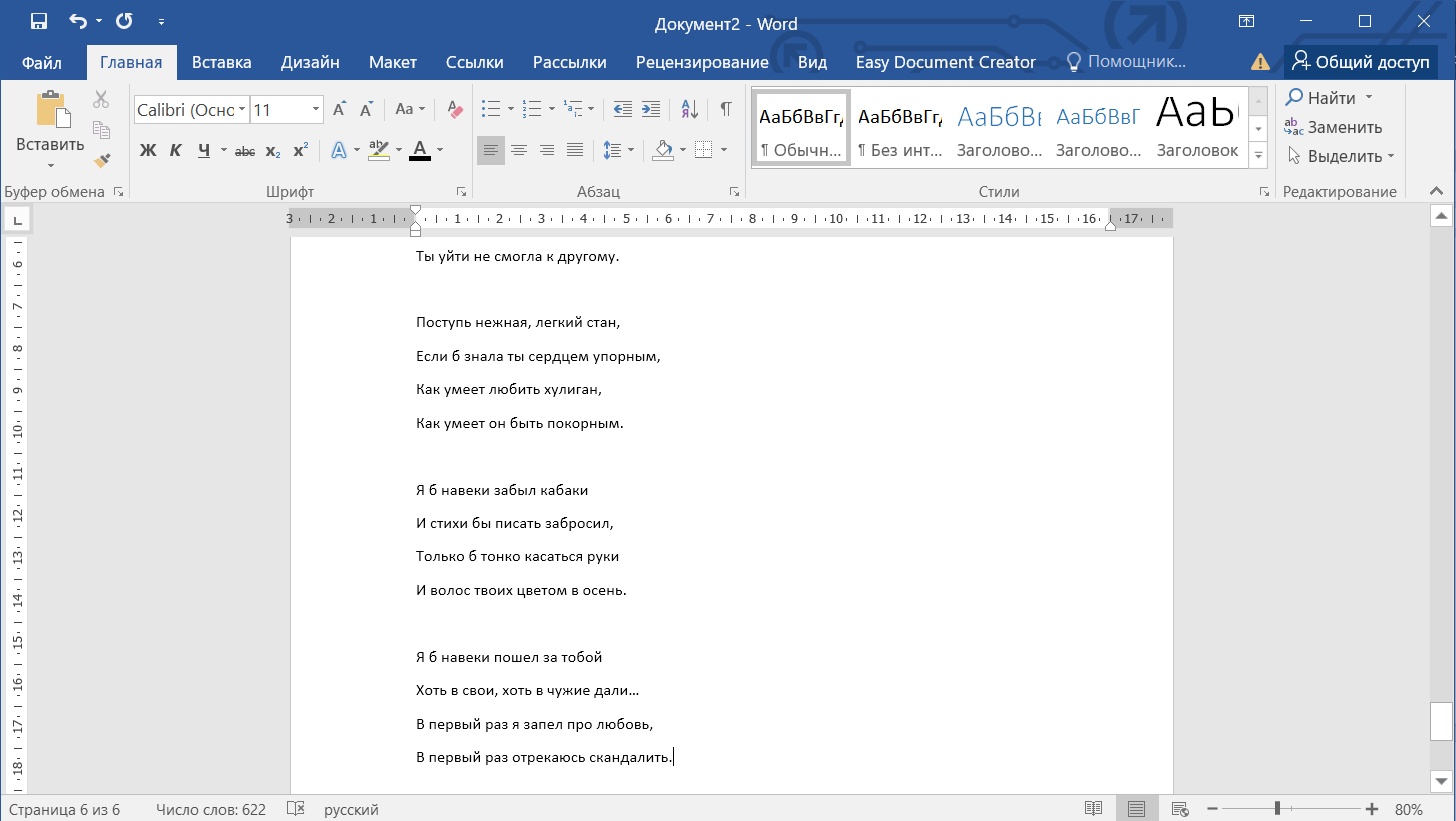


Рисунок 7 - Проверка текста №1

# 

Рисунок 8 - Исходный текст №5

# 

Рисунок 9 - Отсортированный текст №5

# 

Рисунок 10 - Анализ текста №5

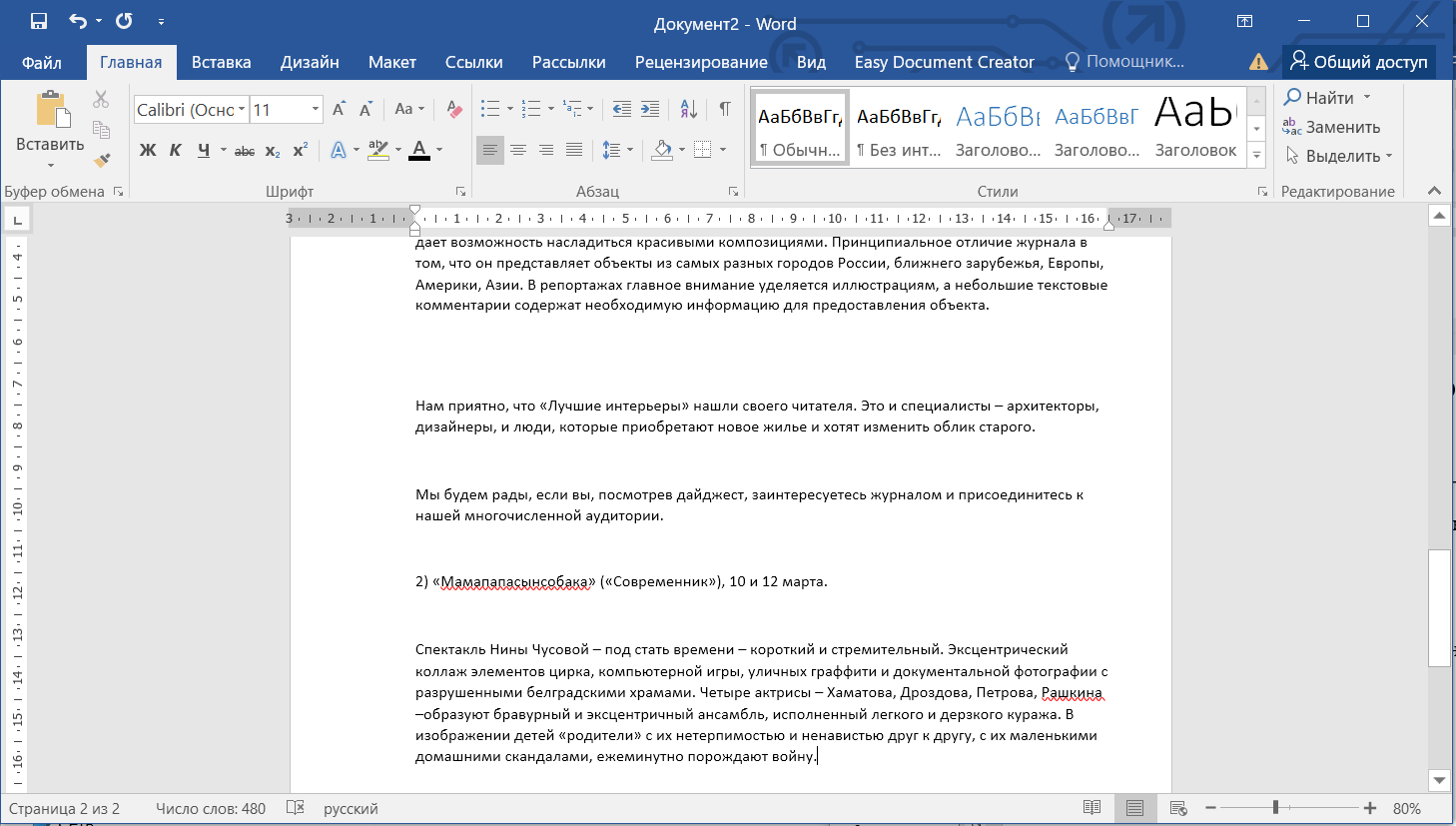


Рисунок 11 - Проверка текста №5

Для текста №3 Microsoft Word насчитал 480 слов, программа насчитала 479.

# 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.

Рассмотрим сложность сортировки вставками. Средняя сложность от O(n ^ 2) в худшем случае до O(n) в лучшем .

# Работа с Github

# Загрузка репозитория

# В процессе написания кода, также была осуществлена практика в выгрузке программы на github. Было загружено 4 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

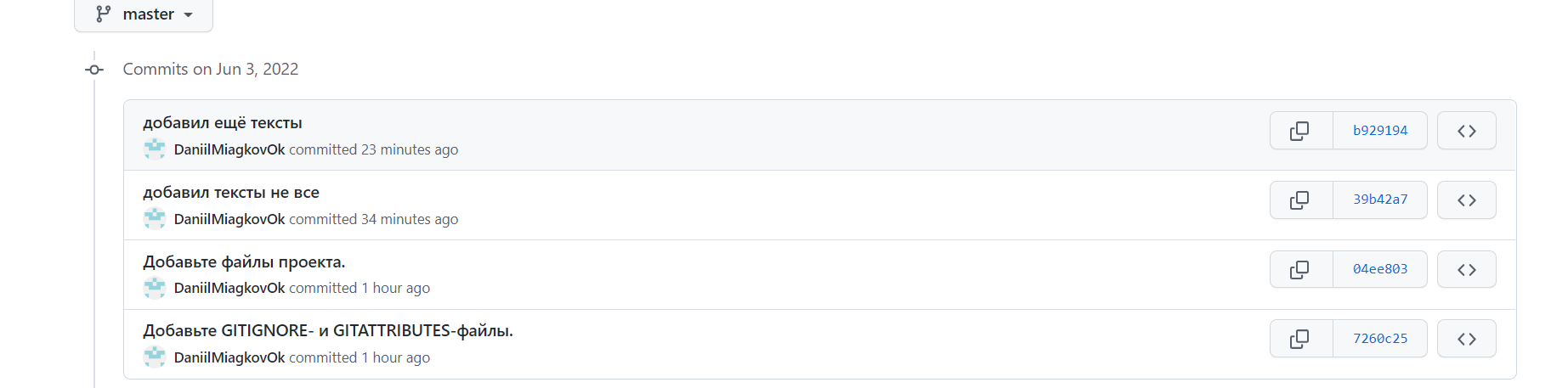


Рисунок 12 – Коммиты на Github

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).

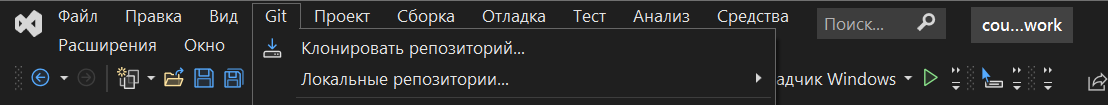


Рисунок 13 – Открытие вкладки «Git»

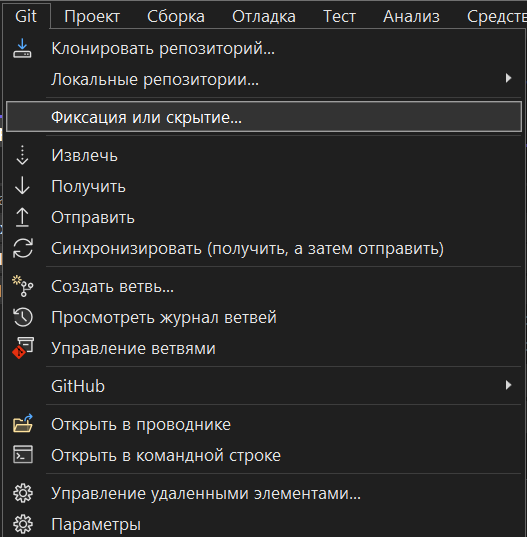


Рисунок 14 – Открытие вкладки «Фиксация или скрытие…»

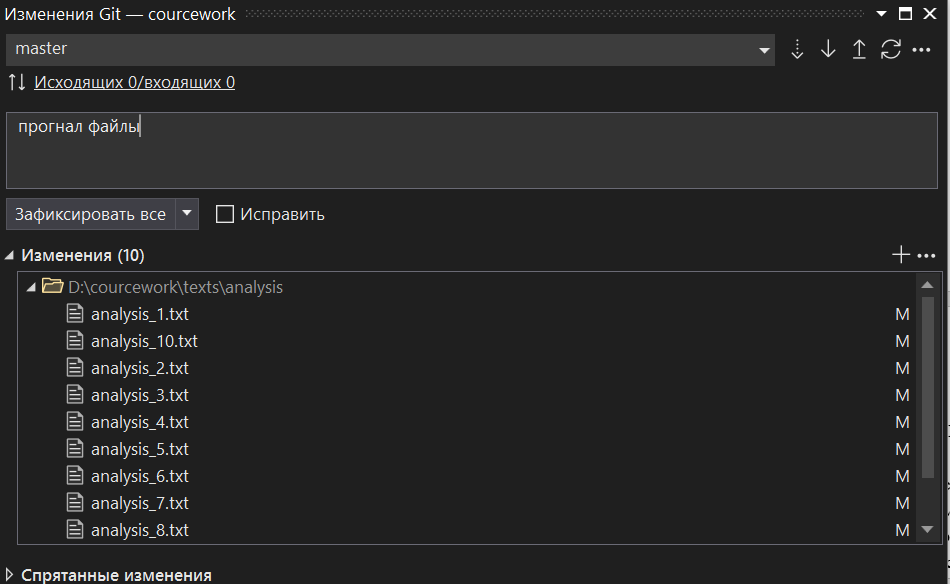


Рисунок 15 – Добавление описания изменений

# 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 16 - 18

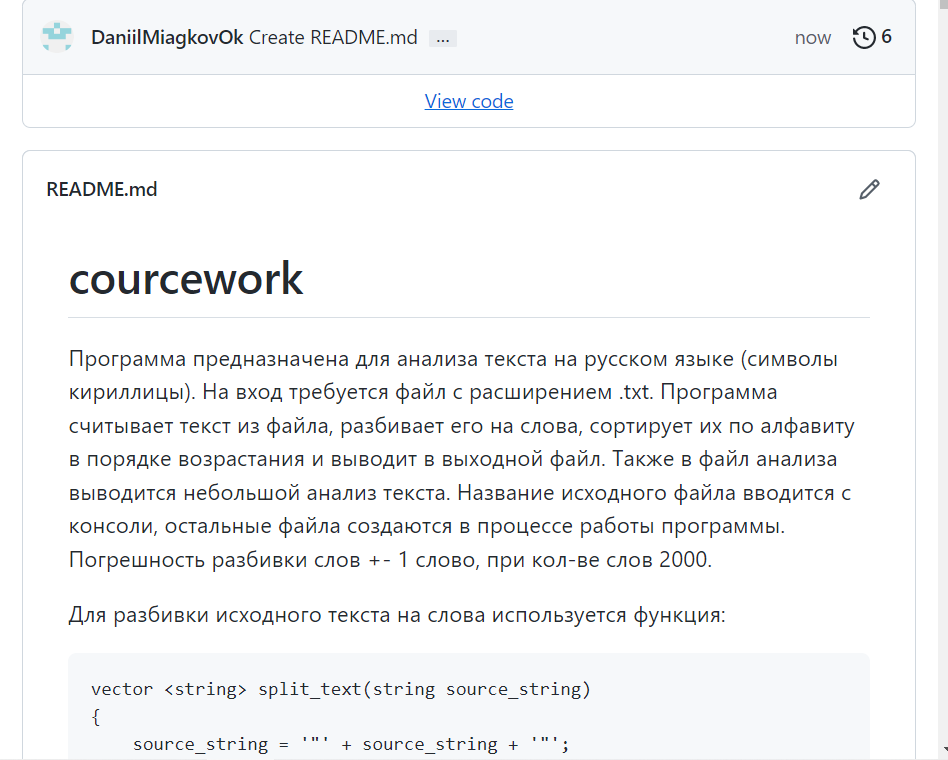


Рисунок 16 – Файл README.md

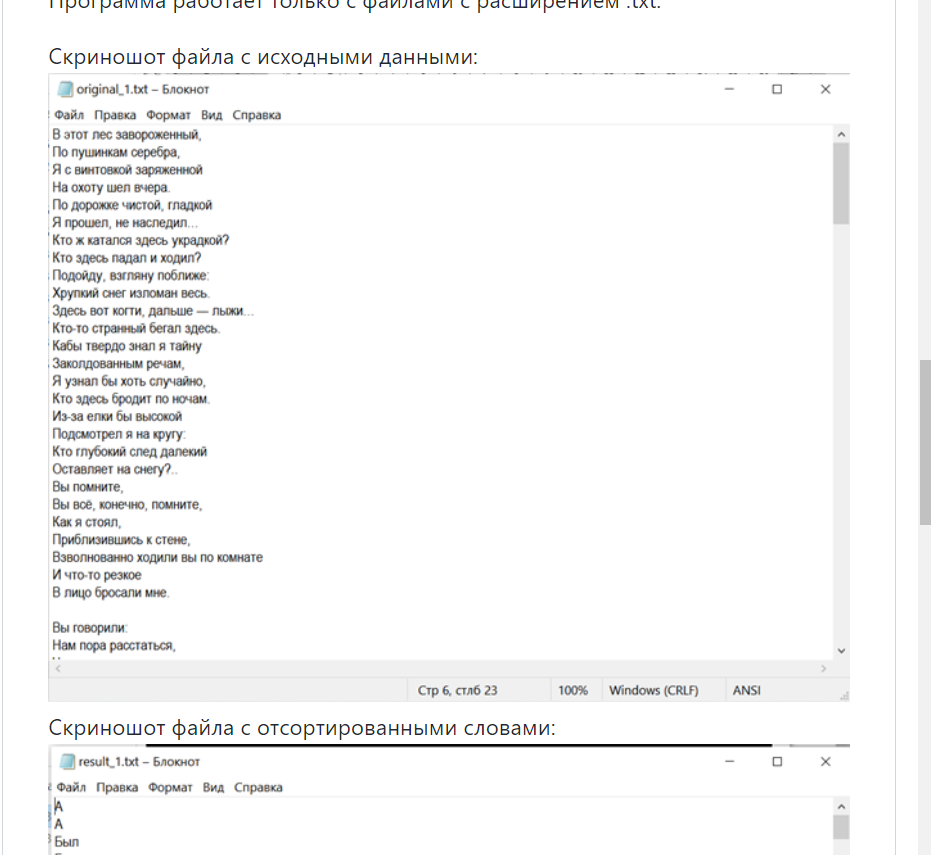


Рисунок 17 – Файл README.md

Ссылка на репозиторий:

https://github.com/DaniilMiagkovOk

# 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 вставками 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, 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.

# 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

Файл main.cpp

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include <vector>

#include <fstream>

#include <clocale>

using namespace std;

int main()

{

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

SetConsoleCP(1251); // поддержка кириллицы в консоли (ввод)

SetConsoleOutputCP(1251); // поддержка кириллицы в консоли (ввод)

string number\_file = get\_number\_file();

string source\_string = get\_string(number\_file);

vector <string> words = split\_text(source\_string);

vector <int> count\_words = counting\_words(words);

unsigned int time = sort(words);

writing\_to\_result(words, number\_file);

writing\_to\_analysis(source\_string, words.size(), count\_words, time, number\_file);

return 0;

}

Файл various\_functions.cpp

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include <vector>

#include <fstream>

#include <string>

using namespace std;

unsigned int sort(std::vector<std::string>& source\_words)

{

unsigned int average\_time = 0;

unsigned int start\_time, end\_time;

int n = 5;

for (int k = 0; k < n; k++)

{

start\_time = clock();

std::vector<std::string> words = source\_words;

for (int i = 1; i < words.size(); i++)

{

for (int j = i; j > 0; j--)

{

if (alphabet().find(words[j - 1][0]) < alphabet().find(words[j][0]))

break;

swap(words[j - 1], words[j]);

}

}

end\_time = clock();

average\_time += end\_time - start\_time;

}

return average\_time / n;

}

std::vector<int> counting\_words(std::vector<std::string> words)

{

std::vector<int> count\_words(33);

for (int i = 0; i < words.size(); i++)

{

count\_words[alphabet().find(words[i][0]) % 33]++;

}

return count\_words;

}

string get\_number\_file()

{

string number\_file = "";

std::cout

<< "Доступные файлы:" << std::endl

<< std::endl

<< " 1. Есенин" << std::endl

<< " 2. Булгаков" << std::endl

<< " 3. Шолохов" << std::endl

<< " 4. 12 стульев" << std::endl

<< " 5. Публицистический стиль" << std::endl

<< " 6. Обломов" << std::endl

<< " 7. Шолохов" << std::endl

<< " 8. Отцы и дети" << std::endl

<< " 9. Преступление и наказание" << std::endl

<< " 10. Отцы и дети" << std::endl

<< std::endl

<< "Введите название файла с текстом: ";

vector <pair <string, int>> text\_names =

{

{"Есенин", 1},

{"Булгаков", 2},

{"Шолохов", 3},

{"12 стульев", 4},

{"Публицистический стиль", 5},

{"Обломов", 6},

{"Шолохов", 7},

{"Отцы и дети", 8},

{"Преступление и наказание", 9},

{"Отцы и дети", 10}

};

while (true)

{

std::cin >> number\_file;

if (number\_file == "-1")

exit(0);

ifstream file\_original; //создаем переменную файла

file\_original.open("D:\\courcework\\texts\\original\\original\_" + number\_file + ".txt"); // открываем файл

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

{

cout << "Вы выбрали " << text\_names[stoi(number\_file) - 1].first << endl;

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

break;

}

else

{

std::cout << "Такого файла нет, введите другое название. Или введите -1, чтобы выйти из программы: " << std::endl;

}

}

return number\_file;

}

string get\_string(string number\_file)

{

ifstream file\_original;

file\_original.open("D:\\courcework\\texts\\original\\original\_" + number\_file + ".txt"); // открываем файл

string source\_string = "";

char ch;

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

{

source\_string+= ch;; //прибавляем символ к строке

}

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

return source\_string;

}

vector <string> split\_text(string source\_string)

{

source\_string = '"' + source\_string + '"';

string word = "";

vector <string> words;

for (int i = 1; i < source\_string.size() - 1; i++)

{

if (alphabet().find(source\_string[i]) != -1)

{

word += source\_string[i];

if (alphabet().find(source\_string[i + 1]) == -1 && source\_string[i + 1] != '-')

{

words.push\_back(word);

word = "";

}

}

else if (source\_string[i] == '-' && alphabet().find(source\_string[i + 1]) >= 33 && alphabet().find(source\_string[i + 1]) <= source\_string.size() - 1 && word.size() != 0)

{

word += source\_string[i];

}

}

return words;

}

string alphabet()

{

string alphabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя";

return alphabet;

}

void writing\_to\_result(vector <string> words, string number\_file)

{

ofstream file\_result;

file\_result.open("D:\\courcework\\texts\\result\\result\_" + number\_file + ".txt"); // открываем файл

for (int i = 0; i < words.size(); i++)

file\_result << words[i] << endl;

}

void writing\_to\_analysis(string source\_string, int count, vector <int> count\_words, int time, string number\_file)

{

ofstream file\_analysis;

file\_analysis.open("D:\\courcework\\texts\\analysis\\analysis\_" + number\_file + ".txt"); // открываем файл

file\_analysis

<< "Исходный текст: " << endl

<< "<<" << source\_string << ">>" << endl

<< "Параметры выбранного варианта (15): кириллица, по алфавиту, по возрастанию, игнорировать числа, сортировка вставками" << endl

<< "Количество слов: " << count << endl

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

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

cout

<< "Исходный текст: " << endl

<< "<<" << source\_string << ">>" << endl

<< "Параметры выбранного варианта (15): кириллица, по алфавиту, по возрастанию, игнорировать числа, сортировка вставками" << endl

<< "Количество слов: " << count << endl

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

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

//вывод количества слов на каждую букву

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

{

file\_analysis << alphabet()[i + 33] << ": " << count\_words[i] << endl;

cout << alphabet()[i + 33] << ": " << count\_words[i] << endl;

}

file\_analysis.close();

}

Файл various\_functions.h

#pragma once

#include <iostream>

#include <vector>

using namespace std;

unsigned int sort(std::vector<std::string>& words);

std::vector<int> counting\_words(std::vector<std::string> array);

string get\_number\_file();

string get\_string(string number\_file);

vector<string> split\_text(string source\_string);

string alphabet();

void writing\_to\_result(vector<string> words, string number\_file);

void writing\_to\_analysis(string source\_string, int count, vector<int> count\_words, int time, string number\_file);