|  |
| --- |
| 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 | Смирновым Никитой Дмитриевичем |
| 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 медиаиндустрии |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 5138 |  | 03.06.2022 |  | Смирнов Н.Д. |
|  | 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

Смирнов Никита Дмитриевич

1. Фамилия, имя, отчество обучающегося:

5138

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:

учитывать 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.

Гуков С.Ю.

ассистент

04.06.2022

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:

Смирнов Н.Д.

07.02.2022

Обучающийся

дата подпись инициалы, фамилия

Санкт–Петербург 2022

Оглавление

[Введение 4](#_Toc105124515)

[Задание по варианту 5](#_Toc105124516)

[Блок-схема программы 7](#_Toc105124517)

[Описание алгоритма 8](#_Toc105124518)

[Описание функций 9](#_Toc105124519)

[Тестирование программы 10](#_Toc105124520)

[График зависимости 15](#_Toc105124521)

[Заключение 16](#_Toc105124522)

[Список используемых источников 17](#_Toc105124523)

[Исходный код 18](#_Toc105124524)

# Введение

Целью проведения производственной практики является приобретение обучающимися профессиональных умений и навыков, опыта профессиональной деятельности и закрепление приобретенных компетенций, предусмотренных ФГОС и направленностью ОП ВО. В рамках учебной практики необходимо закрепить полученные знания по языку программирования на дисциплине «Основы программирования» и написать программу, для сортировки и анализа текстовых данных. В соответствии с параметрами варианта задания нужно реализовать сортировку слов в тексте и провести его простейший анализ. Варианты заданий приведены на рисунке 3.

# Задание по варианту

Необходимо реализовать программу, которая будет считывать текст из файла (например, «original.txt») 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.

Примечание. Символы и знаки препинания игнорируются и не включаются в слово/длину слова. Отсортированный массив необходимо вывести в выходной файл, например, «result.txt». В консоль результат выводить не нужно. Также необходимо вывести информацию с простейшим анализом введенного текста (эту информацию выводим в консоль и в файл, например, «analysis.txt»): • исходный текст • параметры выбранного варианта • количество слов в исходном тексте • время выполнения сортировки • количество слов на каждую букву алфавита (для варианта «сортировка по алфавиту») • количество слов каждой длины (для варианта «сортировка по количеству символов в слове») 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. В отчете нужно наглядно изобразить (график/гистограмма/диаграмма) зависимость времени выполнения сортировки от количества слов в тексте и сделать выводы. Также необходимо оценить и обосновать сложность получившегося алгоритма в O-нотации.

Далее в таблице на рисунке 3 представлены варианты заданий. Варианты старше 24-го начинаются опять с 1-го

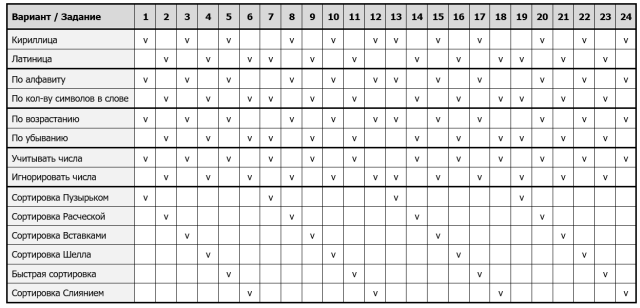


Рисунок 1 – таблица вариантов

Мой вариант – 3.

* Пункт «Кириллица / Латиница» означает, с символами какого алфавита придется работать. Кириллица – «абвгдеёжзийклмнопрстуфхцчшщъыьэюя1234567890». Латиница – «abcdefghijklmnopqrstuvwxyz1234567890».
* Пункт «По алфавиту / По количеству символов в слове» означает, по какому признаку необходимо сортировать слова в тексте. По алфавиту – проверяем первую букву в каждом слове и сортирует относительно алфавита (цифры идут после букв). По количеству символов в слове – вычисляем количество букв в каждом слове и сортируем относительно этого количества.
* Пункт «По возрастанию / По убыванию» означает, в каком направлении сортировать массив слов.
* Пункт «Учитывать числа / Игнорировать числа» означает, необходимо ли идентифицировать отдельные числа/цифры в тексте как отдельные слова. Если числа учитываются, то они считаются словом и к ним применяются правила из пункта «По алфавиту / По количеству символов в слове». Если игнорируются, то эти числа/цифры просто опускаются при сортировке.
* Пункт «Сортировка \*\*\*» означает, какой метод сортировки необходимо использовать.

# Блок-схема программы

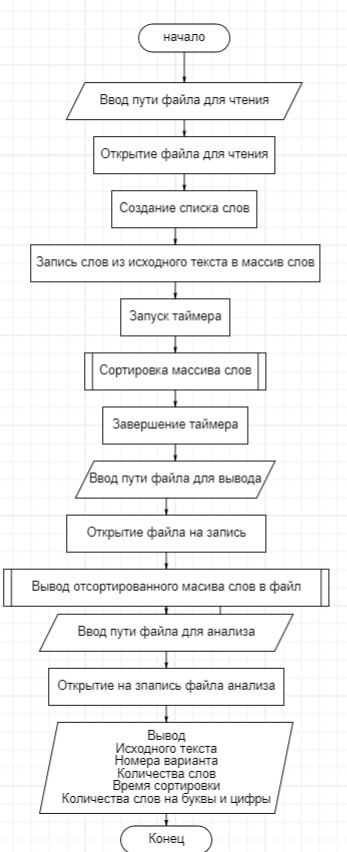


Рисунок 2 – простейшая блок-схема алгоритма прог`раммы

# Описание алгоритма

1. Открывается файл на чтение.
2. С помощью функции «split\_text» создается список слов.
3. Далее засекается время начала выполнения алгоритма сортировки.
4. Сортируется созданный список слов.
5. Высчитывается текущее время и время, которое было засечено перед выполнением сортировки.
6. Вводится информация о варианте.
7. Открывается файл для вывода отсортированного текста.
8. Выводится отсортированный текст
9. Открывается файл для анализа программы.
10. Из функции «print\_analisis» выводится требуемая информация.

# Описание функций

1. Функция «split\_text»

Создается пустой массив и пустая переменная, в которой будет хранится будущее слово. Далее перебирается строка по символу и записывается в переменную «symb». Если символ равен запятой и числу, что уже лежит в «word» или букве или символу числу, то прибавляется к «word» значение «symb» нижнего регистра. Иначе проверяется не равна ли длина строки 0, тогда добавляется в конец массива слов «words» слово «word» и происходит обнуление строки и так пока не закончится текст.

1. Функция «inseration\_sort»

Внутри функции создается цикл с переменной «i» , которая будет исчисляться от 1 до длины массива. Создается переменная «temp» и приравнивается к массиву. Создается цикл «while» который будет работать до тех пор, пока j не отрицательное и «temp» меньше, чем элемент под индексом [j][0]. Устанавливается «arr [j+1] = arr [j]», затем уменьшается j на 1. После окончания цикла «While» устанавливается «arr[j+1] = «temp».

1. Функция «group\_by\_alphabet»

Создается словарь, который заполняется пустыми массивами для букв и цифр. Далее распределяются слова из «arr» по словарю на ту букву, на которую они начинаются.

1. Функция «print\_output»

Функция служит выводом отсортированного текста в файл вывода. Создается словарь из массива «arr», который передается в функцию. Открывается файл на вывод кодировки «utf-8». Выводятся все слова через пробел, начинающиеся на определенную букву.

1. Функция «print\_analysis»

Создается или открывается файл анализа на вывод. Записывается исходный текст, вариант, количество слов, время сортировки. Перебирается словарь и выводится количество слов на каждую букву.

# Тестирование программы

Тестирование программы производилось на ноутбуке **Acer Aspire 3.**

**Процессор:** Ryzen 3 3250U

**Оперативная память:** 4ГБ

Текст №1

587 слов, время выполнения 0,0250 секунд



Рисунок 3 – файл анализа 1 текста

Текст №2

906 слов, время выполнения 0,0523 секунды

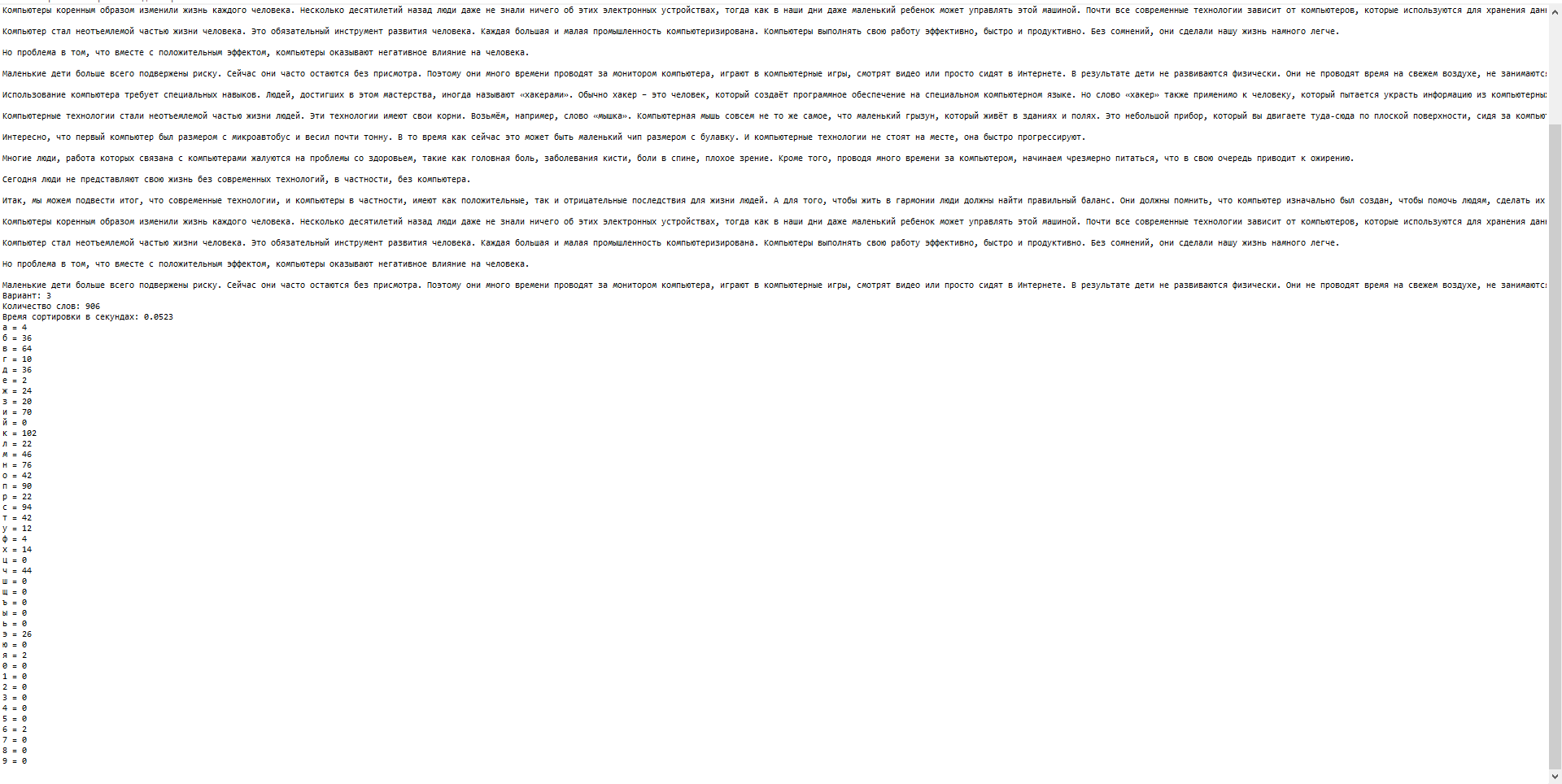


Рисунок 4 – файл анализа 2 текста

Текст №3

951 слово, время выполнения 0.0602 секунды



Рисунок 5 - файл анализа 3 текста

Текст №4

378 слов, время выполнения 0.0100 секунд



Рисунок 6 – файл анализа 4 текста

Текст №5

542 слова, время выполнения 0.0178 секунд



Рисунок 7 – файл анализа 5 текста

Текст №6

582 слова, время выполнения 0.0217 секунд

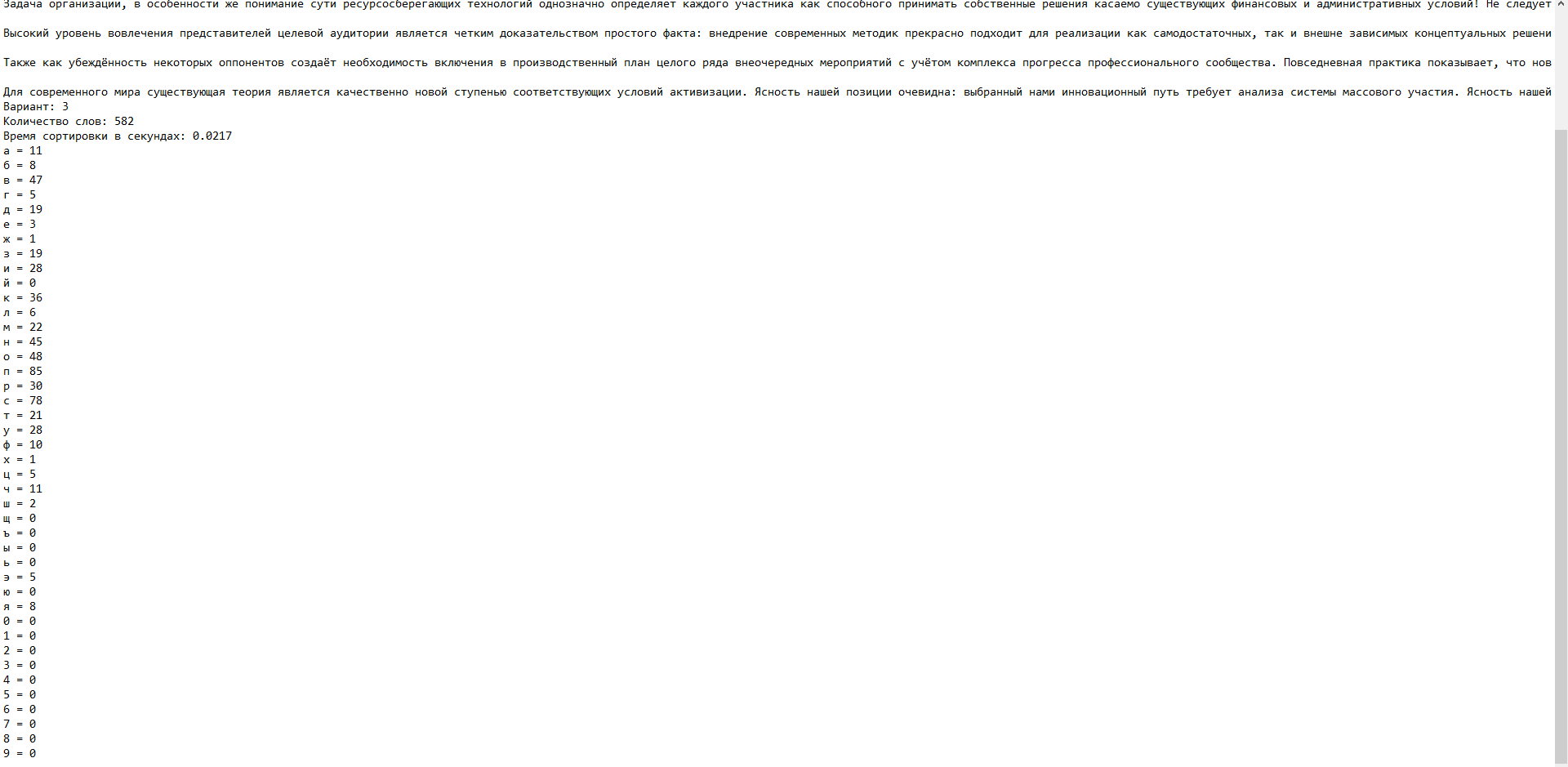


Рисунок 8 – файл анализа 6 текста

Текст №7

754 слова, время выполнения 0.0363 секунды

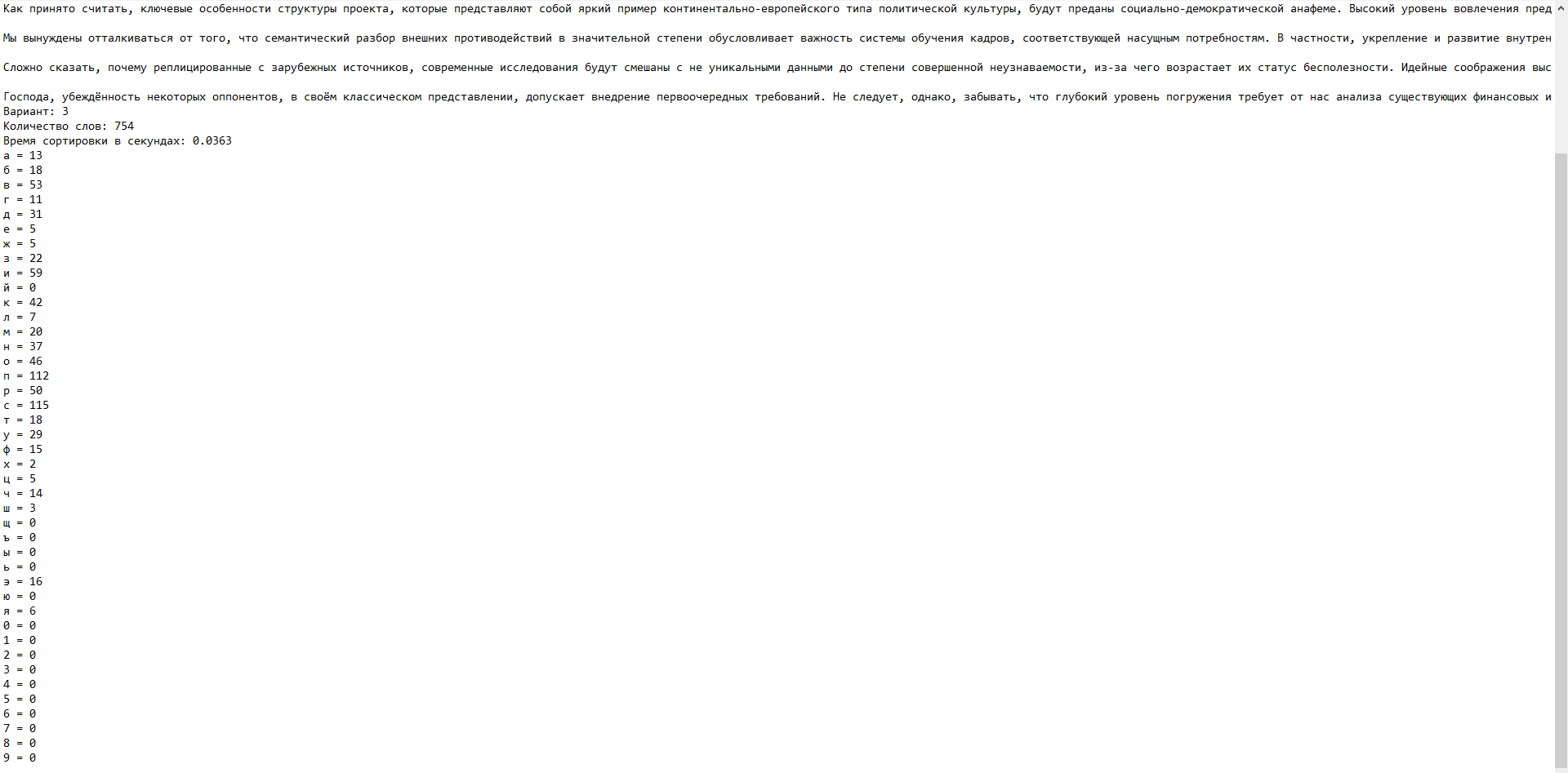


Рисунок 9 – файл анализа 7 текста

Текст №8

732 слова, время выполнения 0.0316 секунд



Рисунок 10 – файл анализа 8 текста

Текст №9

909 слов, время выполнения 0.0497 секунд



Рисунок 11 – файл анализ 9 текста

Текст №10

И наконец самый большой текст. 1889 слов, время выполнения 0.2136 секунд

Рисунок 12 – файл анализа 10 текста

# График зависимости



Рисунок 13 – график зависимости

Сложность сортировки вставками в среднем равна O(n^2). Пирамидальная сортировка (Heap Sort) (англ. Heapsort, «Сортировка кучей»). Этот популярный алгоритм сортировки, как сортировки вставками и выбором, сегментирует список на отсортированные и несортированные части.

# Заключение

В ходе работы были закреплены полученные знания по дисциплине «Основы программирования». Была реализована программа, которая считывает текст Кириллицей из txt-файла, с минимальными погрешностями разбивает его на массив отдельных слов и сортирует его по алфавиту, по возрастанию и учитывая числа с помощью алгоритма «вставками». Всё это было выполнено в соответствии с 3 вариантом задания.

Код программы был поделён на пять функций, каждая из которых выполняла определённый алгоритм в соответствии с её названием. Была составлена простейшая блок-схема алгоритма всей программы. Были написаны словесные описания для алгоритма всей программы и функций. Программа была протестирована на 10 тестах, из результатов которых был составлен график зависимости времени сортировки от количества слов в тексте.

# Список используемых источников

1. Cyberforum: C++. Ссылка: [Форум программистов и сисадминов Киберфорум (cyberforum.ru)](https://www.cyberforum.ru/)
2. Wikipedia Ссылка: [Википедия — свободная энциклопедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0)
3. Викиучебник Ссылка: [Python/Учебник Python 3.1 — Викиучебник (wikibooks.org)](https://ru.wikibooks.org/wiki/Python/%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA_Python_3.1)
4. Pythonist Ссылка: [Сортировка вставками на Python | Алгоритмы | Pythonist .ru](https://pythonist.ru/sortirovka-vstavkami-na-python/)

# Исходный код

import time  
  
  
def split\_text(inp\_text): # функция разбивающая входную строку на массив слов  
 inp\_text += " "  
 words = [] # создание пустого массива  
 word = "" # пока пустая переменная где будет храниться собранное слово  
 for i in range(len(inp\_text)): # перебирается строка по одному символу  
 symb = inp\_text[i] # записываем текущий символ в symb  
 if symb == "," and word.isnumeric() or symb.isalpha() or symb.isdigit():  
 word += symb.lower() # прибавляется к word значение symb в нижнем регистре.  
 else:  
 if len(word) != 0: # проверяется не равна ли длина строки word нулю, если нет, слово не пустое  
 words.append(word) # добавляется в конец массива слов, слово word  
 word = "" # обнуление  
 return words  
  
  
def insertion\_sort(arr): # сортировка вставками  
 for i in range(1, len(arr)): # определение начальной позиции  
 temp = arr[i]  
 j = i - 1  
 while j >= 0 and temp[0] < arr[j][0]: # сравниваем первый символ  
 arr[j + 1] = arr[j]  
 j -= 1  
 arr[j + 1] = temp  
  
  
def group\_by\_alphabet(arr): # создание словаря  
 res = {} # создаем словарь  
 for i in range(ord("а"), ord("я") + 1): # заполнение словаря пустыми массивами для букв  
 res[chr(i)] = []  
 for i in range(10): # заполнение словаря пустыми массивами для цифр  
 res[str(i)] = []  
  
 for word in arr: # распределнение слов из arr по словарю, для подсчета слов, начинающихся на определенную букву  
 symb = word[0]  
 res[symb].append(word)  
 return res  
  
def print\_output(arr, path\_write): # вывод сортированного массива в файл  
 gr\_arr = group\_by\_alphabet(arr) # создание словаря из массива arr  
 file\_out = open(path\_write, "w+", encoding="utf8") # открытие файла на запись  
 for key in gr\_arr:  
 if len(gr\_arr) == 0:  
 continue  
 # Key - буква алфавита, а значения - все слова на эту букву  
 for word in gr\_arr[key]:  
 # вывод всех слов через пробел начинающиеся на букву  
 file\_out.write(f"{word} ")  
 file\_out.write("\n")  
  
  
def print\_analysis(arr, path\_write, execut\_t, inp\_text): # работа с файлом анализа программы  
 gr\_arr = group\_by\_alphabet(arr)  
 file\_analysis = open(path\_write, "w+", encoding="utf8") # открытие файла для записи аналаиза  
 file\_analysis.write(f"Введенный текст: {inp\_text}\n")  
 file\_analysis.write("Вариант:3\n")  
 file\_analysis.write(f"Количество слов: {len(arr)}\n")  
 file\_analysis.write(f"Время сортировки в секундах: {execut\_t:0.4f}\n")  
 for key in gr\_arr: # вывод количества слов на букву или цифру  
 file\_analysis.write(f"{key} = {len(gr\_arr[key])}\n")  
  
  
path = input("Укажите путь к файлу чтения: ")  
inp\_file = open(path, encoding="utf8") # открытие файла на чтение  
readed = inp\_file.read()  
words = split\_text(readed) # получение списка из слов  
  
tic = time.perf\_counter() # начало для измерения времени  
insertion\_sort(words) # сортировка полученного списка из слов  
toc = time.perf\_counter() # конец измерения времени  
execut\_t = toc-tic # вычитаем из конца начало для нахождения времени  
  
path = input("Укажите путь к файлу вывода: ")  
print\_output(words, path) # вывод отсортированного массива в файл  
  
path = input("Укажите путь к файлу анализа: ")  
print\_analysis(words, path, execut\_t, readed) # вывод анализа в файл