

***Кафедра информационных систем***

|  |  |
| --- | --- |
| ***Рейтинговая работа*** | ***Расчетно-аналитическое задание*** |
| ***по дисциплине*** | ***Алгоритмизация и программирование*** |
|  |  |
| ***Задание / Вариант №*** | ***Индивидуальное задание в соответствие с ФИО и ID*** |
|  |  |
| ***Тема*** | ***Алгоритмы типовых задач*** |
|  |  |
| ***Выполнена обучающимся группы*** | о.ИЗДтс 23.2/Б-21 |
| ***ФИО обучающегося*** | Воробьев Кирилл Алексеевич |
|  |  |
| ***Преподаватель*** | Зайцев Сергей Александрович |

Москва – 2022 г.

Оглавление

[ЗАДАНИЕ С ИСХОДНЫМИ ДАННЫМИ 3](#_Toc95745163)

[ОПИСАНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc95745164)

[ОБОСНОВАНИЕ ВЫБОРА АЛГОРИТМОВ СОРТИРОВКИ ДАННЫХ 6](#_Toc95745165)

[ЛИСТИНГ ПРОГРАММНОГО КОДА 7](#_Toc95745166)

[РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ 12](#_Toc95745167)

[СПИСОК ИСТОЧНИКОВ 13](#_Toc95745168)

# ЗАДАНИЕ С ИСХОДНЫМИ ДАННЫМИ

Дан ряд целых чисел, полученный в результате обработки символов Фамилии, Имени, Отчества обучающегося, а также его ID. Необходимо отсортировать набор данных двумя способами, а также выполнить некоторые арифметические расчеты.

Исходные данные:

- ФИО: Воробьев Кирилл Алексеевич

- ID: 70175468

Данные заранее были записаны в файл source\_data.txt в кодировке Unicode (Юникод) в виде, заранее определённом шаблоном задания.

Необходимо написать программу на языке программирования Python, которая выполняет следующие задачи:

1. Считать исходные данные из текстового файла source\_data.txt

2. Вычислить целое значение, получившееся в результате деления ID на количество символов, составляющих ФИО (количество символов подсчитывается **без учета пробелов**).

3. Определить направление сортировки в зависимости от числа, полученного в п.2:

- сортировка по возрастанию, если число чётное;

- сортировка по убыванию, если число нечётное.

4. Сформировать набор данных (список) из кодов Юникода **каждого символа ФИО** (исключая пробелы между словами), переведенных в десятичную форму.

5. Выполнить сортировку набора данных по убыванию или по возрастанию. **Сортировку выполнить двумя различными способами**, например: сортировка выбором и сортировка методом простого обмена (пузырька). В программном коде должны быть представлены комментарии, обозначающие тот или иной алгоритм сортировки.

6. Вычислить среднее арифметическое значение набора данных. При необходимости, результат округлить до третьего знака после запятой.

7. Вычислить среднее квадратическое значение набора данных. При необходимости, результат округлить до третьего знака после запятой.

# ОПИСАНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ ВЫПОЛНЕНИЯ РАБОТЫ

После подготовки исходных данных в файле *source\_data.txt* был создан python-скрипт *Воробьев\_КА.py*, в котором перед началом написания программы была определеная главная функция *main()*, а так же записано стартовое условие

*if \_\_name\_\_ == “\_\_main\_\_”:*

*main()*

Данная конструкция гарантирует выполнение программы исключительно в случае запуска скрипта в качестве исполняемого файла и предотвращает какие-либо действия программы при подключении её в качестве модуля.

Первым делом необходимо получить исходные данные из текстового файла. Для этого была создана функция *get\_source\_data()* и соответствующая результату её выполнения переменная *src\_data* в фукнции *main().* Функция открывает файл с исходными данными в режиме *“r” – режиме чтения –* и возвращает список типа [ФИО, ID], методом *split()* разрезав исходные данные по переносу строки.

Далее необходимо высчитать частное от деления ID на количество символов в ФИО без учёта пробелов. Функция *get\_quotient()* принимает в качестве параметра список с исходными данными, создаёт локальную переменную *\_src\_fio*, которой присваивает первую позицию списка, соответствующую ФИО, и вырезает из неё все пробелы методом *replace()*. После чего делит приведённое к целочисленному типу ID на длину получившейся строки и возвращает полученное значение в переменную *quotient.*

Вопреки изначальному алгоритму из задания, на данном этапе целесообразно составить список символов в кодировке Юникод. Функция *get\_unicode\_list()* принимает в качестве аргумента часть списка с изначальными данными, конкретно – позицию с ФИО, локально избавляет строку от пробелов и, проходя каждый символ в цикле *for-in* и переводя его в кодировку Юникод с помощью метода *ord()*, записывает получившиеся значения в локальный список *\_unicode\_list*, после чего возвращает финальный результат.

Для удобства сразу несколько переменных будут получать свои значения из одной из двух функций сортировки. Отсортированный список, тип частного от деления и тип сортировки списка будут возвращаться функциями сортировки по возрастанию (*sort\_ascending()*) или сортировки по убыванию (*sort\_descending()*), в зависимости от результатов однострочного сравнения остатка от деления частного на 2 – если оно равно нулю, то частное – чётное и над списком символов Юникод будет произведена сортировка по возрастанию, если нечётное – по убыванию.

Обе функции принимают в качестве параметра список закодированных символов. Так же обе определяют локальные значения типа частного и типа сортировки. *sort\_ascending()* использует алгоритм сортировки пузырьком, *sort\_descending()* – алгоритм сортировки слиянием, но перед возвращением результатов реверсирует список, перебирая его срез с шагом -1.

Среднее арифметическое значение набора данных вычисляет функция *get\_average()*, принимающая в качестве параметра массив с кодировкой Юникод. С помощью цикла *for-in* функция складывает каждый элемент массива, приведённый к целочисленному типу, после чего возвращает результат деления суммы на длину массива, округлённый до третьего знака после запятой.

Среднее квадратическое значение набора данный вычисляет функция *get\_square()*, принимающая в качестве параметра массив с кодировкой Юникод. С помощью цикла *for-in* функция складывает квадраты каждого элемента массива, приведённые к целочисленному типу, после чего возвращает квадратный корень частного от деления суммы на длину массива, округляя результат до третьего знака после запятой.

В конце работы программа заполняет шаблон из многострочного блока *f-строки*, подставляя вместо переменных полученные данные и записывает результат работы в файл *result.txt*

# ОБОСНОВАНИЕ ВЫБОРА АЛГОРИТМОВ СОРТИРОВКИ ДАННЫХ

В программе, согласно изначальному заданию, использованы два алгоритма сортировки:

-Пузырьковая сортировка, используется при сортировке по возрастанию

-Сортировка слиянием, используется при сортировке по убыванию

Сортировка простыми обменами, сортировка пузырьком – простой учебный алгоритм с постоянной сложностью O(N2). Хоть вне учебной литературы алгоритм практически не применяется, однако для небольших массивов, как в случае с конкретно этим заданием, он крайне эффективен.

Сортировка слиянием обладает гораздо меньшей сложностью, O(NlogN), однако превосходит другие подобные алгоритмы за счёт возможности асинхронного выполнения, существенно снижая временные затраты.

# ЛИСТИНГ ПРОГРАММНОГО КОДА

*def get\_source\_data(src\_url:str):*

*with open(src\_url, "r", encoding="utf-8") as file:*

*\_src = file.read().split("\n")*

*return \_src*

*def get\_quotient(src\_data:list):*

*"""Вычисление частного от деления ID на количество букв в ФИО без учёта пробелов"""*

*\_src\_fio = src\_data[0].replace(" ", "")*

*return int(src\_data[1]) // len(\_src\_fio)*

*def get\_unicode\_list(src\_string:str):*

*"""Получение списка символов ФИО в кодировке Юникод"""*

*\_unicode\_list:list = []*

*\_src = src\_string.replace(" ", "")*

*for i in range(0, len(\_src)):*

*\_unicode\_list.append(ord(\_src[i]))*

*return \_unicode\_list*

*def merge(left, right):*

*\_sorted\_list = []*

*l\_index = r\_index = 0*

*l\_len, r\_len = len(left), len(right)*

*for i in range(l\_len + r\_len):*

*if l\_index < l\_len and r\_index < r\_len:*

*if left[l\_index] <= right[r\_index]:*

*\_sorted\_list.append(left[l\_index])*

*l\_index += 1*

*else:*

*\_sorted\_list.append(right[r\_index])*

*r\_index += 1*

*elif l\_index == l\_len:*

*\_sorted\_list.append(right[r\_index])*

*r\_index += 1*

*elif r\_index == r\_len:*

*\_sorted\_list.append(left[l\_index])*

*l\_index += 1*

*return \_sorted\_list*

*def merge\_sort(src\_unicode\_list:list):*

*"""Сортировка слиянием"""*

*\_list = src\_unicode\_list.copy()*

*if len(\_list) == 1 or len(\_list) == 0:*

*return \_list*

*left = merge\_sort( \_list[:len(\_list) // 2])*

*right = merge\_sort(\_list[len(\_list) // 2:])*

*return merge(left, right)*

*def bubble\_sort(src\_unicode\_list:list):*

*"""Сортировка пузырьком"""*

*\_sorted\_list = src\_unicode\_list.copy()*

*for i in range(len(\_sorted\_list)-1):*

*for j in range(len(\_sorted\_list)-1):*

*if \_sorted\_list[j] > \_sorted\_list[j+1]:*

*\_sorted\_list[j], \_sorted\_list[j+1] = \_sorted\_list[j+1], \_sorted\_list[j]*

*return \_sorted\_list*

*def sort\_ascending(src\_unicode\_list:list):*

*"""Сортировка по возрастанию"""*

*\_sorted\_list = bubble\_sort(src\_unicode\_list)*

*\_quo\_parity = "чётное"*

*\_sorted\_type = "по возрастанию"*

*return \_sorted\_list, \_quo\_parity, \_sorted\_type*

*def sort\_descending(src\_unicode\_list:list):*

*"""Сортировка по убыванию"""*

*\_sorted\_list = merge\_sort(src\_unicode\_list)*

*\_quo\_parity = "нечётное"*

*\_sorted\_type = "по убыванию"*

*return \_sorted\_list[::-1], \_quo\_parity, \_sorted\_type*

*def get\_average(src\_list:list):*

*"""Среднее арифметическое"""*

*\_sum = 0*

*for i in range(0, len(src\_list)):*

*\_sum += int(src\_list[i])*

*return round(\_sum / len(src\_list), 3)*

*def get\_square(src\_list:list):*

*"""Среднее квадратическое"""*

*\_sum = 0*

*for i in range(0, len(src\_list)):*

*\_sum += int(src\_list[i])\*\*2*

*return round((\_sum / len(src\_list))\*\*0.5, 3)*

*def main():*

*src\_data = get\_source\_data("source\_data.txt")*

*quotient = get\_quotient(src\_data)*

*unicode\_list = get\_unicode\_list(src\_data[0])*

*sorted\_list, quo\_parity, sorted\_type = sort\_ascending(unicode\_list) if quotient % 2 == 0 else sort\_descending(unicode\_list)*

*average = get\_average(unicode\_list)*

*square = get\_square(unicode\_list)*

*with open("result.txt", "w", encoding="utf-8") as file:*

*file.write(f"""*

*1. Исходные данные: {src\_data[0]}; ID: {src\_data[1]}*

*2. {quotient}*

*3. Направление сортировки: {sorted\_type}, так как число {quotient} - {quo\_parity}*

*4. Набор данных: {unicode\_list}*

*5. Отсортированный {sorted\_type} набор данных: {sorted\_list}*

*6. Среднее арифметическое значение: {average}*

*7. Среднее квадратическое значение: {square}"""[1:])*

*if \_\_name\_\_=='\_\_main\_\_':*

*main()*

# РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В результате выполнения программы файл *result.txt* будет иметь следующий вид:

*1. Исходные данные: Воробьев Кирилл Алексеевич; ID: 70175468*

*2. 2923977*

*3. Направление сортировки: по убыванию, так как число 2923977 - нечётное*

*4. Набор данных: [1042, 1086, 1088, 1086, 1073, 1100, 1077, 1074, 1050, 1080, 1088, 1080, 1083, 1083, 1040, 1083, 1077, 1082, 1089, 1077, 1077, 1074, 1080, 1095]*

*5. Отсортированный по убыванию набор данных: [1100, 1095, 1089, 1088, 1088, 1086, 1086, 1083, 1083, 1083, 1082, 1080, 1080, 1080, 1077, 1077, 1077, 1077, 1074, 1074, 1073, 1050, 1042, 1040]*

*6. Среднее арифметическое значение: 1077.667*

*7. Среднее квадратическое значение: 1077.762*

# СПИСОК ИСТОЧНИКОВ

<https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html>

<https://en.wikipedia.org/wiki/Bubble_sort>

<https://ru.wikipedia.org/wiki/Сортировка_слиянием>