Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №5**

**«ИССЛЕДОВАНИЕ АЛГОРИТМОВ СОРТИРОВКИ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРИВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Балахнина Елизавета Ефимовна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

Содержание

1. Цель лабораторной работы
2. Формулировка задания (с вариантом)
3. Описание алгоритма
4. Схема алгоритма с комментариями
5. Код программы
6. Результат выполнения программы
7. Вывод

**1.Цель контрольной домашней работы:** получить базовые сведения о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами.

**2.Формулировка задания (Вариант:2)**

Вариант 2

1.Реализовать сортировку данных с помощью «пузырькового» алгоритма подсчетом.

2.Реализовать сортировку данных с помощью быстрого алгоритма.

3.В обоих случаях необходимо предусматривать возможность изменения компаратора (реализация компаратора в виде передаваемой в подпрограмму функции).

4. Считывание и вывод данных необходимо производить из текстового файла.

5. Для демонстрации работы программных реализаций самостоятельно подготовить варианты входных данных (при этом объём текстовых файлов должен позволять оценить скорость работы скорость работы программ).

**3. Описание алгоритма**

***Задание № 1***

1. Создаем функцию bubble\_sort(arr, comparator), которая принимает список arr и функцию comparator. Функция bubble\_sort будет сортировать список arr с помощью сортировки пузырьком, используя функцию comparator для определения порядка сортировки.

2. Получаем длину списка arr и сохраняем ее в переменную n.

3. Запускаем цикл for i in range(n-1), который проходит по всем элементам списка arr, кроме последнего.

4. Внутри этого цикла запускаем второй цикл for j in range(0, n-i-1), который будет проходить по парам соседних элементов списка arr.

5. Внутри второго цикла проверяем условие if comparator(arr[j], arr[j+1]). Если условие истинно, то меняем местами элементы arr[j] и arr[j+1].

6. Создаем две функции less\_than(a, b) и greater\_than(a, b), которые используются в качестве аргументов функции bubble\_sort для определения порядка сортировки. Функция less\_than(a, b) возвращает True, если элемент a меньше элемента b, и False в противном случае. Функция greater\_than(a, b) возвращает True, если элемент a больше элемента b, и False в противном случае.

7. В блоке кода после if \_\_name\_\_ == "\_\_main\_\_": открываем файл "input.txt" в режиме чтения и читаем его содержимое. Разделяем полученные строки на отдельные элементы и преобразуем их в целые числа, сохраняя в переменную numbers.

8. Вызываем функцию bubble\_sort с аргументами numbers и функция less\_than для сортировки списка numbers в порядке возрастания.

9. Открываем файл "output.txt" в режиме записи и записываем отсортированные числа из списка numbers в каждую строку файла.

10. Завершаем выполнение программы.

***Задание № 2***

1. Определяется функция quick\_sort, которая принимает один аргумент - массив.

2. Если длина массива меньше или равна 1, возвращается сам массив. В противном случае, выбирается опорный элемент (pivot) - средний элемент массива.

3. Создаются три новых массива: less, equal и greater. В less добавляются все элементы из массива, которые меньше опорного элемента. В equal добавляются все элементы из массива, которые равны опорному элементу. В greater добавляются все элементы из массива, которые больше опорного элемента.

4. Вызывается функция quick\_sort для массива less, функция quick\_sort для массива greater. Результаты рекурсивных вызовов объединяются с equal и возвращаются.

5. Открывается файл "input2.txt" для чтения.

12. Создается список numbers, в который добавляются все строки из файла, преобразованные в целые числа. Затем вызывается функция quick\_sort, передавая ей список numbers.

14. Результат сортировки сохраняется в переменной sorted\_numbers.

15. Открывается файл "output2.txt" для записи. В этот файл записывается отсортированный список, преобразованный в строки.

17. Выводится отсортированный список на экран.

**4. Схема алгоритма с комментариями**

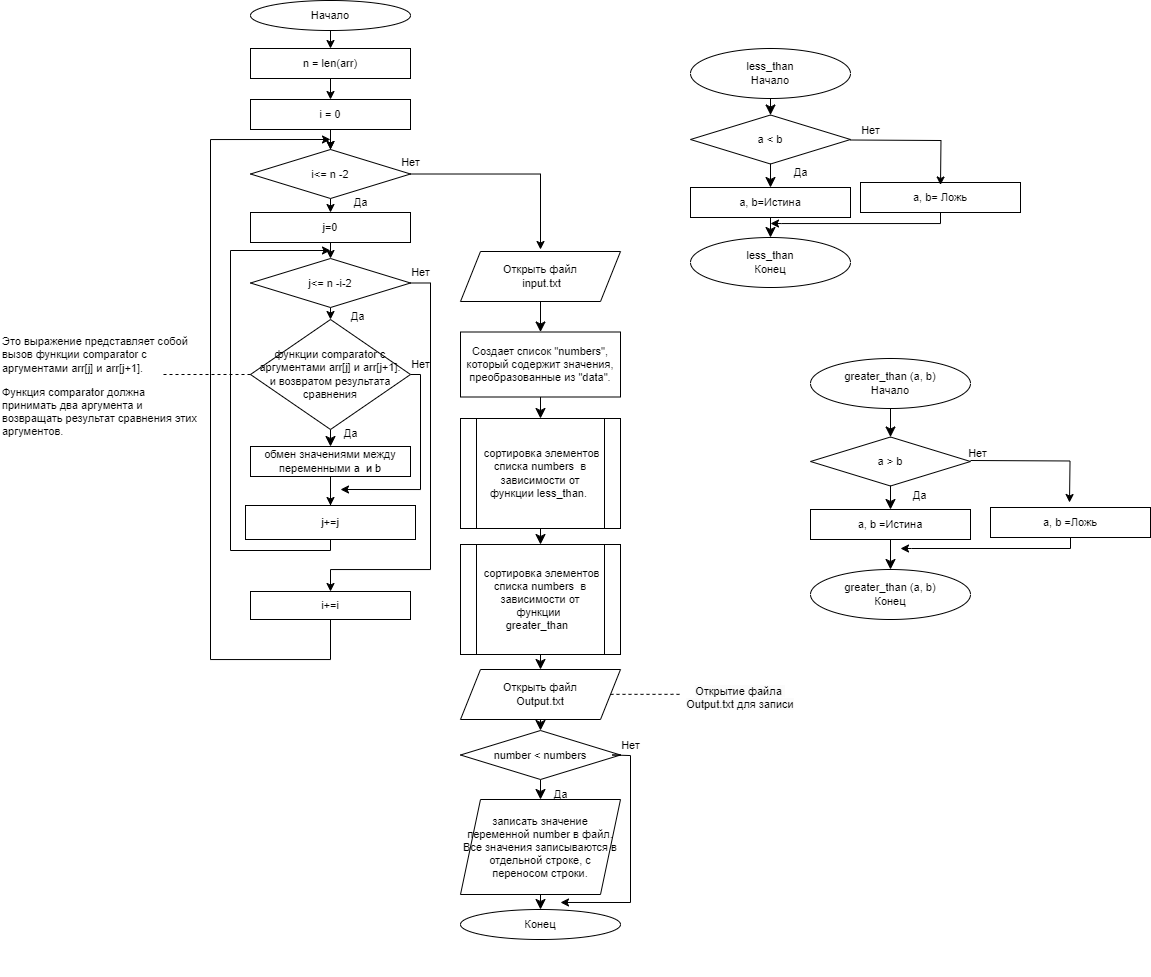
****

Рис 1. Схема алгоритма (задание 1) с комментариями.

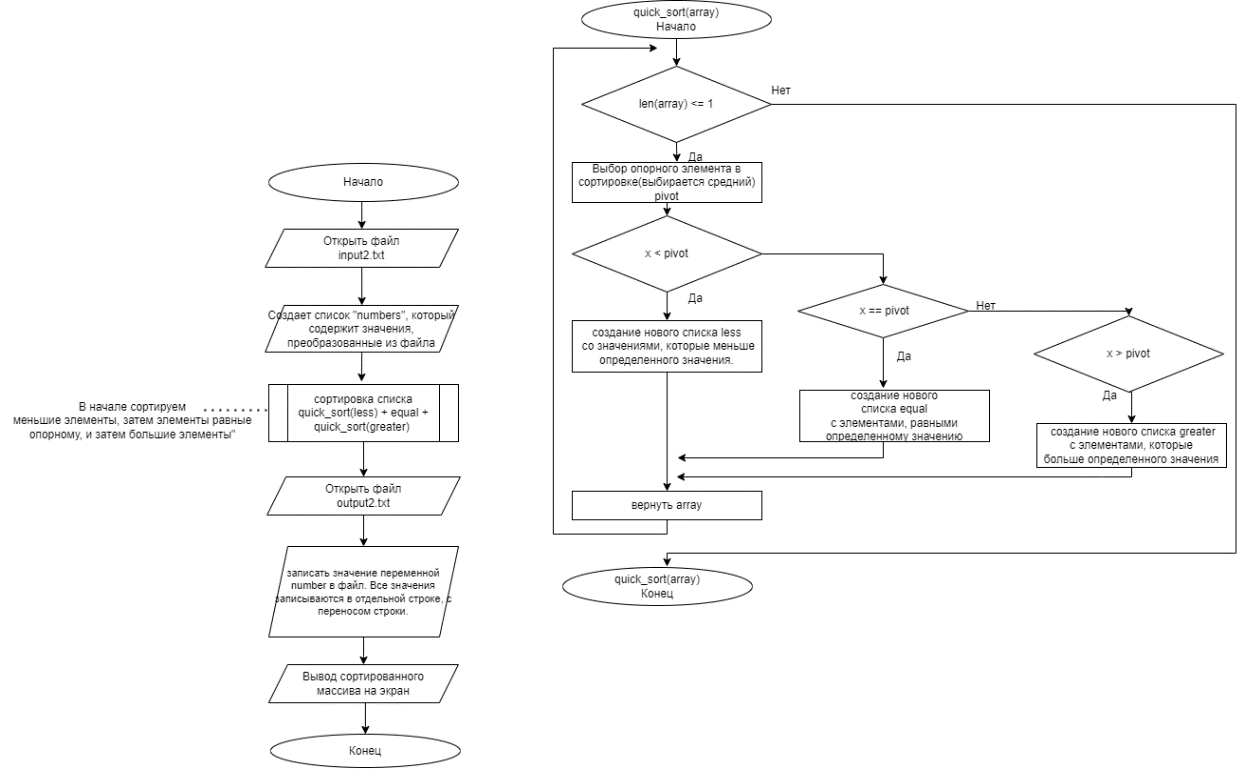


Рис 2. Схема алгоритма (задание 2) с комментариями.

**4. Код программы**

**Задача №1**

def bubble\_sort(arr, comparator):

n = len(arr)

for i in range(n-1):

for j in range(0, n-i-1):

if comparator(arr[j], arr[j+1]):

arr[j], arr[j+1] = arr[j+1], arr[j]

def less\_than(a, b):

return a < b

def greater\_than(a, b):

return a > b

if \_\_name\_\_ == "\_\_main\_\_":

with open("input.txt", "r") as file:

data = file.read().splitlines()

numbers = [int(number) for number in data]

bubble\_sort(numbers, less\_than)

with open("output.txt", "w") as file:

for number in numbers:

file.write(str(number) + "\n")

**Задача №2**

def quick\_sort(array):

if len(array) <= 1:

return array

pivot = array[len(array) // 2]

less = [x for x in array if x < pivot

equal = [x for x in array if x == pivot]

greater = [x for x in array if x > pivot]

return quick\_sort(less) + equal + quick\_sort(greater)

with open("input2.txt", "r") as file:

numbers = [int(line) for line in file.readlines()]

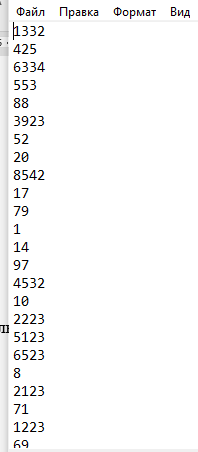
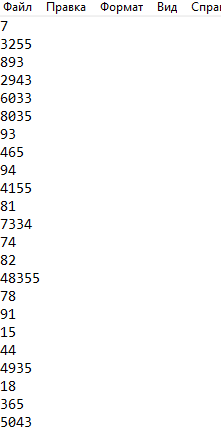
sorted\_numbers = quick\_sort(numbers)

with open("output2.txt", "w") as file:

file.write("\n".join(map(str, sorted\_numbers)))

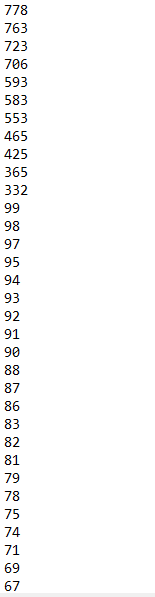
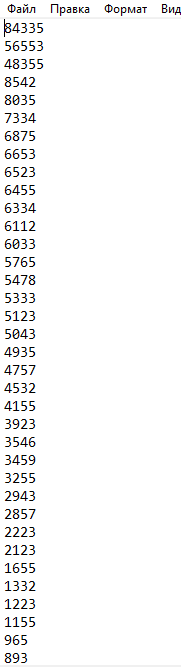
print(sorted\_numbers)

**5. Результат выполнения программы**

**Задача №1**

***(Исходный файл)***

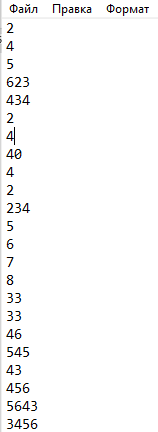
Рис.3. Результат выполнения программы (задание 1)

****

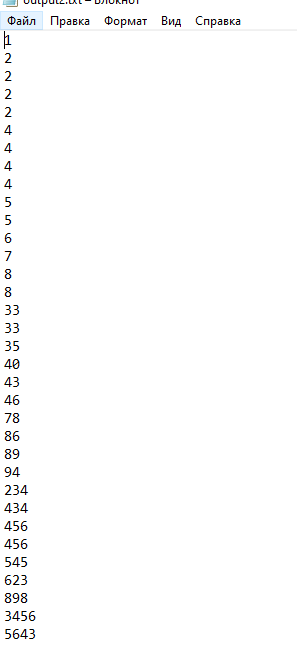
***(Отсортированный файл)***

Рис.4. Результат выполнения программы (задание 1)

***Задача № 2***

******

***(Исходный файл)***

Рис.5. Результат выполнения программы (задание 2)

***(Отсортированный файл)***

Рис.6. Результат выполнения программы (задание 2)

**6. Вывод**

В ходе этой лабораторной работы были изучены базовые структуры организации программы и основные конструкции языка программирования Python. Была создана программа, сортирующая числа, которая берёт числа из файла. Во время сортировки чисел были изучены некоторые алгоритмы сортировки, такие как «пузырьковый» и быстрый.

Пузырьковый алгоритм — это простой алгоритм выполняет итерации по списку, сравнивая элементы попарно и меняя их местами, пока более крупные элементы не «всплывут» в начало списка, а более мелкие не останутся на «дне».

Быстрый алгоритм также относится к алгоритмам «разделяй и властвуй». Его используют чаще других алгоритмов, описанных в этой статье. При правильной конфигурации он чрезвычайно эффективен и не требует дополнительной памяти, в отличие от сортировки слиянием. Массив разделяется на две части по разные стороны от опорного элемента. В процессе сортировки элементы меньше опорного помещаются перед ним, а равные или большие — позади.

Также в процессе выполнения работы, мы познакомились с сайтом diagrams.net. Diagrams.net — это бесплатный онлайн-инструмент для создания диаграмм, схем, графиков и других визуальных элементов. Он представляет удобный интерфейс с возможностью создания различных типов диаграмм, таких как блок-схемы, организационные системы и многие другие.