# **ФГБОУ ВО «Московский политехнический университет»**

## **Лабораторная работа №6**

Файлы

**Задание 1 Вариант №11**

По дисциплине:

Программирование

Выполнил

студент 1 курса

группы 201-321

Колобов К.В.

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_ Никишина И.Н.

# **МОСКВА 2020**

**Цель работы**

Получить практический навык в написании программ, в которых выполняются

операции с текстовыми файлами – чтение, запись.

**Постановка задачи**

В предыдущих заданиях необходимые для программы данные, вводились с

клавиатуры, а результат выводился на экран монитора. Очевидно, что и при отладке

программ, и при вводе большого объема данных в программу такой подход трудозатратен

и потому непригоден. Наиболее подходящее решение - это ввод данных из файла и вывод

результатов работы в файл. При этом входные данные подготавливаются однократно и с

должным многообразием, а результаты работы можно анализировать многократно.

Напишем несколько программ, которые будут считывать входные данные из

заранее подготовленного текстового файла, и выводить результат в текстовый файл. Для

этой цели воспользуемся программами, написанными в предыдущих заданиях.

**Теоретическая часть**

Файловые типы данных введены в языках программирования для работы с

внешними устройствами – файлами на диске, портами, принтерами и т.д. Файловые типы

подразделяются на стандартные типы и типы, определяемые программистом. К

стандартному типу относится текстовый тип файлов, с которым мы и будем работать.

Доступ к файлам может быть последовательным или прямым. При

последовательном доступе каждый следующий элемент может быть прочитан только

после выполнения аналогичной операции с предыдущим элементом. При прямом доступе

операция чтения (записи) может быть выполнена для произвольного элемента с заданным

адресом.

Текстовые файлы относятся к файлам с последовательным доступом. Они

предназначены для хранения информации строкового типа. При этом ввод и вывод

информации сопровождается преобразованием типов данных. При выводе в текстовый

файл данные преобразуются из внутреннего представления в символы, а при вводе

выполняется обратное преобразование.

**Описание программы**

Программа написана на Python 3.8, реализована в среде ОС GNU/Linux Ubuntu 20.04.

**Описание алгоритма**

Алгоритмы делают то же самое, что и в соответствующих лабораторных работах, при

этом изменён поток ввода и выводы на файл.

**Описание входных и выходных данных**

Формат входных и выходных данных аналогичен соответствующим лабораторным

работам, но изменён поток со стандартного на файловый.

**Листинг программы (по з. № 1)**

import math

from pathlib import Path

input\_file = Path("task\_1\_input.txt")

output\_file = Path("task\_1\_output.txt")

def main():

try:

with input\_file.open('r') as inp:

angle = next(inp)

angle\_in\_gradus = float(angle)

except FileNotFoundError:

print("File with input data not found",

file=output\_file.open('w'))

return

except ValueError:

print("Wrong angle given, numerical value was expected",

file=output\_file.open('w'))

return

angle\_in\_rads = math.radians(angle\_in\_gradus)

try:

z1 = ((1 - 2 \* math.sin(angle\_in\_rads)\*\*2) /

(1 + math.sin(angle\_in\_rads \* 2)))

except ZeroDivisionError:

print("It's impossible to calculate the value, division by zero",

file=output\_file.open('w'))

else:

print(f"z1 = {z1:.10f}", file=output\_file.open('w'))

try:

z2 = ((1 - math.tan(angle\_in\_rads)) /

(1 + math.tan(angle\_in\_rads)))

except ZeroDivisionError:

print("It's impossible to calculate the value, division by zero",

file=output\_file.open('a'))

else:

print(f"z2 = {z2:.10f}", file=output\_file.open('a'))

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

main()

**Результат работы программы (файл task\_1\_output.txt)**

z1 = -1.0000000000

z2 = -1.0000000000

**Листинг программы (по з. № 4)**

import random

from pathlib import Path

from typing import List

# skip the values in this range

WRONG\_RANGE\_START = -5

WRONG\_RANGE\_STOP = -3

# values will be generated in this range

VALUE\_START = -5

VALUE\_STOP = 5

input\_file = Path("task\_2\_input.txt")

output\_file = Path("task\_2\_output.txt")

def number\_of\_min\_element(sample: List[float]) -> int:

minimal = number = None

for num, element in enumerate(sample, 1):

if minimal is None or abs(element) < minimal:

minimal = abs(element)

number = num

return number

def index\_of\_first\_negative(sample: List[float]) -> int:

for index, item in enumerate(sample):

if item < 0:

return index

return -1

def sum\_of\_abs(sample: List[float]) -> float:

return sum(

abs(item) for item in sample

)

def change\_sample(sample: List[float]) -> List[float]:

res = []

wrong\_items\_count = 0

for item in sample:

if WRONG\_RANGE\_START <= item <= WRONG\_RANGE\_STOP:

wrong\_items\_count += 1

else:

res += [item]

res += [0] \* wrong\_items\_count

return res

def str\_sample(sample: List[float],

separator: str = ', ') -> str:

res = separator.join(

f"{item:.1f}" for item in sample

)

return res

def main() -> None:

try:

with input\_file.open() as f:

angle = next(f)

length = int(angle)

assert length <= 30

except FileNotFoundError:

print("File with input data not found",

file=output\_file.open('w'))

return

except ValueError:

print("Wrong input", file=output\_file.open('w'))

return

except AssertionError:

print("List length is expected to be less than 30",

file=output\_file.open('w'))

return

sample = [

random.uniform(-5, 5)

for \_ in range(length)

]

num\_of\_min = number\_of\_min\_element(sample)

first\_negative\_index = index\_of\_first\_negative(sample)

if first\_negative\_index == -1:

sum\_of\_abs\_ = "oops, there is no negative element"

else:

sum\_of\_abs\_ = sum\_of\_abs(sample[first\_negative\_index + 1:])

sum\_of\_abs\_ = f"{sum\_of\_abs\_:.1f}"

changed\_sample = change\_sample(sample)

with output\_file.open('w', encoding='utf-8') as f:

f.write(f"Length: {length}\n")

f.write(f"Sample: \n{str\_sample(sample)}\n")

f.write(f"Number of minimal by abs element: {num\_of\_min}\n")

f.write(f"Sum of abs of values after the first negative: {sum\_of\_abs\_}\n")

f.write(f"Changed sample: \n{str\_sample(changed\_sample)}")

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

main()

**Результат работы программы (файл task\_2\_output.txt)**

Length: 4

Sample:

-0.7, -2.3, 2.4, 2.5

Number of minimal by abs element: 1

Sum of abs of values after the first negative: 7.2

Changed sample:

-0.7, -2.3, 2.4, 2.5

**Листинг программы (по з. № 5)**

import random

from pathlib import Path

from typing import Any, List

RANGE\_START = -1

RANGE\_STOP = 1

input\_path = Path('task\_3\_input.txt')

output\_path = Path('task\_3\_output.txt')

class Matrix:

def \_\_init\_\_(self, rows: int, cols: int = None) -> None:

self.\_\_rows, self.\_\_cols = rows, cols or rows

self.\_\_matrix = Matrix.create\_matrix(self.rows, self.cols)

@property

def rows(self) -> int:

return self.\_\_rows

@property

def cols(self) -> int:

return self.\_\_cols

@classmethod

def create\_matrix(cls, rows: int, cols: int) -> Any:

matrix = [

[random.randint(RANGE\_START, RANGE\_STOP) for \_ in range(cols)]

for \_ in range(rows)

]

return matrix

def remove\_zeroes\_rows(self) -> None:

row\_index = 0

for \_ in range(self.rows):

if all(row == 0 for row in self[row\_index]):

del self[row\_index]

row\_index -= 1

row\_index += 1

self.\_\_rows = len(self)

def remove\_zeroes\_cols(self) -> None:

col\_index = 0

for \_ in range(self.cols):

if all(row[col\_index] == 0 for row in self):

for row in range(self.rows):

del self[row][col\_index]

col\_index -= 1

col\_index += 1

try:

self.\_\_cols = len(self[0])

except IndexError:

self.\_\_cols = 0

def num\_of\_positive\_row(self) -> int:

for num, row in enumerate(self, 1):

if any(item > 0 for item in row):

return num

return -1

def \_\_len\_\_(self) -> int:

return len(self.\_\_matrix)

def \_\_getitem\_\_(self, index: int) -> List[int]:

return self.\_\_matrix[index]

def \_\_setitem\_\_(self, index: int, new\_row: List[int]) -> None:

try:

self.\_\_matrix[index] = new\_row

except IndexError:

raise IndexError(f"Wrong row index: {index}")

def \_\_delitem\_\_(self, index: int) -> None:

try:

del self.\_\_matrix[index]

except IndexError:

raise IndexError(f"Wrong row index: {index}")

def \_\_str\_\_(self) -> str:

positive\_row = self.num\_of\_positive\_row()

if positive\_row == -1:

positive\_row = "There is no positive element"

else:

positive\_row = f"Num of the first row " \

f"with a positive item: {positive\_row}"

info = f"Matrix {self.rows} \* {self.cols}"

matrix = '\n'.join(str(row) for row in self)

return f"{info}\n{matrix}\n{positive\_row}"

def main() -> None:

try:

with input\_path.open() as f:

size = next(f).strip()

size = list(map(int, size))

assert 0 < len(size) <= 2

assert all(i > 0 for i in size)

except FileNotFoundError:

print("File with input data not found",

file=output\_path.open('w'))

return

except ValueError:

print("Wrong input",

file=output\_path.open('w'))

return

except AssertionError:

print("One of two natural values were expected",

file=output\_path.open('w'))

return

matrix = Matrix(\*size)

# matrix[0] = matrix[2] = matrix[-1] = [0] \* matrix.cols

with output\_path.open('w', encoding='utf-8') as f:

f.write(f"{matrix}\n")

f.write(f"{'\_' \* 20}\n")

matrix.remove\_zeroes\_rows()

f.write(f"{matrix}\n")

f.write(f"{'\_' \* 20}\n")

matrix.remove\_zeroes\_cols()

f.write(f"{matrix}\n")

f.write(f"{'\_' \* 20}\n")

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

main()

**Результат работы программы (файл task\_3\_output.txt)**

Matrix 5 \* 5

[1, 0, 0, 1, 1]

[-1, 1, -1, 1, 1]

[-1, 0, -1, 0, 1]

[0, 1, 0, 1, 0]

[-1, 0, 0, -1, 1]

Num of the first row with a positive item: 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Matrix 5 \* 5

[1, 0, 0, 1, 1]

[-1, 1, -1, 1, 1]

[-1, 0, -1, 0, 1]

[0, 1, 0, 1, 0]

[-1, 0, 0, -1, 1]

Num of the first row with a positive item: 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Matrix 5 \* 5

[1, 0, 0, 1, 1]

[-1, 1, -1, 1, 1]

[-1, 0, -1, 0, 1]

[0, 1, 0, 1, 0]

[-1, 0, 0, -1, 1]

Num of the first row with a positive item: 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Примечание**

Блок-схемы аналогичны соответствующим лабораторным работам.

**Список используемой литературы**

Документация языка Python: <https://docs.python.org/3/>