**ФГБОУ ВО “МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”**

**Лабораторная работа № 6**

Разветвляющиеся вычислительные процессы

**Задание 1**

по дисциплине:

Основы программирования

Выполнил:

студент 1 курса

группы 201-322

Кириллов Д.П

**МОСКВА 2020**

**Постановка задачи:**

Выполнить корректировку программ, чтобы ввод и вывод данных 1, 4, 5 л.р. осуществлялся с помощью файлов

**Теоретическая часть:**

Для решения задачи были использованы встроенные функции open(), write(), file.write(), file.close()

**Описание программы:**

Программа написана на алгоритмическом языке Python 3.8.5, реализована в среде ОС Windows 10

**Описание алгоритма:**

1. Открыть файл в режиме чтения с помощью функции open()
2. Разбить файл на массив состоящих из строк файлов file.readlines()
3. Высчитать нужную строку и преобразовать ее к числовому типу (массиву)
4. Закрыть файл с помощью метода file.close()
5. Выполнить вычисления
6. Открыть файл в режиме записи с помощью функции open()
7. Записать полученные значения с помощью функции file.write()
8. Закрыть файл с помощью функции file.close()

**Листинг программы (1 л.р.):**

from math import (cos, sin, tan, radians, pi)

try:

fileInput = open('input.txt', 'r')

linesInput = fileInput.readlines()

if (len(linesInput) != 2):

raise ValueError()

aDeg = int(linesInput[1])

a = radians(aDeg)

fileInput.close()

def y1(a):

if (cos(a) - sin(a)) == 0:

raise ZeroDivisionError()

return ((cos(a) + sin(a))/(cos(a) - sin(a)))

def y2(a):

if cos(2 \* a) == 0:

raise ZeroDivisionError()

return tan(2 \* a) + 1/cos(2 \* a)

fileOutput = open('output.txt', 'w', encoding="utf-8")

fileOutput.write('a = {0:.9f}, y1 = {1:.9f}, y2 = {2:.9f}'.format(a, y1(a), y2(a)))

fileOutput.close()

except ZeroDivisionError:

print('Программа не смогла посчитать данное выражение.')

except ValueError:

print('Введите корректное число.')

**Результат тестирования программы (1 л.р., папка task1):**

**input.txt:**

Введите значение аргумента (в градусах) на следующей строке:

12

**output.txt:**

a = 0.209439510, y1 = 1.539864964, y2 = 1.539864964

**Листинг программы (4 л.р.):**

from math import (ceil)

arr = []

try:

fileInput = open('input.txt', 'r')

linesInput = fileInput.readlines()

if (len(linesInput) < 2):

raise ValueError()

strInput = linesInput[1]

arr = strInput.split(' ')

arr = [int(item) for item in arr]

fileInput.close()

# Функция возвращающая минимальное число в массиве

def minArr(arr):

min = abs(arr[0])

for i in range(len(arr)):

if (abs(arr[i]) < min):

min = arr[i]

return min

# Сумма модулей элементов после нуля

def sumAfterZero(arr):

firtZeroIndex = -1

sum = 0

# Находим индекс первого элемента равного нулю

for i in range(len(arr)):

if arr[i] == 0:

firtZeroIndex = i

break;

# Находим сумму модулей элементов расположенных после firtZeroIndex

for i in range(firtZeroIndex, len(arr)):

sum += abs(arr[i])

return sum

# Преобразовывает массив таким образом, что первая половина элементов

# массива стоит на четных позициях, а вторая на нечетных

def changeArr(arr):

newArr = [0] \* len(arr)

indexEven = 0

indexOdd = 1

firstHalf = ceil(len(arr)/2)

arrFirst = arr[:firstHalf]

arrSecond = arr[firstHalf:]

for i in range(len(arrFirst)):

newArr[indexEven] = arrFirst[i]

indexEven += 2

for i in range(len(arrSecond)):

newArr[indexOdd] = arrSecond[i]

indexOdd += 2

return newArr

fileOutput = open('output.txt', 'w', encoding="utf-8")

fileOutput.write('Исходный массив: {}\n'.format(arr))

fileOutput.write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n')

fileOutput.write('Минимальное по модулю число: {}\n'.format(minArr(arr)))

fileOutput.write('Сумма модулей элементов после нуля: {}\n'.format(sumAfterZero(arr)))

fileOutput.write('Преобразованный массив: {}\n'.format(changeArr(arr)))

fileOutput.close()

except ValueError:

print('Введите корректное число.')

**Результат тестирования программы (4 л.р., папка task4):**

**input.txt:**

Введите элементы массива на следующей строке(через пробел):

1 2 0 3 -5

**output.txt:**

Исходный массив: [1, 2, 0, 3, -5]

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

Минимальное по модулю число: 0

Сумма модулей элементов после нуля: 8

Преобразованный массив: [1, 3, 2, -5, 0]

**Листинг программы (5 л.р):**

from random import (randint)

# Возвращает двумерный массив Row x Col со случайными значениями

# в диапазоне от min до max

def makeRandomArr(row, col, min, max):

arr = [[0] \* col for i in range(row)]

for Row in range(row):

for Col in range(col):

arr[Row][Col] = randint(min, max)

return arr

# Выводит количество локальных минимумов

def findLocalMin(arr):

rowLength = len(arr)

colLength = len(arr[0])

count = 0

for Row in range(rowLength):

for Col in range(colLength):

# Текущий элемент который нужно проверить

el = arr[Row][Col]

# Проверка первой строки

if (Row == 0):

if (Col == 0):

# Проверка элемента в левом верехнем углу

if (el < arr[Row][Col + 1])\

and (el < arr[Row + 1][Col + 1])\

and (el < arr[Row + 1][Col]):

count += 1

elif (Col == colLength - 1):

# Проверка элемента в правом верхнем углу

if (el < arr[Row][Col - 1])\

and (el < arr[Row + 1][Col - 1])\

and (el < arr[Row + 1][Col]):

count += 1

else:

# Проверка остальных элементов в первой строке

if (el < arr[Row][Col - 1])\

and (el < arr[Row + 1][Col - 1])\

and (el < arr[Row + 1][Col])\

and (el < arr[Row + 1][Col + 1])\

and (el < arr[Row][Col + 1]):

count += 1

# Проверка последней строки

if (Row == rowLength - 1):

if (Col == 0):

# Проверка элемента в левом нижнем углу

if (el < arr[Row - 1][Col])\

and (el < arr[Row - 1][Col + 1])\

and (el < arr[Row][Col + 1]):

count += 1

elif (Col == colLength - 1):

# Проверка элемента в правом верхнем углу

if (el < arr[Row - 1][Col])\

and (el < arr[Row - 1][Col - 1])\

and (el < arr[Row][Col - 1]):

count += 1

else:

# Проверка остальных элементов в поселедней строке

if (el < arr[Row][Col - 1])\

and (el < arr[Row - 1][Col - 1])\

and (el < arr[Row - 1][Col])\

and (el < arr[Row - 1][Col + 1])\

and (el < arr[Row][Col + 1]):

count += 1

# Проверка остальных строк (промежуточных)

if (Row != rowLength - 1) and (Row != 0):

# Проверка левого крайнего элемента

if (Col == 0):

if (el < arr[Row - 1][Col])\

and (el < arr[Row + 1][Col])\

and (el < arr[Row][Col + 1])\

and (el < arr[Row + 1][Col + 1])\

and (el < arr[Row - 1][Col + 1]):

count += 1

# Проверка правого крайнего элемента

elif (Col == colLength - 1):

if (el < arr[Row - 1][Col])\

and (el < arr[Row + 1][Col])\

and (el < arr[Row][Col - 1])\

and (el < arr[Row + 1][Col - 1])\

and (el < arr[Row - 1][Col - 1]):

count += 1

# Проверка любых не крайних элементов

else:

if (el < arr[Row - 1][Col - 1])\

and (el < arr[Row - 1][Col])\

and (el < arr[Row - 1][Col + 1])\

and (el < arr[Row + 1][Col - 1])\

and (el < arr[Row + 1][Col])\

and (el < arr[Row + 1][Col + 1])\

and (el < arr[Row][Col - 1])\

and (el < arr[Row][Col + 1]):

count += 1

return count

# Находит сумму элементов выше главной диагонали

def sumAbs(arr):

rowLength = len(arr)

colLength = len(arr[0])

sum = 0

startIndex = 1

for Row in range(rowLength - 1):

for Col in range(startIndex, colLength):

sum += abs(arr[Row][Col])

startIndex += 1

return sum

arr = makeRandomArr(10, 10, 0, 9)

fileOutput = open('output.txt', 'w', encoding="utf-8")

fileOutput.write('\_\_\_\_\_\_\_ Исходный массив \_\_\_\_\_\_\_\n')

for Row in range(len(arr)):

fileOutput.write('{0}\n'.format(arr[Row]))

fileOutput.write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n')

fileOutput.write('Количество локальных минимумов: {0}\n'.format(findLocalMin(arr)))

fileOutput.write('Сумма модулей элементов выше главной диагонали: {0}\n'.format(sumAbs(arr)))

**Результат тестирования программы (5 л.р., папка task5):**

**output.txt:**

\_\_\_\_\_\_\_ Исходный массив \_\_\_\_\_\_\_

[2, 5, 0, 3, 4, 0, 5, 1, 2, 1]

[7, 5, 4, 4, 1, 9, 5, 5, 1, 6]

[9, 0, 1, 4, 3, 1, 3, 8, 2, 7]

[1, 6, 5, 1, 4, 8, 1, 7, 2, 9]

[5, 8, 9, 1, 4, 3, 2, 5, 1, 5]

[5, 2, 3, 6, 5, 1, 6, 2, 6, 7]

[3, 8, 8, 6, 5, 2, 2, 9, 5, 2]

[1, 4, 1, 0, 5, 0, 8, 5, 3, 9]

[1, 1, 4, 9, 9, 0, 8, 3, 9, 9]

[8, 1, 4, 4, 8, 4, 9, 6, 7, 4]

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

Количество локальных минимумов: 10

Сумма модулей элементов выше главной диагонали: 189

**Использованная литература**

pythonworld.ru Справочник для начинающих. Python 3