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

## **Лабораторная работа №4**

Одномерные массивы

**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 обработки одномерных

массивов: поиск максимумов и минимумов, сортировка.

**Постановка задачи**

Сформировать одномерный список, состоящий из N вещественных чисел,

полученных генератором случайных чисел. Количество элементов списка (N)

запрашивается у пользователя, но не превышает 30. Диапазон значений элементов от -5.0

до 5.0. Вычислить:

1. Номер минимального по модулю элемента.

2. Сумму модулей элементов, расположенных после первого отрицательного

элемента. Сжать массив, удалив из него все элементы, величина которых находится

в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

**Теоретическая часть**

Массив - это структурированный набор данных одного типа.

В решении этого задания для хранения однотипных данных (массивов данных)

предлагается использовать список. Список представляет собой последовательность элементов, пронумерованных от 0. Элементы списка могут иметь различные типы. Список можно задать

перечислением элементов списка в квадратных скобках.

При обращении к элементам списка можно использовать как положительные

индексы, так и отрицательные. Если индекс положительный, то счет ведется от нуля до

максимального элемента, слева направо. Если индекс отрицательный, то счет ведется

справа налево.

Количество элементов в списке (длину списка), можно получить при помощи

функции len.

Существует несколько способов работы со списком. Разработчики предлагают различные

варианты от специальных модулей до библиотек. Стандартные решения языка для нашего примера достаточны.

Рассмотрим один из способов создания списка и его наполнения:

* Запросить размер списка у пользователя.
* Создать пустой список (не содержащий элементов, длины 0);
* Наполним список случайными числами, применяя метод append.

**Описание программы**

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

**Описание алгоритма**

1. Получаем валидное натуральное значение длины списка, не превосходящее 30. Передаём сообщение об ошибке в *sys.stderr* и завершаем выполнение программы, если что-то пошло не так.
2. Создаём список указанной длины, состоящий из случайных вещественных чисел из некоторого диапазона, при помощи списочного выражения.
3. Ищем индекс первого отрицательного элемента списка. Если отрицательных элементов нет, то на печать будет выведено сообщение об этом.
4. Если же отрицательный элемент в списке есть, то считаем сумму модулей элементов после него, не включая его.
5. Изменяем список, заменяя попавшие в неверный промежуток значения нулями, которые ставим в конец списка.
6. Печатаем: длину списка, значения списка, ограничив количество знаков после запятой единицей, номер минимального по модулю элемента, сумму модулей элементов после первого отрицательного элемента (или сообщение о том, что отрицательных элементов в списке нет), изменённый список.

**Описание входных и выходных данных**

Программа получает на вход натуральное число, не превосходящее 30, печатает

вещественные числа.

**Листинг программы**

Листинг приведён в Приложении 1.

Блок-схема приведена в Приложении 2.

**Результат работы программы**

>>> 5

Length: 5

Sample:

4.2, -0.6, 5.0, 0.6, -4.9

Number of minimal by abs element: 2

Sum of abs of values after the first negative: 10.5

Changed sample:

4.2, -0.6, 5.0, 0.6, 0.0

**Список используемой литературы**

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

**Приложение 1**

import random

import sys

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

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:

length = int(input("Enter the list length: "))

assert 0 < length <= 30

except ValueError:

print("Wrong input", file=sys.stderr)

return

except AssertionError:

print("List length is expected to be in (0; 30]",

file=sys.stderr)

return

sample = [

random.uniform(VALUE\_START, VALUE\_STOP)

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)

print(f"Length: {length}")

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

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

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

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

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

main()

**Приложение 2**

