Homework

COMPUTING SYSTEMS ARCHITECTURE

Жулин Артем Германович | БПИ204 |

25,10,2021

Вариант - 131 - (5, 10)

Условие задачи

Обобщенный артефакт, используемый в задании	Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив)	Общие для всех альтернатив функции
Квадратные матрицы с действительными числами	1. Обычный двумерный массив 2. Диагональная (на основе одномерного массива) 3. Нижняя треугольная матрица (одномерный массив с формулой пересчета)	Вычисление среднего арифметического (действительное число)

Описание структуры ВС

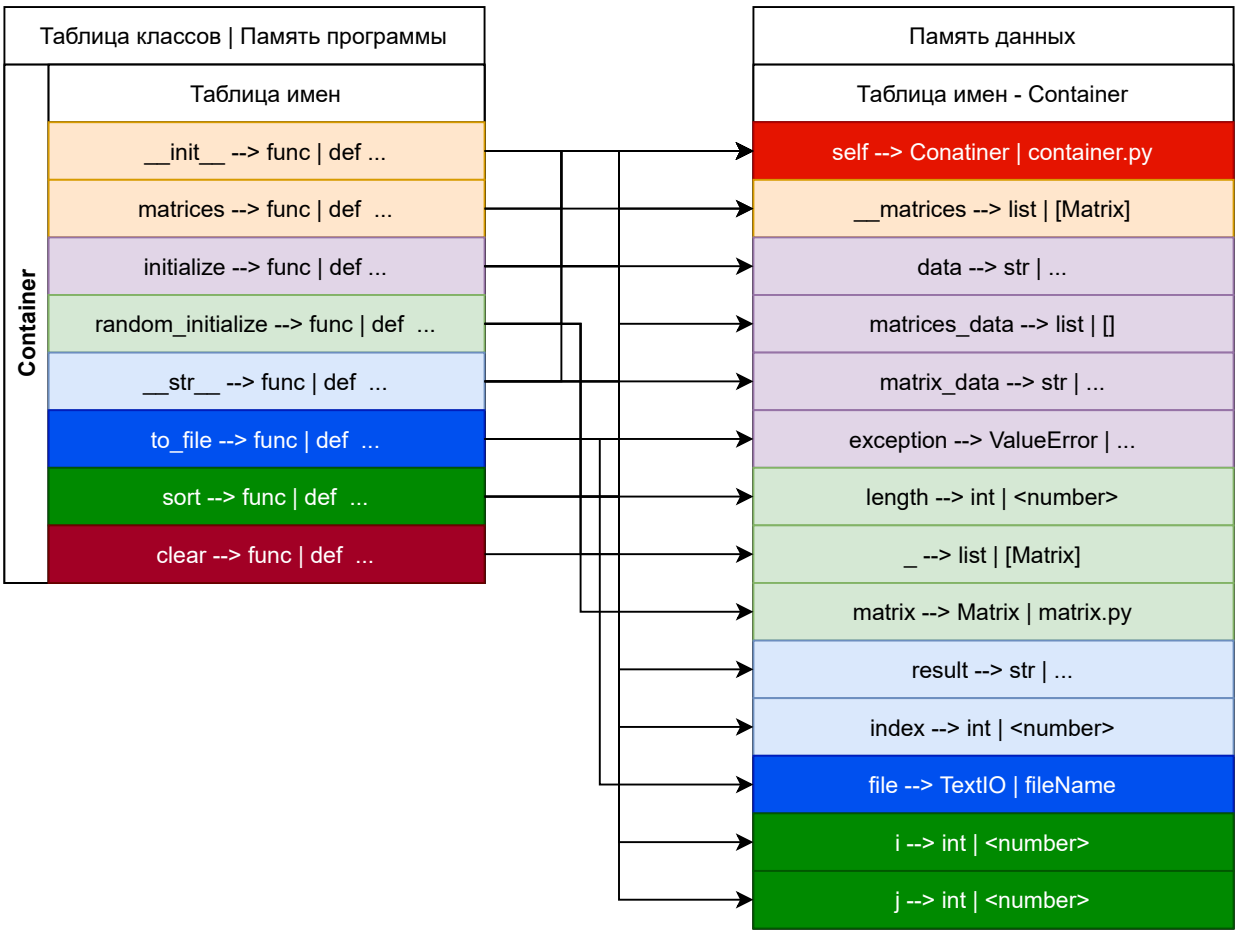
Таблица классов | Память программы
Таблица имен
Container
__init__ --> func | def ...
matrices --> func | def ...
initialize --> func | def ...
random_initialize --> func | def ...
__str__ --> func | def ...
to_file --> func | def ...
sort --> func | def ...
clear --> func | def ...
Память данных
Таблица имен - Container
self --> Conatiner | container.py
__matrices --> list | [Matrix]
data --> str | ...
matrices_data --> list | []
matrix_data --> str | ...
exception --> ValueError | ...
length --> int | <number>
_ --> list | [Matrix]
matrix --> Matrix | matrix.py
result --> str | ...
index --> int | <number>
file --> TextIO | fileName
i --> int | <number>
j --> int | <number>

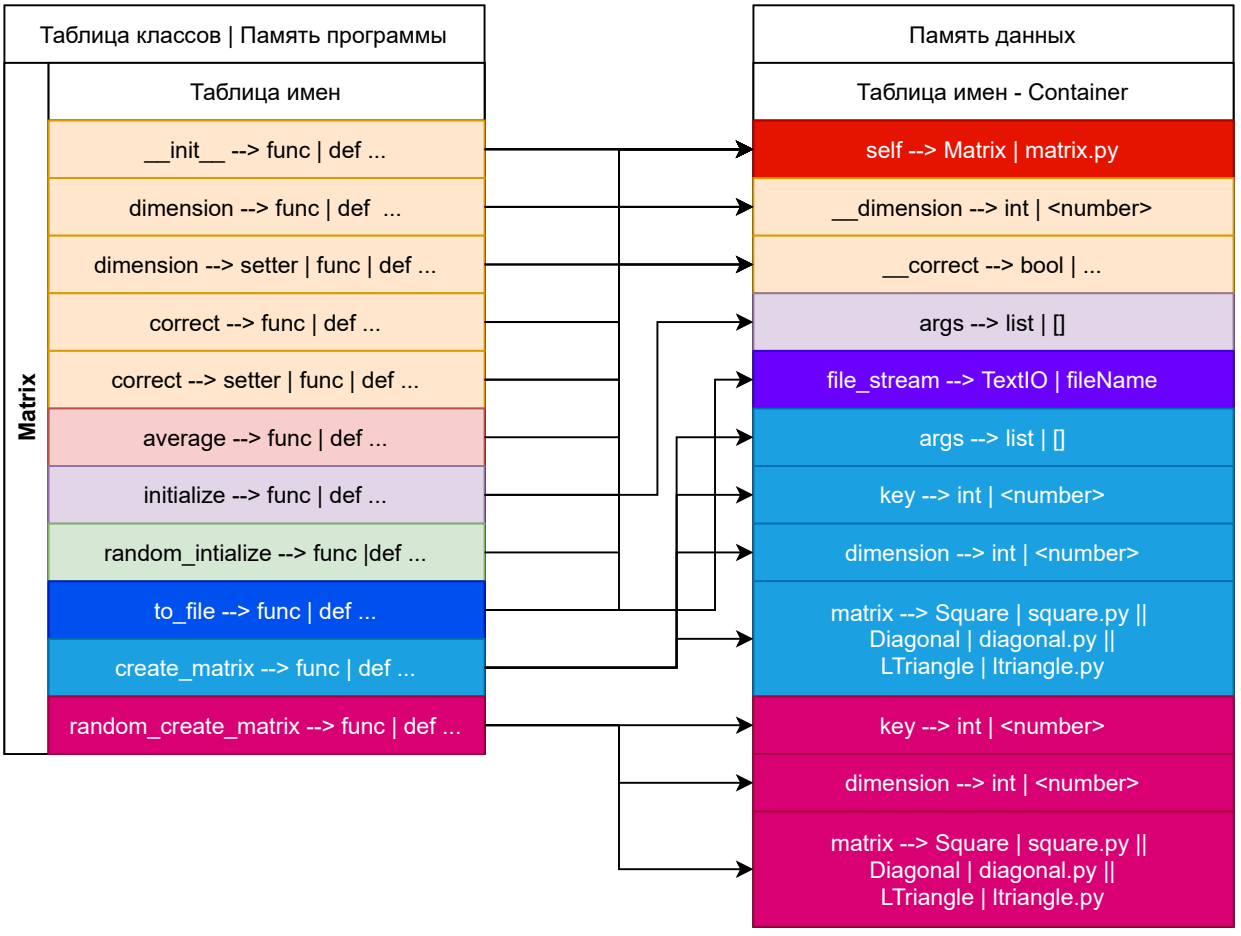
Таблица классов | Память программы
Таблица имен
Matrix
__init__ --> func | def ...
dimension --> func | def ...
dimension --> setter | func | def ...
correct --> func | def ...
correct --> setter | func | def ...
average --> func | def ...
initialize --> func | def ...
random_intialize --> func |def ...
to_file --> func | def ...
create_matrix --> func | def ...
random_create_matrix --> func | def ...
Память данных
Таблица имен - Container
self --> Matrix | matrix.py
__dimension --> int | <number>
__correct --> bool | ...
args --> list | []
file_stream --> TextIO | fileName
args --> list | []
key --> int | <number>
dimension --> int | <number>
matrix --> Square | square.py ||
Diagonal | diagonal.py ||
LTriangle | ltriangle.py
key --> int | <number>
dimension --> int | <number>
matrix --> Square | square.py ||
Diagonal | diagonal.py ||
LTriangle | ltriangle.py

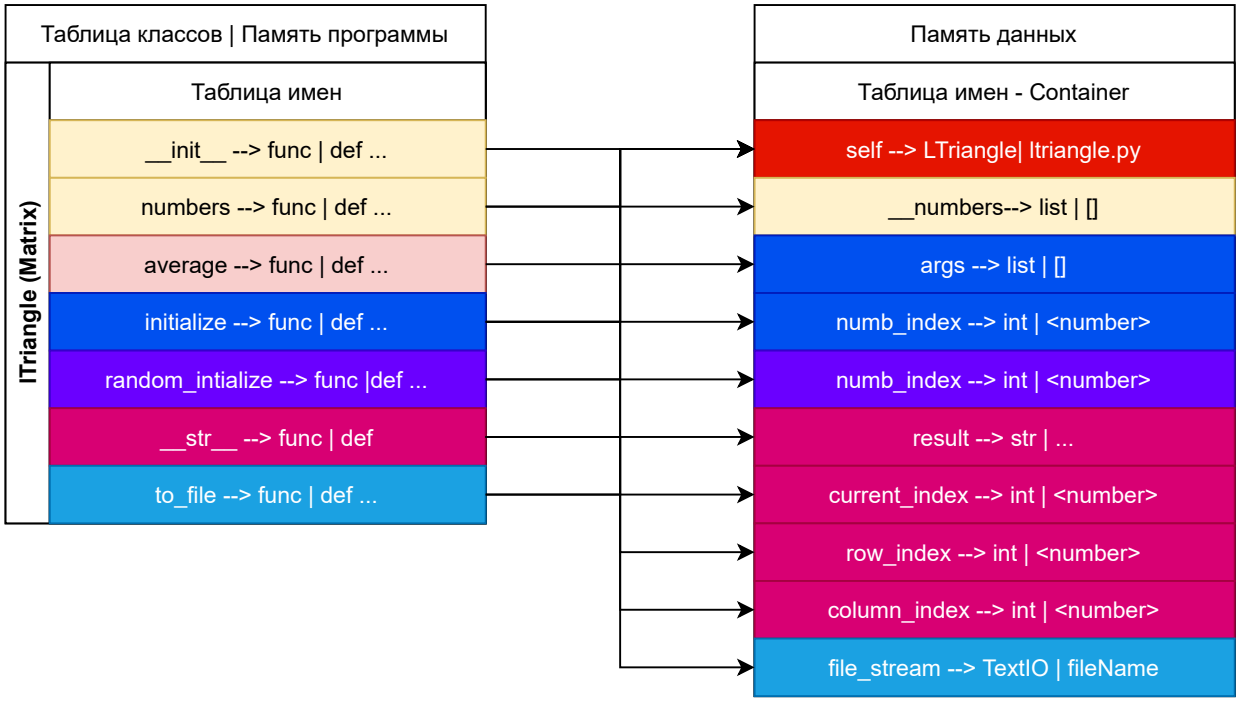
Таблица классов | Память программы
ITriangle (Matrix)
Таблица имен
__init__ --> func | def ...
numbers --> func | def ...
average --> func | def ...
initialize --> func | def ...
random_intialize --> func |def ...
__str__ --> func | def
to_file --> func | def ...
Память данных
Таблица имен - Container
self --> LTriangle| ltriangle.py
__numbers--> list | []
args --> list | []
numb_index --> int | <number>
numb_index --> int | <number>
result --> str | ...
current_index --> int | <number>
row_index --> int | <number>
column_index --> int | <number>
file_stream --> TextIO | fileName

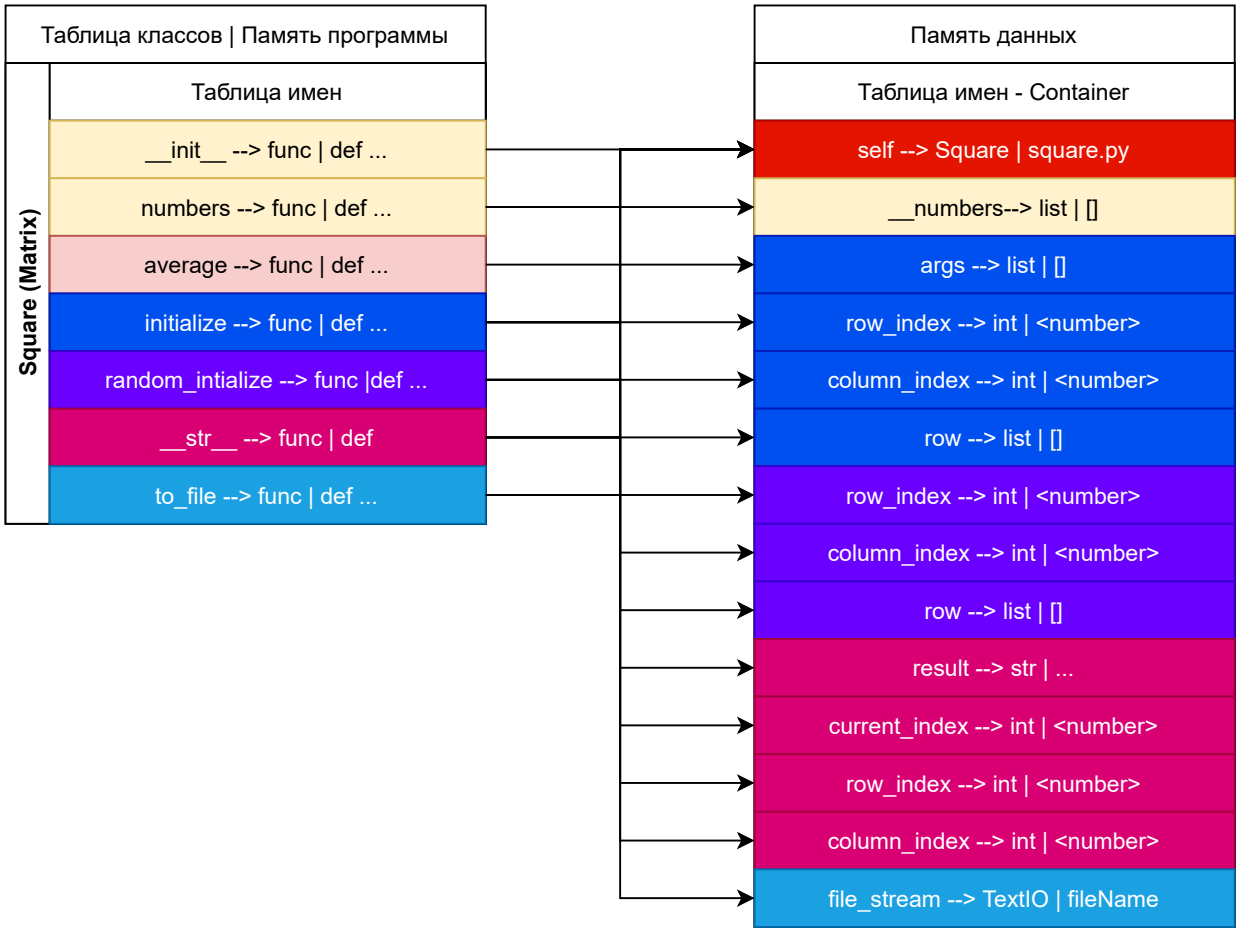
Таблица классов | Память программы
Square (Matrix)
Таблица имен
__init__ --> func | def ...
numbers --> func | def ...
average --> func | def ...
initialize --> func | def ...
random_intialize --> func |def ...
__str__ --> func | def
to_file --> func | def ...
Память данных
Таблица имен - Container
self --> Square | square.py
__numbers--> list | []
args --> list | []
row_index --> int | <number>
column_index --> int | <number>
row --> list | []
row_index --> int | <number>
column_index --> int | <number>
row --> list | []
result --> str | ...
current_index --> int | <number>
row_index --> int | <number>
column_index --> int | <number>
file_stream --> TextIO | fileName

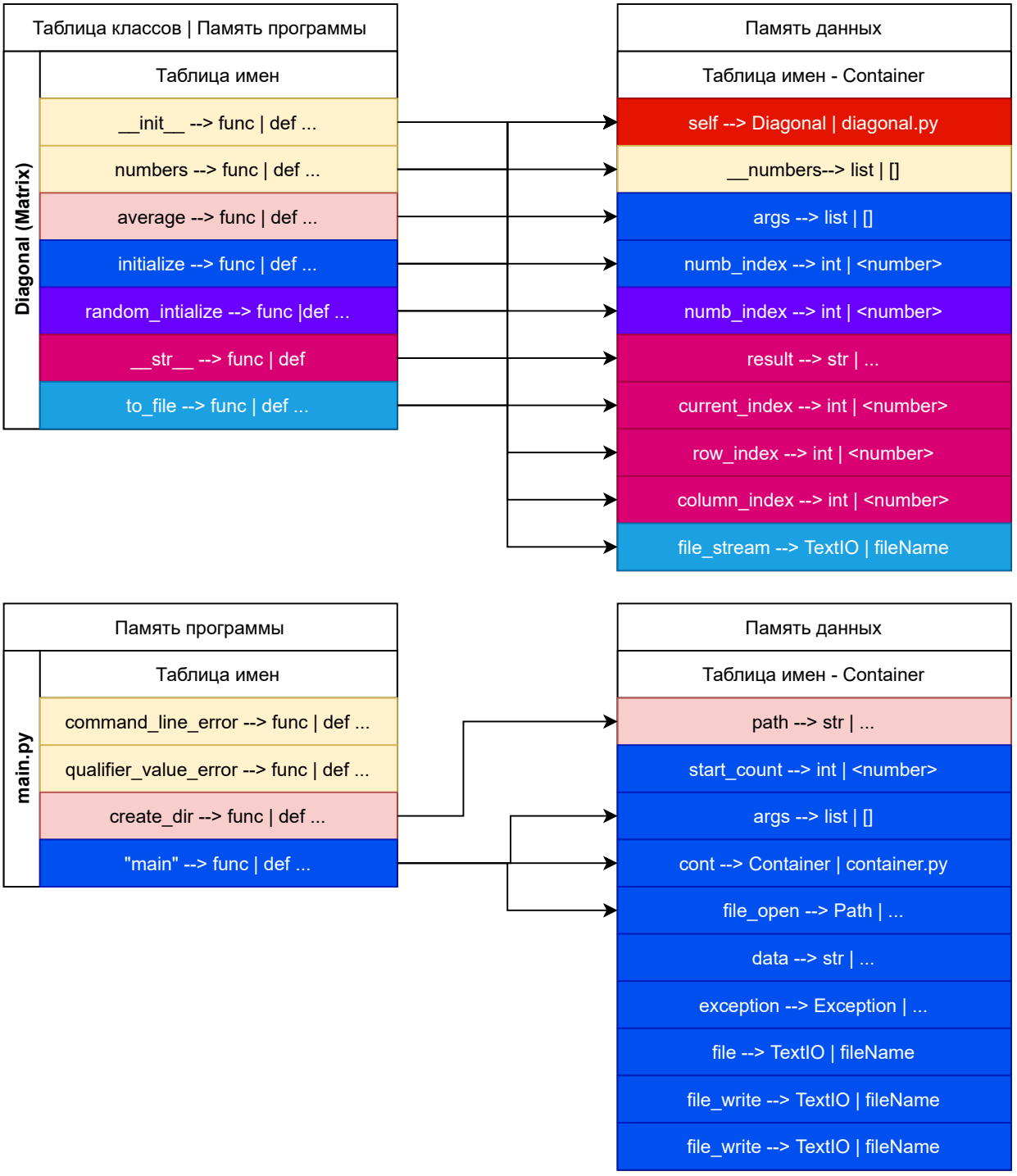
Таблица классов | Память программы
Diagonal (Matrix)
Таблица имен
__init__ --> func | def ...
numbers --> func | def ...
average --> func | def ...
initialize --> func | def ...
random_intialize --> func |def ...
__str__ --> func | def
to_file --> func | def ...
Память данных
Таблица имен - Container
self --> Diagonal | diagonal.py
__numbers--> list | []
args --> list | []
numb_index --> int | <number>
numb_index --> int | <number>
result --> str | ...
current_index --> int | <number>
row_index --> int | <number>
column_index --> int | <number>
file_stream --> TextIO | fileName
Память программы
main.py
Таблица имен
command_line_error --> func | def ...
qualifier_value_error --> func | def ...
create_dir --> func | def ...
"main" --> func | def ...
Память данных
Таблица имен - Container
path --> str | ...
start_count --> int | <number>
args --> list | []
cont --> Container | container.py
file_open --> Path | ...
data --> str | ...
exception --> Exception | ...
file --> TextIO | fileName
file_write --> TextIO | fileName
file_write --> TextIO | fileName

Входные и выходные данные

1. В консоль поступает команда следующего типа:
Ввод из файла: "python -f "input_file" "output_file_1" "output_file_2""
Случайный ввод: "python -n <number> "output_file_1" "output_file_2""

2. Образец входных данных:
<begin
1
2
1 2
3 4
end>
Где "begin" и "end" означают начало и конце информации о матрице. 1 (2 строка) - это тип матрицы, 2 (3 строка) - это размер матрицы, далее идут элементы самой матрицы

3. Матрица является некорректной в случае отсутствия хотя бы одна из команд "begin" или "end", некорректного типа матрицы (от 1 до 3), некорректной размерности матрицы (> 1), некорректных элементов матирцы. В случае, если матрица является некорректной программа переходит к следующим данным.

4. После обработки данных, программа выводит их в выходной файл, после чего сортирует и выводит данные во второй выходной файл. Пример данных:
<Container contains 1 elements:
1: It's diagonal matrix. Dimension = 3
1.0 0 0
0 2.0 0
0 0 3.0
Average: 0.6666666666666666

Time: 0.0012 s>
В выходном файле также указывается количество времени (в секундах), затраченное программой на обработку данных.

Краткий отчет

1. Количество заголовочных файлов: 0 шт

2. Количество модулей реализации: 6

3. Время, затраченное на каждый тест, указано в выходном файле соответствующего теста

Сравнение реализаций

C++ процедурный подход	C++ ООП подход	Python динамическая типизация, ООП подход
Время на обработку 10000 элементов после каждого этапа: Генерация: 0,884 сек Сортировка: 17,343 сек Удаление: 17,351 сек	Время на обработку 10000 элементов после каждого этапа: Генерация: 0,717 сек Сортировка: 16,283 сек Удаление: 16,284 сек	Время на обработку 10000 элементов после каждого этапа: Генерация: 4,9794 сек Сортировка: 156,2371 сек Удаление: 156,2541 сек
Отсутствие удобства в реализации, плохая читаемость кода, уступает времени исполнения ООП подходу	Простая реализация, высокая читабельность кода, возможность простой поддержки кода.	Самая простая реализация, однако сильно проигрывает во времени строго типизированным языкам.