Архитектура вычислительных систем.

Вариант 305

Задание 11, Функция 22

Проломов Виталий Алексеевич БПИ208

**Задача**

**Сущности**

1. Комплексные (действительная и мнимая части – пара действительных чисел)
2. Простые дроби (числитель, знаменатель – пара целых чисел)
3. Полярные координаты (угол [радиан] – действительное; координаты конечной точки на плоскости)

### **Общие для всех альтернатив функции:**

Приведение каждого значения к действительному числу, эквивалентному записанному

### Обобщенная структура - число

### Функционал

1. Провести отладку и тестирование разработанной программы на заранее подготовленных тестовых наборах данных. При необходимости, программа должна правильно обрабатывать переполнение по данным. Тестовые наборы до 10 элементов должны вводиться из заранее подготовленных тестовых файлов с использованием сведений из рассматриваемой в задаче предметной области. Количество тестовых наборов данных (тестов, сделанных вручную), в которых четко представлены параметры предметной области (корректные названия, имена, фразы) – не менее пяти. Число уникальных элементов в тестовых наборах должно варьироваться от нуля до 10000. Тестовые данные с большим числом элементов должны порождаться программно с использованием генераторов случайных наборов данных. Данные, формируемые генератом случайных наборов должны поддерживать допустимые для обработки значения. Управление вводом данных задается из командной строки.
2. Использование модульной структуры и классов для представления программных объектов дает дополнительные бонусы к базовой оценке.
3. Описать архитектуру виртуальной Python машины с наложением на нее обобщенной схемы разработанной программы.
4. Зафиксировать для отчета основные характеристики программы, такие как: число интерфейсных модулей (заголовочных файлов) и модулей реализации (фалов с определением программных объектов), общий размер исходных текстов, полученный размер исполняемого кода (если он формируется), время выполнения программы для различных тестовых наборов данных.
5. Привести результаты сравнительного анализа полученных характеристик с теми, которые были получены для предыдущей программы. Сделать выводы о достоинствах и недостатках этого и предшествующего решения относительно друг друга.

--1--

1) Модули реализации

complex\_number.py

rational\_number.py

random\_number\_generator.py

polar\_coordinates.h

container.py

number.py

main.py

build\_container.py

Всего: 8

1. Интерфесные модули:

Отсутствуют

3) общий размер исходных текстов = 2 + 1\*8= 10КБ

1. время выполнения программы для различных тестовых наборов данных: (тесты рассчитаны на значения в диапазоне от 1 до 20 вводимых эл-тов (чисел))

ТЕСТЫ ДО 20 ЭЛ-ТОВ:

1 тест:

0.0002 sec

2 тест:

0.0003 sec

3 тест:

0.0003 sec

4 тест:

0.0004 sec

5 тест:

0.0004 sec

6 тест:

0.0003 sec

7 тест :

0.0007 sec

ТЕСТЫ ОТ 20 ДО 10000 ЭЛ-ТОВ:

8 тест (50 эл-тов):

0.0061

9 тест (200 эл-тов):

0.0268 sec

10 тест (1000 эл-тов):

0.1349 sec

11 тест (9999 эл-тов):

0.725 sec

--2--.

Программа загружена в систему Github.

--3--.

Папка с тестами лежит рядом с exe файлом.

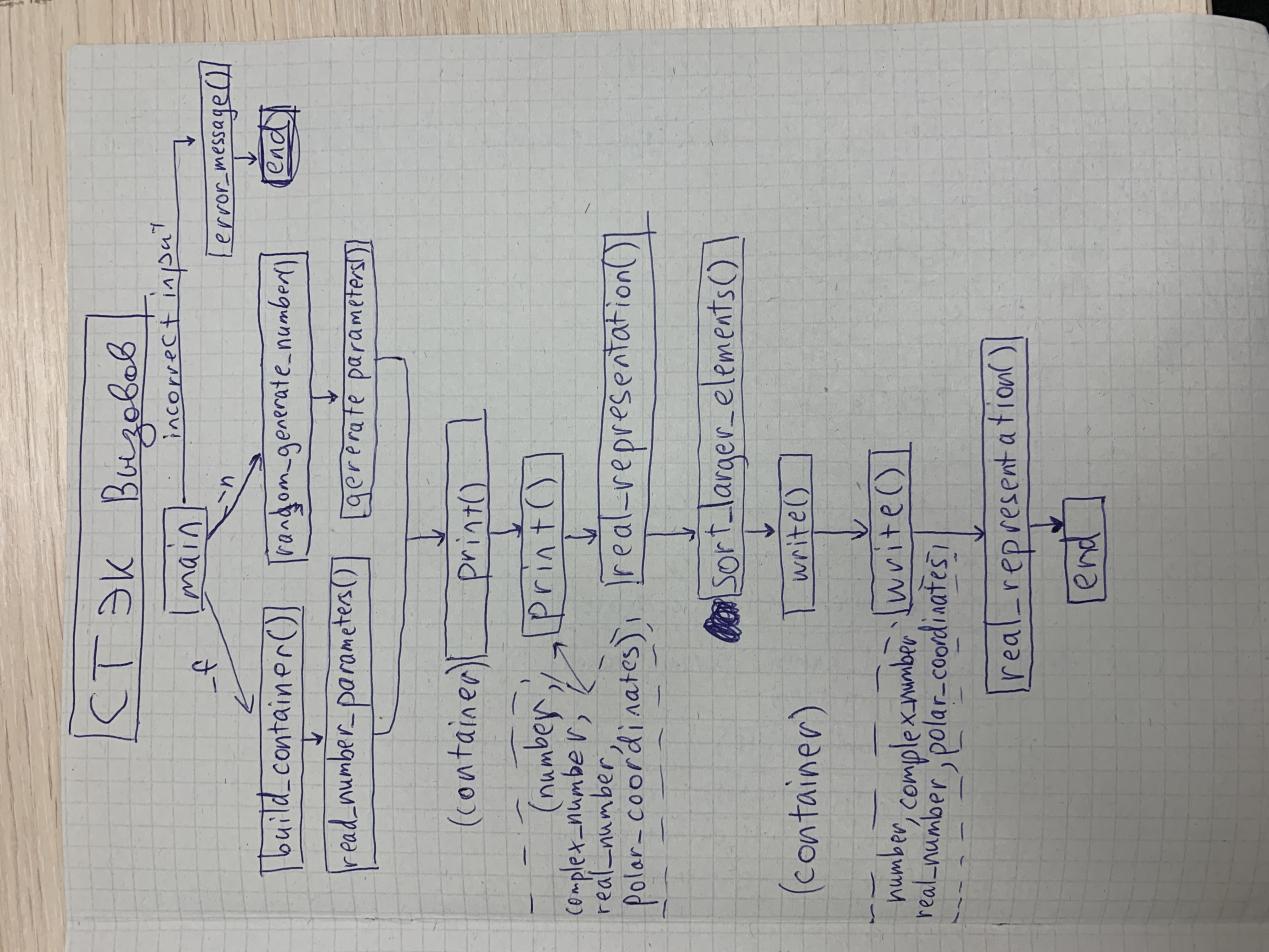
Вот путь:

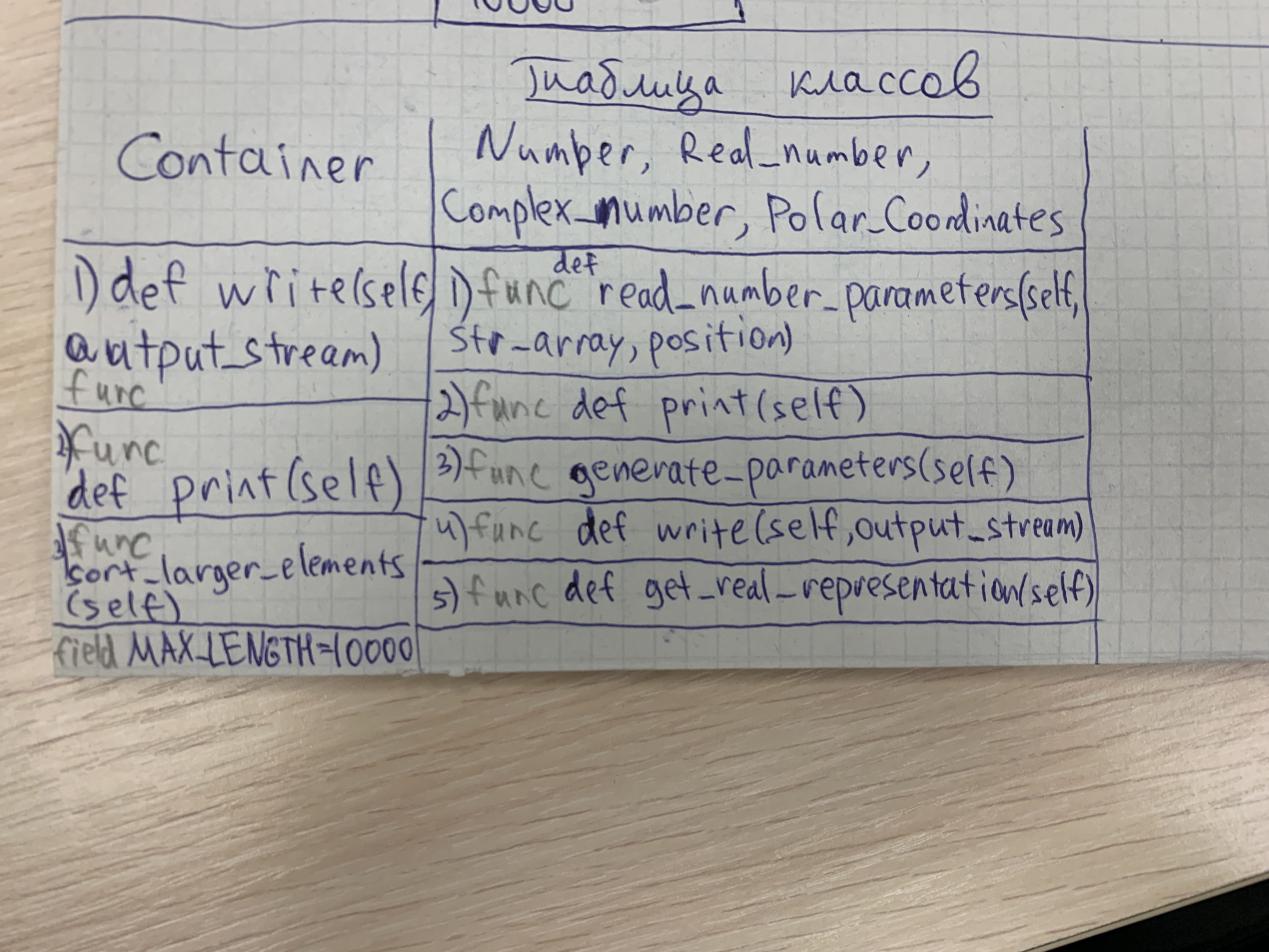
HW1/cmake-build-debug/tests

--4--.

1. Отображение на память методов классов (таблица для класса Container):

|  |  |  |
| --- | --- | --- |
| Память программы | Таблица имён | Память данных |
| 1. main.py | container | Container container |
|  | input\_stream | file file\_name |
|  | input\_string | file file\_name |
|  | Str\_array | List [...] |
|  | size | int <…> |
|  | input\_file\_name | string “…” |
|  | output\_file\_name1 | string “…” |
|  | output\_file\_name2 | string “…” |
|  | output\_stream1 | file file\_name |
|  | output\_stream2 | file file\_name |
|  |  |  |
| 2)Container.print | store | List [..] |
|  | new\_store | List[…] |
|  |  |  |
| 3)Container.sort\_larger\_elements | store | List [..] |
|  | all\_real\_representations | List […] |
|  | average\_real\_representation | double <…> |
|  | new\_store | list […] |
| ….… |  |  |

****

****

**Формат ввода:**

Для ввода одного числа следует ввести следующие данные:

X1

f11 f12

X1 - тип вводимого числа (1 - для комплексного числа, 2 - для рационального числа, 3 - для полярных координат)

В следующей строке вводятся параметры для указанного числа (для каждого типа числа ровно 2 параметра)

Для комплексных - реальную и мнимую часть

Для рациональных - числитель и знаменатель

Для полярных координат - угол в радианах и длину до начала координат

Расположение входных и выходных данных

Входные данные лежат в:

HW1/cmake-build-debug/tests/input

Выходные лежат в:

HW1/cmake-build-debug/tests/output

Различия с предыдущими подходами:

По времени работы с тестами видно, что программа с динамической типизацией и ООП работают сильно медленней, особенно на больших тестах (на маленьких тестах разницы почти нет), чем программа со статитеской типизацией. Объяснить это можно тем, что интерпретатором выделяется больше времени на определение всех типов в программе. Однако работа программы с динамической типизацинй и ООП отработала быстрее на небольших тестах ( с 1 по 9) - до 50 эл-тов.

В то же время, сравнивая размер исполняемого кода, можно отметить, что исходные тесты в предыдущих программах занимают больше памяти. Скорее всего это связано с тем, что при динамической типизации код становится более емким, занимая меньше места в памяти.