НИУ ВШЭ ФКН

Домашнее задание №3

Васин Егор Данилович, БПИ205

Описание задания:

Номер задания: 11. Номер обработки: 12

Базовые альтернативы:

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

Функция:

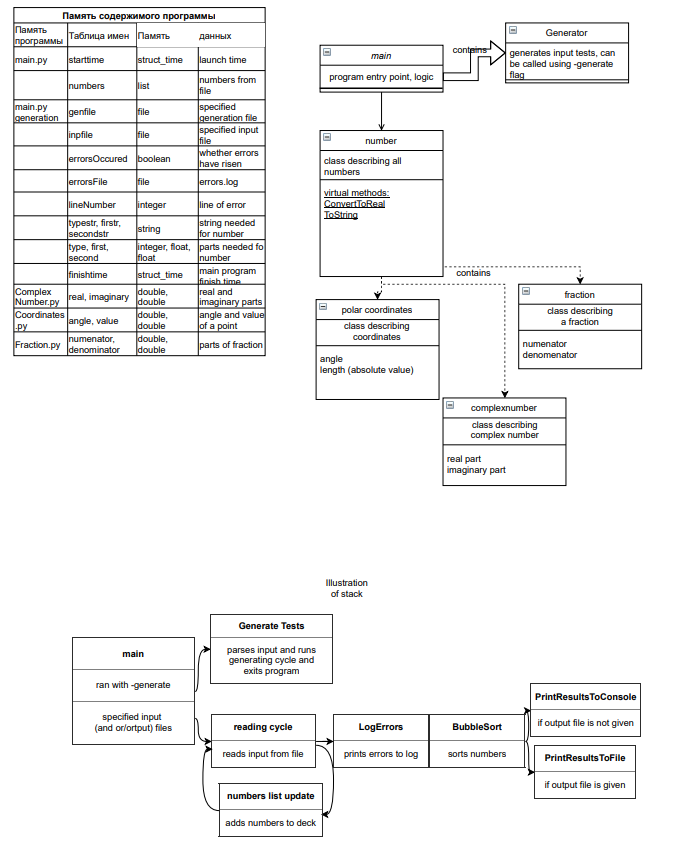
Приведение каждого значения к действительному числу, эквивалентному записанному. Например, для комплексного числа осуществляется по формуле: sqrt(d^2+i^2)), а для полярных координат - расстояние.

Обработка:

Упорядочить элементы контейнера по убыванию используя сортировку с помощью прямого обмена или пузырька (Bubble Sort). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Реализованный доп. Функционал: проверка деления на ноль и вывод возникающих ошибок в лог, генератор входных данных, запускаемый из командной строки простым добавлением флага, вывод измерений быстродействия программы.

Структурная схема:



Метрики программы:

Время на тестах (без времени на вывод результатов) для базового функционала (статический массив):

ManuallyGenerated.txt: 0.0009994506 s. (- 0.0002 s в сравнении с первой реализацией)

ManuallyGenerated2.txt: 0.0009984s. (+ 0.0007 s в сравнении с первой реализацией)

RandomlyGenerated1.txt 2.52011s. (+ 2.12 s в сравнении с первой реализацией)

RandomlyGenerated2. 0.0030210s. (- 0.0002 s в сравнении с первой реализацией)

RandomlyGenerated3.txt: 0.29422s. (+ 0.279 s в сравнении с первой реализацией)

RandomlyGenerated4\_LARGE.txt: 3.11436s.(+ 2.124 s в сравнении с первой реализацией)

Интерфейсных модулей 3, модулей реализации 1, общий размер – 4кб, размер исполняемого кода – 12кб

Исходный код находится в корневой папке репозитория, в файлах

main.py, ComplexNumber.py, Coordinates.py, Fraction.py

Логи ошибок находятся в корневой папке репозитория в файле errors.log

Тестовые данные находятся в директории tests в файлах

Анализ полученных результатов:

Хотя на больших объемах входных данных архитектура с динамической типизацией, ожидаемо, начинает стабильно работать гораздо медленнее, на небольших тестах (до 20 элементов), такое замедление не заметно. На них время работы программы может варьироваться в пределах 40% от среднего и сложно оценить точно, насколько та или иная архитектура опережает другие в производительности. Однако это не объясняет совпадающих доверительных интервалов времени работы программы на всех трех архитектурах. Это явление объясняется тем, что на небольших наборах данных наибольшая часть времени тратится не на сортировку и не на чтение из файла, а на его открытие, которое происходит примерно за одно время вне зависимости от архитектуры (проверено скриптом, многократно открывающим случайный из набора файлов)