Пояснительная записка

Описание полученного задания:

Номер варианта - 232.

1. Номер основного задания - 8.
    - Обобщенный артефакт, используемый в задании:
        Языки программирования.
    - Базовые альтернативы:
        1.1. Процедурные (наличие, отсутствие абстрактных типов данных – булевская величина)
        1.2. Объектно-ориентированные (наследование: одинарное, множественное, интерфейса – перечислимый тип)
        1.3. Функциональные языки (типизация – перечислимый тип = строгая, динамическая; поддержка «ленивых» вычислений – булевский тип)
    - Общие для всех альтернатив переменные:
        Популярность в процентах (TIOBI) — действительное
        Год создания – целое
    1. Общие для всех альтернатив функции:
        Частное от деления года создания на количество символов в названии

2. Номер доп. задания (обработка данных в контейнере) - 17.
    Упорядочить элементы контейнера по убыванию используя сортировку с помощью разделения (Quick Sort). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Для выполнения задания был использован PyCharm - интегрированная среда разработки для языка программирования Python.м

Структура программы:

Это программный продукт с использованием динамической проверки типов во время выполнения программы.

Всего имеется 7 модулей реализации:

1. language.py
2. procedural.py
3. objectOriented.py
4. functional.py
5. languageNew.py
6. container.py
7. main.py

Файл language.py содержит описание класса Language, обобщающего все имеющиеся языки программирования. От него наследуются классы Procedural (содержит описание процедурных языков), ObjectOriented (содержит описание Объектно-ориентированных языков) и Functional (содержит описание функциональных языков). Каждый из перечисленных классов записан в своем одноименном файле, и у каждого из перечисленных классов есть переменные, соответствующие характеристикам языка, описанным в задании, и методы, выполняющие следующие функции:

- Ввод параметров языка из файла
- Случайный ввод параметров языка
- Вывод параметров языка в файл
- Вычисление частного языка

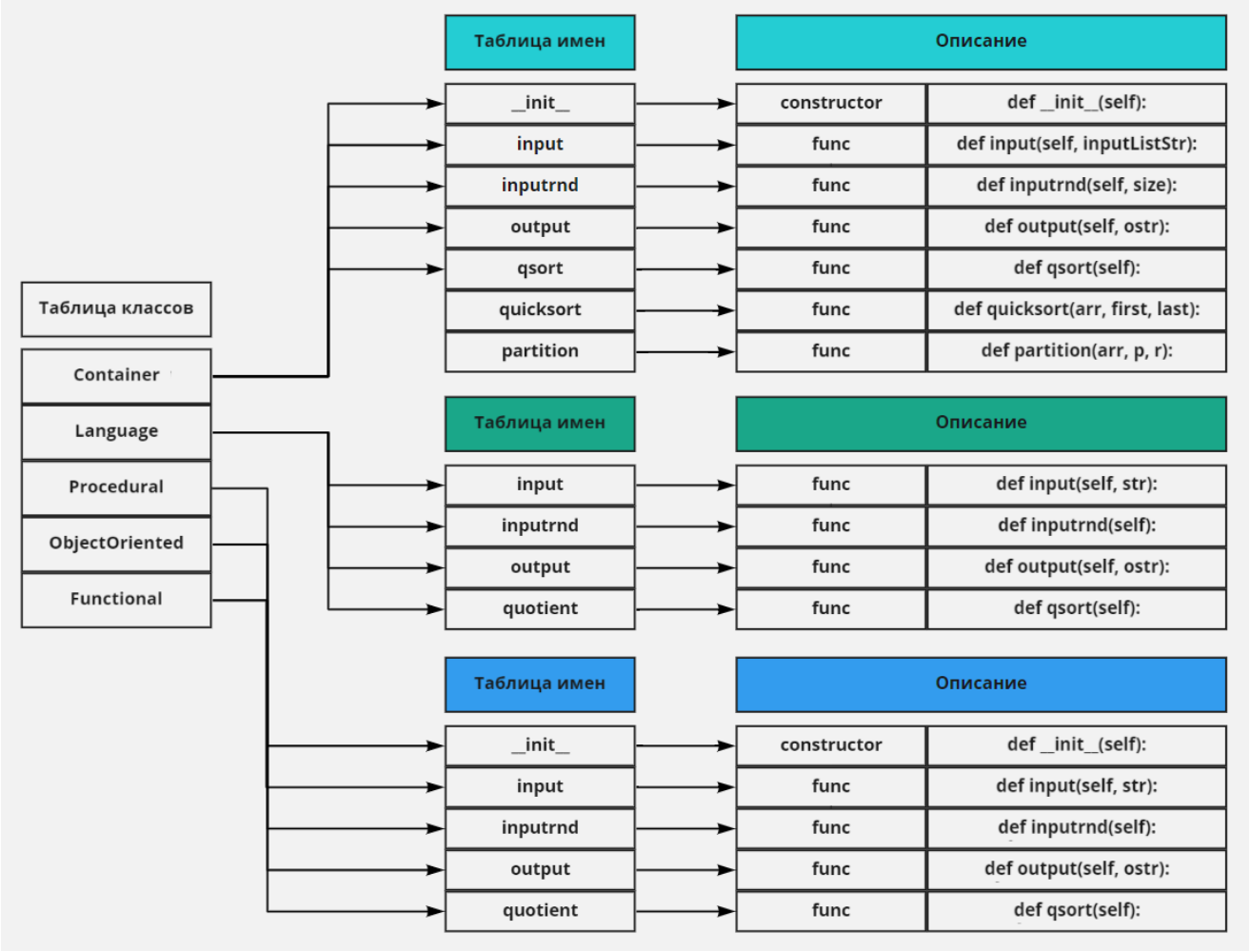
Также присутствует файл languageNew.py, в котором обрабатывается создание определенного типа языка программирования (Procedural/ObjectOriented/ Functional) в зависимости от информации из входного файла или же от случайной генерации рандомайзера. Созданный язык программирования возвращается функцией и добавляется в контейнер.

В файле container.py описан класс Container, в котором при инициализации создается пустой список. Container содержит функции, выполняющие следующие действия:

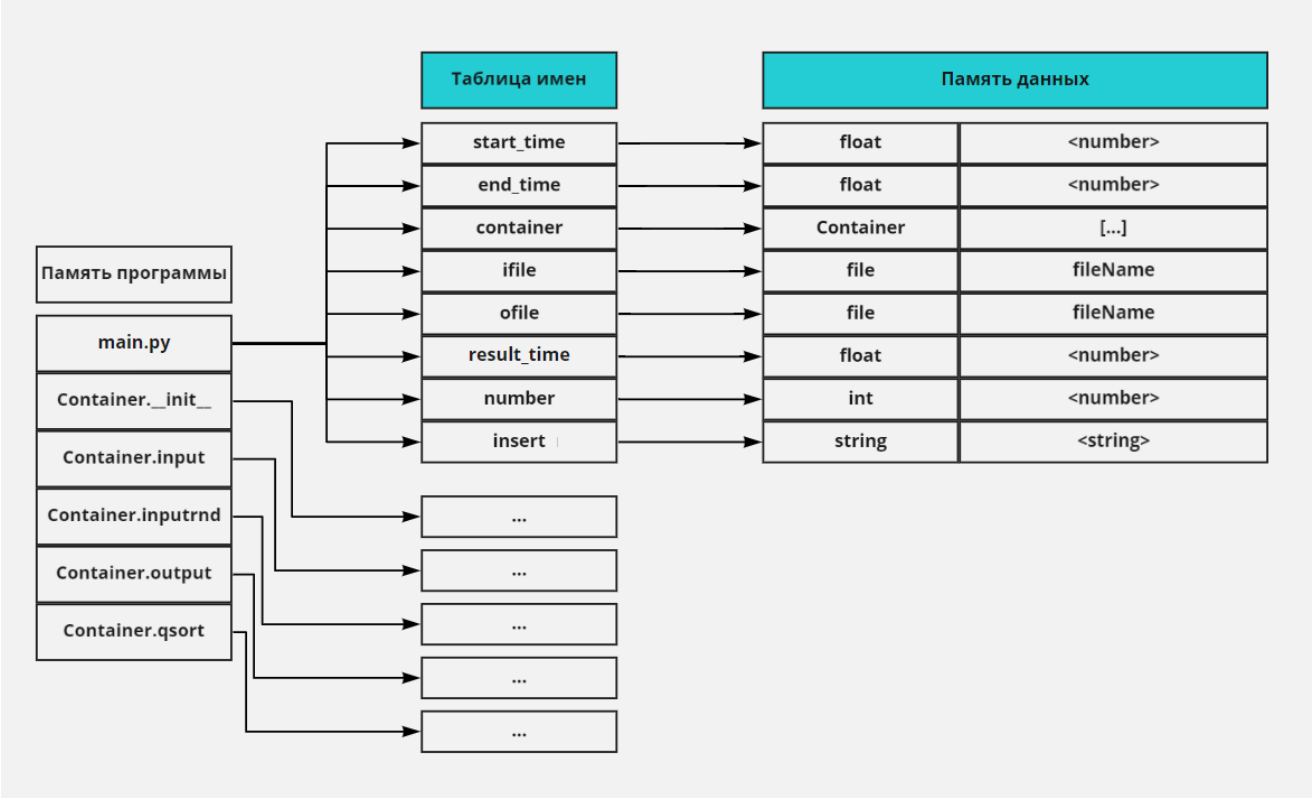
- Заполнение контейнера по информации из файла
- Заполнение контейнера случайными данными
- Вывод содержимого контейнера в файл
- Сортировка языков программирования по их частному

В коде присутствуют подробные комментарии.
Также программа успешно обрабатывает некорректный ввод и указывает, где именно была совершена ошибка при вводе.

Отображение содержимого классов



Отображение на память методов классов



Ссылка на онлайн доску, где были сделаны обе этих таблицы:
https://miro.com/welcomeonboard/ZFV6S2t4dHRRa2REcDU3dXdnUmZkSnA3NTY2VWNiMzdCUzZ1YmVmVncySkVVZDdyeVU3MlBJR054VjZTaFR0OXwzMDc0NDU3MzY2NDI3MjEyMzU0?invite_link_id=716588919363

Размеры программы:

Размер текста:

- container.py: 53 строки. Объем текста (Unicode UTF-8) = 1,89 КБ (1 941 байт)
- language.py: 21 строка. Объем текста (Unicode UTF-8) = 743 байт (743 байт)
- languageNew.py: 35 строк. Объем текста (Unicode UTF-8) = 854 байт (854 байт)
- procedural.py: 42 строки. Объем текста (Unicode UTF-8) = 1,74 КБ (1 785 байт)
- objectOriented.py: 44 строки. Объем текста (Unicode UTF-8) = 1,87 КБ (1 917 байт)
- functional.py: 49 строк. Объем текста (Unicode UTF-8) = 2,04 КБ (2 093 байт)
- main.py: 64 строки. Объем текста (Unicode UTF-8) = 1,91 КБ (1 959 байт)

Итого: 308 строк.
Объем текста (Unicode UTF-8) в байтах = 11 292
Объем текста (Unicode UTF-8) в КБ = 11,08

Вес файлов на диске:

- container.py: 4,00 КБ (4 096 байт)
- language.py: 4,00 КБ (4 096 байт)
- languageNew.py: 4,00 КБ (4 096 байт)
- procedural.py: 4,00 КБ (4 096 байт)
- objectOriented.py: 4,00 КБ (4 096 байт)
- functional.py: 4,00 КБ (4 096 байт)
- main.py: 4,00 КБ (4 096 байт)

Итого: 28 КБ (28 672 байт)

Тесты:

Всего имеется 2 рукописных теста (т.е. два In.txt файла) и 3 теста с использованием генератора случайных значений. Файлов с результатами всего 10 - по два на каждый тест. В файле out<x>_1 находится содержимое контейнера сразу после ввода данных. В файле out<x>_2 находится содержимое уже отсортированного контейнера.

Правила ввода рукописных тестов:

(Процедурные языки)

1 <name(строка)> <abstract_types (есть или нет (1 или 0))> <popularity(вещественное число)> <year(целое число)>

(Объектно-Ориентированные языки)

2 <name> <key_of_inheritance (наследовение: одинарное-1, множественное-2, интерфейсы - 3)> <popularity(вещественное число)> <year(целое число)>

(Функциональные языки)

3 name(строка)> <typing(типизация: строгая-1, динамическая-2)> <lazy_calculations(поддержка "ленивых" вычислений: есть-1, нет- 0)> <popularity(вещественное число)> <year(целое число)>

Время выполнения программы при разных тестах:

- Ввод из файла 9 элементов – примерно 1 миллисекунда.
- Ввод из файла 12 элементов – примерно 1 миллисекунда.
- Заполнение случайными элементами (50шт.) – примерно 2 миллисекунды.
- Заполнение случайными элементами (2000 шт.) – примерно 53 миллисекунды
- Заполнение случайными элементами (5000 шт.) – примерно 113 миллисекунд
- Заполнение случайными элементами (7500 шт.) – примерно 242 миллисекунды
- Заполнение случайными элементами (10000 шт.) – примерно 302 миллисекунды

Сравнительный анализ с предыдущими версиями программы:

У новой версии программы есть ряд весомых преимуществ, по сравнению с предыдущими версиями. Во-первых, при полном сохранении функциональности программы исходный код новой версии занимает более чем в два раза меньше

места по сравнению с обеими предыдущими версиями. Это было достигнуто в том числе за счет упрощения модульной структуры программы. Во-вторых, очень сильно уменьшилось время работы программы. Ввод из файлов стал на порядок быстрее, а заполнение контейнера случайными значениями стали быстрее в 1,5-2 раза. Это очень хороший результат.