Отчёт по программе, разработанной на Python3

Структура проекта

```
project/
       |
       +-container/       # файлы, описывающие структуру container
       +-objects/         # файлы, описывающие все объекты(структуры, описывающие
числа)
       +-docs/            # документация
       +-tests/           # файлы с тестовыми входными данными.
       +-utils/           # заголовочный файл с общедоступными статическими функциями
       +-load-testing.sh  # шелл-скрипт для тестирования времени работы проекта.
       +-main.cpp         # точка входа в приложение
```

Спецификация

Спецификация ВС

- **Operating System**: Arch Linux
- **Kernel**: Linux 5.14.7-arch1-1
- **Architecture**: x86-64
- **RAM**: 16Gb

Спецификация средств разработки

- **IDE**: PyCharm(v2021.2.2)
- **Библиотеки**:
    - abc
    - typing
    - enum
    - math
    - random
    - time
    - sys
- **Средство интерпретации**: Python interpreter 3.10

Дополнительный флаг `--random-input`

Была реализована функция сохранения сгенерированных входных данных в файл для дальнейшей отладки, для того, чтобы воспользоваться данной функцией необходимо указать флаг `--random-input` и предоставить название файла.

Пример: `python main.py -r 100 --random-input generated_input.txt -o output.txt` - при данном вводе контейнер заполнится 100 случайно сгенерированными объектами и этот ввод запишится в файл `generated_input.txt`, а вывод программы - в файл `output.txt`. Это позволяет быстро генерировать входные тесты и сразу записывать и входные данные и выходные в нужные файлы.

Характеристики проекта

- Количество программых объектов: 11
- Размер исходных файлов: ~ 20 Kb
- Время выполнения программы для различных входных данных

Флаг (-r)	Время выполнения(sec)
10	0.000300
100	0.002100
1000	0.141000
10000	0.940000

Расчет времени выполнения программы

Для расчета времени работы берется среднее арифметическое от 10 запусков программы. Шел скрипт для проверки расположен в корне проекта в файле `load-testing.sh`

Сравнение с предыдущей программой

Различия во времени работы программы для 1 и 2 проектов

Флаг (-r)	Время выполнения(sec) для 1 проекта	Время выполнения(sec) для 2 проекта	Время выполнения(sec) для 3 проекта
10	0.000208	0.000218	0.000300
100	0.001341	0.001552	0.002100
1000	0.011584	0.013051	0.141000
10000	0.537891	0.524102	0.940000

Программа, написанная на Python работает заметно медленнее аналогов на языке C(C++). Это может быть объяснено архитектурой языка Python. Т.к. Python - это интерпретируемый язык, то не существует этапа компиляции, когда компилятор может оптимизировать работу программы и заранее просчитать некоторые значения. Также, Python является динамически типизируемым языком, что усложняет программу и добавляет время на определение типов в run-time. С другой стороны, Python очень прост в разработке и удобен для программистов. Python позволяет совершать привычные операции по типу поиска элемента в списке(массиве) за 1 строчку(см. метод `main::is_flag_valid`). Также, язык Python позволяет обращаться к любому объекту во время исполнения - это позволяет гибко настраивать работу программ. Например, можно передавать функции как аргументы в другие функции не тратя время на создание указателя на функцию или делегата. За многие удобства языка Python приходится платить скоростью работы, как можно увидеть в сравнительной таблице. Ко всему прочему, динамическая типизация может усложнять тестирование и дебаг программ, когда они разрастаются и становится очень сложно находить места в коде, где происходит ошибка.

В текущей реализации класс `number` является абстрактным и не реализует никаких общих методов, но если бы класс `number` содержал общие параметры для всех чисел, то `number` имел бы виртульные методы, которые переопределялись бы в наследниках.

Структура текущего архитектурного решения