**Бот ребилдер**

Бот ребилдер(далее БР) - алгоритм, который умеет исправлять “примитивный код” уменьшая объем занимаемой памяти, уменьшая количество строк, исправлять синтаксические ошибки, изменять названия массивов данных(переменных, списков и др.), делать код читаемый(если слиплись несколько строк кода, или весь код программы в одной строке), если это необходимо или возможно.

Примитивный код - это код, который состоит из простых синтаксических конструкций языка программирования(в данном случае пока что один язык Python)

Пример простых конструкций:

1. переменные и их типы
2. списки, словари, кортежи, матрицы(двумерные массивы)
3. циклы, ветвления
4. функции
5. классы(только стандартные!!! методы и переменные, никаких паттернов,, которые специфически работают)

Проще говоря данный БР похож на задание курса, которое мы делали: “ребилд кода telegram”

Пример работы:

Предположим у нас есть некий код, состоящий как из одного файла, так и возможно из нескольких(все библиотеки с которыми работает бот, должны находится в каталоге проекта и быть представлены в текстовом формате, никаких символов из левых кодировок, шифрования и прочего)

После считывания и загрузки(указываем корневую директорию в которой лежит проект=код), бот выполняет ряд функций, изучая и преобразовывая код. После выполнения программы возможны два варианта действия:

1. Бот выдает ребилд кода, который он сделал с записью всех правок внутри кода
2. визуально отражает все возможные правки, которые можно применить к проекту, после чего, программист выбирает необходимые правки по своему желанию, после чего происходит сохранение копии проекта

К сожалению с frontend’ом мне работать доводилось не так часто, так что первый пункт вывода наиболее реален. С docker я тоже не работал, так что frontend будет решаться в процессе написания кода проекта.

Если останется время и будет возможность что-то дописать, будут реализованы дополнительный функционал:

1. Сделан более приятный глазу интерфейс, который убирает проблему пролистывания 100+ строк скролл баром и переходами между разными файлами проекта в поисках единственной переменной, функции
2. test code - тестирование кода на основе входных данных и полученного результата работы кода

а) общее тестирование кода методом черного ящика

б)тест по семантическому заданию:

**Например:**

**Задание:**"Создай массив натуральных чисел кратных двум"

<какой-то код, написанный программистом выше> - **файлы проекта, код**

Алгоритм будет распознавать задание и тестировать код в зависимости от него. Конечно задания будут примитивными, т.к. для более сложного анализатора требуется больше месяца времени.

1. Добавлены новые языки программирования(возможно в будущем можно будет писать одну программу на разных языках)
2. интерпретатор, транслятор
3. новые синтаксические замены(синтаксический сахар) custom-language

как пример в языке с++ используются {} - для отображения кода, который используется в цикле или ветвлении, а в python - табуляция со сдвигом(абзац).

В custom-language программист сможет выбрать, какой тип выделения под-кода ему нужен, просто поставив галочку или сделав свои обозначения

1. написан компилятор для всех языков, которые будут добавлены к распознаванию
2. и другие