Групове завдання 16

Треба розробити (доробити) модулі tokenizer, syntax_analyzer_ext, code_generator згідно специфікації.

Це завдання продовжує завдання 15.

Треба

1. У модулі tokenizer забезпечити повернення токену для символу ‘=’ (тип “equal”) та перевірити правильність виконання.
2. Реалізувати модуль syntax_analyzer_ext, де описати клас SyntaxAnalyzerExt як нащадок SyntaxAnalyzer, у ньому описати метод check_assignment_syntax для перевірки синтаксису присвоєння.
3. Модуль storage та клас Storage не змінюються.
4. Реалізувати модуль , code_generator та клас CodeGenerator для генерації низькорівневого програмного коду для обчислення виразів та реалізації присвоєнь.

Синтаксична діаграма виразу, близька до нашої задачі, зображена на рисунку нижче. Зокрема, у нас по-іншому позначаються змінні та константи (що не впливає на побудову коду). Все інше – відповідає.

Код програми розраховано на виконання з використанням стеку. Стек – це список, до якого ми можемо додавати елементи у кінець та забирати елементи з кінця. У стеку будуть зберігатись числа: константи та значення змінних, які вказано у виразі. Тому ці числа потрібно завантажити (додати) до стеку.

Для завантаження існують команди

```
("LOADC", <число>) - завантажити число у стек
("LOADV", <змінна>) - завантажити значення змінної у стек
                      (використовується storage)
```

Арифметичні дії будуть виконуватись командами

```
("ADD", None) - обчислити суму двох верхніх елементів стеку
("SUB", None) - обчислити різницю двох верхніх елементів стеку
("MUL", None) - обчислити добуток двох верхніх елементів стеку
("DIV", None) - обчислити частку від ділення двох верхніх елементів
стеку
```

Результат кожної арифметичної дії має завантажуватись у стек.

Присвоєння буде виконуватись командою

```
("SET", <змінна>) - встановити (присвоїти) значення змінної
                    у пам'яті (storage) рівним
                    значенню останнього елементу стеку
```

При присвоєнні елемент забирається зі стеку

Специфікація для пришвидшення розробки надається у вигляді «кістяка» модулів у файлах .py у окремому архіві.

Після розробки кожного модуля його треба запустити окремо та досягти виведення значення

Success = True

```
Основну частину кожного модуля після
if __name__ == "__main__":
```

не змінювати (можна вставляти print для налагодження)

Усі функції модуля мають бути реалізовані

Заголовки функцій не змінювати

Можна додавати власні внутрішні функції у модулі, якщо потрбіно.

Після завершення розробки модулів запустити модуль main та впевнитись, що виводиться

Success = True

Для неповних команд (з 2 або одного студента) достатньо реалізувати частину модулів по порядку (syntax_analyzer_ext, code_generator)