Lab 3

Wprowadzenie

Zadanie polega na stworzeniu i wypisaniu abstrakcyjnego drzewa składni (ang. abstract syntax tree, AST). Drzewo składni powinno uwzględniać w swoich węzłach następujące konstrukcje:

- wyrażenia binarne,
- wyrażenia relacyjne,
- instrukcje przypisania,
- instrukcje warunkowe `if-else`,
- pętle: `while` oraz `for`,
- instrukcje `break`, `continue` oraz `return`,
- instrukcje `print`,
- instrukcje złożone,
- tablice oraz ich zakresy.

Przykładowo, dla poniższego kodu:

```
A = zeros(5); # create 5x5 matrix filled with zeros
D = A.+B' ;   # add element-wise A with transpose of B

for j = 1:10
    print j;
```

translator powinien stworzyć odpowiadające mu drzewo składni (AST) oraz wypisać jego tekstową reprezentację na standardowym wyjściu:

```
=
|  A
|  zeros
|  |  5
=
|  D
|  .+
|  |  A
|  |  TRANSPOSE
|  |  |  B
FOR
|  j
|  RANGE
|  |  1
|  |  10
|  PRINT
|  |  j
```

Przykładowe pliki wejściowe: example1.m, example2.m, example3.m
oraz odpowiadające wyjściowe drzewa składni: example1.tree, example2.tree, example3.tree

- Do rozwiązania zadania można użyć generatora parserów `PLY`.
- Rozpoznawany język powinien być zgodny ze stworzoną gramatyką. Obecność białych znaków, sposób sformatowania tekstu nie ma wpływu na postać drzewa.
- Drzewo abstrakcyjne składni (AST) powinno być wypisywane tylko dla syntaktycznie poprawnego wejścia.
- Translator powinien wykrywać niepoprawne syntaktycznie wejście. W takim przypadku lepiej nie tworzyć drzewa syntaktycznego, ale dla niepoprawnych linii wypisywać numer linii wraz z informacją że wystąpił błąd.
- Do stworzenia translatora, można użyć poprzednio stworzonych plików scanner.py, Mparser.py. Nowe pliki, które zostaną stworzone, to AST.py, TreePrinter.py oraz main.py..
- W trakcie parsingu należy stworzyć drzewo syntaktyczne. Różnica między (konkretnym) drzewem parsingu (ang. *parse tree*) a (abstrakcyjnym) drzewem syntaktycznym (ang. *syntax tree*) opisana jest tutaj.
- W zależności od rodzaju nieterminala należy zdefiniować specyficzne dla nich klasy lub struktury danych (stała, zmienna, przypisanie, wyrażenie arytmetyczne, wyrażenie porównania, instrukcja warunkowa, lista instrukcji, itp.).
- Dla każdej klasy należy zdefiniować funkcję `printTree` wypisującą odpowiadającą danemu węzłowi część drzewa syntaktycznego.
- Wszystkie funkcje `printTree` należy zdefiniować odpowiednio w klasie `TreePrinter`. Funkcje te zostaną zainstalowane w klasach odpowiadającym węzłom drzewa abstrakcyjnego przy pomocy dekoratora `@addToClass`.

`@addToClass`. Dekoratory opisane są szczegółowo w *M. Lutz, Learning Python, 4th Ed., Chapt. 38*.