Analisador Léxico - *Python PLY*

Brendon Vicente Rocha Silva
Graduando em Ciências da Computação
Universidade Federal de Santa Catarina - UFSC
Florianópolis, SC, Brasil
<brendon.vicente@grad.ufsc.br>

Para este trabalho foi necessário o desenvolvimento de um analisador léxico para a linguagem *LCC-2022-2*, gerada pela gramática *CC-2022-2*.

O projeto foi desenvolvido em linguagem de programação *Python 3.10*, com auxílio da ferramenta *PLY*.

TOKENS UTILIZADOS

Um *token* é um segmento de texto ou símbolo que pode ser manipulado por um analisador sintático, que fornece um significado ao texto.

Dentro da linguagem *LCC-2022-2*, foram utilizados os seguintes:

PALAVRAS RESERVADAS

def	*int*	*float*	*string*	*break*
print	*read*	*return*	*if*	*else*
for	*new*	*null*		

***TOKENS* LITERAIS**

{ } ( ) [ ] ; , < > = + - * / %

***TOKENS* NÃO TRIVIAIS**

LESS_EQUAL_THAN	<=
GREATER_EQUAL_THAN	>=
EQUAL	==
DIFFERENT	*!=*
INT_CONSTANT	*[0-9]+(E[\+-]?[0-9]+)?*
FLOAT_CONSTANT	*[0-9]+\.[0-9]+(E[\+-]?[0-9]+(\.[0-9]+)?)?*
STRING_CONSTANT	*("."*"|\'."*\')*
ID	*[a-zA-Z_][a-zA-Z_0-9]*

Para especificação e identificação dos *tokens* foram utilizadas expressões regulares (*RegEx*). Abaixo seguem os *tokens* especificados e suas respectivas expressões, assim como as regras definidas por elas:

`INT_CONSTANT = r'[0-9]+(E[\+-]?[0-9]+)?'`

Strings formadas por um ou mais algarismos, podendo conter indicador de exponencial (caractere *E*).

Exemplo: 1234E+123

`FLOAT_CONSTANT = r'[0-9]+\.[0-9]+(E[\+-]?[0-9]+(\.[0-9]+)?)?'`

Strings formadas por um ou mais algarismos, com separador decimal (caractere .), podendo conter indicador de exponencial (caractere *E*).

Exemplo: 1.234E+1.23

`STRING_CONSTANT = r('".*"|'.*')`

Strings quaisquer, cercadas por aspas ou aspas duplas (incluindo *strings* vazias).

Exemplo: 'teste'

`ID = r'[a-zA-Z_][a-zA-Z_0-9]*'`

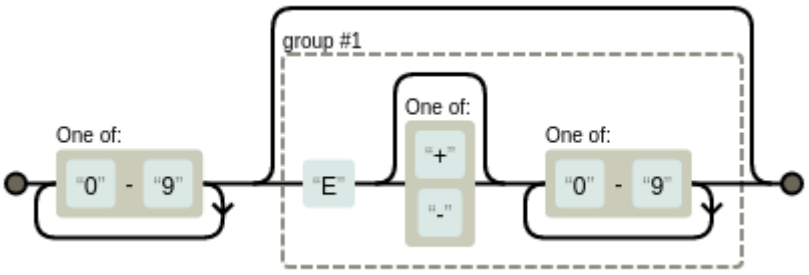
Strings iniciadas por uma letra (de A à Z, ignorando capitalização) ou *underline* (caractere _), seguido por *N* letras, números ou *underlines*.

Exemplo: multiplicar_matrizes

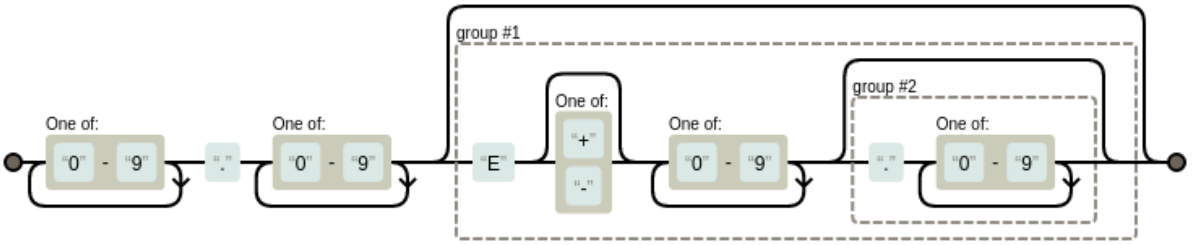
DIAGRAMAS DE TRANSIÇÃO

Segue, abaixo, os diagramas de transição utilizados nos *tokens* não triviais, cujo *RegEx* é complexo (os demais *tokens* serão detalhados no fim do relatório):

INT_CONSTANT



FLOAT_CONSTANT



STRING_CONSTANT

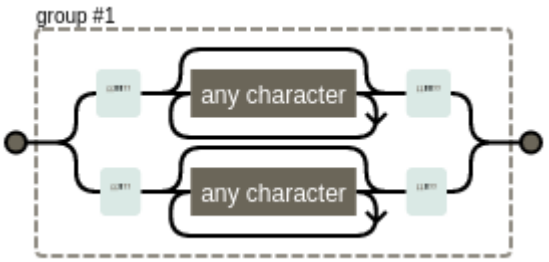


TABELA DE SÍMBOLOS

A tabela de símbolos foi implementada utilizando a estrutura de dicionários da linguagem *Python*, onde as chaves correspondem aos lexemas armazenados e o conteúdo de cada entrada é uma lista de tuplas, contendo as linhas e colunas correspondentes às posições dos mesmos, no código fonte.

Na versão atual, apenas lexemas da classe *ID* (correspondente a nomes de variáveis e funções) são armazenados na tabela.

FERRAMENTA *PLY* - ENTRADA EXIGIDA

A ferramenta *PLY* foi utilizada para desenvolvimento do analisador léxico. Como entrada para análise léxica, o projeto exige definições de variáveis e funções no escopo global do programa para que a instância da classe *lex* faça o devido processamento.

As definições são as seguintes:

- **Variáveis *t_[token]* -** devem conter as expressões regulares correspondentes aos *tokens* utilizados;
- **Funções *t_[token](t)* -** contêm, além dos *RegEx* correspondentes, instruções adicionais, para que o programa execute, ao se deparar com tais *tokens*;
- **Variável "tokens" -** tupla com a identificação de cada *token*;

Outras definições podem ser feitas, como a utilização de literais (*tokens* correspondentes à exatamente um caractere) ou a opção de ignorar cadeias de caracteres específicas.

Um exemplo de como fornecer entradas para a ferramenta pode ser encontrado aqui.

FERRAMENTA *PLY* - SAÍDA ESPERADA

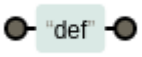
A ferramenta fornece funções que, quando alimentadas, produzem a identificação dos *tokens* automaticamente. No projeto em questão, foi utilizada a função *input*, que recebe uma cadeia de caracteres, correspondente ao código que se deseja analisar.

Um exemplo de saída esperada e utilização da função *input* pode ser encontrado aqui.

DIAGRAMAS DE TRANSIÇÃO - *TOKENS* TRIVIAIS

Para os *tokens* triviais e palavras regulares, os diagramas de transição correspondem a exatamente o conjunto de caracteres definido por tal palavra. Como todos os diagramas possuem o mesmo comportamento, apenas três serão exemplificado a seguir:

def



string

return