

Compilador Lexico

Doc. Victor de la Cueva

Diseño de compiladores

Isaias Martinez Vieyra

A00988525

Tokens:

Se componen de palabras reservadas e identificadores

reserved = {

'if': 'IF',

'else': 'ELSE',

'int': 'INT',

'return': 'RETURN',

'void': 'VOID',

'while': 'WHILE'

}

tokens = [

'PLUS',

'MINUS',

'TIMES',

'DIVIDE',

'LTHAN',

'LOEQU',

'MTHAN',

'MOEQU',

'EQUALS',

'DIFF',

'ASSIGN',

'SEMICOLON',

'COMA',

'LPAREN',

'RPAREN',

'LBRACKET',

'RBRACKET',

'LKEY',

'RKEY',

'COMMENTS',

'COMMENTE',

'ID',

'NUM',

'LETTER',

'DIGT',

'ENDFILE'

]+list(reserved.values())

Mis tokens se revisan con las siguientes expresiones regulares.

# Regular expression rules for simple tokens

t\_PLUS = r'\+'

t\_MINUS = r'-'

t\_TIMES = r'\\*'

t\_DIVIDE = r'/'

t\_LTHAN = r'<'

t\_LOEQU = r'<='

t\_MTHAN = r'>'

t\_MOEQU = r'>='

t\_EQUALS = r'=='

t\_DIFF = r'!='

t\_ASSIGN = r'='

t\_SEMICOLON = r';'

t\_COMA = r'\,'

t\_LPAREN = r'\('

t\_RPAREN = r'\)'

t\_LBRACKET = r'\['

t\_RBRACKET = r'\]'

t\_LKEY = r'\{'

t\_RKEY = r'\}'

eof = '$'

t\_ENDFILE = r'\$'

t\_ignore = ' \t'

def t\_NUM(t):

r'\d+'

try:

t.value = int(t.value)

return t

except ValueError:

print("Integer value too large %d", t.value)

# Quitamos los espacios vacios y tabs

def t\_newline(t):

r'\n+'

t.lexer.lineno += len(t.value)

# Detectamos Errores, los impimimos y nos saltamos al sig token

def t\_error(t):

print("\nError: ")

print(f"Linea {t.lexer.lineno}: String Unknown \"{t.value[0]}\" ")

arr = t.lexer.lexdata.split('\n')

linea = arr[t.lexer.lineno-1].strip()

print(linea)

print(" "\*linea.find(t.value[0])+"^")

t.lexer.skip(1)

def t\_ID(t):

r'[a-zA-Z][a-zA-Z]\*\d+'

t.type = reserved.get(t.value, 'ID')

return t

def t\_COMMENT(t):

r'(/\\*(.|\n+)\*?\\*/)'

pass

DFA para deteccion de errores:

Estoy ocupando una Libreria similar a Lex, se llama ply, y es Lex adaptado a python ( <https://www.dabeaz.com/ply/> ), por lo que no hay necesidad de hacer otra implementacion o adaptar.

Dentro de la libreria de ply, existe el metodo para detectar errores “t\_error(t)”. El cual recibe un parametro tipo Token, este metodo se manda a llamar cuando no pasa ninguno de las expresiones regulares previamente definidas.

def t\_error(t):

print("\nError: ")

print(f"Linea {t.lexer.lineno}: String Unknown \"{t.value[0]}\" ")

arr = t.lexer.lexdata.split('\n')

linea = arr[t.lexer.lineno-1].strip()

print(linea)

print(" "\*linea.find(t.value[0])+"^")

t.lexer.skip(1)

Lo que hace ply para reconocer es tomar la string mas larga hasta que exista un espacio y evalua si es Token lo acepta si no sale error.

