Universidad La Salle

Compiladores

Informe de la Práctica 07

Analizador Léxico con Ply.lex

Karlo Emigdio Pacha Curimayhua

Sexto Semestre - Ingeniería de Software

2023

Ejercicio 1

Enunciado

Implemente el analizador léxico para el lenguaje propuesto. Su programa deberá leer el código fuente de archivo en disco (debe proporcionar varios ejemplos) y luego deberá mostrar todos los tokens de manera similar al ejemplo mostrado en la Sección 5.

Prueba 1

Código 1

```
import ply.lex as lex

reserved = {
  'main' : 'main',
  'var'  : 'var',
  'null' : 'null',
  'return': 'return',
  'func': 'funcion',
  'print' : 'print',
  'read' : 'read',
  'if' : 'if',
  'elif' : 'elseif',
  'else' : 'else',
  'for' : 'for',
  'while' : 'while'
}

# List of token names. This is always required
tokens = ['FLOAT', 'INT','STRING', 'PLUS', 'MINUS', 'TIMES', '
    DIVIDE', 'LPAREN', 'RPAREN','LKEY', 'RKEY', 'EQUAL', 'GREATER',
  'LESS', 'EQUALEQUAL', 'LESSEQUAL', 'GREATEREQUAL', 'DIFERENT', '
    COMMENT', 'COMMENTOPEN', 'COMMENTCLOSE', 'IDENTIFIER', 'COMMA',
     'OR', 'AND' ] + list(reserved.values())

# Regular expression rules for simple tokens
t_PLUS = r'\+'
t_MINUS = r'-'
t_TIMES = r'\*'
t_DIVIDE = r'/'
t_LPAREN = r'\('
t_RPAREN = r'\)'

t_LKEY = r'{'
t_RKEY = r'}'
t_EQUAL = r'='
t_GREATER = r'>'
t_LESS = r'<'
t_EQUALEQUAL = r'=='
t_LESSEQUAL = r'<='
t_GREATEREQUAL = r'>='
t_DIFERENT = r'!='
t_COMMENT = r'//.*'
t_COMMENTOPEN = r'/\*'
```

```
#t_COMMENTOPEN = r'/\*(\n| |.)*'
t_COMMENTCLOSE = r'\*/'
t_COMMA = r','
t_OR = r'\|'
t_AND = r'&'


# A regular expression rule with some action code
def t_FLOAT(t):
  r'\d+\.\d+'
  t.value = float(t.value)
  t.type = reserved.get(t.value,'FLOAT')
  return t

def t_INT(t):
  r'\d+'
  t.value = int(t.value)
  t.type = reserved.get(t.value,'INT') #
  return t

def t_STRING(t):
  r"(\"|\') (\n|\t||.|(\d)|\n|)*(\"|\')"
  t.type = reserved.get(t.value,'STRING') #
  return t


# A regular expression rule with some action code
def t_IDENTIFIER(t):
  r'[a-zA-Z]([a-zA-Z]|(\d+)|(_))*'

  t.type = reserved.get(t.value,'IDENTIFIER') # guardamos el valor
    del lexema
  return t

# Define a rule so we can track line numbers
def t_newline(t):
  r'\n+'
  t.lexer.lineno += len(t.value)

# A string containing ignored characters (spaces and tabs)
t_ignore = '\t\n '

# Error handling rule
def t_error(t):
  print("Illegal character     % s  " % t.value[0])
  t.lexer.skip(1)

# Build the lexer
lexer = lex.lex()

# Test it out
data = ""

with open('example.ds', 'r') as file: data = file.read()
# Give the lexer some input
lexer.input(data)

# Tokenize
```

```
out = open("out1.txt", "w")
while True:
  tok = lexer.token()
  if not tok: break
  print(tok)
  out.write(str(tok)+"\n")
out.close()
```

Ejemplo de Drachen Script 1

```
func Square (a) {
  return a*a
}

func main() {
    var a = 0
    var b = 10.5
    var c = null
    var d = "abc"
    var e = 'abc'
    var f = false

    if a < b & c == null {
        for i = 0, b, 1 {
            a += 1
        }
    }
    elif a == b | f == false {
        for i = a, 2*b {
            a += 1
        }
    }
    else {
        while a > b {
            a -= 1
        }
    }


    Square(a)

}
```

Salida 1

```
LexToken(funcion,'func',1,0)
LexToken(IDENTIFIER,'Square',1,5)
LexToken(LPAREN,'(',1,12)
LexToken(IDENTIFIER,'a',1,13)
LexToken(RPAREN,')',1,14)
LexToken(LKEY,'{',1,16)
LexToken(return,'return',1,20)
LexToken(IDENTIFIER,'a',1,27)
LexToken(TIMES,'*',1,28)
LexToken(IDENTIFIER,'a',1,29)
LexToken(RKEY,'}',1,31)
LexToken(funcion,'func',1,34)
LexToken(main,'main',1,39)
```

```
LexToken(LPAREN,'(',1,43)
LexToken(RPAREN,')',1,44)
LexToken(LKEY,'{',1,46)
LexToken(var,'var',1,52)
LexToken(IDENTIFIER,'a',1,56)
LexToken(EQUAL,'=',1,58)
LexToken(INT,0,1,60)
LexToken(var,'var',1,66)
LexToken(IDENTIFIER,'b',1,70)
LexToken(EQUAL,'=',1,72)
LexToken(FLOAT,10.5,1,74)
LexToken(var,'var',1,83)
LexToken(IDENTIFIER,'c',1,87)
LexToken(EQUAL,'=',1,89)
LexToken(null,'null',1,91)
LexToken(var,'var',1,100)
LexToken(IDENTIFIER,'d',1,104)
LexToken(EQUAL,'=',1,106)
LexToken(STRING,'"abc"',1,108)
LexToken(var,'var',1,118)
LexToken(IDENTIFIER,'e',1,122)
LexToken(EQUAL,'=',1,124)
LexToken(STRING,"'abc'",1,126)
LexToken(var,'var',1,136)
LexToken(IDENTIFIER,'f',1,140)
LexToken(EQUAL,'=',1,142)
LexToken(IDENTIFIER,'false',1,144)
LexToken(if,'if',1,155)
LexToken(IDENTIFIER,'a',1,158)
LexToken(LESS,'<',1,160)
LexToken(IDENTIFIER,'b',1,162)
LexToken(AND,'&',1,164)
LexToken(IDENTIFIER,'c',1,166)
LexToken(EQUALEQUAL,'==',1,168)
LexToken(null,'null',1,171)
LexToken(LKEY,'{',1,176)
LexToken(for,'for',1,186)
LexToken(IDENTIFIER,'i',1,190)
LexToken(EQUAL,'=',1,192)
LexToken(INT,0,1,194)
LexToken(COMMA,',',1,195)
LexToken(IDENTIFIER,'b',1,197)
LexToken(COMMA,',',1,198)
LexToken(INT,1,1,200)
LexToken(LKEY,'{',1,202)
LexToken(IDENTIFIER,'a',1,216)
LexToken(PLUS,'+',1,218)
LexToken(EQUAL,'=',1,219)
LexToken(INT,1,1,221)
LexToken(RKEY,'}',1,231)
LexToken(RKEY,'}',1,238)
LexToken(elseif,'elif',1,247)
LexToken(IDENTIFIER,'a',1,252)
LexToken(EQUALEQUAL,'==',1,254)
LexToken(IDENTIFIER,'b',1,257)
LexToken(OR,'|',1,259)
LexToken(IDENTIFIER,'f',1,261)
LexToken(EQUALEQUAL,'==',1,263)
```

```
LexToken(IDENTIFIER,'false',1,266)
LexToken(LKEY,'{',1,272)
LexToken(for,'for',1,282)
LexToken(IDENTIFIER,'i',1,286)
LexToken(EQUAL,'=',1,288)
LexToken(IDENTIFIER,'a',1,290)
LexToken(COMMA,',',1,291)
LexToken(INT,2,1,293)
LexToken(TIMES,'*',1,294)
LexToken(IDENTIFIER,'b',1,295)
LexToken(LKEY,'{',1,297)
LexToken(IDENTIFIER,'a',1,311)
LexToken(PLUS,'+',1,313)
LexToken(EQUAL,'=',1,314)
LexToken(INT,1,1,316)
LexToken(RKEY,'}',1,326)
LexToken(RKEY,'}',1,332)
LexToken(else,'else',1,338)
LexToken(LKEY,'{',1,343)
LexToken(while,'while',1,353)
LexToken(IDENTIFIER,'a',1,359)
LexToken(GREATER,'>',1,361)
LexToken(IDENTIFIER,'b',1,363)
LexToken(LKEY,'{',1,365)
LexToken(IDENTIFIER,'a',1,379)
LexToken(MINUS,'-',1,381)
LexToken(EQUAL,'=',1,382)
LexToken(INT,1,1,384)
LexToken(RKEY,'}',1,394)
LexToken(RKEY,'}',1,400)
LexToken(IDENTIFIER,'Square',1,412)
LexToken(LPAREN,'(',1,418)
LexToken(IDENTIFIER,'a',1,419)
LexToken(RPAREN,')',1,420)
LexToken(RKEY,'}',1,423)
```

Prueba 2

Código 2

```
import ply.lex as lex

reserved = {
   'main' : 'main',
   'var'  : 'var',
   'null' : 'null',
   'return': 'return',
   'func': 'funcion',
   'print' : 'print',
   'read' : 'read',
   'if' : 'if',
   'elif' : 'elseif',
   'else' : 'else',
   'for' : 'for',
   'while' : 'while'
}

# List of token names. This is always required
```

```
tokens = ['FLOAT', 'INT','STRING', 'PLUS', 'MINUS', 'TIMES', 'DIVIDE', 'LPAREN', 'RPAREN','LKEY', 'RKEY', 'EQUAL', 'GREATER', 'LESS', 'EQUALEQUAL', 'LESSEQUAL', 'GREATEREQUAL', 'DIFERENT', 'COMMENT', 'COMMENTOPEN', 'COMMENTCLOSE', 'IDENTIFIER', 'COMMA', 'OR', 'AND' ] + list(reserved.values())

# Regular expression rules for simple tokens
t_PLUS = r'\+'
t_MINUS = r'-'
t_TIMES = r'\*'
t_DIVIDE = r'/'
t_LPAREN = r'\('
t_RPAREN = r'\)'

t_LKEY = r'{'
t_RKEY = r'}'
t_EQUAL = r'='
t_GREATER = r'>'
t_LESS = r'<'
t_EQUALEQUAL = r'=='
t_LESSEQUAL = r'<='
t_GREATEREQUAL = r'>='
t_DIFERENT = r'!='
t_COMMENT = r'//.*'
t_COMMENTOPEN = r'/\*'
#t_COMMENTOPEN = r'/\*(\n| |.)*'
t_COMMENTCLOSE = r'\*/'
t_COMMA = r','
t_OR = r'\|'
t_AND = r'&'


# A regular expression rule with some action code
def t_FLOAT(t):
  r'\d+\.\d+'
  t.value = float(t.value)
  t.type = reserved.get(t.value,'FLOAT')
  return t

def t_INT(t):
  r'\d+'
  t.value = int(t.value)
  t.type = reserved.get(t.value,'INT') #
  return t

def t_STRING(t):
  r"(\"|\') (\n|\t||.|(\d)|\n|)*(\"|\')"
  t.type = reserved.get(t.value,'STRING') #
  return t


# A regular expression rule with some action code
def t_IDENTIFIER(t):
  r'[a-zA-Z]([a-zA-Z]|(\d+)|(_))*'

  t.type = reserved.get(t.value,'IDENTIFIER') # guardamos el valor del lexema
  return t
```

```
73
74 # Define a rule so we can track line numbers
75 def t_newline(t):
76   r'\n+'
77   t.lexer.lineno += len(t.value)
78
79 # A string containing ignored characters (spaces and tabs)
80 t_ignore = '\t\n '
81
82 # Error handling rule
83 def t_error(t):
84   print("Illegal character     % s  " % t.value[0])
85   t.lexer.skip(1)
86
87 # Build the lexer
88 lexer = lex.lex()
89
90 # Test it out
91 data = ""
92
93 with open('example2.ds', 'r') as file: data = file.read()
94 # Give the lexer some input
95 lexer.input(data)
96
97 # Tokenize
98 out = open("out2.txt", "w")
99 while True:
100   tok = lexer.token()
101   if not tok: break
102   print(tok)
103   out.write(str(tok)+"\n")
104 out.close()
```

Ejemplo de Drachen Script 2

```
1  func Square (a) {
2    return a*a
3  }
4
5  func main() {
6      var a = 0
7      var b = 10.5
8      var c = null
9      var d = "abc"
10     var e = 'abc'
11     var f = false
12
13     if a < b & c == null {
14         for i = 0, b, 1 {
15             a += 1
16         }
17     }
18     elif a == b | f == false {
19         for i = a, 2*b {
20             a += 1
21         }
22     }
23     else {
24         while a > b {
```

```
            a -= 1
        }
    }


    Square(a)

}
```

Salida 2

```
LexToken(COMMENT,'// esto es un comentario de una linea',1,0)
LexToken(COMMENTOPEN,'/*',1,38)
LexToken(IDENTIFIER,'comentario',1,41)
LexToken(IDENTIFIER,'multilinea',1,52)
LexToken(IDENTIFIER,'mas',1,63)
LexToken(IDENTIFIER,'lineas',1,67)
LexToken(IDENTIFIER,'del',1,74)
LexToken(IDENTIFIER,'comentario',1,78)
LexToken(COMMENTCLOSE,'*/',1,89)
LexToken(funcion,'func',1,93)
LexToken(main,'main',1,98)
LexToken(LPAREN,'(',1,102)
LexToken(RPAREN,')',1,103)
LexToken(LKEY,'{',1,105)
LexToken(print,'print',1,111)
LexToken(LPAREN,'(',1,116)
LexToken(INT,1,1,117)
LexToken(RPAREN,')',1,118)
LexToken(var,'var',1,124)
LexToken(IDENTIFIER,'a',1,128)
LexToken(EQUAL,'=',1,130)
LexToken(read,'read',1,132)
LexToken(LPAREN,'(',1,136)
LexToken(INT,2,1,137)
LexToken(RPAREN,')',1,138)
LexToken(var,'var',1,144)
LexToken(IDENTIFIER,'MiIdentificador',1,148)
LexToken(EQUAL,'=',1,164)
LexToken(INT,10,1,166)
LexToken(var,'var',1,173)
LexToken(IDENTIFIER,'mi_segundo_identificador',1,177)
LexToken(EQUAL,'=',1,202)
LexToken(INT,20,1,204)
LexToken(var,'var',1,211)
LexToken(IDENTIFIER,'a',1,215)
LexToken(EQUAL,'=',1,217)
LexToken(IDENTIFIER,'true',1,219)
LexToken(if,'if',1,228)
LexToken(IDENTIFIER,'a',1,231)
LexToken(LKEY,'{',1,233)
LexToken(for,'for',1,243)
LexToken(IDENTIFIER,'i',1,247)
LexToken(EQUAL,'=',1,249)
LexToken(INT,0,1,251)
LexToken(COMMA,',',1,252)
LexToken(INT,10,1,254)
LexToken(COMMA,',',1,257)
LexToken(INT,2,1,259)
```

```
LexToken(LKEY,'{',1,261)
LexToken(print,'print',1,275)
LexToken(LPAREN,'(',1,280)
LexToken(IDENTIFIER,'i',1,281)
LexToken(RPAREN,')',1,282)
LexToken(RKEY,'}',1,292)
LexToken(RKEY,'}',1,298)
LexToken(if,'if',1,305)
LexToken(IDENTIFIER,'a',1,308)
LexToken(EQUALEQUAL,'==',1,310)
LexToken(IDENTIFIER,'false',1,313)
LexToken(LKEY,'{',1,319)
LexToken(RKEY,'}',1,325)
LexToken(if,'if',1,332)
LexToken(IDENTIFIER,'b',1,335)
LexToken(LESS,'<',1,337)
LexToken(INT,11,1,339)
LexToken(LKEY,'{',1,342)
LexToken(RKEY,'}',1,348)
LexToken(if,'if',1,355)
LexToken(IDENTIFIER,'a',1,358)
LexToken(DIFERENT,'!=',1,360)
LexToken(IDENTIFIER,'false',1,363)
LexToken(LKEY,'{',1,369)
LexToken(RKEY,'}',1,375)
LexToken(var,'var',1,382)
LexToken(IDENTIFIER,'entero',1,386)
LexToken(EQUAL,'=',1,393)
LexToken(INT,123,1,395)
LexToken(var,'var',1,403)
LexToken(IDENTIFIER,'flotante',1,407)
LexToken(EQUAL,'=',1,416)
LexToken(FLOAT,1.23,1,418)
LexToken(var,'var',1,427)
LexToken(IDENTIFIER,'booleano',1,431)
LexToken(EQUAL,'=',1,440)
LexToken(IDENTIFIER,'true',1,442)
LexToken(var,'var',1,451)
LexToken(IDENTIFIER,'nulo',1,455)
LexToken(EQUAL,'=',1,460)
LexToken(null,'null',1,462)
LexToken(return,'return',1,472)
LexToken(INT,0,1,479)
LexToken(RKEY,'}',1,481)
```