

# Práctica 11

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Compiladores

PRÁCTICA	TEMA	DURACIÓN
11	LL(1) Parsing	3 horas

## 1. Datos de los estudiantes

- Grupo: 1
- Git Hub: <https://github.com/CrazyDani17/Practica11-Compiladores>
- Integrantes:
  - Guillermo Aleman
  - Marvik Del Carpio
  - Daniel Mendiguri
  - Daniela Vilchez

## 2. Preguntas

1. Diseñe la tabla sintáctica de gramática de su lenguaje.

Revisar archivo adjunto de excel: <https://github.com/CrazyDani17/Practica11-Compiladores/blob/main/tablita.xlsx>

[illegible]

## 2. Implemente el algoritmo LL(1) del analizador sintáctico.

- Input: Archivo de texto con código fuente.
- Output: Arbol sintáctico, debe utilizar graphviz.

Las fases del programa serían:

- Leer el código fuente y generar una lista de tokens (analizador léxico).
- Leer la tabla sintáctica, puede ser desde un archivo csv o puede estar de manera estática en el código.
- Con la tabla sintáctica y la lista de tokens, generar el arbol sintáctico y validar si el código pertenece a la gramática.

Solución:

Analizador léxico código

```
import ply.lex as lex
# List of token names. This is always required
tokens = ('mision', 'si', 'sino', 'pinocho', 'numero', 'lnumero', 'decimal',
          'ldecimal', 'mostrar', 'texto', 'ltexto', 'descansito', 'yoyo', 'devuelve', 'chacha',
          'lchacha', 'mas', 'menos', 'por', 'y', 'o', 'dividir', 'mayor', 'menor', 'pizquierdo',
          'pderecho', 'cizquierdo', 'cderecho', 'coma', 'igual', 'menor_igual', 'mayor_igual',
          'diferente', 'identificador', 'lizquierdo', 'lderecho', 'division',
          'verdad', 'mentira', 'modulo', 'llama')

# Regular expression rules for simple tokens
t_mision = r'mision'
t_si = r'si'
t_sino = r'sino'
t_pinocho = r'pinocho'
t_numero = r'numero'
t_decimal = r'decimal'
t_mostrar = r'mostrar'
t_texto = r'texto'
t_descansito = r'descansito'
t_yoyo = r'yoyo'
t_devuelve = r'devuelve'
t_chacha = r'chacha'
t_llama = r'llama'

t_mas = r'\+'
t_menos = r'\-'
t_por = r'\*'
t_division = r'\/'
t_modulo = r'\%'

t_y = r'\#y'
t_o = r'\#o'

t_mayor = r'\>'
t_menor = r'\<'

t_pizquierdo = r'\('
```

Página 3

```
def t_ltexto(t):
    r'\\"(\W|\w)+\''
    t.value = t.value # guardamos el valor del lexema
    return t

def t_comments(t):
    r'\~(\W|\W)*\~'

# 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'
# 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

file=open('archivo.txt','r')
texto=file.readlines()
file.close()
data = texto
tokens=[]
for renglon in texto:
    # Give the lexer some input
    lexer.input(renglon)
    # Tokenize
    while True:
        tok = lexer.token()
        if not tok:
            break # No more input
        tokens.append(tok.type)
        #print(tok.type, tok.value, tok.lineno, tok.lexpos)
```

---

## Algoritmo LL(1) código

---

```
import analizador_lexico
import xlrd
import pandas as pd
import numpy as np
from graphviz import Digraph
dot = Digraph()
filas = {"Inicio" :1 ,
"E" :2 ,
"E'" :3 ,
"DeclaracionFuncion" :4 ,
"FuncionPrincipal" :5 ,
"K" :6 ,
"Parametros" :7 ,
"Y'" :8 ,
"Y" :9 ,
"TipoDato" :10 ,
"C" :11 ,
"CuerpoF" :12 ,
"J" :13 ,
"Cuerpo" :14 ,
"D'" :15 ,
"DeclaracionVariables" :16 ,
"Dc'" :17 ,
"Sentencias" :18 ,
"OPS" :19 ,
"MuchasSentenciasYoyo" :20 ,
"M'" :21 ,
"SentenciasYoyo" :22 ,
"MuchasSentencias" :23 ,
"S'" :24 ,
"DeclaracionVariable" :25 ,
"Corchetes" :26 ,
"OPI" :27 ,
"Expresion" :28 ,
"Ex" :29 ,
"Ex'" :30 ,
"Tx" :31 ,
"Literal" :32 ,
"Op" :33 ,
"ParaLLamados" :34 ,
"PL'" :35 ,
"PL" :36 ,
"Simbolo" :37
}
columnas={"mision" :1 ,
"identificador" :2 ,
"pizquierdo" :3 ,
"pderecho" :4 ,
"lizquierdo" :5 ,
"lderecho" :6 ,
"llama" :7 ,
"coma" :8 ,
"pinocho" :9 ,
```

```
"numero" :10 ,
"decimal" :11 ,
"texto" :12 ,
"chacha" :13 ,
"cizquierdo" :14 ,
"cderecho" :15 ,
"devuelve" :16 ,
"si" :17 ,
"sino" :18 ,
"yoyo" :19 ,
"mostrar" :20 ,
"=" :21 ,
"igual" :22 ,
"descansito" :23 ,
"lpinocho" :24 ,
"lnumero" :25 ,
"ltexto" :26 ,
"ldecimal" :27 ,
"lchacha" :28 ,
"verdad" :29 ,
"mentira" :30 ,
"mas" :31 ,
"menos" :32 ,
"por" :33 ,
"o" :34 ,
"y" :35 ,
"mayor_igual" :36 ,
"diferente" :37 ,
"menor_igual" :38 ,
"comparacion" :39 ,
"menor" :40 ,
"mayor" :41 ,
"modulo" :42 ,
"dividir" :43 ,
"$" :44
}
df = pd.read_excel("tablita.xlsx", 'Hoja1', header=None)
tablita_parse = df.values

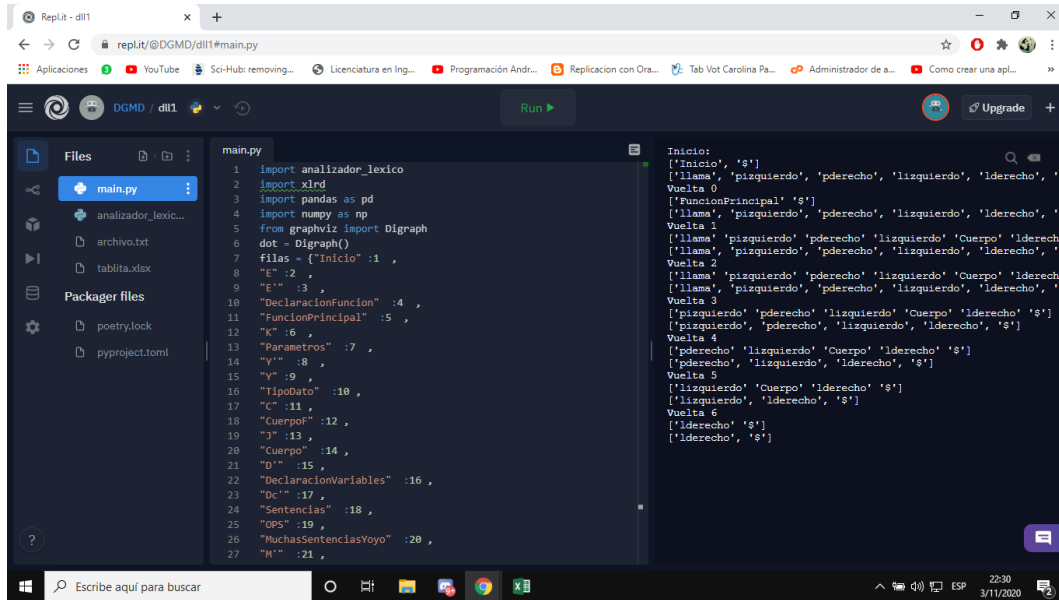
pila = ["Inicio","$"]
entrada = analizador_lexico.tokens
entrada.append("$")

continuar=True
i=0
j=0
p=0
aux=[]
dot.node(str(j), pila[0])
padres=[0,-1]
print("Inicio:")
print(pila)
print(entrada)

while continuar:
    print("Vuelta",i)
```

```
if pila[0]=="$" and entrada[0]=="$":
    continuar=False
    print(pila)
    print(entrada)
    print("Cadena aceptada")
elif pila[0] == entrada[0]:
    print(pila)
    print(entrada)
    pila = pila[1:]
    entrada.pop(0)
elif pila[0][0]==(pila[0][0]).lower() and entrada[0][0]==(entrada[0][0]).lower():
    continuar=False
    print(pila)
    print(entrada)
    print("Error: Cadena rechazada")
else:
    if entrada[0] in columnas:
        reemplazo = tablita_parse[filas[pila[0]]][columnas[entrada[0]]]
        p=int(padres[0])
    else:
        continuar=False
        print(pila)
        print(entrada)
        print("Error: Cadena rechazada")
        break
    if reemplazo == "vacio":
        j=j+1
        h=j
        dot.node(str(h), "' '")
        dot.edge(str(p),str(h))
        pila=pila[1:]
        padres=padres[1:]
        print(pila)
        print(entrada)
    else:
        array_aux=reemplazo.split()
        pila = np.concatenate((array_aux,pila[1:]),axis=0)
        hijos = len(array_aux)
        aux.clear()
        for e in range(hijos):
            j=j+1
            if array_aux[e][0]==array_aux[e][0].upper():
                aux.append(j)
            h=j
            dot.node(str(h), array_aux[e])
            dot.edge(str(p),str(h))
        padres = np.concatenate((aux,padres[1:]),axis=0)
        print(pila)
        print(entrada)
    i=i+1
print(dot.source)
```

## Pruebas del código



```

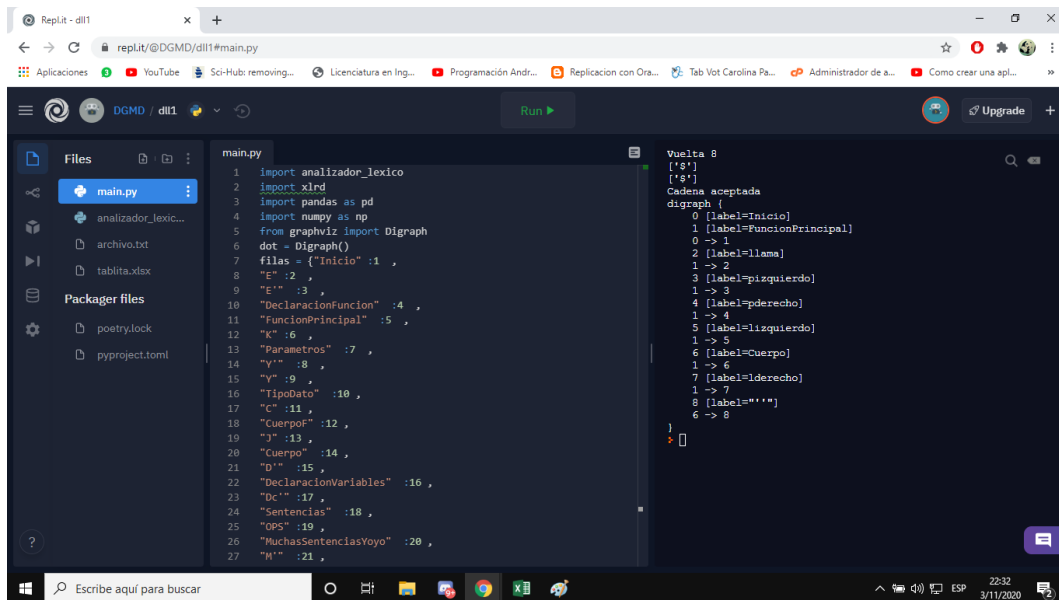
1 import analizador_lexico
2 import xlrld
3 import pandas as pd
4 import numpy as np
5 from graphviz import Digraph
6 dot = Digraph()
7 filas = {"Inicio" :1 ,
8 "E" :2 ,
9 "E'" :3 ,
10 "DeclaracionFuncion" :4 ,
11 "FuncionPrincipal" :5 ,
12 "K" :6 ,
13 "Parametros" :7 ,
14 "Y'" :8 ,
15 "Y" :9 ,
16 "TipoDato" :10 ,
17 "C" :11 ,
18 "CuerpoP" :12 ,
19 "J" :13 ,
20 "Cuerpo" :14 ,
21 "D" :15 ,
22 "DeclaracionVariables" :16 ,
23 "Dc" :17 ,
24 "Sentencias" :18 ,
25 "OPS" :19 ,
26 "MuchasSentenciasYoyo" :20 ,
27 "M" :21 ,

```

```

Inicio:
["Inicio", '$']
["llama", 'pizquierdo', 'pderecho', 'lizquierdo', 'lderecho', '$']
Vuelta 0
["FuncionPrincipal", '$']
["llama", 'pizquierdo', 'pderecho', 'lizquierdo', 'lderecho', '$']
Vuelta 1
["llama", 'pizquierdo', 'pderecho', 'lizquierdo', 'Cuerpo', 'lderecho', '$']
["llama", 'pizquierdo', 'pderecho', 'lizquierdo', 'lderecho', '$']
Vuelta 2
["llama", 'pizquierdo', 'pderecho', 'lizquierdo', 'Cuerpo', 'lderecho', '$']
["llama", 'pizquierdo', 'pderecho', 'lizquierdo', 'lderecho', '$']
Vuelta 3
["pizquierdo", 'pderecho', 'lizquierdo', 'Cuerpo', 'lderecho', '$']
["pizquierdo", 'pderecho', 'lizquierdo', 'lderecho', '$']
Vuelta 4
["pderecho", 'lizquierdo', 'Cuerpo', 'lderecho', '$']
["pderecho", 'lizquierdo', 'lderecho', '$']
Vuelta 5
["lizquierdo", 'Cuerpo', 'lderecho', '$']
["lizquierdo", 'lderecho', '$']
Vuelta 6
["lderecho", '$']
["lderecho", '$']

```



```

1 import analizador_lexico
2 import xlrld
3 import pandas as pd
4 import numpy as np
5 from graphviz import Digraph
6 dot = Digraph()
7 filas = {"Inicio" :1 ,
8 "E" :2 ,
9 "E'" :3 ,
10 "DeclaracionFuncion" :4 ,
11 "FuncionPrincipal" :5 ,
12 "K" :6 ,
13 "Parametros" :7 ,
14 "Y'" :8 ,
15 "Y" :9 ,
16 "TipoDato" :10 ,
17 "C" :11 ,
18 "CuerpoP" :12 ,
19 "J" :13 ,
20 "Cuerpo" :14 ,
21 "D" :15 ,
22 "DeclaracionVariables" :16 ,
23 "Dc" :17 ,
24 "Sentencias" :18 ,
25 "OPS" :19 ,
26 "MuchasSentenciasYoyo" :20 ,
27 "M" :21 ,

```

```

Vuelta 8
['$']
['$']
Cadena aceptada
digraph {
  0 [label=Inicio]
  1 [label=FuncionPrincipal]
  0 --> 1
  2 [label=llama]
  1 --> 2
  3 [label=pizquierdo]
  1 --> 3
  4 [label=pderecho]
  1 --> 4
  5 [label=lizquierdo]
  1 --> 5
  6 [label=Cuerpo]
  1 --> 6
  7 [label=lderecho]
  1 --> 7
  8 [label=""]
  6 --> 8
}

```



Árbol generado

