

GRAMATICA DEL COMPILADOR

OMAR ALEJANDRO QUIROZ TRUJILLO

- **DESARROLLO**

Un compilador es un tipo de software que traduce el código fuente de un lenguaje de programación a un código ejecutable en otro lenguaje, como el lenguaje de máquina o un código intermedio. En términos más simples, un compilador toma el código escrito por un programador en un lenguaje de programación específico y lo convierte en un formato que la computadora puede entender y ejecutar. Este proceso se realiza en varias etapas, que incluyen análisis léxico, análisis sintáctico, generación de código intermedio y optimización, entre otras. Los compiladores son herramientas fundamentales en el desarrollo de software, ya que permiten a los programadores escribir programas en lenguajes de alto nivel y luego ejecutarlos en diferentes plataformas y arquitecturas de hardware.

FUENTE:

<https://immune.institute/blog/que-es-un-compilador/>

- **GRAMATICA PARA EL COMPILADOR**

identificador 0

entero 1

real 2

cadena 3

tipo 4

opSuma 5

opMul 6

opRelac 7

opOr 8

opAnd 9

opNot 10

opIgualdad 11

; 12

, 13

(14

) 15

{ 16

} 17

= 18

if 19

while 20

return 21

else 22

\$ 23

R1 <programa> ::= <Definiciones>

R2 <Definiciones> ::= \e

R3 <Definiciones> ::= <Definicion> <Definiciones>

R4 <Definicion> ::= <DefVar>



R5 <Definicion> ::= <DefFunc>

R6 <DefVar> ::= tipo identificador <ListaVar> ;

R7 <ListaVar> ::= \e

R8 <ListaVar> ::= , identificador <ListaVar>

R9 <DefFunc> ::= tipo identificador (<Parametros>) <BloqFunc>

R10 <Parametros> ::= \e

R11 <Parametros> ::= tipo identificador <ListaParam>

R12 <ListaParam> ::= \e

R13 <ListaParam> ::= , tipo identificador <ListaParam>

R14 <BloqFunc> ::= { <DefLocales> }

R15 <DefLocales> ::= \e

R16 <DefLocales> ::= <DefLocal> <DefLocales>

R17 <DefLocal> ::= <DefVar>

R18 <DefLocal> ::= <Sentencia>

R19 <Sentencias> ::= \e

R20 <Sentencias> ::= <Sentencia> <Sentencias>

R21 <Sentencia> ::= identificador = <Expresion> ;

R22 <Sentencia> ::= if (<Expresion>) <SentenciaBloque> <Otro>

R23 <Sentencia> ::= while (<Expresion>) <Bloque>

R24 <Sentencia> ::= return <ValorRegresa> ;

R25 <Sentencia> ::= <LlamadaFunc> ;

R26 <Otro> ::= \e

R27 <Otro> ::= else <SentenciaBloque>

R28 <Bloque> ::= { <Sentencias> }

R29 <ValorRegresa> ::= \e

R30 <ValorRegresa> ::= <Expresion>

R31 <Argumentos> ::= \e

R32 <Argumentos> ::= <Expresion> <ListaArgumentos>

R33 <ListaArgumentos> ::= \e

R34 <ListaArgumentos> ::= , <Expresion> <ListaArgumentos>

R35 <Termino> ::= <LlamadaFunc>

R36 <Termino> ::= identificador

R37 <Termino> ::= entero

R38 <Termino> ::= real

R39 <Termino> ::= cadena

R40 <LlamadaFunc> ::= identificador (<Argumentos>)

R41 <SentenciaBloque> ::= <Sentencia>

R42 <SentenciaBloque> ::= <Bloque>

R43 <Expresion> ::= (<Expresion>)

R44 <Expresion> ::= opSuma <Expresion>

R45 <Expresion> ::= opNot <Expresion>

R46 <Expresion> ::= <Expresion> opMul <Expresion>

R47 <Expresion> ::= <Expresion> opSuma <Expresion>

R48 <Expresion> ::= <Expresion> opRelac <Expresion>

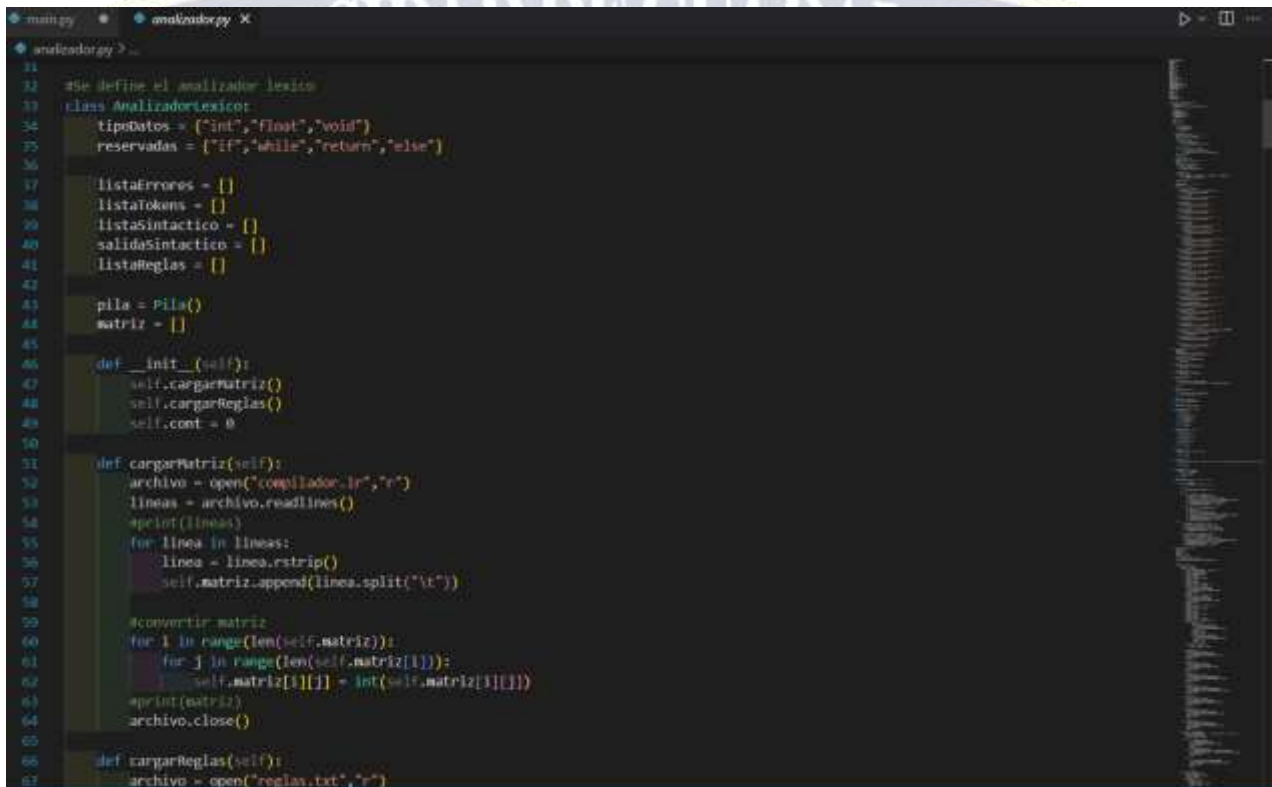
R49 <Expresion> ::= <Expresion> oplgualdad <Expresion>

R50 <Expresion> ::= <Expresion> opAnd <Expresion>

R51 <Expresion> ::= <Expresion> opOr <Expresion>

R52 <Expresion> ::= <Termino>

- **PRUEBAS DE QUE EL PROGRAMA FUNCIONA**



```
11
12 #Se define el analizador lexico
13 class AnalizadorLexico:
14     tiposDatos = ["int", "float", "void"]
15     reservadas = ["if", "while", "return", "else"]
16
17     listaErrores = []
18     listaTokens = []
19     listaSintactico = []
20     salidaSintactico = []
21     listaReglas = []
22
23     pila = Pila()
24     matriz = []
25
26     def __init__(self):
27         self.cargarMatriz()
28         self.cargarReglas()
29         self.cont = 0
30
31     def cargarMatriz(self):
32         archivo = open("compilador.lex", "r")
33         lineas = archivo.readlines()
34         print(lineas)
35         for linea in lineas:
36             linea = linea.rstrip()
37             self.matriz.append(linea.split("\t"))
38
39         #convertir matriz
40         for i in range(len(self.matriz)):
41             for j in range(len(self.matriz[i])):
42                 self.matriz[i][j] = int(self.matriz[i][j])
43
44         print(matriz)
45         archivo.close()
46
47     def cargarReglas(self):
48         archivo = open("reglas.txt", "r")
```



```
main.py • analizador.py
main.py > ...
1 import analizador
2 print ("-----")
3 texto = input("Ingresa el texto que quieres analizar: ")
4 print ("-----")
5 analizador = analizador.AnalizadorLexico()
6 analizador.analizador(texto)
7 analizador.getListTokens()
8 print ("-----")
9 print("Esta es la salida bro")
10 print ("-----")
11 analizador.imprimirSalida()
12 print ("-----")
13 print("La pila es la siguiente bro")
14 print ("-----")
15 analizador.imprimirSintactico()
16 print ("-----")
```

```
-----
Ingresa el texto que quieres analizar: omar quiroz
-----
omar es un identificador
quiroz es un identificador
$ es un signo de $
-----
Esta es la salida bro
-----
R2
R1
R0
-----
La pila es la siguiente bro
-----
$5
$5 $2
$5 programa1
-----
PS. C:\Users\Omar\OneDrive\Documentos\Escritorio\Septimo Semestre\SEM TRADUCTORES II\GRAMATICA COMPILADOR> []
```

```
-----  
Ingresa el texto que quieres analizar: hola + mundo  
-----
```

```
hola es un identificador
```

```
+ operador suma
```

```
mundo es un identificador
```

```
$ es un signo de $
```

```
-----  
Esta es la salida bro  
-----
```

```
R2
```

```
R1
```

```
R0
```

```
-----  
La pila es la siguiente bro  
-----
```

```
$5
```

```
$5 $2
```

```
$5 programa1  
-----
```

• CONCLUSIÓN

Los compiladores juegan un papel fundamental en la programación al traducir código de alto nivel a lenguaje de máquina, lo que permite a los programadores escribir en un formato más comprensible y portable. Además de facilitar el desarrollo de software, los compiladores también contribuyen a la optimización del rendimiento y la detección de errores, lo que los convierte en herramientas indispensables en el proceso de desarrollo de software moderno.