22 de Junio del 2018

Maestría:

Sistemas Inteligentes Multimedia

Materia:

Compiladores y Librerías

Alumno:

Renato Armando De S. Palomares Navarro

Proyecto:

Proyecto final – Mi compilador

Introducción

Un compilador es un programa informático que traduce un programa que ha sido escrito en un lenguaje de programación a un lenguaje común, reúne diversos elementos o fragmentos en una misma unidad, sualmente lenguaje de máquina, aunque también puede ser traducido a un código intermedio (bytecode) o a texto. Este proceso de traducción se conoce como compilación, compilar consiste en traducir un programa escrito en un cierto lenguaje a otro.

Alcance

Este proyecto solo comprende e integra el analizador léxico del compilador, que es el contexto de este reporte. El analizador léxico soporta la validación de:

- Variables
- For loop
- Funciones
- Condicional if
- Llamadas a función
- Tipo de constantes

El lenguaje

La gramática del lenguaje de entrada es una mezcla de C y Fortran. Integra dos tipo de datos para variables, INT para definir variables de tipo entero y FLOAT para definir variables de tipo punto flotante. Para el programa principal se usa MAIN para el comienzo de programa y ENDMAIN para finalizar programa.

Así como lo hace Python, este compilador de igual forma analiza las constantes dentro del código y les asigna un tipo de dato, pues de sr decimal, hexadecimal o punto flotante.

Para los identificadores solo se usan caracteres alfabéticos en minúsculas y guión bajo. Para las palabras clave solo se usan caracteres alfabéticos en mayúsculas.

Variables

<type> <identifier> <=> <data> <;>

For loop

FOR <(> <index> <,> <condition> <,> <step> <)>

ENDFOR

Funciones

FUNC <func_name> <(> <parameter> <)> <(> <return_value> <)>

ENDFUNC

Condicional If

IF <(> <condition> <)>

ENDIF

Llamada a función

CALL <func_name> <(> <parameter> <)> <;>

Tabla de símbolos

- Tipo decimal para constantes [DECI]
- Tipo hexadecimal para constantes [HEXA]
- Tipo de punto flotante para constantes [FLOT]
- Tipo entero para variables [INT]
- Tipo flotante para variables [FLOAT]
- Operador “+” [PLUS]
- Operador “++” [PLUSPLUS]
- Operador “-“ [MINUS]
- Operador “- -“ [MINUSMINUS]
- Operador “=” [ASSIGN]

- Operador “==” [EQUALTO]
- Operador “/” [DIV]
- Operador “*” [MULT]
- Operador “<” [LESSTHAN]
- Operador “>” [MORETHAN]
- Operador “!” [EXCLA]
- Operador “&” [AND]
- Operador “|” [OR]
- Operador “^” [XOR]
- Operador “ (“ [LPAREN]
- Operador “)” [RPAREN]
- Operador “;” [SEMICOLON]
- Operador “:” [COLON]
- Operador “,” [COMMA]
- Palabra clave MAIN [MAIN]
- Palabra clave ENDMAIN [ENDMAIN]
- Palabra clave INT [INT]
- Palabra clave FLOAT [FLOAT]
- Palabra clave FOR [FOR]
- Palabra clave FUNC [FUNC]
- Palabra clave IF [IF]
- Palabra clave ENDIF [ENDIF]
- Palabra clave ENDFOR [ENDFOR]
- Palabra clave ENDFUNC [ENDFUNC]
- Palabra clave CALL [CALL]
- Palabra clave IDENT [IDENT]
- Palabra clave EOFT [EOFT]
- Palabra clave ERROR [ERROR]

22 de Junio del 2018

Analizador Léxico

El analizador léxico consta de tres partes: tokenización de constantes, tokenización de operadores y tokenización de palabras clave (kwywords). Para generar un token, el programa toma una línea del código y la analiza carácter por carácter, cuando encuentra un espacio, se genera un token.

Tokenización de constantes

La parte de la tokenización de constantes se implementó usando un diseño de un autómata finito no-determinístico, el cual consta de 9 estados.

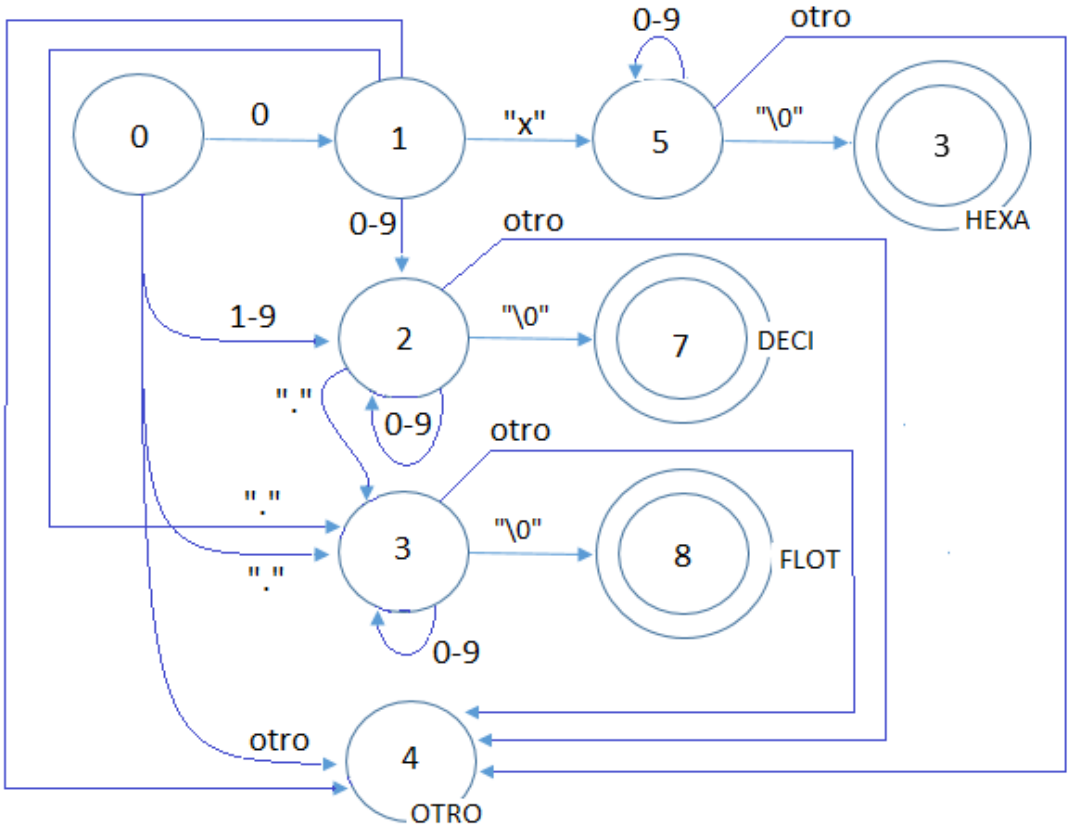


Figura 1. Autómata finite no-determinístico para la tokenización de constantes.

Tokenización de operadores

La parte de la tokenización de operadores se implementó usando un diseño de un autómata finito no-determinístico, el cual consta de 15 estados.

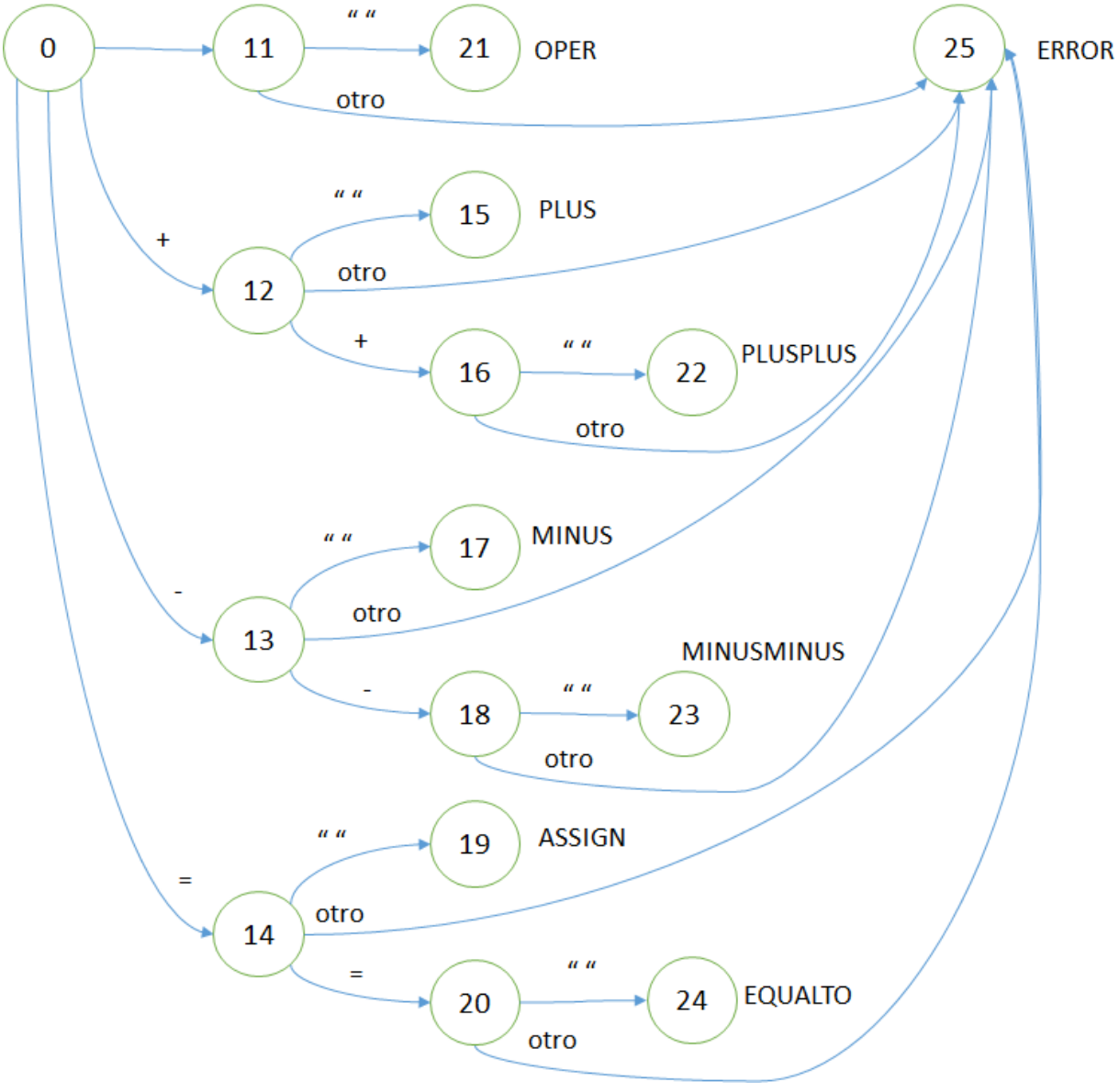


Figura 2. Autómata finito no-determinístico para la tokenización de operadores.

Tokenización de operadores

La parte de la tokenización de las palabras claves e identificadores se implementó usando un diseño de un autómata finito no-determinístico, el cual consta de 58 estados.

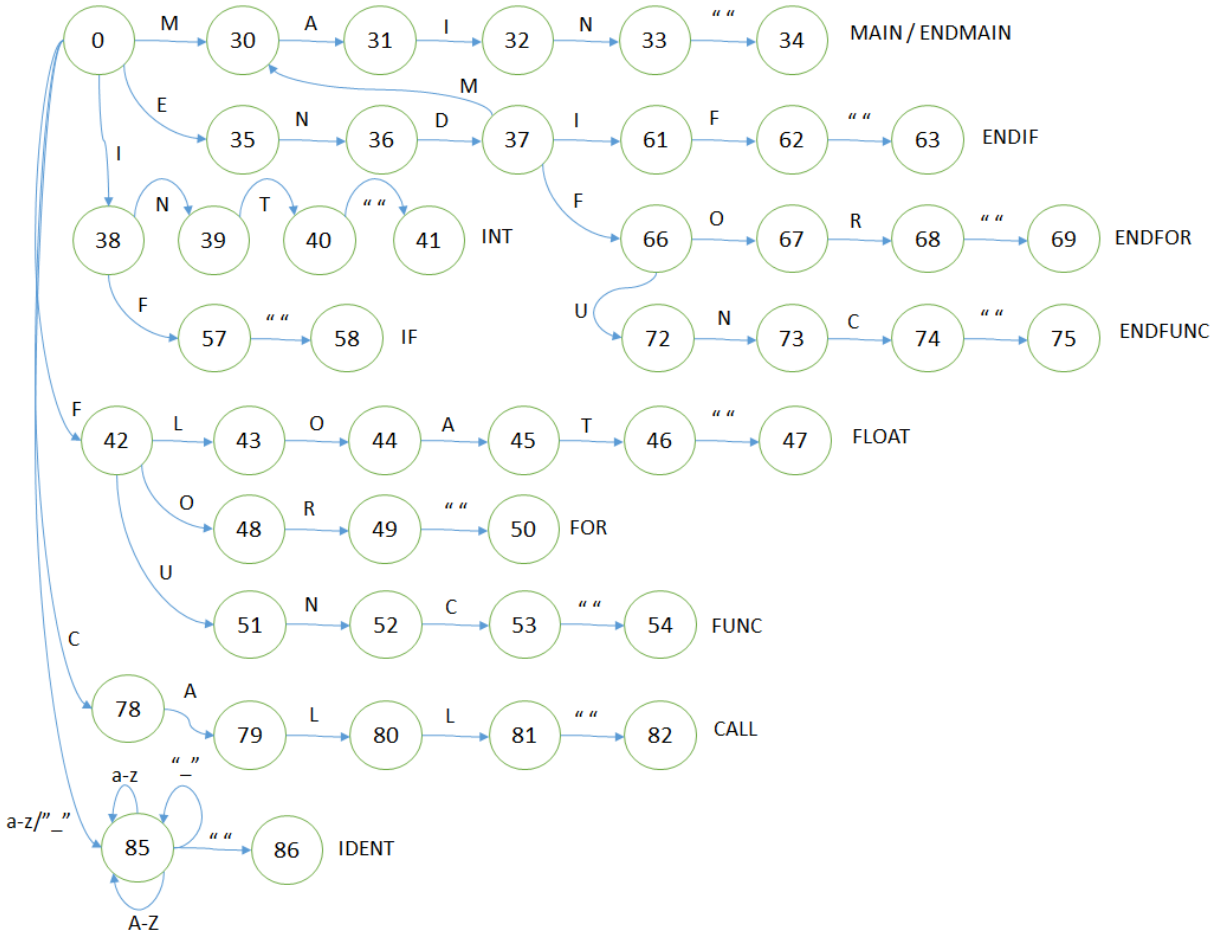


Figura 3. Autómata finito no-determinístico para la tokenización de palabras clave e identificadores.

22 de Junio del 2018

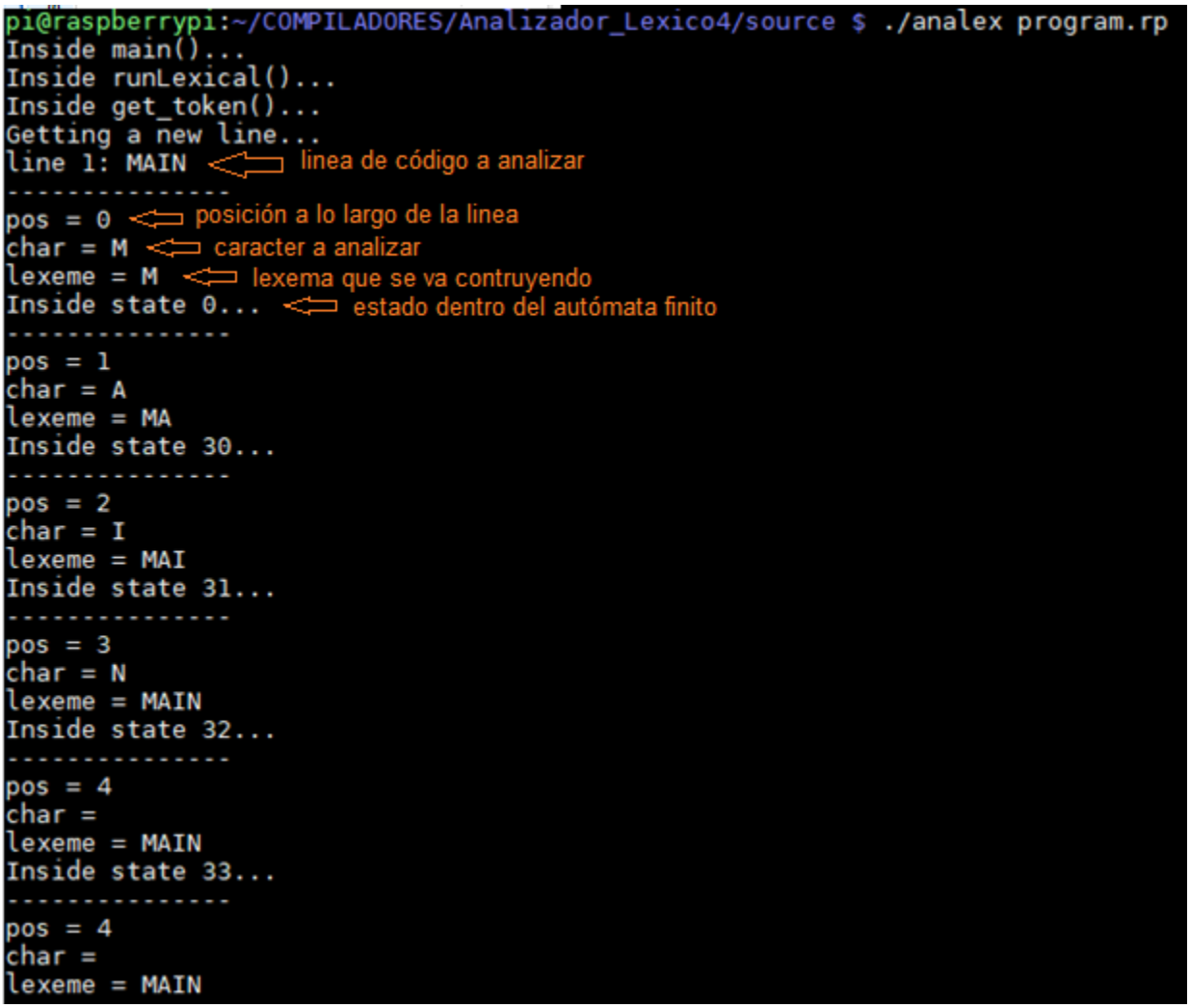
Pruebas del programa

Prueba 1. PASS

Código de entrada:

```
MAIN
   FLOAT var ;
ENDMAIN
```

El programa inicia con la palabra clave MAIN y termina con ENDMAIN. Declara una variable de tipo FLOAT llamada “var”. Las instrucciones del lenguaje están delimitadas con un punto y coma “;”. La corrida del programa es la siguiente:



```
Inside state 33...
---------------
pos = 4
char =
lexeme = MAIN
Inside state 34...
        MAIN : MAIN
Inside get_token()...
Getting a new line...
line 2:     FLOAT var ;
---------------
pos = 4
char = F
lexeme = F
Inside state 0...
---------------
pos = 5
char = L
lexeme = FL
Inside state 42...
---------------
pos = 6
char = O
lexeme = FLO
Inside state 43...
---------------
pos = 7
char = A
lexeme = FLOA
Inside state 44...
---------------
pos = 8
char = T
lexeme = FLOAT
Inside state 45...
---------------
pos = 9
char =
```

```
pos = 9
char =
lexeme = FLOAT
Inside state 46...
---------------
pos = 9
char =
lexeme = FLOAT
Inside state 47...
        FLOAT : FLOAT  token
Inside get_token()...
---------------
pos = 10
char = v
lexeme = v
Inside state 0...
---------------
pos = 11
char = a
lexeme = va
Inside state 85...
---------------
pos = 12
char = r
lexeme = var
Inside state 85...
---------------
pos = 13
char =
lexeme = var
Inside state 85...
---------------
pos = 13
char =
lexeme = var
Inside state 86...
        IDENT : var  token
Inside get_token()...
```

```
---------------
pos = 14
char = ;
lexeme = ;
Inside state 0...
---------------
pos = 15
char =
lexeme = ;
Inside state 11...
---------------
pos = 15
char =
lexeme = ;
Inside state 21...
       SEMICOLON : ;  token
Inside get_token()...
Getting a new line...
line 3: ENDMAIN
---------------
pos = 0
char = E
lexeme = E
Inside state 0...
---------------
pos = 1
char = N
lexeme = EN
Inside state 35...
---------------
pos = 2
char = D
lexeme = END
Inside state 36...
---------------
pos = 3
```

```
char = D
lexeme = END
Inside state 36...
---------------
pos = 3
char = M
lexeme = ENDM
Inside state 37...
---------------
pos = 4
char = A
lexeme = ENDMA
Inside state 30...
---------------
pos = 5
char = I
lexeme = ENDMAI
Inside state 31...
---------------
pos = 6
char = N
lexeme = ENDMAIN
Inside state 32...
---------------
pos = 7
char =
lexeme = ENDMAIN
Inside state 33...
---------------
pos = 7
char =
lexeme = ENDMAIN
Inside state 34...
       ENDMAIN : ENDMAIN   token
Inside get_token()...
Getting a new line...
End of file reached.
pi@raspberrypi:~/COMPILADORES/Analizador_Lexico4/source $
```

Prueba 2. FAIL

Código de entrada:

MAIN

FLOT var ;

ENDMAIN

El programa es el mismo que el anterior, excepto que el tipo usado para declarar la variable es FLOT envés de FLOAT, este es un tipo incorrecto y no lo reconoce el compilador, por lo tanto lanzará un error léxico para este token. Veamos los resultados de la corrida.

```
---------------
pos = 7
char = T
lexeme = FLOT
Inside state 44...
---------------
pos = 8
char =
lexeme = FLOT
CASE 100
Error at 2,8: Invalid character found: FLOT
        ERROR : FLOT
Inside get_token()...
---------------
pos = 9
char = v
lexeme = v
Inside state 0...
---------------
```

El programa tiene la capacidad para manejar errores. En esta imagen el programa reporta que hay un error en la línea 2, posición 8 y que tenemos un token inválido. Al final se reporta que la compilación falló en la fase del análisis léxico, como se muestra en la siguiente imagen.

```
---------------
pos = 7
char =
lexeme = ENDMAIN
Inside state 34...
        ENDMAIN : ENDMAIN
Inside get_token()...
Getting a new line...
End of file reached.
Compilation failed in Lexical Analizer.
pi@raspberrypi:~/COMPILADORES/Analizador_Lexico4/source $
```

Conclusión

Al principio del curso realmente me costó mucho trabajo entender el proceso de diseño para implementar un compilador de lenguajes de programación, los conceptos de analizador léxico, sintáctico y semántico, y las técnicas de diseño como los autómatas finitos y los arboles de parseo. Pero al final, después de ver cómo trabaja y lo que puede hacer mi diseño del analizador léxico, estoy muy sorprendido de los resultados, sobre todo del gran potencial que ofrecen los autómatas finitos para dar un nivel de inteligencia al programa. Me emosiona pensar en todas las posibilidades para aplicar esta técnica.

22 de Junio del 2018