

Seminario de Solución de Problemas de Traductores de Lenguaje II

Tarea 5:

Analizador Sintáctico Con Tabla

Nombre del maestro:

López Franco Michel Emanuel

Nombre del alumno:

Ramos Calderón Christian Daniel

Código del alumno:

216577014

Índice

Contents

Índice2

Introducción3

Desarrollo.....3

Capturas de pantalla5

Conclusión7

Bibliografía7

Introducción

Para esta actividad el maestro nos proporciono una tabla para el uso del analizador Sintáctico y de esta manera tomarla mediante un archivo de texto y tenemos también la matriz que de igual manera nos la dio el, mediante lo explicado en clase generar un programa a partir de lo proporcionado.

Desarrollo

Así que tenemos que desarrollar tanto como el analizador léxico como el analizador Sintáctico siendo posible que el programa pueda leer la siguiente gramática que son las reglas proporcionadas por el maestro.

Básicamente para el desarrollo del programa anteriormente utilizamos la clase EP con la cual manejamos los Terminales, No terminales y el Final de la cadena, de esta manera solo cambiamos el cómo están las reglas porque teníamos otras para una gramática en especial ahora con esto podríamos integrar cualquier tipo de gramática de un lenguaje C.

Las reglas proporcionadas por el maestro son las siguientes:

- 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

4 | Seminario de Solución de Problemas de Traductores de Lenguaje II

- 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 Terminio ::= LlamadaFunc
- R36 Terminio ::= identificador
- R37 Terminio ::= entero
- R38 Terminio ::= real
- R39 Terminio ::= 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 opIgualdad Expresion
- R50 Expresion ::= Expresion opAnd Expresion
- R51 Expresion ::= Expresion opOr Expresion
- R52 Expresion ::= Terminio

Capturas de pantalla

```

1  using System.Text;
2
3  namespace Analizador_Sintactico
4  {
5      3 referencias
6      public partial class Form1 : Form
7      {
8          1 referencia
9          public Form1()
10         {
11             InitializeComponent();
12         }
13
14         3 referencias
15         class Pila...
16
17         3 referencias
18         class AnalizadorLexico...
19
20         5 referencias
21         class Reglas...
22
23         12 referencias
24         class ElementoPila...
25
26         4 referencias
27         class Terminal...
28
29         5 referencias
30         class NoTerminal...
31
32         1 referencia
33         class Estado...
34
35         1 referencia
36         private void btnLimpiar_Click(object sender, EventArgs e)...
37
38         1 referencia
39         private void btnAnalizar_Click(object sender, EventArgs e)...
40
41         1 referencia
42         private void mostrarSintactico(List<String> lista, List<String> salida)...
43     }
44 }
    
```

Ingrese Expresión

```
int b;
int a;
x = a+b;
```

Analizador Léxico

```
int es un tipo de dato
b es un identificador
; es un punto y coma (;)
int es un tipo de dato
a es un identificador
; es un punto y coma (;)
x es un identificador
= es un operador de igual (=)
a es un identificador
+ es un operador de suma
b es un identificador
; es un punto y coma (;)
```

Errores

Pila	Salida
\$0int5b8	R7
\$0int5b8ListaVar9	D12
\$0int5b8ListaVa...	R6
\$0int5b8ListaVa...	D5
\$0int5b8ListaVa...	D8
\$0int5b8ListaVa...	R7
\$0int5b8ListaVa...	D12
\$0int5b8ListaVa...	R6
\$0int5b8ListaVa...	R2
\$0int5b8ListaVa...	R1
\$0programa1	R0
*	

6 | Seminario de Solución de Problemas de Traductores de Lenguaje II

Form1

Analizar

Limpiar

Ingrese Expresión

int a;
char b;

Analizador Léxico

int es un tipo de dato
a es un identificador
; es un punto y coma (;)
char es un identificador
b es un identificador
; es un punto y coma (;)
\$ es el signo \$

Errores

	Pila	Salida
►	\$0	D5
	\$0int5	D8
	\$0int5a8	R7
	\$0int5a8ListaVar9	D12
	\$0int5a8ListaVa...	R6
	\$0int5a8ListaVa...	R2
	\$0int5a8ListaVa...	R1
	\$0programa1	R0
*		

Form1

Analizar

Limpiar

Ingrese Expresión

float b;

Analizador Léxico

float es un tipo de dato
b es un identificador
; es un punto y coma (;)
\$ es el signo \$

Errores

	Pila	Salida
►	\$0	D5
	\$0float5	D8
	\$0float5b8	R7
	\$0float5b8Lista...	D12
	\$0float5b8Lista...	R6
	\$0DefVar4	R4
	\$0Definicion3	R2
	\$0Definicion3D...	R3
	\$0Definiciones2	R1
	\$0programa1	R0
*		

Conclusión

La realidad con esta actividad es que fue un poco compleja al momento de como tomar las reglas y la matriz pero una vez resolviendo eso la actividad fluyo por si sola, desde mi punto de vista aun que realmente es un poco tediosa pero sirve para darnos cuenta de todo el proceso que hace un compilador.

Bibliografía

No valida en este trabajo