

Seminario de Solución de Problemas de Traductores de Lenguaje II

Tarea 3:

Analizador Léxico Completo

Nombre del maestro:

López Franco Michel Emanuel

Nombre del alumno:

Ramos Calderón Christian Daniel

Código del alumno:

216577014

Índice

Contents

Índice2

Introducción3

Desarrollo.....4

Capturas de pantalla4

Conclusión6

Bibliografía6

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

Introducción

Durante esta actividad pasamos a la práctica del analizador léxico después de generar el mini analizador pasaremos a generar el completo el cual va a determinar los siguientes campos:

Símbolo	Tipo
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

Desarrollo

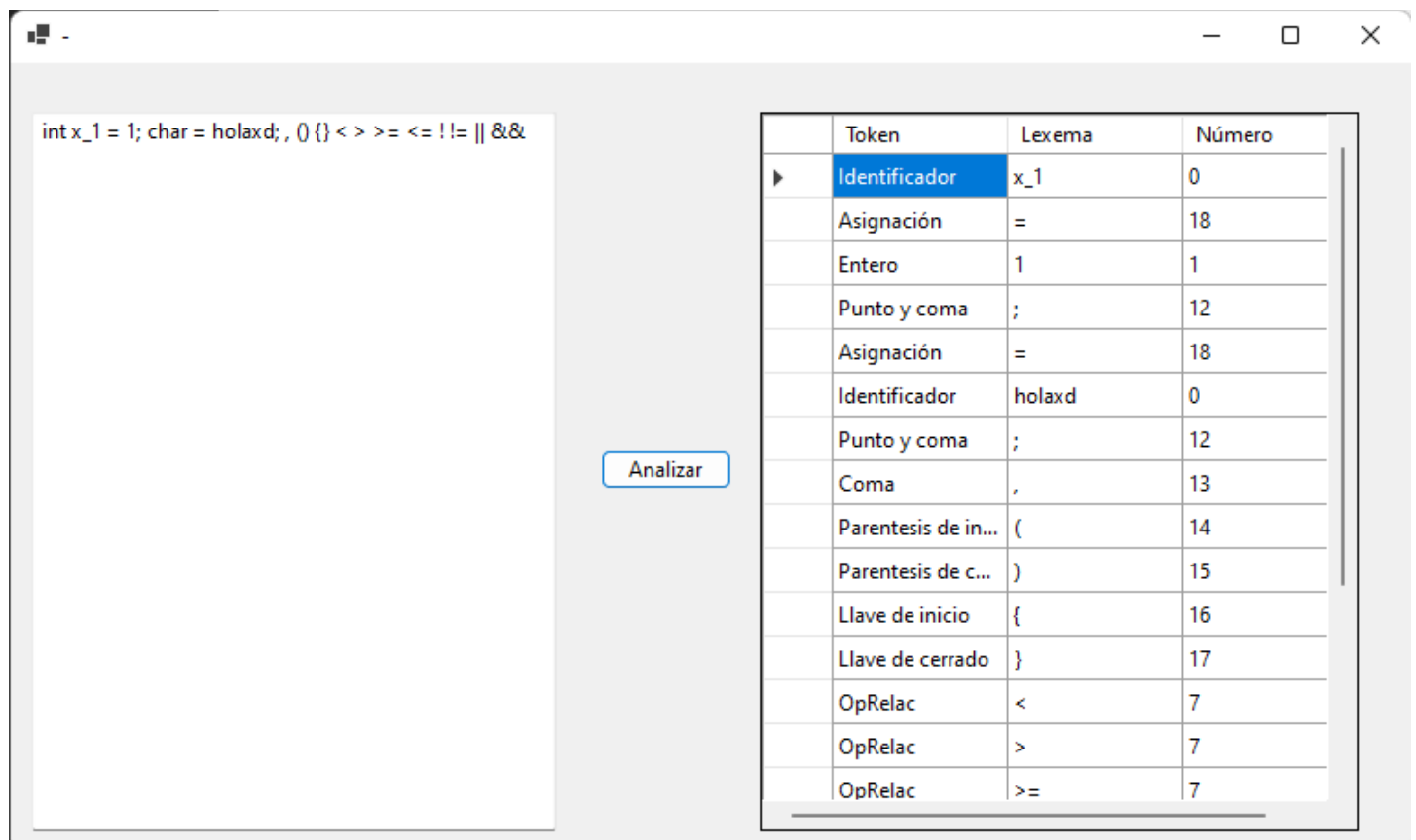
Durante esta práctica modifique mi código de tal manera que fuera más fácil de digerir, de tal manera que al evaluar el carácter individualmente entrando a un estado puedo evaluar el siguiente para corroborar que símbolo es así para no tener errores con los símbolos de igualdad o los operadores And o el Or.

Utilizo una función para encontrar las palabras reservadas como son while, if, else, return y de esta manera secciono el código así permitiendo revisarlo paso a paso, como lo mencione anteriormente el código es simple voy recorriendo carácter por carácter hasta encontrar un espacio y así seccionar la palabra que se guardó, mediante if puedo ir evaluando el carácter de esta manera encontrar los símbolos para asignar la palabra guardada y por último todo esto se muestra en un dataview.

NOTA:

No muestro el código de tal manera que se encuentra en el repositorio de github.

Capturas de pantalla



The screenshot shows a software interface for code analysis. On the left, a text area contains the code snippet: `int x_1 = 1; char = holaxd; , () { } < > >= <= != || &&`. Below the text area is a button labeled "Analizar". On the right, a table displays the results of the analysis, listing tokens, their corresponding lexemes, and their assigned numbers.

	Token	Lexema	Número
▶	Identificador	x_1	0
	Asignación	=	18
	Entero	1	1
	Punto y coma	;	12
	Asignación	=	18
	Identificador	holaxd	0
	Punto y coma	;	12
	Coma	,	13
	Parentesis de in...	(14
	Parentesis de c...)	15
	Llave de inicio	{	16
	Llave de cerrado	}	17
	OpRelac	<	7
	OpRelac	>	7
	OpRelac	>=	7

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

```
int x_1 = 1; char = holaxd; , () {} < > >= <= != || &&
```

Analizar

	Token	Lexema	Número
	Coma	,	13
	Parentesis de in...	(14
	Parentesis de c...)	15
	Llave de inicio	{	16
	Llave de cerrado	}	17
	OpRelac	<	7
	OpRelac	>	7
	OpRelac	>=	7
	OpRelac	<=	7
	OpNot	!	10
	OpRelac	!=	7
	OpOr		8
	OpAnd	&&	9
*			

```
if while return else
```

Analizar

	Token	Lexema	Número
►	condicional SI	if	19
	Ciclo	while	20
	Retornar valor	return	21
	Sino	else	22
*			

Conclusión

Con esta actividad terminaremos de ver lo complejo que es realmente el analizador y todas las partes que lo integran, con partes me refiero a todas las condicionales por las cuales tiene que pasar para evaluar todas y cada una de las palabras reservadas tanto como los valores posibles y si tiene un error lo detecta.

Básicamente es una practica para ver su funcionamiento y ver desde otro punto lo que pasa dentro de los compiladores como un proceso de dos que primero es el analizador léxico y después pasa al sintáctico.

Bibliografía

No valida en este trabajo