Memoria Final

Entrega parcial

Analizador lenguaje BSL

Realizado por:
Marcelino Tena Blanco; NIA: 100383266

Índice

1. Introducción

El siguiente documento trata sobre la realización y el procedimiento para llegar a la creación de un analizador léxico para el lenguaje BSL, el cual está especificado su formato en aula global. Para explicar el proceso llevado para crear el analizador léxico, iré comentándolo sus estructuras según el siguiente orden:

- Los dos tipos de comentarios
- Operadores Aritmeticológicos
- Declaración de variables e inicialización de variables
- Declaración de funciones
- Bucles y sentencias
- Estructuras structs

Una vez que realicé el analizador léxico, he estado haciendo parte del analizador sintáctico pero todavía no es funcional, ya que faltan bastantes estructuras. Por ahora reconoce hasta declaración de variables e inicialización, aunque no hace nada con ellas, pero puede reconocer su estructura y saber si está bien o no.

Para acabar con la introducción, hay que añadir que se han implementado los ejemplos dados en el enunciado de la práctica. Conozco que puede ser que me deje algunas estructuras más explícitas sin comprobar, pero por varias circunstancias no he podido agregar más. El programa principal de esta entrega parcial es AnalizadorLexico, el cual es el programa dado en el enunciado para comprobar el correcto funcionamiento del analizador léxico sin tener que ejecutar el parser del cup.

2. Construcción del analizador léxico

A continuación iré estructura por estructura detallando cada una de las partes compuestas por el analizador léxico:

2.1 Comentarios

Para los comentarios debemos tener en claro que existen dos formas para realizarlos, una es mediante el símbolo # que marca como comentario todo lo que queda de línea, y la otra es mediante un símbolo de abrir comentario, que es un conjunto de símbolos (<!--) y un símbolo de cerrar comentario (- ->). Para que el analizador léxico sea capaz de reconocer tanto uno como otro, tenemos una expresión regular que tiene dos opciones que son dos expresiones regulares, donde una es la que permite que sea comentario todo lo que falte de línea a partir del símbolo # y el otro solo comprende que es comentario lo que hay dentro de los símbolos antes dichos de abierto y cerrado.

2.2 Operadores Aritmeticológicos

En el caso para que el analizador reconozca los símbolos aritmeticológicos lo único que he realizado es agregarlos cada uno como tokens, ya que no es necesario hacer otra cosa. En el caso de, por ejemplo, exista un mayor igual o un similar, será el analizador sintáctico quien tendrá la capacidad de reconocer este conjunto de símbolos, pero en el reconocedor léxico solo los añadimos como nuevos tokens por separado. Los tokens añadidos en este caso son:

- Símbolos aritméticos
  - “+”, “-“, “/”, “*”, “=”
- Símbolos lógicos
  - “<”, “>”, “NOT”, “AND”, “OR” , “!”, “&”, “|”

2.3 Declaración de variables e inicialización

Para esta parte debemos comprender que existen dos estructuras:

- Declaración de variable:

Una variable se declara cuando tiene delante un tipo y después un nombre, es decir, el identificador de la variable. Para reconocer el tipo se usa las palabras claves: ENTERO, REAL, BULEANO y CARACTER (puede estar indistintamente en mayúsculas y minúsculas ya que el analizador léxico no discrimina entre ellas), donde cada una especifica un tipo. Después, para reconocer el identificador es necesario saber que solo puede empezar por una letra y solo puede contener símbolos alfanuméricos. Para ello, lo que he hecho es hacer una expresión regular capaz de detectar los identificadores. Se ha agregado también el símbolo “,”, en el caso de que se quieran declarar varias variables.

- Inicializar una variable

Para inicializar una variable debemos tener en cuenta que se necesita los símbolos “:”, “=” y una expresión matemática, buleana o un carácter. Por lo que lo que reconoce el analizador es: los dos puntos por una parte, el igual por otra, y la expresión se reconoce mediante los operadores aritméticos y según si los números están escritos correctamente. En este caso, los números pueden ser enteros, reales, exponenciales en base diez y hexadecimales. Todos los números se devuelven al analizador sintáctico como double, donde será necesario en posteriores trabajos reconvertirlo en el caso de que se quiera almacenar como entero el número o como double. En el caso de los caracteres, se identifica que es un carácter si está entre comillas simples y se devuelve solo el carácter sin las comillas y por el caso de los buleanos se devuelve si es TRUE o FALSE (se tiene en cuenta en todas las palabras reservadas pueden ir en mayúsculas o minúsculas).

2.4 Declaración de funciones

El analizador léxico solo detecta los tokens, por lo que detecta las palabras: FUNCION, RETURN (como palabras reservadas) y los símbolos “{“,”}”,”(“ y ”)”.

2.5 Bucles y sentencias

Para los bucles y las sentencias se han agregado las palabras reservadas: SI, SINO, FINSI, MIENTRAS, ENTONCES y FINMIENTRAS.

2.6 Estructuras structs

Se analiza la palabra reservada STRUCTS, además que contiene los paréntesis ya añadidos antes y las declaraciones que también están añadidos.

3. Conclusión

Realizar el analizador léxico ha sido sencillo y no me ha supuesto ningún problema, aunque no sé si voy a ser capaz de poder realizar el analizador sintáctico porque parece bastante complejo. Por ahora detecta declaración de variables e inicialización de estas, pero no funciona de la mejor forma. Por parte de las pruebas, tendría que haber agregado algunas más, pero no he tenido el tiempo suficiente para poder examinar más a fondo si mi analizador léxico es correcto o tiene alguna falla que no se observa a simple vista.