Compiladores Proyecto Final

Guerrero Chávez Diana Lucía
Sánchez del Moral Adriana
Gallardo Valdez José Jhovan

Junio 2019

1 Análisis Léxico

1.1 Introducción

Para este curso crearemos nuestro propio compilador que procese un lenguaje definido por la gramática dada en la especificación del proyecto. En especıfico para esta primera entrega (Analizador Léxico) se necesitan de- finir las reglas de la gramática y construir un analizador léxico utilizando el lenguaje de programación lex y yacc.

1.2 Análisis del problema

El analizador léxico debe recibir como entrada un conjunto de tokens desde un archivo de texto plano, consumir todos los tokens del archivo y como salida indicar si pertenecen o no a la gramática.

Dada la gramática anterior, se deben realizar ciertas modificaciones respecto a las palabras reservadas para evitar conflictos del lenguaje en el que programemos nuestro analizador léxico.

1.2.1 Expresiones regulares de los componentes léxicos

Para las palabras reservadas (int, float, double, char, void etc) se utilizará su traducción abreviada al español.

A continuacion se muestran los nombres de las palabras reservadas utilizadas en el programa:

- ent - Entero (int)
- flot - Flotante (float)
- doble - Doble (double)
- car - Caracter (char)

- vacio - Vacío (void)
- estruct - Estructura (struct)
- si - if
- sino - else
- mientras - while
- hacer - do
- retornar - return
- cambiar - switch
- romper - break
- imprimir - print
- caso - case
- defecto - default
- verdadero - true
- falso - false

1.3 Diseño e implementación

El lexer, devuelve el valor de un token encontrado, el cual será utilizado posteriormente por el analizador sintáctico.

Compilar con:
flex analizadorLexico.lex
gcc lex.yy.c -o analizadorLexico -lfl

Ejecutar con:
./analizadorLexico programita.txt

2 Análisis Sintáctico

EL analizador Sintáctico recibe tokens provenientes del analizador léxico y devuelve como resultado un árbol sintáctico.

La construcción del árbol sintáctico se realiza a través de una tabla de simbolos.

2.1 Descripción del problema

Debemos identificar cada token y su precedencia para la creación del árbol con el fin de hacer un análisis y validar que se cumplan las reglas de la gramática.

2.2 Análisis del problema

Se necesita elegir una estructura eficiente y sencilla para la tabla de simbolos. Para nuestro objetivo la estructura dee árbol es la que mejor se adapta a nustro proyecto.

2.3 Eliminar la ambigüedad de la gramática

PENDIENTE: tabla de gramática reducida

2.4 Diseño e implementación

En el archivo "analizadorSintáctico.lex" se realizó una losta de todos los tokens que se pueden recibir como entrada por parte del analizador léxico.

Por cada uno de estos, se definieron reglas