Diseño de Compiladores

Trabajo Práctico de Cursada – Primera Entrega

Grupo Número 14

Bianco, Martin ([martinbianco995@gmail.com](mailto:martinbianco995@gmail.com))

Di Pietro, Esteban ([dipietroesteban@gmail.com](mailto:dipietroesteban@gmail.com))

Serrano, Francisco ([francisco.serrano372@gmail.com](mailto:francisco.serrano372@gmail.com))

Ayudante asignado: Antonela Tommasel

# Introducción

En el siguiente informe se describirán las acciones llevadas a cabo para implementar un analizador léxico con determinadas especificaciones definidas por la cátedra. Codificamos un Lexer. Esto permite reconocer los tokens que son: identificadores, constantes, operadores, palabras reservadas, entre otros. Adicionalmente, el Lexer realizará un informe de los errores léxicos detectados a lo largo del código fuente, con el número de línea asociado a cada token, junto con la descripción asociada a los errores, así también como los contenidos presentes en la tabla de símbolos.

Con respecto a la segunda parte del trabajo, se especificarán las acciones realizadas para construir un Parser que trabaje junto al Analizador Léxico implementado durante la primera parte del trabajo. Dicho Parser reconoce un lenguaje que incluye un programa conformado por un conjunto de sentencias, que pueden ser declarativas o ejecutables. Para el desarrollo del mismo se utilizó la herramienta YACC, en su versión para Java. Adicionalmente se realizaron adaptaciones del Lexer a fin de compatibilizar la salida de Yacc con el trabajo previamente realizado.

# Temas Particulares Asignados

A continuación, se detallan los temas asignados por la cátedra, junto con la identificación de cada uno.

2. Enteros sin signo: agregar como palabra reservada UINT.

4. Enteros largos sin signo: agregar como palabra reservada ULONG.

7. Incorporar a la lista de palabras reservadas las palabras WHILE y DO.

13. Incorporar a la lista de palabras reservadas las palabras FUNCTION, RETURN y MOVE.

16. Se definirá en el trabajo práctico 3.

18. Comentarios multilínea: Comentarios que comienzan con “[” y terminan con “]” (estos comentarios pueden ocupar más de una línea).

19. Cadenas de 1 línea: Cadenas de caracteres que comiencen y terminen con “ ‘ ” (estas cadenas no pueden ocupar más de una línea).

# Decisiones de Diseño e Implementación

Para llevar a cabo la realización de este trabajo decidimos escoger el lenguaje de programación Java. Inicialmente disponemos una clase llamada main donde se definen los nombres de los archivos que almacenan dos matrices correspondientes al autómata hecho previamente, que indican las acciones semánticas y los estados para reconocer o no un token. También se crea al analizador léxico que va a operar sobre un archivo de entrada específico.

En la clase Analizador Léxico se realiza una lectura caracter por caracter del archivo de entrada y se van insertando en dos listas. Una de ellas es simplemente el archivo, mientras que en la segunda se inserta el tipo de carácter que se leyó (letra, dígito, salto de línea, parentesis, etc). Además, durante la lectura se lleva un contador de numero de lineas.

En esta clase, las matrices de estados y de acciones semánticas nos proporcionan información acerca de los estados y acciones semánticas que se deben ir tomando dependiendo solamente del carácter leído y del estado.

Se definió la función yylex que a medida que se solicitan los tokens, estos se van calculando y retornando.

También se desarrolló una clase TablaSimbolos donde se almacenan y se agregan las cadenas de caracteres, identificadores y constantes. Se trata de una lista con las palabras reservadas y un MultiMap en el cual se van incorporando dinámicamente los identificadores, constantes y cadenas con sus respectivos lexemas. Esta clase tiene métodos para agregar, buscar y devolver los lexemas que se necesiten.

# Implementación de Matrices (estados y acciones semánticas)

# Descripción de Acciones Semánticas

# Gramática del Lenguaje

# Construcción del Parser

# Descripción de los Errores Considerados

# Casos de Prueba

# Conclusión