Diseño de Compiladores

Cursada 2014

N° DE GRUPO: 20

INTEGRANTES: Julieta González Poume – [julipoume@gmail.com](mailto:julipoume@gmail.com)

Eliana Mira – [elitaamira@gmail.com](mailto:elitaamira@gmail.com)

AYUDANTE: Dr. José Massa.

Introducción

La cátedra solicito la realización un analizador léxico que reconozca diferentes tokens leídos a partir de un archivo de código fuente, también se deben reconocer comentarios, caracteres especiales, errores léxicos y los temas particulares que nos fueron designados.

Para llevar a cabo la implementación realizamos un diagrama de transición de estados el cual nos facilito la definición de los cambio de estados y acciones semánticas que se producen a lo largo del análisis.

Este analizador será parte de trabajos posteriores para poder realizar un compilador del lenguaje especificado por la cátedra, por lo que debemos tener en cuenta que sea fácil de integrarse a las especificaciones futuras.

Tokens a reconocer

El analizador léxico debe reconocer los siguientes tokens:

• Identificadores cuyos nombres pueden tener hasta 12 caracteres de longitud. El primer carácter debe ser una letra, el resto pueden ser letras o dígitos. Los identificadores con longitud mayor serán truncados y esto se informará como Warning.

• Operadores aritméticos: “+”, “-” , “\*”, “/”

• Operador “:=” para asignación.

• Operadores relacionales: “>=”, “<=”, “>”, “<”, “=”, “^[”

• “{” , “}”, “[”, “]”, “(”, “)”, “,”, y “;”

• Palabras reservadas: if then else print int

Temas Particulares

* **Flotantes**: Números reales con signo y parte exponencial. El exponente comienza con la letra F (mayúscula o minúscula) y puede tener signo. La ausencia de signo implica positivo. La parte exponencial puede estar ausente. Considerar el rango 1.17549435 f-38 < |x| < 3.40282347 f38. Se debe incorporar a la lista de palabras reservadas la palabra **flotante**.
* **Iteración 3**: Incorporar a la lista de palabras reservadas la palabra **para.**
* **Comentarios multilínea**: Comentarios que comiencen con “ [- ” y terminen con “-]” (estos comentarios pueden ocupar más de una línea).
* **Cadenas de 1 línea**: Cadenas de caracteres que comiencen y terminen con “ “ ” (estas cadenas no pueden ocupar más de una línea).
* **Incorporar** a la lista de palabras reservadas las palabras **vector** y **de**, y **reconocer** los caracteres “[“ y “]”.

Decisiones de Diseño e Implementación

Para llevar a cabo la implementación del Compilador, decidimos utilizar Java como lenguaje de programación, porque consideramos los beneficios de los lenguajes de programación orientados a objetos.

El proyecto está dividido en dos paquetes: dataModel y accionesSemanticas.

En paquete dataModel, contendrá las clases relacionadas con el compilador: Compilador, Analizador Léxico y aquellas que representen las estructuras utilizadas, como: Matriz, Tabla de Símbolos, y Palabras Reservadas.

El paquete accionesSemanticas, contiene todas las Acciones Semánticas utilizadas por el compilador.

Compilador

Contendrá a las diferentes etapas de la compilación: Análisis Léxico y Análisis Sintáctico. Contiene las estructuras que se utilizaran en diferentes etapas, como la Tabla de Símbolos.

Es el encagardo de instanciar la clase Parser, y desde esta clase, Parser ejecuta su función “run”, que realizara la llamada a la función yyparse().

Diagrama de transición de estados

Analizador Léxico

Incluido en la primer etapa de la compilación, recibe como entrada un código fuente (lee los caracteres del código fuente) y produce una salida de tokens. Estos tokens serán utilizados en una etapa posterior, siendo la entrada para el analizador sintáctico.

Se encarga de completar la Tabla de Símbolos con los identificadores, constantes y cadenas de caracteres encontradas en el código.

Al crear una instancia de esta clase, se inicializan: la Tabla de Palabras

Reservadas y dos matrices estáticas: una contiene la transición de estados y la otra las funciones que implementan las acciones semánticas. Dichas tablas se obtuvieron a partir del Autómata de Transición de Estados creado previamente, cada vez que se lee un carácter, se obtiene la columna de la matriz que corresponde al mismo. Con este valor y el estado actual, se ejecuta el proceso correspondiente y se obtiene el próximo estado; si este corresponde al estado final, significa que se encontró un token.

PONER MAS FRUTA DE CÓMO FUNCIONA PERO NO ME ACUERO

Matrices

Acciones Semánticas

AccionSemantica0: Su ejecución es la encargada de informar si el carácter leído no es válido, agregando un mensaje de error en el archivo errores\_lexico.log.

AccionSemantica1: Se encarga de guardar en el stringBuffer los caracteres que se van leyendo.

AccionSemantica2: Esta acción semántica se encarga de buscar el lexema leído en la tabla de Palabras Reservadas. También verifica que el lexema no exceda la longitud máxima permitida que debe poseer un Identificador. Si cumple con la longitud, se agrega en la Tabla de Símbolos.

AccionSemantica3: Esta acción semántica se ejecuta cuando se lee un número entero. Verifica el rango y en caso de no cumplirse la condición, retorna un error.

AccionSemantica4: Es la encargada de realizar la verificación cuando se encuentra un numero de tipo Float. Verifica que este dentro del rango, de no ser así, lo informa con un mensaje de error.

AccionSemantica5: Esta acción retorna los lexemas para los símbolos que son Comparadores y el de Asignación (<=, >=, !=, :=).

AccionSemantica6: Guarda el lexema y retorna el ultimo carácter leído. Además retrocede una posición en el Lector, para continuar con la lectura.

AccionSemantica7: Al ejecutarse guarda el símbolo leído ({, }, +, -, etc) y lo retorna.

AccionSemantica8: La acción numero 8 elimina el lexema leído que fue formado por un comentario.

AccionSemantica9: Cuando se lee un carácter espacio en blanco, esta acción se ejecuta y retorna un null.

AccionSemantica11: La acción guarda el lexema VECTOR.

AccionSemantica13: Se encarga de guardar las cadenas de una línea, verifica que esta no tenga más de una línea, en caso de ser así retorna un error, indicando que la cadena no cumple con la longitud máxima.

chauuu

