PRUEBA DE DOCUMENTO

**Imagen que contiene Logotipo

Descripción generada automáticamenteUna caricatura de una persona

Descripción generada automáticamente con confianza baja UNIVERSIDAD AUTÓNOMA DE CAMPECHE**

*Facultad de ingeniería*

*Ingeniería en Sistemas Computacionales*

*Unidad de aprendizaje:*

**COMPILADORES**

*Semestre y grupo:*

**7MO SEMESTRE, GRUPO B**

*Alumnos:*

**EDSON CRISTOBAL COJ UC**

*Docente:*

**JOSE A. AGUILAR CANEPA**

*Actividad:*

**PRACTICA 3: Análisis semántico**

*Fecha de entrega:*

**SAN FRANCISCO DE CAMPECHE A 19 DE NOVIEMBRE DE 2023**

**Primero creamos el archivo .flex:**

Texto

Descripción generada automáticamente

En donde:

**Definición de Tokens:**

TerminadorLinea: Define los caracteres de terminación de línea.

EspacioBlanco: Define los espacios en blanco, que incluyen caracteres de terminación de línea, tabulaciones, y otros espacios en blanco.

Numero: Define un número, que puede ser entero o decimal, positivo o negativo.

Variable: Define una variable, que comienza con una letra y puede contener letras, números y guiones bajos.

Cadena: Define una cadena de caracteres, delimitada por comillas dobles, que puede contener caracteres regulares o secuencias de escape.

**Código Java en las Acciones:**

%{ ... %}: Contiene código Java que será insertado en la sección de definición de la clase generada por Java Cup.

Se definen dos métodos getToken que crean instancias de la clase Symbol con información sobre el tipo, la línea, la columna y, en el segundo caso, un valor asociado al token.

Reglas de Tokens:

Dentro de la sección %%, se definen las reglas para reconocer los diferentes tokens. Por ejemplo:

"numero" y "cadena" son palabras clave que se corresponden con tokens específicos.

";", "<-", "->", "\*", y "+"` son operadores o separadores que también se corresponden con tokens específicos.

{ Variable }, { Cadena }, y { Numero } son patrones que coinciden con las definiciones previas de variables, cadenas y números.

**Acciones asociadas a los Tokens:**

Después de cada patrón, hay una acción en Java que se ejecuta cuando se reconoce el patrón correspondiente. Por ejemplo, return getToken(sym.NUMERO, Integer.parseInt(yytext())); devuelve un token de número con el valor convertido a un entero.

**Manejo de Caracteres no Reconocidos:**

La sección [^] maneja caracteres que no coinciden con ninguna de las reglas anteriores y lanza una excepción indicando la línea, la columna y el carácter no reconocido.

Ahora, en la creación del archivo CUP:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En donde:

**Acciones Globales:**

En la sección action code, se declara una Hashtable llamada tablaDeSimbolos que se utiliza como una tabla de símbolos para almacenar variables y sus valores.

**Métodos de Manejo de Errores:**

Se proporcionan los métodos report\_error y report\_fatal\_error para manejar errores durante el análisis sintáctico. Estos métodos imprimen mensajes de error que incluyen información sobre la ubicación del error.

**Terminales y No Terminales:**

Se definen terminales (tokens) como CADENA\_ID, NUMERO\_ID, SEPARADOR, OP\_ASIGNACION, OP\_IMPRESION, PLUS, TIMES, CADENA, NUMERO, y VARIABLE.

También se definen no terminales como Principal, programa, nada, listaExp, expr, dec\_string, dec\_entero, expr\_asignacion, expr\_impresion, expr\_suma, y expr\_multi.

**Reglas de Producción:**

start with programa; establece que la producción inicial es programa.

Se definen reglas de producción que describen cómo se pueden construir diferentes partes del programa a partir de otras partes. Por ejemplo:

programa ::= listaExp; indica que un programa es una lista de expresiones.

listaExp ::= listaExp SEPARADOR expr | expr; indica que una lista de expresiones puede ser la concatenación de otra lista con una expresión separada por un separador, o simplemente una expresión.

**Acciones asociadas a las Producciones:**

En las reglas de producción, se incluyen acciones escritas en Java que se ejecutan cuando se reduce la producción. Estas acciones manipulan la tabla de símbolos y realizan operaciones específicas según la construcción sintáctica encontrada.

**Expresiones Aritméticas y de Impresión:**

Se definen expresiones para la asignación, impresión, suma y multiplicación. Cada una de estas expresiones incluye acciones específicas que operan en la tabla de símbolos.

**Manejo de Errores:**

Se incluyen mensajes de error en las acciones cuando se detectan situaciones incorrectas. Por ejemplo, se imprime un mensaje de error si se intenta concatenar cadenas idénticas.

**PARA LA EJECUCIÓN:**

Utilizamos Lexer.flex Lexer.cup y Principal.java.

Generamos el lexer.java con el jflex.

Y posteriormente utilizamos los siguientes comandos (en este orden):

**java -cp ".;java-cup-11b.jar" java\_cup.Main < Practica3.cup**

**javac -cp ".;java-cup-11b.jar" Principal.java**

**java -cp ".;java-cup-11b-runtime.jar" Principal codigoPrueba.txt**

**Texto

Descripción generada automáticamente**