Teoría de la Computación

Práctica 2. Analizador léxico.

Objetivo: Realizar la implementación de AFD´s que aceptan lenguajes regulares, y aplicarlos para el diseño e implementación de un analizador léxico. Cualquier lenguaje de programación puede ser utilizado (C, C++, C#, Java, Python).

Desarrollo:

Parte I. Diseñar e implementar un Analizador léxico que, dado un programa fuente escrito en lenguaje Java, identifique los valores numéricos (constantes) en sus diferentes tipos (int, float, double) y notaciones.

Tipo	Notación	Representación	Caracter (n)
Entero	Decimal	[+,-] nnn…n	0…9
	Octal	[+,-] 0nnn…n	0…7
	Hexadecimal	[+,-] 0xnnn…n	0…F
Real	Decimal. s/exp	[+,-] n…n.n…n	0…9
	Decimal. c/exp	[+,-] n.n…nE[+,-]nn	0…9

Tabla 1. Representación de constantes numéricas en Java

Para resolver el problema se sugiere construir un AFD que opere los símbolos especiales definidos en la tabla 2, y con tantos estados de aceptación como las diferentes formas numéricas que deben ser reconocidas (aceptadas), que en este caso son cinco: tres formas enteras y dos reales (tabla 3).

Simbología especial	
Símbolo	Significado
b	blanco, <cr>, <lf>,<tab>
;	fin de sentencia
.	punto decimal
+, -	signos de valor numérico
β	Cualquier símbolo diferente de: 0…9, +, -

Tabla 2. Símbolos especiales reconocidos

Estados de aceptación	
Símbolo	Significado
C0	Entero decimal
C1	Entero hexadecimal
C2	Entero octal
C3	Real sin exponente
C4	Real con exponente

Tabla 3. Estados de aceptación.

Partiendo de nuestra definición formal de AFD:

$$AFD = (\Sigma, Q, q_0, \delta, F)$$

Tenemos que:

$$\Sigma = \{ 0, \dots , F, +, - , b , ; , . , E, x \}$$
$$F = \{ C0, C1, C2, C3, C4, C5 \}$$

Encontrar: Q, q_0, δ (o el diagrama de transición).

Entrada al Analizador léxico: La entrada al programa será un archivo de texto con la extensión .java el cual contiene un programa fuente en Java en donde tenemos constantes numéricas.
Por ejemplo:

```
8   public class EjemploPracticaAnalizador {
9       public static void main(String[] args) {
            int octal1 = -0123, octal2 = 0381;
            short dato = 12A12;
12          double PI = 3.1416, CteGrav = -6.674E-19;
13          float prom = (float) 0.0;
14          double val;
15          /* Calculos Generales */
16          for ( int i = 1; i < 100 ; i ++ ){
17              prom += i;
18          }
            double pot = 7.34E+1.4;
20          val = CteGrav * dato * pot;
            System.out.println ( "Prom = " + ( prom / PI ) + "Result = " + (val * 0xAxB) );
22          }
23  }
```

Programa **EjemploPracticaAnalizador.java** usado como entrada al Analizador léxico.

Salida del analizador léxico: La salida del programa serán mensajes de texto que indiquen si hay algún error en el uso de una constante numérica desde el punto de vista léxico.
La salida del programa, teniendo como entrada el archivo fuente anterior, es la siguiente:

Error en línea 10.
Error en línea 11.
Error en línea 19.
Error en línea 21.

Si el archivo **EjemploPracticaAnalizador.java** no tuviera ningún error la salida del Analizador sería:

No hay errores de análisis léxico en el archivo EjemploPracticaAnalizador.java.

Nota: No se aceptarán programas que no usen como entrada un archivo de código fuente java y una salida en pantalla tal como se indica arriba.

Parte 2. Agregar los siguientes estados de aceptación al AFD del analizador léxico:

Estados de aceptación (Parte II)		
Símbolo	**Significado**	**Ejemplos**
C5	Identificador válido en Java	*var1, dato, prom_final, info,* etc.
C6	Comentarios	Para una sola línea: // Para más de una línea: /* … */
C7	Palabras reservadas de Java	*class, main, public, private, void, int*, etc.

Tabla 4. Estados de aceptación Parte II.

Estados de aceptación (Parte III) Manejo de Sintaxis		
Símbolo	**Significado**	**Ejemplos**
C8	Símbolos especiales I	Operadores de comparación (<,>,=, >=, <=, ==, !=) y operador de asignación (=). Cadenas válidas: var1 < var2 dato == 3.14 190 > 23 valor = 89 Cadenas no válidas: int = 12 20 = 34 public > 100 class = 85.23
C9	Símbolos especiales II	Operadores aritméticos (+, -, *, /. %) Cadenas válidas temp1 * 8.23 var12 + temp5 123 % 3 cadenas no válidas: static * 2 int + char

Tabla 5. Estados de aceptación Parte III.

Evaluación:

Cada estado de aceptación (tabla 3, tabla 4 y tabla 5), que funcione correctamente, vale 1 punto.

Presentación de la práctica:

- Presentar el programa en ejecución: todos los estados de aceptación deben estar en el mismo programa.
- Presentar el diagrama de estados del AFD utilizado en esta práctica (se puede usar JFLAP, un archivo PDF, o JPEG). **No se revisarán prácticas que no tengan su diagrama de estados correspondiente.**
- Presentar la definición del AFD usado en esta práctica:

$$AFD = (\Sigma, Q, q_0, \delta, F)$$

  - **El diagrama de estados del AFD se tomará como la función de transición (δ).**
  - **El diagrama del AFD utilizado en esta práctica debe corresponder con el diseño del código de otro modo se restarán puntos (5).**
  - **No utilizar expresiones regulares para esta práctica (clases de Java, Python, etc.); es requisito programar el AFD del inciso anterior.**
- Sustentar un breve examen oral acerca del código y de los conceptos de Teoría Computacional empleados en esta práctica.
- Es requisito presentar el diagrama de estados del AFD usado en esta práctica, de otro modo se restarán puntos (5) a la calificación.
- Prácticas copiadas serán canceladas.
- Presentar en forma individual o por equipo (máximo dos personas).
- Presentar práctica en laboratorio, en salón de clase, o en cubículo de profesor de acuerdo al calendario de presentaciones de la práctica 2.

Nota: No se aceptarán prácticas de otras materias como Compiladores.

Fecha de entrega:

Semana del 20 al 24 de mayo. No habrá prórroga.