**Teoría Computacional**

**Práctica 4. Autómata finito determinista (AFD)**

**Parte II**

**Objetivo:** Realizar la implementación de AFD´s que aceptan expresiones regulares, y aplicarlos para el diseño e implementación de un analizador léxico.

**Desarrollo:** 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.



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).





Tabla 3. Estados de aceptación.

Partiendo de nuestra definición formal de AFD:

AFD = ( Σ, Q, q0, ,F)

Tenemos que:

Σ = { 0, … , F, +, - , b , ; , . , E, x }

F = { C0, C1, C2, C3, C4, C5 }

Encontrar: Q, q0, (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:



Programa Numeros.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 Numeros.java, es la siguiente:

**Error en línea 4.**

**Error en línea 6.**

**Error en línea 8.**

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

**No hay errores de análisis léxico en las constantes numéricas del archivo Numeros.java.**

**Evaluación:**

Cada estado de aceptación, que funcione correctamente, vale 2 puntos.

**Presentación de la práctica:**

* Presentar el programa en ejecución: todos los estados de aceptación deben estar en el mismo programa.
* Sustentar un breve examen oral acerca del código y de los conceptos de Teoría Computacional empleados en esta práctica.
* No es necesario entregar reporte escrito,
* Se debe mostrar la definición formal del AFD (archivo PDF o JPG) usado en el Analizador léxico:

AFD = ( Σ, Q, q0, ,F) , se puede entregar el diagrama de estados en lugar de la función de transferencia.

* Prácticas copiadas serán canceladas.
* Presentar práctica en laboratorio, en salón de clase, o en cubículo de profesor.

**Fecha de entrega:**

Fecha de caducidad: viernes 27 de octubre.

**Actividades Opcionales para la Práctica 3**

**A) Incluir en el Analizador léxico, el estado de aceptación C5, para aceptar identificadores válidos de Java. El programa debe ser capaz de diferenciar cualquier palabra reservada de Java de un identificador definido por el programador. Valor: 5 participaciones extra.**

**B) Hacer una versión gráfica del AFD del Analizador léxico para alguno de los estados de aceptación. Por ejemplo, para el estado de aceptación C0 se deberá ver la ejecución del AFD de un estado a otro, de acuerdo con el símbolo del alfabeto que lleve a cabo la transición. Al final de procesar una palabra se debe observar el estado final (aceptación o no aceptación) Valor: de 5 a 10 participaciones extra.**