Tecnológico de Monterrey

Desarrollo

El alumno deberá diseñar e implementar un analizador léxico de acuerdo con los requerimientos que se presentan más adelante. Además, deberá elaborar un reporte de práctica.

Archivo de prueba

La implementación deberá ofrecer la opción de leer el nombre del archivo que habrá de ser analizado, ya sea que el programa solicite esta información o que deba ser proporcionada durante el lanzamiento del mismo.

Familias léxicas

Usaremos la siguiente nomenclatura:

- Caracteres *alfabéticos*: `a`, `b`, …, `z`, y `A`, `B`, …, `Z`.
- Caracteres *numéricos*: `0`, `1`, …, `9`.
- Caracteres *octales*: `0`, `1`, …, `7`
- Caracteres *hexadecimales*: `0`, `1`, …, `9`, `a`, `b`, …, `f`, y las letras `A`, `B`, …, `F`, `x` y `X`.

Identificadores

1. Consisten solo de caracteres alfabéticos, numéricos y el guión bajo (_).
2. No pueden comenzar con un carácter numérico.
3. Deben tener al menos un carácter alfabético.
4. No son palabras reservadas.

Números naturales

Consisten en una *secuencia no vacía* (de al menos una letra) de caracteres numéricos que no comienzan con `0` (*cero*).

Números octales

Consisten en secuencias de caracteres octales que comienzan con `0`(*cero*).

Números hexadecimales

1. Solo pueden contener caracteres hexadecimales.
2. Comienzan con el prefijo `0x` ó `0X`.
3. Finalizan con una secuencia no vacía de caracteres hexadecimales.

Números de punto flotante

1. Consisten solo de caracteres numéricos, las letras `e` y `E`, así como los símbolos: `.` (*punto*), `+` (*suma*) y `-`(*resta*).
2. Un número de punto flotante consiste en dos partes: la primera llamada **mantisa** y la segunda **exponente**.
3. La **mantisa** consiste en una secuencia no vacía de caracteres numéricos, seguida de un punto (`.`) y finalizada en otra secuencia no vacía de caracteres numéricos.
4. El **exponente** es opcional.
5. El **exponente** inicia con la letra `e` (ó `E`), seguida de un carácter `+` ó `-`, el cual es opcional.
6. El **exponente** finaliza con una secuencia no vacía de caracteres numéricos.

Comentarios

Un comentario inicia con la marca `//` (*doble diagonal*), continúa con cualquier secuencia de caracteres distintos del salto de línea y finaliza con el primer salto de línea.

Caracteres de delimitación

Son los siguientes: `(` (*paréntesis izquierdo*), `)` (*paréntesis derecho*), `[` (*corchete izquierdo*) y `]` (*corchete derecho*).

Operadores aritméticos

Se contemplan los siguientes: `+` (*suma*), `-` (*resta*), `*` (*multiplicación*) y `/` (*división*).

Operadores lógicos

Se contemplan los siguientes: `and` (*conjunción*), `or` (*disyunción*) y `not` (*negación*). Estos se consideran un tipo especial de palabras reservadas.

Operadores de relación

Se contemplan los siguientes: `==` (*igualdad*), `!=` (*desigualdad*), `<` (*mayor que*), `>` (*menor que*), `<=` (*menor ó igual que*) y `>=` (*mayor ó igual que*).

Signos de puntuación

Se contemplan los siguientes: `.` (*punto*), `,` (*coma*) y `;` (*punto y coma*).

Operador de asignación

Solo se contempla el siguiente: `=`.

Palabras reservadas

Se contemplan solo las siguientes: `program`, `var`, `begin`, `end`, `if`, `then`, `else`.

Espacios en blanco

Se contempla el uso de *espacios*, *tabuladores* y *saltos de línea*. El analizador deberá ignorarlos cada vez que sean localizados.

La marca de fin de archivo (EOF)

Su analizador léxico deberá ignorar (además de los comentarios) los espacios en blanco: *espacio* (*spc*), *tabulador* (*tab*) y *salto de línea* (*nl*).

Presentación de resultados

El analizador léxico deberá mostrar la siguiente información para cada lexema encontrado:

- El tipo de lexema (Eg. *identificador*, *operador igualdad*, *número de punto flotante*, etc.).
- El lexema correspondiente (Eg. `abc`, `==`, `34.0e-10`, etc.).

Los comentarios no deberán de ser informados. Al terminar su análisis, deberá señalar que este ha terminado, así como el número de líneas que conforman el archivo de prueba.

Informe de errores

El analizador léxico deberá informar la ocurrencia de algún error cuando suceda alguna secuencia que no pueda ser clasificada, así como la línea donde este ocurre. Además, cuando un error sea localizado deberá terminar la ejecución del programa.

Diseño de la implementación

El alumno deberá diseñar un autómata para modelar cada uno de los reconocedores que sean implementados para la construcción de su analizador léxico. Se sugiere:

- Emplear la técnica de la *máquina de estados* para la implementación de los reconocedores
- de lexemas.
- Implementar el manejador del analizador como un *buscador secuencial* de lexemas.
- Utilizar un *diseño modular*, ya que esto facilitará el mantenimiento de su proyecto para futuras aplicaciones.

Cualquier solución alterna a las anteriores deberá ser señalada y justificada. Las siguientes cláusulas son obligatorias:

a) Deberá implementar un reconocedor universal para los números naturales, octales, hexadecimales y de punto flotante.
b) Deberá implementar un reconocedor universal para cada una de las familias siguientes: operadores aritméticos, (2) operadores lógicos, (3) operadores de relación y asignación, (4) operadores de puntuación. Considere el uso de un tipo de *token* de *error*, para simplificar el uso de estos reconocedores.
c) Deberá implementar el reconocimiento de palabras reservadas mediante la técnica de diccionario.

Tecnológico de Monterrey

Requerimientos para la evaluación de la practica

Los siguientes requerimientos serán solicitados para la evaluación de esta práctica:

1. **Verificación de la aplicación**. Se proporcionará un archivo de texto como prueba. El programa deberá ser capaz de reconocer cada uno de los lexemas que aparezcan a lo largo de este archivo, de acuerdo con lo expuesto en las secciones **presentación de resultados** e **informe de errores**. Se verificará que no ocurran conflictos de interpretación, por ejemplo, entre los números enteros y los números reales. Al final se proporciona una secuencia de prueba.
2. **Comprensión del problema**. El alumno deberá tener una clara comprensión del problema, la cual podrá ejercitar durante el desarrollo de esta práctica. Este punto será evaluado mediante una examinación rápida que el profesor hará durante la verificación de la aplicación, así como la revisión del código fuente.

Ejemplo de un archivo de prueba

```
// Comentario
begin
x = 098;        // asigna x
y = 0x3F5A;     // comentario
z = 0.32e-4;    // otro comentario
if x + 34 > y or z == 0.4 then imprime(x,y);
else imprime(y,5z);
end
```

La salida esperada para su programa debe ser similar a la siguiente:

```
Palabra reservada [begin]
Identificador [x]
Asignación [=]
Número octal [098]
Punto y coma [;]
Identificador [y]
Asignación [=]
Numero hexadecimal [0x3F5A]
Punto y coma [;]
Identificador [z]
Asignación [=]
Número real [0.32e-4]
Punto y coma [;]
Palabra reservada if [if]
Identificador [x]
Suma [+]
Numero natural [34]
Operador GE [>]
Identificador [y]
Operador disyunción [or]
Identificador [z]
Operador igualdad [==]
Número real [0.4]
Palabra reservada then [then]
Identificador [imprime]
Paréntesis izquierdo [(]
Identificador [x]
Coma [,]
Identificador [y]
Paréntesis derecho [)]
Punto y coma [;]
Palabra reservada else [else]
Identificador [imprime]
Paréntesis izquierdo [(]
Identificador [y]
Coma [,]
Error en la línea 7
Fin de la ejecución
```