Programación de Sistemas

Práctica No. 1: **Diseño e Implementación de un Analizador Léxico**

Objetivo de la práctica

Diseñar e implementar un analizador léxico que permita reconocer y clasificar una serie de lexemas que se proporcionan mediante un archivo de texto.

Fecha límite de entrega

Miércoles 21 de Septiembre de 2011.

Descripción del problema

El alumno deberá diseñar e implementar un analizador léxico de acuerdo a los requerimientos que se presentan más adelante. Además deberá elaborar un reporte de práctica, según el esquema que se menciona en el Manual para la elaboración de los reportes 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[1]:

- *Caracteres alfabéticos*: Conjunto de los caracteres en los rangos `a`, `b`, …, `z` y A, B, …, Z.
- *Caracteres numéricos*: Conjunto de los caracteres comprendidos en el rango `0`, `1`, …, `9`.
- *Caracteres octales*: Conjunto de los caracteres comprendidos en el rango `0`, `1`, …, `7`
- *Caracteres hexadecimales*: Conjunto de los caracteres comprendidos en los rangos `0`, `1`, …, `9` (*caracteres numéricos*), `a`, `b`, …, `f` y A, B, …, F.

Identificadores
1. Consisten solo de *caracteres alfabéticos*, *caracteres 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
Un número natural es una *secuencia no vacía* (de al menos una letra) de *caracteres numéricos* que no comienza con `0` (*cero*).
Números octales
Un número octal es una secuencia de *caracteres octales* que comienza con `0` (*cero*).

[1] En este documento, usaremos una fuente `monoespaciada` para referirnos a las literales **ASCII** en la descripción de los lexemas ó sus partes, además de los mensajes que son generados por el programa.

Números hexadecimales

1. Solo pueden contener *caracteres hexadecimales* y las letras `x` y `X`.
2. Comienzan con el prefijo `0x` ó `0X`.
3. Finalizan con una secuencia de longitud par pero no vacía de *caracteres hexadecimales*.

Números de punto flotante

1. Consisten solo de *caracteres numéricos*, además de las letras `e` y `E`, así como los símbolos: `.` (*punto*), + (símbolo *suma*) y – (símbolo *resta*).
2. Un número de punto flotante consiste de dos partes: la primera llamada **mantisa** y la segunda **exponente**.
3. La **mantisa** consiste de 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 es cualquier secuencia de caracteres que inicia con la marca `{` (*llave izquierda*) y finaliza con la marca `}` (*llave derecha*). Esta secuencia tiene mayor precedencia que cualquier otra familia léxica. Es decir, que comprende lo que en otro contexto podría ser un espacio en blanco, identificadores, etc. Particularmente, puede comprender múltiples líneas.

Caracteres de delimitación

Se contemplan 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: && (*conjunción*), || (*disyunción*) y ! (*negación*).

Operadores relacionales
Se contemplan los siguientes: == (*igualdad*), != (*desigualdad*), < (*menor que*), > (*mayor 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 las siguientes: `begin`, `end`, `if`, `then`, `else`, `do` y `while`.

Espacios en blanco
Se contempla solo el uso de *espacios*, *tabuladores* y *saltos de línea*.

La marca de fin de archivo (EOF**)**

Su analizador léxico deberá informar la ocurrencia de cada uno de estos lexemas, excepto *los comentarios* y los *espacios en blanco*, según se señala mas adelante.

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, solo cuando se trate de un lexema numérico ó de un identificador (Eg. `abc`, `34.0e-10`, etc.).

Al terminar el análisis (al finalizar la lectura del archivo), su programa deberá señalar que ya ha terminado. Además deberá informar el número de líneas que conforman el archivo de prueba. Recuerde en este punto que los lexemas *comentarios* pueden comprender diversas líneas.

Informe de errores

El analizador léxico deberá informar (mediante un *token de error*) la ocurrencia de algún error cuando suceda alguna secuencia que no pueda ser clasificada así como la línea donde este ocurre. Cuando un error sea localizado deberá informar el número de línea donde este sucede. Posteriormente, deberá continuar el análisis con el *carácter de sincronización* más próximo.

Se considerará un carácter de sincronización cualquiera no alfanumérico (por ejemplo un espacio en blanco, un carácter de delimitación, etc.). Cualquier secuencia alfanumérica o que use el guión bajo que no pueda ser clasificada como *identificador*, *número natural*, *número de punto flotante*, *número octal* ó *número hexadecimal* deberá ser origen de un error. Por ejemplo:

- Una secuencia de *caracteres numéricos* (no octales) inicializados en cero (Eg. `029` ó `083`).
- Una secuencia inicializada con _ que no contenga caracteres alfabéticos (Eg. `_234` ó `__`).
- Una secuencia inicializada en `0x` (ó `0X`) que continúe con una secuencia vacía o impar de caracteres hexadecimales (Eg. `0x3f6` ó `0x`)

Lineamientos generales para la realización de su solución

1. **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 cada uno 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 de carácter obligatorio:

   a) Deberá implementar un *reconocedor universal* para los números naturales, octales, hexadecimales y de punto flotante (1).
   b) Deberá implementar un reconocedor universal para cada una de los grupos siguientes: operadores aritméticos (2), operadores lógicos (3), operadores de relación y asignación (4), y operadores de puntuación (5). Considere el uso del *token de error* para simplificar el diseño de estos reconocedores.
   c) Deberá implementar una búsqueda de diccionario para hacer el reconocimiento de las palabras reservadas.

2. **Lenguaje de implementación**. Se exhorta al uso del lenguaje ANSI C/C++ como lenguaje de desarrollo. Atienda los siguientes puntos:

   a) El uso de interfaces gráficas y/o entornos de desarrollo será opcional y su uso se hará bajo responsabilidad del alumno. Cualquier inconveniente originado por el uso de ellos que impida la revisión del proyecto será causa de la anulación de la misma.
   b) No podrá hacer uso de herramientas auxiliares o ajenas al contenido del curso que suplanten las técnicas estudiadas para el desarrollo de su implementación. Cualquier solución alterna deberá ser señalada y justificada, y no deberá alterar en ningún caso los aspectos a evaluar. El abuso del no apego a este punto podrá ser causa de anulación de su proyecto.

3. **Asesoría para su elaboración**. El alumno podrá en todo momento asesorarse con el profesor durante el desarrollo de su proyecto, empleando para ello preferentemente los horarios de laboratorio destinados a este fin.

Requerimientos para la evaluación de la práctica
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 la descripción del problema. Se verificará que su implementación cumpla además con los lineamientos generales.

2. **Comprensión del problema**. El alumno deberá tener una clara comprensión teórica y práctica del problema, la cual podrá ejercitar durante el desarrollo de esta práctica. Este punto será evaluado mediante una breve evaluación que el profesor hará durante la verificación de la aplicación así como la revisión del código fuente.

3. **Documentación del proyecto**. Deberá elaborarse un reporte de práctica, apegándose al esquema que se expone en el *Manual para la elaboración de los reportes de práctica*. Particularmente, deberá informar:

    a) Qué tipo de solución empleó.
    b) Cuál es el diseño de sus máquinas de estados (autómatas).
    c) Qué soluciones empleó para el almacenamiento del lexema, el informe de errores, el control del fin de archivo, etc.

    Recuerde que la entrega de la documentación impresa será un requisito necesario para dar inicio a la revisión de su proyecto.

Ejemplo
Suponga una entrada como la siguiente:

```
begin
  _x34 = (23 + 043.3e-2);
  hexa = 0x34f * 09wer;
  { Comentario multilinea if else
  }
  if (_x34 > hexa || hexa == .23) then hexa = 0
end
```

Entonces, su programa deberá generar un resultado similar al que se muestra enseguida:

```
Palabra reservada begin
Identificador: [_x34]
Asignación
Paréntesis izquierdo
Número natural: [23]
Operador suma
Número PF: [043.3e-2]
Paréntesis derecho
Punto y coma
Identificador: [hexa]
Asignación
Error en la línea 3
Operador multiplicación
Error en la línea 3
Punto y coma
Palabra reservada if
Paréntesis izquierdo
Identificador: [_x34]
Operador mayor que
Identificador: [hexa]
Operador disyunción
Identificador: [hexa]
Operador igualdad
Punto
Número natural: [23]
Paréntesis derecho
Palabra reservada then
Identificador: [hexa]
Asignación
Número octal: [0]
Palabra reservada end

Fin del análisis
Líneas analizadas: 7
```