Prácticas de Lenguajes, Gramáticas y Autómatas

Cuarto cuatrimestre (primavera) de Ingeniería en Informática

Curso 2010-2011

http://webdiis.unizar.es/asignaturas/LGA

Profesor Responsable: Jorge Júlvez
Dpto. Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Práctica 1

Requisitos: *Haberse leído la introducción a Flex que acompaña este guión, y que también está en la página Web de la asignatura (para esta primera práctica no es necesario mirar la parte de condiciones de arranque ni la de interacción con Bison, así que son sólo unas seis páginas)*

Objetivo: *El objetivo principal de esta primera práctica de la asignatura es que el alumno se familiarice con la herramienta de creación de analizadores léxicos* Flex*, y de paso que reflexione sobre algunas cuestiones de teoría sobre lenguajes y expresiones regulares. Para ello, tendrá que crear una serie de pequeños procesadores de texto haciendo uso de dicha herramienta.*

Flex y Bison en hendrix-ssh.cps.unizar.es

Las prácticas de LGA se realizarán sobre hendrix-ssh.cps.unizar.es. En esta máquina los ejecutables correspondientes a flex (todas las prácticas) y bison (prácticas 3 y 4) están en /opt/csw/bin/flex y /opt/csw/bin/bison. Para obtener un ejecutable, se debe utilizar:

```
flex nombre_fichero_fuente
gcc lex.yy.c –lfl –o nombre_ejecutable
```

Equivalencia de la notación vista en teoría y la de Flex

El alfabeto **Σ** que maneja Flex es el compuesto por los caracteres manejables por el sistema, p.ej., todos los símbolos del código ASCII. En las prácticas trabajaremos con letras (las mayúsculas y las minúsculas son distintos símbolos del alfabeto), números y signos de puntuación.

Algunas equivalencias entre la notación de clase y la de Flex se muestran en la siguiente tabla:

Teoría	**Flex**
· (concatenación de e.r., aunque generalmente no ponemos nada)	No hay símbolo. Las e.r. a concatenar se escriben pegadas como en la notación de teoría.
+ (unión de e.r.)	|
* (cerradura de Kleene)	*
+ (cerradura positiva)	+
( ) (paréntesis)	( )
{a,b,c,d,0,1,2} (conjunto formado por los símbolos a,b,c,d,0,1 y 2)	[abcd012] ó también [a-d0-2]
ε + a	a? El símbolo a, cero o una veces.
ε	No hay equivalente directo en Flex. Para usarlo como parte de una expresión, se usan los símbolos * ó ? que permiten repetir algo cero veces (ver línea anterior).

Ejercicio 1

Diseñar un programa con Flex que elimine los caracteres blancos del final de cada línea del texto de entrada y reduzca a un solo blanco cualquier grupo de blancos del interior de cada línea (se entiende por blanco el espacio en blanco y el carácter de tabulación). Ejemplo de entrada:

```
Este      es    un texto        <EOL>
con varios        blancos         <EOL>
<EOF>
```

Salida:

```
Este es un texto<EOL>
con varios blancos<EOF>
```

Ejercicio 2

Elaborar un programa en Flex que copie el archivo de entrada en uno de salida, sustituyendo todo numero entero positivo que sea múltiplo de 7 por el carácter '*'.
Este ejercicio tiene al menos dos aproximaciones posibles en Flex:

- Construir una e.r. que denote el lenguaje de las palabras sobre {0,1,2,3,4,5,6,7,8,9} que en decimal son un múltiplo de 7 y buscarla en Flex. ¿Se puede construir una e.r. que reconozca los números decimales múltiplos de 7? ¿Cómo sería la expresión?
- Construir una e.r. que lea números en decimal, y después en la acción asociada en Flex comprobar si éstos son múltiplos de 7.

Ejercicio 3

Elaborar un programa en Flex que cuente el número de palabras del texto de entrada que contengan alguna vez la letra 'w'.

- ¿Cómo sería la e.r.? ¿Y como sería si nos piden las palabras que contengan sólo una w?

Ejercicio 4

Elaborar un programa en Flex que dada una secuencia de entrada, realice lo siguiente para cada uno de los casos:

- Informar del número de caracteres, de palabras y de líneas de texto.
- Reconocer todos los números enteros y calcular e informar de la media de los mismos.
- Reconocer todos los números reales e informar cuántos de ellos están comprendidos en el rango entre cero y uno.

La principal dificultad de este ejercicio, está en notar que los números son también palabras, y además están compuestos por caracteres, sin embargo un carácter del fichero de entrada en Flex sólo se puede emparejar con, como mucho, un patrón (una e.r.).

Ejercicio 5

Elaborar un programa con Flex que dado un fichero de texto, verifique que siempre que se abre un paréntesis, se cierra. Si detecta algún error, informará de la línea donde se produce dicho error. Este ejercicio tiene una dificultad:

- ¿Es el lenguaje de las palabras sobre el alfabeto {(,)} que son pares de paréntesis correctamente anidados regular? ¿Podéis demostrarlo?
- Si no es regular, ¿podemos hacer algo?

Práctica 2

Requisitos: *Haber hecho la práctica 1 y mirar la parte de condiciones de arranque de la introducción a Flex.*

Objetivo: *El objetivo principal de esta segunda práctica de la asignatura es que el alumno "asiente" el manejo de la herramienta de creación de generadores léxicos Flex. Para ello creará analizadores léxicos de mayor complejidad y tendrá la oportunidad de aplicar* ***condiciones de arranque****. Las condiciones de arranque nunca son imprescindibles, siempre se pueden "emular" mediante código C en las acciones de los patrones, pero pueden facilitar la escritura, y la lectura, de patrones/acciones de Flex que sólo deben aplicarse en condiciones determinadas.*

Ejercicio 1

Implementar un programa en Flex que tome como entrada un fichero fuente PASCAL y devuelva la siguiente información:

- Número de líneas y caracteres del programa fuente.
- Número de comentarios (entre llaves) y longitud media expresada en forma de número de caracteres.
- Número de instrucciones de asignación del programa (sintaxis del operador :=).
- Número de instrucciones de composición condicional (if, case) y de composición iterativa (while, for, repeat).

Nota: Se proporcionan diferentes programas de prueba para poder comprobar el correcto funcionamiento del analizador léxico construido. En el directorio "/export/home/practicas/Practicas/LGA/pascal" podéis encontrar programas PASCAL de prueba.

Ayuda: en este ejercicio, las condiciones de arranque pueden venir bien, por ejemplo, para no contar palabras clave (p.ej. while) que estén dentro de comentarios o de constantes de tipo cadena. La idea es definir una condición de arranque que esté activa dentro de un comentario, otra para las cadenas y contar las palabras clave sólo cuando estas condiciones no estén activas (por ejemplo, contarlas sólo cuando esté activa la condición de arranque INITIAL).

Ejercicio 2

Es corriente encontrar páginas Web donde aparecen ejemplos de código fuente. Implementar un programa Flex que dada una página Web, en la que aparece código fuente, elimine todos los TAGs propios de la sintaxis HTML (están delimitados por < y >). Es necesario tener en cuenta que un TAG puede contener una cadena de caracteres (entre comillas) de forma que alguno de sus caracteres sea un >. En este caso, no se trataría como la finalización del TAG, sino como un carácter de la cadena.

Utilizando el analizador léxico construido, procesa alguna de las páginas Web que se te proporcionan e intenta compilar el código fuente (en C) obtenido.

Nota: Se proporcionan diferentes páginas Web de prueba para poder comprobar el correcto funcionamiento del analizador léxico construido. En el directorio "/export/home/practicas/Practicas/LGA/web" podéis encontrarlas.

Ayudas:

- Notad que dentro del código C puede haber símbolos de menor qué y de mayor qué (<,>) por ejemplo en las directivas #include de bibliotecas estándar, o en operaciones de comparación de ciertos tipos de datos. En los ejemplos que se os dan, estos casos están sustituidos (pone *<* para el símbolo < y *>* para el símbolo > (son códigos HTML para caracteres especiales)).
- En este ejemplo, las condiciones de arranque se pueden usar para distinguir los casos en los que se está dentro de una cadena que aparece dentro de un TAG para tratar los símbolos (>) que puedan aparecer allí como cierres del TAG.