Universidad Nacional Autónoma de México
Facultad de Ciencias
Compiladores. 2025-1 | 7008
Práctica 1: Análisis léxico con Flex
Kevin Steve Quezada Ordoñez
stevequezada@ciencias.unam.mx
30/08/2024

1. Transcribir el código anterior a un archivo con extensión .flex dentro de la carpeta.

src/Primer programa en Lex/

```
/**
* Escáner que detecta números y palabras
*/

%%

%public
%class Lexer
%standalone

digito=[0-9]
letra=[a-zA-Z]
palabra={letra}+
espacio=[ \t\n]

%%

{espacio} {/* La acción léxica puede ir vacía si queremos que el escáner
    ignore la regla*/}
{digito}+ { System.out.print("Encontré un número: "+yytext()+"\n"); }
{palabra} { System.out.print("Encontré una palabra: "+yytext()+"\n"); }
```

2. Compilar mediante la instrucción: jflex archivo.flex

```
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma$ jflex practica1.flex
Reading "practica1.flex"
Constructing NFA : 14 states in NFA
Converting NFA to DFA :
......
8 states before minimization, 5 states in minimized DFA
Writing code to "Lexer.java"
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma$ ls Lexer.java  practica1.flex
```

c. Comprobar se genero el archivo Lexer.java

```
steve-quezada@steve-quezada-VMware-Virtual-Platform:
            ~/Desktop/Compiladores/PrimerPrograma$ ls
Lexer.java  practica1.flex
```

d. Compilar mediante: javac Lexer.java

```
steve-quezada@steve-quezada-VMware-Virtual-Platform:
            ~/Desktop/Compiladores/PrimerPrograma$ javac Lexer.java
steve-quezada@steve-quezada-VMware-Virtual-Platform:
            ~/Desktop/Compiladores/PrimerPrograma$ ls
Lexer.class  Lexer.java  practica1.flex
```

e. Crear un archivo de texto que será la entrada: echo "Hay 100 nubes en el cielo hoy» input.txt

```
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma$
        echo "Hay 100 nubes en el cielo hoy" > input.txt
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma$ cat input.txt
Hay 100 nubes en el cielo hoy
```

f. Ejecutar mediante: java Lexer input.txt

```
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma$ java Lexer input.txt
        Encontré una palabra: Hay
        Encontré un número: 100
        Encontré una palabra: nubes
        Encontré una palabra: en
        Encontré una palabra: el
        Encontré una palabra: cielo
        Encontré una palabra: hoy
```

Ejercicios

1. ¿Qué ocurre si en la primera sección se quitan las llaves al nombre de la macro letra? (0.5 pts)

   Si se quitan las llaves a la macro `letra`, `Flex` deja de tratarla como una expresión regular independiente. Esto implica que las palabras reconocidas no se detectan correctamente.

```
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma/1.PrimerPrograma$ java Lexer input.txt
        HayEncontré un número: 100
```

2. ¿Qué ocurre si en la segunda sección se quitan las llaves a las macros? (0.5 pts)

   Al quitar las llaves a las macros `Flex` las trata como texto literal en lugar de expandirlas como expresiones regulares, lo que genera advertencias de macros no utilizadas y provoca que el análisis léxico no funcione correctamente, esto resulta en que el texto de entrada se procese sin reconocer ni ejecutar las acciones definidas para las expresiones regulares.

```
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma/2. PrimerPrograma$ jflex practica1.flex
        Reading "practica1.flex"

        Warning : Macro "espacio" has been declared but never used.

        Warning : Macro "palabra" has been declared but never used.

        Warning : Macro "letra" has been declared but never used.

        Warning : Macro "digito" has been declared but never used.
        Constructing NFA : 46 states in NFA
        Converting NFA to DFA :
        ......................
        23 states before minimization, 22 states in minimized DFA
        Writing code to "Lexer.java"
steve-quezada@steve-quezada-VMware-Virtual-Platform:
     ~/Desktop/Compiladores/PrimerPrograma/2. PrimerPrograma$ javac Lexer.java
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma/2. PrimerPrograma$
        echo "Hay 100 nubes en el cielo hoy" > input.txt
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma/2. PrimerPrograma$ java Lexer
        input.txt       Lexer.class     Lexer.java      practica1.flex
steve-quezada@steve-quezada-VMware-Virtual-Platform:
    ~/Desktop/Compiladores/PrimerPrograma/2. PrimerPrograma$ java Lexer input.txt
        Hay 100 nubes en el cielo hoy
```

3. ¿Cómo se escribe un comentario en flex?

   Los comentarios se escriben de la siguiente manera:

```
// Comentario de una línea

/*
Comentario
de
múltiples
líneas
*/
```

4. ¿Qué se guarda en yytext? (0.5 pts

   Guarda el texto que coincide con la expresión regular que se está evaluando en ese momento. Cada vez que el escáner sencuentra una coincidencia en el texto de entrada, `yytext` contiene la cadena exacta que se ha reconocido, lo que permite que se realicen acciones con ese texto, como mostrarlo o manipularlo.

5. ¿Qué pasa al ejecutar el programa e introducir cadenas de caracteres y de dígitos en el archivo de entrada? (0.5 pts)

   Cuando se ejecuta el programa y se ponen cadenas de texto y números en el archivo de entrada, el escáner los reconoce según las reglas que se definieron. Identifica las palabras y las imprime como tales, y lo mismo hace con los números. El programa separa y procesa las palabras y números de acuerdo con las expresiones regulares.

6. ¿Qué ocurre si introducimos caracteres como "*.^{en} el archivo de entrada? (0.5 pts)

   Si se introduce en el archivo de entrada y no se tienes una regla específica, el escáner no sabrá cómo procesarlo. Dependiendo de cómo esté configurado el programa, se ignora el carácter, se imprime tal cual, o se muestra un error, en general, no lo reconocerá ni lo procesará como lo hace con las palabras o números.

7. Modificar al código anterior en un archivo nuevo, de tal manera que reconozca lo siguiente: (2 pts)

   i. La expresión regular para los hexadecimales en lenguaje Java.

   ii. 5 palabras reservadas del lenguaje Java.

   iii. Los identificadores válidos del lenguaje Java, con longitud máxima de 32 caracteres (Sugerencia: use el operador m,n).

   iv. Los espacios en blanco.

```
/**
 * Escáner que detecta números, palabras, hexadecimales, palabras reservadas e
   identificadores en Java
 */

%%

%public
%class Lexer
%standalone

// Expresiones regulares básicas
digito=[0-9]
letra=[a-zA-Z]
letraDigito=[a-zA-Z0-9]
espacio=[ \t\n]

// Expresiones regulares adicionales
hexadecimal=0[xX][0-9a-fA-F]+
reservadas=(if|else|while|return|class)
identificador={letra}({letraDigito}|_){0,31}
identificadorlargo={letra}({letraDigito}|_){32,64}

%%

{espacio} { System.out.print("Se encontró un espacio en blanco: "+yytext()+"\n"); }
{hexadecimal} { System.out.print("Se encontró un hexadecimal: "+yytext()+"\n"); }
{reservadas { System.out.print("Se encontró una palabra reservada: "+yytext()+"\n"); }
{identificador} { System.out.print("Se encontró un identificador: "+yytext()+"\n"); }
{identificadorlargo} { System.out.print("Se encontró  un identificador no valido: "+
    yytext()+"\n"); }
{digito}+ { System.out.print("Se encontró un número: "+yytext()+"\n"); }
```