



Universidad de Málaga

Departamento de Lenguajes y
Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

TRADUCTORES, COMPILADORES E INTÉRPRETES. 05/06
EJERCICIOS DE LEX. TEMA 2.

- 1.) Hacer un programa LEX que convierta todos los caracteres de su entrada que sean LF (ascii 10), por LF y CR (ascii 10 y ascii 13).
- 2.) Hacer un programa LEX que tras leer un texto nos diga el número de caracteres, palabras y líneas de dicho texto, entendiéndose por palabra toda secuencia de caracteres que no posea ni espacios ni tabuladores ni retornos de carro. Se supone que toda línea está acabada por un retorno de carro (\n).
- 3.) Hacer un programa LEX que reconozca números múltiplos de dos. No utilizar la función **módulo** de C.
- 4.) Hacer un programa LEX que reconozca números múltiplos de cuatro. No utilizar la función **módulo** de C.
- 5.) Hacer un programa LEX que reconozca números múltiplos de tres. No utilizar la función **módulo** de C.
- 6.) Hacer un programa LEX que lea un fichero escrito en Java e indique lo siguiente:
 - * Cuántos comentarios hay (sólo multilínea).
 - * Cuántas veces aparece la palabra **public**.
 - * Cuántas veces aparece la palabra **private**
 - * Cuántos espacios y tabuladores hay.
- 7.) Hacer un programa LEX que lea un fichero escrito en Java e indique la profundidad de anidamiento de llaves más grande. Cada vez que unas llaves están contenidas dentro de otras, aumenta el nivel de anidamiento.
- 8.) Hacer un programa LEX que lea un programa en MODULA-2 e indique lo siguiente:
 - * Cuántos comentarios hay.
 - * Cuántas funciones hay.
 - * Cuántos procedimientos hay.
 - * Cuántos espacios y tabuladores hay.
- 9.) Mediante un programa LEX, sacar un listado con todos los identificadores de usuario de un fichero escrito en Java. Para ello, hay que reconocer todas las palabras reservadas, y todo identificador que no sea una palabra reservada se asume que es un identificador de usuario.
- 10.) Mediante un programa LEX, sacar un listado con todos los identificadores de usuario de un fichero escrito en Java. Los identificadores no se deben sacar por pantalla sobre la marcha, sino que deben almacenarse en una lista y, al final del funcionamiento de **yylex()**, emitir por pantalla el contenido de dicha lista.
- 11.) Mediante un programa LEX, sacar un listado con todos los nombres de procedimientos (PROCEDURE) de un programa en MODULA-2.

12.) Hacer un programa en LEX, de manera que se cifre el texto de entrada, convirtiendo cada palabra en su inversa. El concepto de palabra es el mismo que en el ej. 2).

13.) Hacer un cifrado ligeramente más complicado que el anterior:

* Si una palabra tiene 4 o menos letras, cambiarla por su inversa. Ej.: niño --> oñin.

* Si tiene 5 ó 6 letras, cambiarla por su inversa en bloques de dos caracteres. Ej.: comida --> damico.

* Si tiene 7, 8 ó 9 letras, cambiarla por su inversa en bloques de tres caracteres. Ej.: botellín --> líntelbo.

* Si tiene más de 9 letras, cambiarla por su inversa en bloques de 4 caracteres. Ej.: ferretería --> eríarretfe.

14.) Mediante un programa LEX, hacer la lectura de un programa fuente en MODULA-2. Extraer todos los identificadores definidos por el usuario. Visualizar al final una lista con todos ellos, a ser posible en orden alfabético.

15.) Análogo al anterior, pero además de visualizar los identificadores al final, indicar para cada uno de ellos, todos los números de línea en que han aparecido.

16.) Supuesto que se tiene un diccionario de palabras en formato texto, (almacenado en un fichero con una palabra por línea), procesar mediante un programa LEX, cualquier texto de entrada, visualizando por pantalla todas las palabras que no estén en dicho diccionario. El diccionario puede ser volcado a memoria justo antes de comenzar el procesamiento.

17.) Modificar el programa anterior, de manera que cada vez que se encuentre una palabra que no está en el diccionario, se consulte al usuario, que tendrá las siguientes opciones:

- Ignorar la palabra.

- Meterla en el diccionario.

- Modificar la palabra. En tal caso, cada vez que se vuelva a encontrar la palabra original, se sustituirá por la nueva.

Se obtendrá como salida el mismo texto de entrada (con las palabras modificadas), y el mismo fichero diccionario de entrada, pero enriquecido con las nuevas palabras.

18.) Hacer un programa LEX que tras leer su entrada, nos indique el número de palabras leídas que poseen un diptongo cuya primera letra es **u**, y la segunda no es una **a**. No se considerará diptongo aquella subcadena que forme parte de un triptongo. De hecho, en español sólo existen tres triptongos: *-uai-*, *uei-*, *-iai-*, *-iei-*. Del total de palabras leídas con el diptongo indicado decir cuantas son de cada forma: *-ue-*, *-ui-*, *-uo-*, *-uu-*. Si una cadena posee más de uno de estos diptongos se contabilizará una vez para cada diptongo diferente que posea.

19.) La Secretaría de la E.T.S.I. de Informática recibe de los Departamentos las notas de los alumnos en un fichero formato texto, en el que cada línea posee tres campos separados por espacios: **D.N.I.** del alumno, **Convocatoria** del examen (Diciembre, Febrero o Junio), y la **Nota**. Hacer un programa LEX que efectúe la lectura de dicho fichero, emitiendo al final la media de las notas recuperadas. Asimismo informará de los errores de formato que se pueda encontrar: líneas en las que falten campos, valores incorrectos, etc..

20.) Hacer con LEX un formateador de programas en MODULA-2:

* Cada vez que se encuentre una palabra de comienzo de bloque (BEGIN, WHILE, IF, LOOP, etc.), indenta 4 espacios a la derecha a partir de la siguiente línea.

- * Cuando encuentre el END o el UNTIL correspondiente, elimina la última indentación efectuada.
- * Coloca los comentarios en cursiva, (anteponer la palabra *Cursiva*, y acabar el comentario con la palabra *Fin Cursiva*).
- * Coloca las cabeceras de los procedimientos en negrita, (igual que el anterior pero con las palabras **Negrita** y **Fin Negrita**).