



LEX

Segunda parte



Variables predefinidas de Lex

	Tipo	Utilidad
yytext	Cadena de caracteres	Almacena la última cadena procesada
yytext	Entero	Longitud de yytext
yyin, yyout	Ficheros de texto	Ficheros estándar de entrada y salida del analizador léxico



Funciones predefinidas de Lex

	Tipo	Utilidad
yylex()	Procedimiento	Analizador léxico
yywrap()	Procedimiento	Cuando <code>yylex()</code> alcanza el final del fichero de entrada, esta función retorna 0 o 1 según haya más entradas o no
ECHO	Acción predefinida	Escribe en <code>yyout</code> la cadena procesada



Acción por omisión

La acción que se ejecuta por omisión (cuando una cadena no *encaja* con ninguna de las e.r. definidas) consiste en **escribir** en el fichero de salida la cadena procesada

● ● ● | Acción por omisión. Ejemplo:

`[a-zA-Z0-9] cout << "*";`

el programa escribirá un * por cada letra o dígito encontrado, el resto de los caracteres leídos (incluidos los caracteres en blanco) se escriben en el fichero de salida

Entrada

¿Hay 5 perros, 2 gatos y
1 loro?.

Salida

¿*** * *****, * ***** *
* ****?.

● ● ● | Acción por omisión. Solución:

```
[a-zA-Z0-9]    cout << "*";  
.  
;
```

Entrada

¿Hay 5 perros, 2 gatos y
1 loro?.

Salida

Definición de e.r.

- La unión entre e.r., se representa con el símbolo $|$

Ejemplo:

$ab|cd^*$ (representa a la cadena ab o a cualquier cadena que *encaje* con cd^*)

$\{entero\}|\{real\}$



Definición de e.r. Complementario

- Si se utiliza \wedge como primer símbolo dentro de los corchetes se define el **complementario** del conjunto indicado

- Ejemplos:

$[\wedge aA]$ (define cualquier carácter diferente de la **a**, minúscula o mayúscula)

$[\wedge a-zA-Z]$ (define cualquier carácter que no sea una letra)

$[a\wedge A]$ (representa a los caracteres: a, A, \wedge)



Definición de e.r. Complementario

- Necesitamos una E.R. para reconocer una cadena de caracteres en C/C++ (caracteres encerrados entre dobles comillas), por ejemplo “Hola, mundo”.
- Una propuesta muy sencilla (pero que no siempre funciona bien):

`["].*["]`

```
cout << “Total” << x << “euros” ;
```



Definición de e.r. Complementario

- Ejemplo que permite representar una cadena de caracteres en C/C++ (caracteres encerrados entre dobles comillas):

`[\"^[^\"]*\"]`

```
cout << "Total" << x << "euros" ;
```



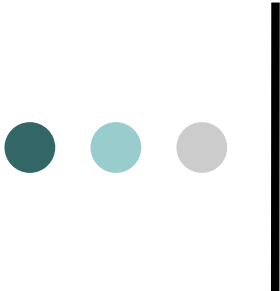
Definición de e.r. Diferencia

- Si se utiliza la cadena **{-}** entre dos conjuntos de valores, se representa la **diferencia** entre ambos conjuntos: caracteres que pertenecen al primero y no al segundo

- Ejemplos:

[a-z]{-}[aeiou] (cualquier minúscula consonante)

[a-zA-Z]{-}[xX] (cualquier letra salvo la X)



Definición de e.r.

Condiciones de contexto

- Se utiliza \wedge para indicar que la e.r. debe aparecer al comienzo de la línea
- Se utiliza $\$$ para indicar que la e.r. debe aparecer al final de la línea
- $er1/er2$ indica que la $er1$ sólo es válida si va seguida de $er2$



Definición de e.r. Condiciones de contexto

Ejemplos:

Expr. Regular

Cadena procesada

[^]el

el coche era rojo

jo\$

el coche era rojo

ro/jo

el coche era rojo



Definición de e.r. Prioridades

1. Paréntesis	()
2. Unión de caracteres	[]
3. Repeticiones	* + ? { }
4. Concatenación	Cc
5. Unión de e.r.	
6. Condiciones de contexto	^ \$/

Ejemplos:

E.R.

ab*

(ab)*

Cadena validada

abbbbbbb

abababab



Acciones

- Si una cadena encaja con varias e.r., se ejecuta la acción asociada a la e.r. que valide la cadena más larga

Ejemplo:

[0-9]	acción1;
[0-9]+	acción2;

Si en el fichero de entrada aparece la cadena **24** se ejecuta la acción2, ya que la segunda regla valida dos caracteres y la primera solo uno

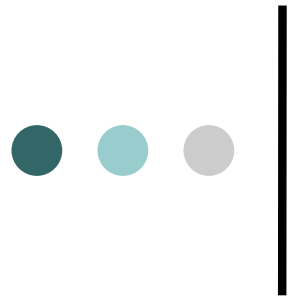


Acciones

- Si una cadena encaja con varias e.r.'s y todas ellas tienen la misma longitud se ejecuta la acción asociada a la e.r. que se ha definido antes
- Ejemplo:

```
for      acción1;  
[a-z]+   acción2;
```

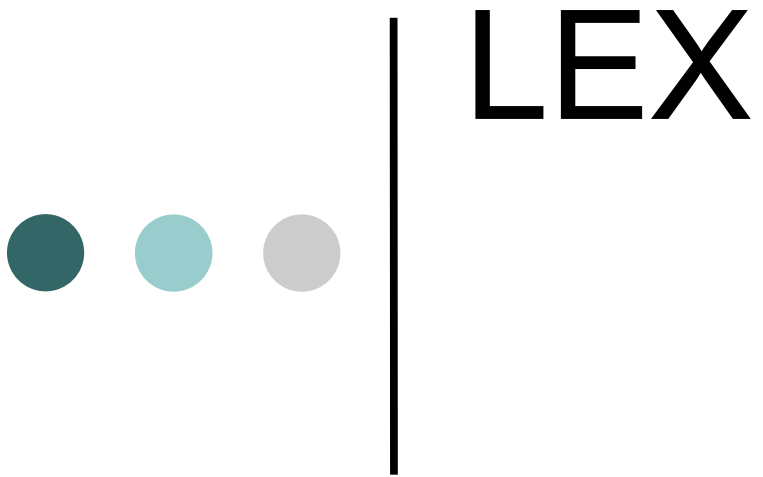
Si en el fichero de entrada aparece la cadena **for** (las dos reglas validarían 3 caracteres) se ejecuta la acción1, simplemente por estar escrita en primer lugar



Ejemplo: simple.l

Este ejemplo elimina de un fichero de texto aquello que se considera inútil: líneas en blanco, algunos tabuladores y compacta los espacios en blanco.

```
%%  
[\\t ]+$          ;  
[ ]+             cout << " ";  
^[\\t\\n]         ;  
                  ECHO;  
\\n               ECHO;  
%%
```



LEX

Segunda parte