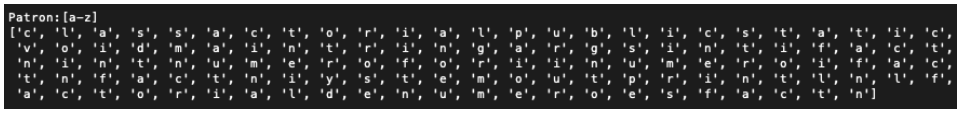
Compiladores, 14 de febrero de 2026

Martínez Buenrostro Jorge Rafael

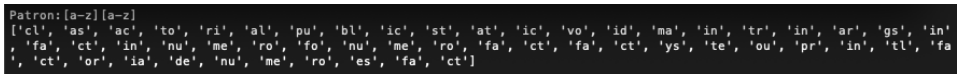
correo, *cbi*2203040824@*izt.uam.mx*
Universidad Autónoma Metropolitana
Unidad Iztapalapa, México

Ejemplo de análisis léxico sobre un archivo Java

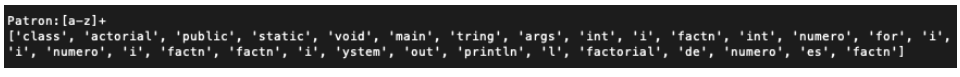
- **Expresión regular:** [**a-z**] Este patrón reconoce cualquier carácter individual que sea una letra entre la 'a' y la 'z' minúscula. En la siguiente imagen se puede ver la salida en la que se puede ver una lista de todos los caracteres que son letras minúsculas en el archivo en orden de aparición.



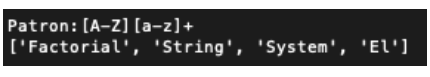
- **Expresión regular:** [**a-z**][**a-z**] Este patrón reconoce pares de caracteres en el que ambos son cualquier letra entre la 'a' y la 'z' minúsculas. En la siguiente imagen se puede ver la salida en la que se puede ver una lista de todas las letras minúsculas entre la 'a' y la 'z' del archivo agrupadas en pares.



- **Expresión regular:** [**a-z**]+ Este patrón busca una o más letras minúsculas consecutivas, sin mayúsculas, números ni otros caracteres). En la salida podemos ver las "palabras"que se encuentran en el archivo que tienen letras minúsculas o todas las letras minúsculas de una palabra sin contar las mayúsculas, números o otros caracteres.



- **Expresión regular:** [**A-Z**][**a-z**]+ El patrón busca palabras que comienzan con una letra mayúscula seguida de una o más letras minúsculas. En la salida podemos ver todas las palabras que cumplen con este patrón dentro del archivo



- **Expresión regular: [A-Za-z][A-Za-z0-9_]*** El patrón busca una letra seguida de cero o más caracteres que pueden ser letras, dígitos o espacios.

```
Patron:[A-Za-z][A-Za-z0-9]*
['class Factorial ', 'public static void main', 'String args ', 'int i', 'factn ', 'int numero ', 'for ', 'i ', 'i ', 'numero
', 'i', 'factn ', 'factn ', 'i', 'System', 'out', 'println', 'El', 'factorial', 'de', 'numero ', 'es', 'factn']
```

- **Expresión regular:** <= | >= | < | > El patrón busca los operadores relacionales <= | >= | < | > dentro del archivo Java

```
Patron:<=|>=|<|>
['<=']
```

Reconocimiento de elementos de un programa en C

Para poder identificar y clasificar siguiente elementos léxicos se usaron las siguientes expresiones regulares:

```
("IDENTIFICADOR", r'(?<=[ \t])[A-Za-z_][A-Za-z0-9_]*(?=[ ,;])'),
("NUMERO_ENTERO", r"(?<=[ \t(])\d+(?=[ ,;)])"),
("NUMERO_REAL",   r"[0-9]+\.[0-9]+"),
("CADENA",        r'"[^"]*"'),
("COMENTARIO_LINEA",  r"//[^\n]*"),
("COMENTARIO_MULTI",  r"/\*.*?\*/"),
("OPERADOR_ARITMETICO", r"[+\-*/%=]"),
("OPERADOR_RELACIONAL", r"<=|>=|>|<"),
("OPERADOR_LOGICO",     r"&&|\|\|"),
```

Identificadores

La expresión regular la podemos dividir en tres partes: inicio de la cadena, contenido y fin de la cadena.

1. **Inicio de la cadena** $(? <= [\ \backslash t])$ Quiere decir que aceptamos la cadena si antes de la cadena hay un espacio o un tabulador.

2. **Contenido** $[A-Za-z_][A-Za-z0-9_]*$ Aceptamos cualquier cadena que empiece con alguna letra (minúscula o mayúscula) seguida de cero o más repeticiones de letras (minúsculas o mayúsculas), dígitos o guión bajo.

3. **Fin de la cadena** $(? = [\ ,;])$ Aceptamos la cadena si el caracter después del final es un espacio, una coma o un punto y coma.

Número entero

Para los números enteros se reutilizó el inicio y fin de cadena de los identificadores. Solo se modifica el contenido deseado con $\backslash d+$ lo que significa que el contenido tiene al menos un dígito

Número real

La expresión regular para los números reales tiene la siguiente forma $[0-9]+\backslash.[0-9]+$. Esta expresión indica que un número real comienza con al menos un dígito seguido por un punto y termina con al menos un dígito.

Cadena delimitada por comillas dobles

La expresión regular para las cadenas delimitadas por comillas dobles es `"[^"]*"`. Esta expresión nos dice que las cadenas inician y terminan con comillas dobles, el detalle está en `[^"]`, esto nos indica que se acepta cualquier carácter excepto una comilla doble.

Comentario en una línea

El caso del comentario en una línea es similar al anterior en el que indicamos como inicia la cadena aceptada y ponemos una restricción al contenido. En este caso queda `//[^\n]*` lo que significa que aceptamos que empiece con //; sin embargo, para el contenido no aceptamos el carácter para le salto de línea `\n`

Comentario multilínea

Este es un caso similar a los números reales. La expresión regular es `/\*.*?\*/` nos indica que la cadena puede empezar con `/\*`, para luego poder aceptar cualquier carácter cero o más veces, pero lo mínimo posible deteniendo en la primera coincidencia que se encuentra `.*?`, para terminar con `\*/`

Operador aritmético

Creamos una clase que contiene todos los símbolos de los operadores lógicos. Por lo que la expresión regular resultante es `[+\-*/%=]`

Operador relacional

Se crea una expresión regular que contenga los símbolos de los operadores relaciones para que resulte de esta manera `<=|>=|<|>`

Operador lógico

Al igual que con los operadores relaciones solo agregamos los símbolos de los operadores lógicos de tal modo que la expresión regular resultante es `&&|\|\|`

Por último esta es la salida al ejecutar el código en python

```
--- IDENTIFICADOR ---
Patrón: (?<=[ \t])[A-Za-z_][A-Za-z0-9_]*(?=[ ,;])
Todas las coincidencias: ['float', 'radio', 'area', 'area', 'radio', 'de', 'un', 'entrada', 'de', 'el', 'radio', 'proceso', '
para', 'calcular', 'el', 'Area', 'area']

--- NUMERO_ENTERO ---
Patrón: (?<=[ \t(])\d+(?=[ ,;)])
Todas las coincidencias: ['0', '0', '2']

--- NUMERO_REAL ---
Patrón: [0-9]+\.[0-9]+
Todas las coincidencias: ['3.1416', '5.2']

--- CADENA ---
Patrón: "[^"]*"
Todas las coincidencias: ['"Area de un ciruculo"', '"\\nIntroduce el radio =>"', '"%f"', '"\\n\\nArea = %5.2f"']

--- COMENTARIO_LINEA ---
Patrón: //[^\n]*
Todas las coincidencias: ['// entrada de datos', '// salida']

--- COMENTARIO_MULTI ---
Patrón: /\*.*?\*/
Todas las coincidencias: ['/* proceso para calcular el Area total*/']

--- OPERADOR_ARITMETICO ---
Patrón: [+\-*/%=]
Todas las coincidencias: ['=', '=', '/', '/', '=', '%', '/', '*', '*', '/', '=', '*', '/', '/', '=', '%']

--- OPERADOR_RELACIONAL ---
Patrón: <=|>=|>|<
Todas las coincidencias: ['<', '>', '<', '>', '>']

--- OPERADOR_LOGICO ---
Patrón: &&|\|\|
```

Expresiones regulares en BASH

Para esta sección se desarrolló un script en BASH para mostrar las particiones cuyo uso de disco sea mayor o igual a 80 %. El script muestra el uso de disco del sistema utilizando el comando `df -h`, usa expresiones regulares para identificar los porcentajes de uso de disco en la salida del comando y al final filtra y muestra únicamente las particiones cuyo uso de disco sea mayot o igual al 80 %.

- La salida del comando `df -h` nos muestra el uso de disco del sistema como se puede ver en la siguiente imagen.

```
+------------------------+
|USO DE DISCO DEL SISTEMA|
+------------------------+
Filesystem       Size   Used  Avail Capacity iused     ifree %iused  Mounted on
/dev/disk1s5s1  119Gi   14Gi   44Gi    25%  502388 457153560    0%   /
devfs           190Ki  190Ki    0Bi   100%     658          0  100%   /dev
/dev/disk1s4    119Gi  4.0Gi   44Gi     9%       4 457153560    0%   /System/Volumes/VM
/dev/disk1s2    119Gi  314Mi   44Gi     1%    1414 457153560    0%   /System/Volumes/Preboot
/dev/disk1s6    119Gi  1.0Mi   44Gi     1%      18 457153560    0%   /System/Volumes/Update
/dev/disk1s1    119Gi   56Gi   44Gi    57%  717169 457153560    0%   /System/Volumes/Data
map auto_home     0Bi    0Bi    0Bi   100%       0          0  100%   /System/Volumes/Data/home
```

- La expresión regular creada para poder identificar las líneas en las que hay porcentajes es `df␣-h␣|␣grep␣-E␣"[0-9]{1,3}%"`. Esta expresión busca números que tengan entre 1 y 3 dígitos seguidos del símbolo %.

```
+------------------------------+
|Identificaciones de porcentajes|
+------------------------------+
/dev/disk1s5s1  119Gi   14Gi   44Gi    25%  502388 457153560    0%   /
devfs           190Ki  190Ki    0Bi   100%     658          0  100%   /dev
/dev/disk1s4    119Gi  4.0Gi   44Gi     9%       4 457153560    0%   /System/Volumes/VM
/dev/disk1s2    119Gi  314Mi   44Gi     1%    1414 457153560    0%   /System/Volumes/Preboot
/dev/disk1s6    119Gi  1.0Mi   44Gi     1%      18 457153560    0%   /System/Volumes/Update
/dev/disk1s1    119Gi   56Gi   44Gi    57%  717169 457153560    0%   /System/Volumes/Data
map auto_home     0Bi    0Bi    0Bi   100%       0          0  100%   /System/Volumes/Data/home
```

- Para obtener las particiones que tienen uso de disco del sistema mayor o igual al 80 % es `df␣-h␣|␣grep␣-E␣␣"([8-9][0-9]|100)%"`