

Compiladores

Carlos Eduardo Atencio Torres

Semestre 2020-I

Práctica 2 – Regex e introducción a FLEX

Objetivo:

Construir un analizador léxico utilizando la herramienta FLEX

Herramientas:

FLEX: Es una herramienta que nos permite generar analizadores léxicos. Tiene su propia sintaxis y pide como requisito tener un conocimiento básico sobre manipulación de expresiones regulares.

Para instalar en linux:

```
sudo apt-get install flex
```

Para instalar en windows:

<http://gnuwin32.sourceforge.net/packages/flex.htm>

1. EXPRESIONES REGULARES

También llamadas REGEX, se trata de una secuencia de caracteres que conforman un patrón de búsqueda. Usualmente usadas para buscar o buscar y reemplazar cadenas.

Wikipedia

Símbolos más comunes

?	Ninguna o una ocurrencia
*	Ninguna o varias ocurrencias
+	Mínimo una ocurrencias
{n}	Exactamente n ocurrencias
	“o”
[]	Opciones
()	Agrupación

- . El punto significa “cualquier cosa”
- \$ Fin de línea.

Símbolos comodines (*):

- \d dígito
- \s cadena

(*) Los símbolos comodines pueden o no estar presentes en el lenguaje de programación escogido.

Caracteres especiales

Si queremos por ejemplo reconocer “.” como punto, debemos de utilizar el caracter “\”. Por ejemplo: \. \? \+

OBS: Tener cuidado al colocarlo en el patrón de un lenguaje de programación pues el “\” debe colocarse con “\\” (doble), así: “\\.” “\\?” “\\+”

Las expresiones regulares pueden ser programadas en diferentes lenguajes de programación. Por ejemplo para C++ a partir de la std 11 ya tiene soporte para regex:

1. Verificar si una cadena está conformada por números:

```
import re

p = re.compile('[0-9]+$')

if re.match(p):
    print("SI ES UN NUMERO")
else:
    print("NO ES UN NUMERO")
```

2. Obtener todas las palabras de un texto:

```
import re

regstr = '[a-z]+'
text = 'abc cde 777ghi jkl999mno'
groups = re.findall(regstr, text)
```

```
for group in groups:  
    print( group )
```

Ejercicios:

1. Reconocer una dirección ip.
2. Reconocer el nombre de una variable (que no comience por número)

2. FLEX

1) Crearemos un ejemplo que será un programa para borrar todos los espacios en blanco y tabuladores de los extremos de las líneas.

- En un archivo llamado test.l (test punto ele), colocaremos la siguiente sintaxis:

```
1. %%  
2. [b\t]+$ ;
```

(desconsidere los números)

Entonces la línea 1, %% significa que el programa comienza allí.

2) Luego ejecutaremos en consola:

flex test.l

si no hubiera errores aparecerán el archivo **lex.yy.c**

3) Compilares el archivo lex como una librería:

gcc lex.yy.c -L/lib -lfl

4) Esto nos creará un archivo ejecutable (**a.out** en linux o **a.exe** en windows)

5) Creamos un archivo de prueba.txt:

```
Este es un _ _ _ _  
programa con _ _ _  
varias tabulaciones _ _  
al final _
```

(los sub-guiones en realidad son tabulaciones, solo se colocaron para mejorar la visualización)

6) Ejecutar:

```
./a.out < prueba.txt > salida.txt
```

7). Verificamos que el archivo de salida no tenga tabulaciones al final

Ejercicio

1. En un ejercicio similar, quite los números al final de la línea
2. En un ejercicio similar, cambie todas las letras repetidas por una sola.