

Análisis léxico con Flex

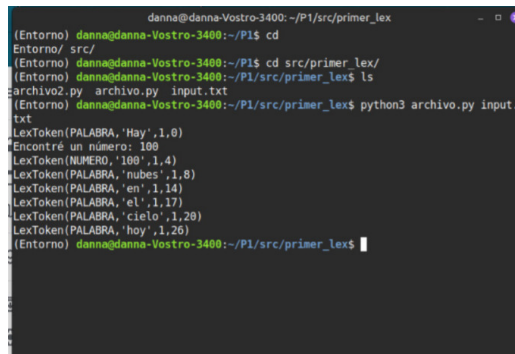
Márquez Corona Danna Lizette

- **¿Qué ocurre si agregamos una regla simple como `t_espacio = r' +'` y nada más? (0.5 pts)**
Pues tenemos que agregar "espacio" a la lista de tokens también para poder ejecutarlo, ya que si pones una regla tienes que ponerlo en la lista de tokens.
- **¿Qué ocurre si quitamos algún elemento de la lista de tokens? (0.5 pts)**
Nos da un error que dice que se definió una regla, pero que no hay un token para esta.
Si eliminamos un elemento de la lista de tokens, como por ejemplo NUMERO, el analizador léxico ya no será capaz de reconocer ese tipo de token en la entrada, por lo que cualquier número presente en el código fuente no se procesaría.
- **¿Cómo podemos calcular la posición en columna en caso de un error léxico? (0.5 pts)**
.
- **¿Qué significa el valor que se aloja en `t.value`? (0.5 pts)**
Contiene el texto que corresponde al token que se ha reconocido en el código fuente. Por ejemplo, si el analizador encuentra un número, `t.value` almacenará ese número en forma de cadena y así para cada tipo de token.
- **¿Qué pasa al ejecutar el programa e introducir cadenas de caracteres y de dígitos sin espacios en el archivo de entrada? (0.5 pts)**
Se intentará tokenizar cada uno de los caracteres, ya que las reglas `t_PALABRA` y `t_NUMERO` están definidas para detectar esos patrones y no se generaría un error, ya que puede manejar ambos tipos sin la necesidad de espacios.

- ¿Qué ocurre si introducimos caracteres como "*" en el archivo de entrada? (0.5 pts)

Si no hay una regla para manejar el asterisco se genera un error léxico, ya que el carácter no coincide con ninguna de las expresiones definidas para los tokens, así la función `t_error` se ejecutaría y el carácter `*` sería ignorado.

- Modificar al código anterior en un archivo nuevo, de tal manera que ejecute una acción léxica al detectar lo siguiente: (2 pts)
 - La expresión regular para los hexadecimales en lenguaje C.
 - 5 palabras reservadas del lenguaje Python.
 - Los identificadores válidos del lenguaje Java, con longitud máxima de 32 caracteres (Sugerencia: use el operador `m,n`).
 - Los espacios en blanco.



```
danna@danna-Vostro-3400: ~/P1/src/primer_lex
(Entorno) danna@danna-Vostro-3400:~/P1$ cd
Entorno/ src/
(Entorno) danna@danna-Vostro-3400:~/P1$ cd src/primer_lex/
(Entorno) danna@danna-Vostro-3400:~/P1/src/primer_lex$ ls
archivo2.py  archivo.py  input.txt
(Entorno) danna@danna-Vostro-3400:~/P1/src/primer_lex$ python3 archivo.py input.
txt
LexToken(PALABRA,'Hay',1,0)
Encontre un numero: 100
LexToken(NUMERO,'100',1,4)
LexToken(PALABRA,'nubes',1,8)
LexToken(PALABRA,'en',1,14)
LexToken(PALABRA,'el',1,17)
LexToken(PALABRA,'cielo',1,20)
LexToken(PALABRA,'hoy',1,26)
(Entorno) danna@danna-Vostro-3400:~/P1/src/primer_lex$
```

Figure 1: Ejercicio ejecutado en entorno