**Compiladores Laboratorio**
**4**
**Escáner basado en autómata**

**Programa**

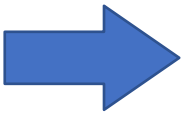
Se presenta un escáner léxico basado en autómatas, diseñado para reconocer los siguientes tokens:

| Categoría | Token | Descripción | Ejemplo |
|---|---|---|---|
| **Operadores** | PLUS | Suma + | 3 + 2 |
| | MINUS | Resta - | 5 - x |
| | MULT | Multiplicación * | 2 * y |
| | DIV | División / | 10/2 |
| | POW | Potencia ** | 2 ** 3 |
| **Funciones** | SIN | Función seno sin | sin(x) |
| | COS | Función coseno cos | cos(y) |
| | LOG | Función logaritmo log | log(10) |
| **Constantes** | PI | Constante pi | Pi |
| | E | Constante e | e |
| **Identificadores** | NUM | Números (int o float) | 3.14, 42 |
| | ID | Identificadores (variables) | x, var1 |
| **Símbolos** | LPAREN | Paréntesis izquierdo ( | (x + 1) |
| | RPAREN | Paréntesis derecho ) | sin(x) |
| | SEMICOLON | Punto y coma ; (fin de expresión) | x = 5; |
| **Especiales** | END | Fin de la entrada (\0) | - |
| | ERR | Error léxico (carácter desconocido) | @, #, etc. |

Ejemplo:

**input.txt**

```
123.45+67
2**10
(5+6)
log(pi/2);
```



**nextToken()**

```
next token NUM(123.45)
next token PLUS(+)
next token NUM(67)
next token NUM(2)
next token POW(**)
next token NUM(10)
next token LPAREN
next token NUM(5)
next token PLUS(+)
next token NUM(6)
next token RPAREN
next token LOG
next token LPAREN
next token PI
next token DIV(/)
next token NUM(2)
next token SEMICOLON(;)
last token END
```

**nextTokenWithLine()**

```
=== TABLA DE TOKENS ===
+------------+------------+-------+
| Lexema     | Token      | Linea |
+------------+------------+-------+
| 123.45     | NUM        | 1     |
| +          | PLUS       | 1     |
| 67         | NUM        | 1     |
| 2          | NUM        | 2     |
| **         | POW        | 2     |
| 10         | NUM        | 2     |
| (          | LPAREN     | 3     |
| 5          | NUM        | 3     |
| +          | PLUS       | 3     |
| 6          | NUM        | 3     |
| )          | RPAREN     | 3     |
| log        | LOG        | 4     |
| (          | LPAREN     | 4     |
| pi         | PI         | 4     |
| /          | DIV        | 4     |
| 2          | NUM        | 4     |
| )          | RPAREN     | 4     |
| ;          | SEMICOLON  | 4     |
+------------+------------+-------+
```

**Problema 1**

Agregue comentarios de línea utilizando el símbolo #.

```
123.45+67 #hola mundo
2**10
(5+6) #esto debe ser ignorado
log(pi/2);
```

**Problema 2**

Modifique el escáner para permitir la entrada de números en formato binario, octal y hexadecimal. Debe crear los tokens BIN, OCT y HEX.

```
0b1001 + 0b11
0o123 + 0o15
0x11 + 0x56;
0xA3 + 0xB15;
```

Ojo: Considere en el sistema hexadecimal A=10, B=11 ,etc.

**Problema 3**

Agregue tokens para manejar la asignación de variables (=), impresión (print), expresiones booleanas (True, False) y operadores de relación (<, >, ==, !=). Deben crear los tokens correspondientes para cada uno de estos casos.

```
x = 5
y = True
z = (3<5) + (4 !=5)
print(x+6);
```

**OBS**: Solo modificar el método nextToken() o el método nextTokenWithLine(); y eliminar sus respectivos scanners en la función main().