

- 1) Crear un analizador Léxico en el Lenguaje de programación de su preferencia, pero utilizando FLEX.

```
ejemplo1.l
You, hace 8 minutos | 1 author (You)
1  %{
2  #include <stdio.h>
3  %}
4
5  %%
6
7  "if"      { printf("Palabra reservada: IF\n"); }
8  "else"    { printf("Palabra reservada: ELSE\n"); }
9  "[0-9]+"  { printf("Numero entero: %s\n", yytext); }
10 "[a-zA-Z]+" { printf("Identificador: %s\n", yytext); }
11 "=="      { printf("Operador de igualdad\n"); }
12 "="      { printf("Operador de asignacion\n"); }
13 "("       { printf("Parentesis de apertura\n"); }
14 ")"       { printf("Parentesis de cierre\n"); }
15 ";"       { printf("Punto y coma\n"); }
16 "{"       { printf("Llave de apertura\n"); }
17 "}"       { printf("Llave de cierre\n"); }
18 "+"       { printf("Operador de suma\n"); }
19 "-"       { printf("Operador de resta\n"); }
20 "*"       { printf("Operador de multiplicacion\n"); }
21 "/"       { printf("Operador de division\n"); }
22 [ \t\n]   { /* Ignorar espacios en blanco, tabulaciones y saltos de línea */ }
23 .         { printf("Caracter no reconocido: %s\n", yytext); }
24
25  %%
26
27 int main() {
28     yylex();
29     return 0;
30 }
31 You, hace 18 minutos • Cambio final ...
32
```

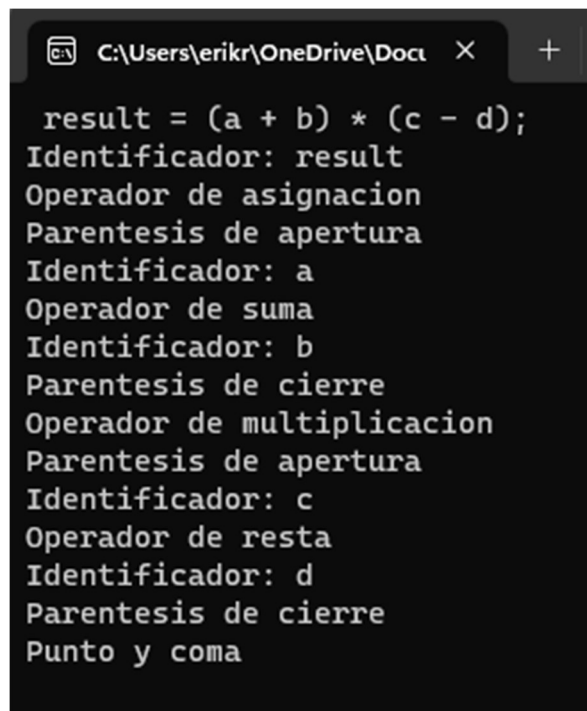
2) Documentar Lenguaje utilizado del analizador Léxico para realizar pruebas.

El lenguaje reconocido por este analizador léxico sigue las siguientes reglas básicas:

1. Palabras Reservadas: Se reconocen las palabras reservadas "if" y "else".
2. Identificadores: Se reconocen secuencias de letras como identificadores.
3. Números Enteros: Se reconocen secuencias de dígitos como números enteros.
4. Operadores de Comparación y Asignación: Se reconocen los operadores "==" (igualdad) y "=" (asignación).
5. Paréntesis y Llaves: Se reconocen los símbolos "(", ")", "{", y "}".
6. Punto y Coma: Se reconoce el símbolo ";".
7. Operadores Aritméticos: Se reconocen los operadores "+", "-", "*", y "/".
8. Ignorar Espacios en Blanco: Se ignoran espacios en blanco, tabulaciones y saltos de línea.

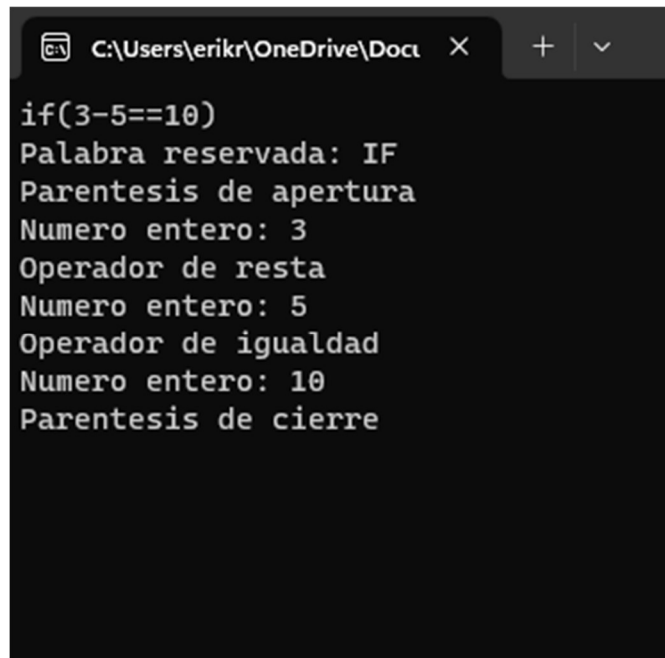
Prueba1:

```
result = (a + b) * (c - d);
```



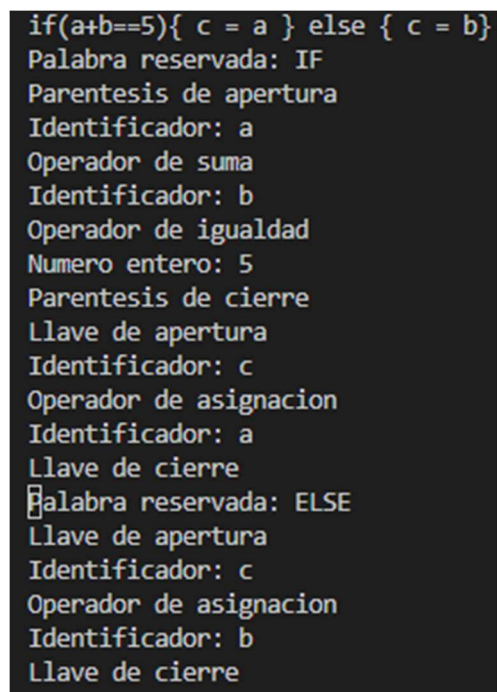
```
C:\Users\erikr\OneDrive\Docu  X  +  
result = (a + b) * (c - d);  
Identificador: result  
Operador de asignacion  
Parentesis de apertura  
Identificador: a  
Operador de suma  
Identificador: b  
Parentesis de cierre  
Operador de multiplicacion  
Parentesis de apertura  
Identificador: c  
Operador de resta  
Identificador: d  
Parentesis de cierre  
Punto y coma
```

Prueba 2: `if(3-5==10)`



```
if(3-5==10)
Palabra reservada: IF
Parentesis de apertura
Numero entero: 3
Operador de resta
Numero entero: 5
Operador de igualdad
Numero entero: 10
Parentesis de cierre
```

Prueba 3:



```
if(a+b==5){ c = a } else { c = b}
Palabra reservada: IF
Parentesis de apertura
Identificador: a
Operador de suma
Identificador: b
Operador de igualdad
Numero entero: 5
Parentesis de cierre
Llave de apertura
Identificador: c
Operador de asignacion
Identificador: a
Llave de cierre
Palabra reservada: ELSE
Llave de apertura
Identificador: c
Operador de asignacion
Identificador: b
Llave de cierre
```

Ejecución del Analizador Léxico

- 1) Debemos compilar el programa con **flex ejemplo1.l**
- 2) Compilar el código C generado con `gcc lex.yy.c -lfl`
- 3) Ejecutar el analizador léxico desde la consola y abriendo el ejecutable generado

Ejemplo:

```
PS C:\Users\erikr\OneDrive\Documentos\ESTUDIOS\UTESA\UTESA 3-2023\COMPILADORES\Analixador-Lexico> flex ejemplo1.l
PS C:\Users\erikr\OneDrive\Documentos\ESTUDIOS\UTESA\UTESA 3-2023\COMPILADORES\Analixador-Lexico> gcc lex.yy.c -lfl
PS C:\Users\erikr\OneDrive\Documentos\ESTUDIOS\UTESA\UTESA 3-2023\COMPILADORES\Analixador-Lexico> ./a
```

Github: <https://github.com/kelok3rik/Analixador-Lexico>