UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Traductores de Lenguajes II

Reporte de práctica

Nombre del alumno:	Luis Jaime Portillo Correa
Profesor:	Erasmo Gabriel Martínez Soltero
Título de la práctica:	"Analizador Léxico"
Fecha:	28 de febrero del 2023

Introducción

Un analizador léxico o analizador lexicográfico (en inglés scanner o tokenizer) es la primera fase de un compilador, consistente en un programa que recibe como entrada el código fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de tokens (componentes léxicos) o símbolos. Estos tokens sirven para una posterior etapa del proceso de traducción, siendo la entrada para el analizador sintáctico (en inglés parser).

La especificación de un lenguaje de programación a menudo incluye un conjunto de reglas que definen el léxico. Estas reglas consisten comúnmente en expresiones regulares que indican el conjunto de posibles secuencias de caracteres que definen un token o lexema.

Resultados

Para llevar a cabo esta práctica me basé principalmente en diccionarios pues se me hizo la manera menos complicada de implementarlo, sin embargo, no dejé las comparaciones totalmente en manos de las keys de los diccionarios, sino que también comparé las características de los lexemas para definir que tipo de token podía corresponderle antes de ingresar a su tipo de diccionario.

```
count=0
program = a.split("\n")
for line in program:
    count = count + 1
    print("line #" , count, "\n" , line)

    status=0
    tokens=line.split(' ')
    print("Tokens are " , tokens)
    print("Line#", count, "properties \n")
    for token in tokens:
        # Comprobar si el nombre de la variable comienza con una letra o un guión bajo (_)
        if not token[0].isalpha() and token[0] != '_':

            if token[0]=="<" or token[0]==">" or token[0]=="=" or token[0]=="!":
                if token in operatorsRel_key:
                    print("lexema: ",token, "is", operatorsRel[token])
                    tabla.insert("",END,text=token,values=(operatorsRel[token],codigos[operatorsRel[token]]))
                    status = 1
            if token in operators_key:
                print("lexema: ",token, "is", operators[token])
                tabla.insert("",END,text=token,values=(operators[token],codigos[operators[token]]))
                status=1
            if token in punctuation_symbol_key:
                print ("lexema: ",token, "is", punctuation_symbol[token])
                tabla.insert("",END,text=token,values=(punctuation_symbol[token],codigos[punctuation_symbol[token]]))
                status=1
            if token.isdigit() or isNumeric(token):
                print("lexema: ",token, "is", "constante")
                tabla.insert("",END,text=token,values=("constante","13"))

        # Comprobar si el nombre de la variable solo contiene letras, números y guiones bajos (_)
        elif not all(caracter.isalnum() or caracter == '_' for caracter in token):
            print ("lexema: ", token, "error")
            tabla.insert("",END,text=token,values=("Error","Error"))

        # Comprobar si el nombre de la variable es una palabra reservada de Python
```

Figura 1:

Aqui es donde realizó la mayor parte de las comparaciones, también tengo un diccionario que cuenta con las claves de cada token, que son las que el profe proporcionó en classroom.

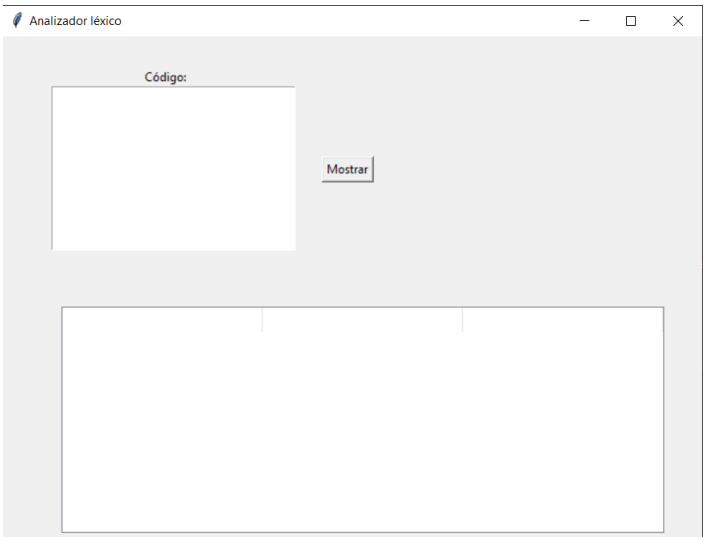

Figura 2: Para la parte gráfica implementé una especie de robot tuerto que recibirá el código en una cuadro de texto y tras presionar el botón mostrar, nos imprimirá en la tabla de abajo nuestro código posterior al análisis léxico.

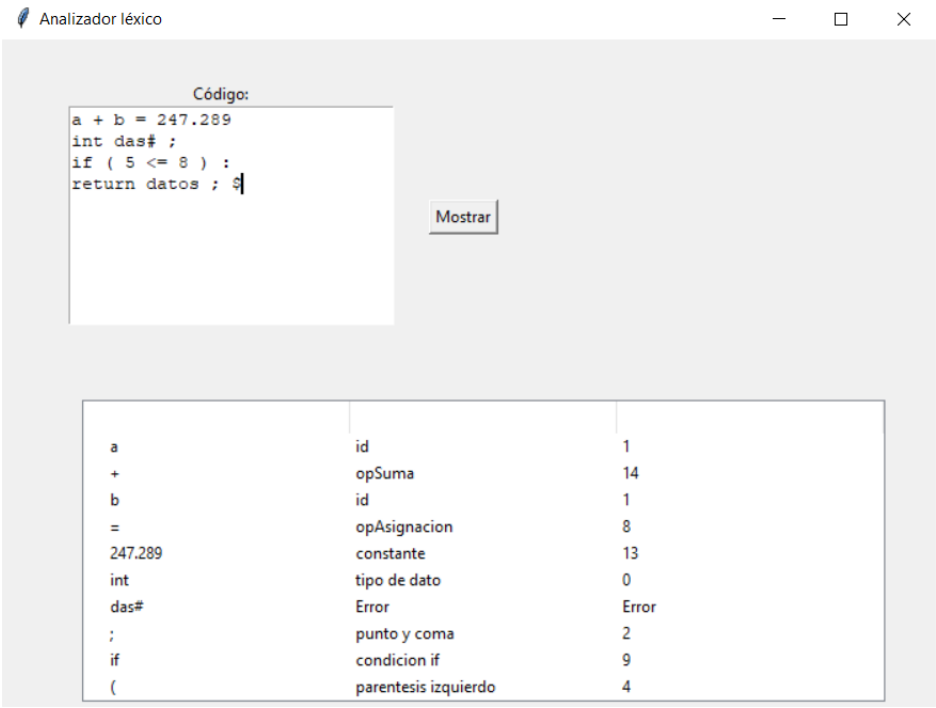

Figura 3: Ahora agregaré un fragmento de código para ponerlo a prueba y mostraré los resultados en pantalla. Podemos observar que los datos han sido separados mostrando en la tabla su lexema, token y código.

Figura 4: En esta parte de la ejecución se muestran en pantalla los datos restantes del anterior código, que serían los de la tercera línea hacía abajo, se observa que el programa está funcionando correctamente.

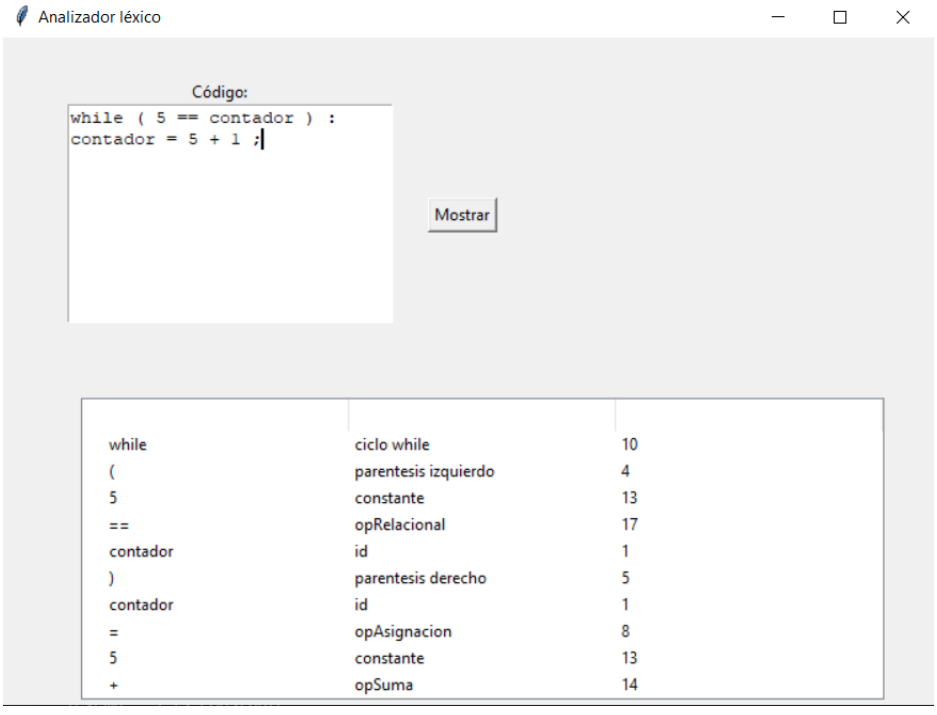

Figura 5: Ejemplo de ejecución con otro código.

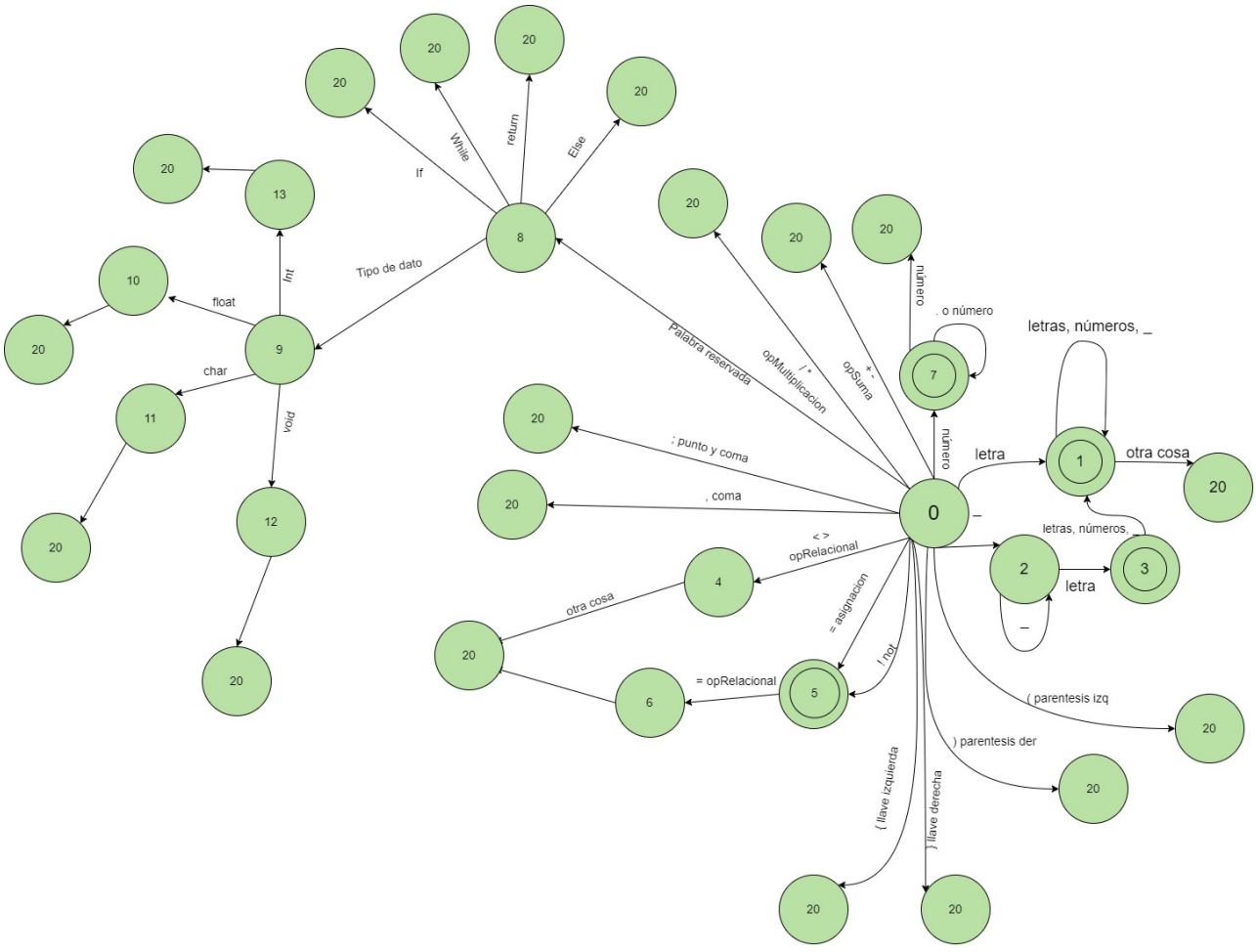

Figura 6: Autómata de Analizador Léxico.