

UNIVERSIDAD DE GUADALAJARA



CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Traductores de Lenguajes II

Reporte de práctica

Nombre del alumno:	Guillermo Ortiz Macías
Profesor:	Erasmus Gabriel Martínez Soltero
Título de la práctica:	"Análisis de cadena con expresiones regulares"
Fecha:	3 Septiembre 2019

Introducción

Esta tarea busca desarrollar un analizador léxico que sea capaz de detectar tokens y clasificarlos. El programa recorre código ingresado por el usuario y clasifica cada uno de los tokens tales como while, if, else, o bien operadores relacionales y matemáticos.

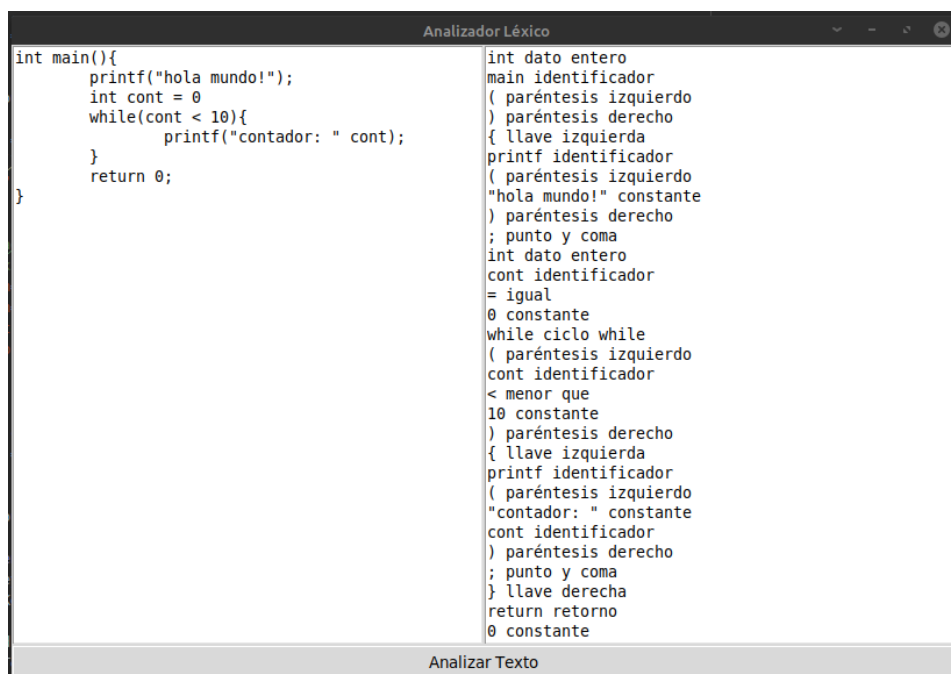
Metodología

El analizador se realizó con ayuda del lenguaje de programación Python. El programa funciona mediante un ciclo principal el cual va a recorrer todo el código que ingresó el usuario, formando palabras y buscando tokens. El programa lee el siguiente carácter del código y busca si es un token de un solo carácter (+, -, *, /, =, etc), si no lo es concatena el carácter a una palabra que se va formando a cada iteración del ciclo, pero si sí es un token primero analiza la palabra que ya se formó y la clasifica, y después clasifica el carácter que está leyendo tomando en cuenta que puede ser un token

de dos caracteres (!=, ==, etc). De esta forma el código va analizando y clasificando de forma secuencial los tokens del código.

Resultados

El usuario ingresa código en el cuadro izquierdo, le da click al botón en la parte inferior y el programa identifica y clasifica los tokens del código:



```
int main(){
    printf("hola mundo!");
    int cont = 0;
    while(cont < 10){
        printf("contador: " cont);
    }
    return 0;
}
```

```
int dato entero
main identificador
( paréntesis izquierdo
) paréntesis derecho
{ llave izquierda
printf identificador
( paréntesis izquierdo
"hola mundo!" constante
) paréntesis derecho
; punto y coma
int dato entero
cont identificador
= igual
0 constante
while ciclo while
( paréntesis izquierdo
cont identificador
< menor que
10 constante
) paréntesis derecho
{ llave izquierda
printf identificador
( paréntesis izquierdo
"contador: " constante
cont identificador
) paréntesis derecho
; punto y coma
} llave derecha
return retorno
0 constante
```

Analizar Texto

Conclusiones

En análisis léxico es la primera fase de un compilador y puedo ver la utilidad del mismo pues el compilador necesita identificar las distintas partes del código para darles su tratamiento adecuado y también determinar si la sintaxis del mismo es correcta o no. El compilador en una etapa siguiente podría ver errores de sintaxis con sucesiones de tokens no validas, como un tipo de dato seguido de una constante, sin tener un identificador en medio.

Referencias

Diseño de compiladores:

https://www.tutorialspoint.com/compiler_design/compiler_design_lexical_analysis.htm

Consultado el día 9 de septiembre del año 2019.