**INSTITUTO POLITÉCNICO NACIONAL**

ESCUELA SUPERIOR DE CÓMPUTO

Compiladores

Alumnos:

García Gasca Axel Gabriel

Taboada Montiel Enrique

Profesora:

Albortante Morato Cecilia

**Práctica 2 – Analizador Léxico**

**Grupo: 5CM4**

# Introduccion

Un analizador léxico, también conocido como scanner o tokenizador, es una de las primeras fases del proceso de compilación o interpretación de un lenguaje de programación. Su objetivo principal es tomar una secuencia de caracteres (es decir, el código fuente) y dividirla en componentes más pequeños llamados tokens, que son las unidades mínimas de significado para un lenguaje de programación. Estos tokens se pasan luego a la siguiente fase del compilador o intérprete, que normalmente es el analizador sintáctico\*o parser.

Un analizador léxico lee el código fuente carácter por carácter, formando y reconociendo secuencias válidas (tokens) de acuerdo con las reglas de un lenguaje. Los tokens pueden ser palabras clave, identificadores, operadores, números, símbolos especiales, etc. Por ejemplo, en el lenguaje de programación C, el código `int x = 10;` podría descomponerse en los siguientes tokens:

- `int`: palabra clave

- `x`: identificador

- `=`: operador de asignación

- `10`: número entero

- `;`: símbolo especial

Componentes básicos de un analizador léxico

1. Tabla de símbolos: Una estructura de datos donde se almacenan identificadores y palabras clave, permitiendo distinguir entre ellos cuando se encuentran en el código fuente.

2. Autómata finito: La mayoría de los analizadores léxicos se construyen utilizando autómatas finitos deterministas (AFD), que permiten reconocer patrones de cadenas (como palabras clave o números) mediante un conjunto de estados y transiciones.

3. Tokens: Los resultados que devuelve el analizador léxico. Cada token tiene un tipo (p. ej., identificador, número, operador) y un valor asociado (p. ej., el nombre del identificador o el valor del número).

Proceso de construcción de un analizador léxico

1. Definir el alfabeto y los tokens: El primer paso es identificar qué símbolos forman parte del alfabeto (p. ej., letras, dígitos, operadores) y los tokens válidos que deben reconocerse (p. ej., palabras clave, números, operadores).

2. Construir las reglas léxicas: Estas reglas se definen mediante expresiones regulares o diagramas de estados que describen cómo reconocer tokens. Por ejemplo:

- Identificadores: cualquier secuencia de letras y dígitos que comience con una letra.

- Números: una secuencia de dígitos.

- Operadores: símbolos como `+`, `-`, `\*`, `/`.

3. Implementar el autómata finito: El analizador léxico se puede modelar como un autómata finito determinista (AFD), donde:

- Los estados representan la etapa actual de reconocimiento.

- Las transiciones representan el paso de un estado a otro basado en el carácter de entrada.

- Los estados finales indican que se ha reconocido un token completo.

4. Gestionar los errores: Es crucial que el analizador léxico tenga mecanismos para detectar secuencias de caracteres que no pertenecen a ningún token válido. Esto es conocido como \*\*manejo de errores léxicos\*\*.

# Desarrollo

El constructor \_\_init\_\_ inicializa las estructuras necesarias para un automáta: un diccionario de transiciones, un conjunto de estados, el alfabeto, el estado inicial, los estados de aceptación y el estado actual.

Texto

Descripción generada automáticamente

Luego llama a cargar automata para leer la configuración desde un archivo, el archivo contiene toda la configuración del automáta. Es importante mencionar que a partir de la quinta línea, carga las transiciones. Cada transición tiene un estado actual, una entrada y un estado siguiente. Estas transiciones se almacenan en el diccionario self.transiciones

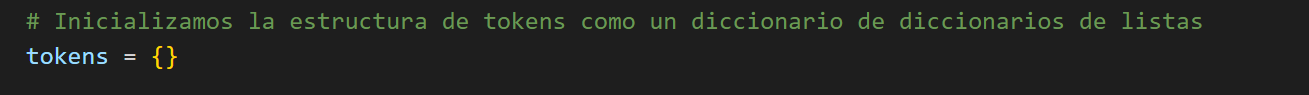
Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

También se define un diccionario global tokens, que almacenará las cadenas procesadas, clasificadas por tipo



Después tenemos la funcion que añade una cadena evaluada al diccionario de tokens, categorizándola bajo el tipo correspondiente.

Texto

Descripción generada automáticamente

La funcion transición maneja las transiciones del automata. Dado un carácter de entrada, verifica si existe una transición válida y cambia el estado del automáta en consecuencia. También clasifica la entrada como número, letra, operador, etc.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

La funcion de inicio procesa la cadena de entrada carácter por carácter, realizando transiciones en el automáta y almacenando las cadenas evaluadas (tokens) según el tipo. Reinicia el automáta cuando se completa un token y lo almacena usando agregar\_token.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

# Resultados

Ejemplo con números

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

# Conclusión

Al realizar esta práctica reforzamos y aprendimos más sobre el analizador léxico, ya que es un elemento importante de un compilador, que su función es reconocer números, cadenas, palabras propias del lenguaje.

Continuamos avanzando para poder crear nuestro propio compilador y ya tenemos una parte importante de éste mismo, gracias a la práctica realizada y las diferentes clases que se impartieron en el aula de clase.

# Referencias

Marciszack, M. M. (s.f.). *Máquinas de Estados: Autómatas Finitos Deterministas (AFD)*. Obtenido de GHD: https://www.institucional.frc.utn.edu.ar/sistemas/ghd/T-M-AFD.htm