JMVL

Avances en la Construcción de tu Traductor II

Alumno: Villalobos Lázaro Juan Manuel

Código: 216109185

14/10/2024

Sección D06

Maestra: MICHEL EMANUEL LOPEZ FRANCO

Materia: Seminario de Traductores de Lenguajes 2

Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingeniería

Villalobos Lázaro Juan Manuel

Introducción

Este documento presenta una descripción del diseño y la implementación del analizador léxico y sintáctico desarrollado en Python utilizando la biblioteca Tkinter para la interfaz gráfica. El analizador se encarga de procesar cadenas de texto que representan código fuente y de identificar los componentes léxicos y su estructura sintáctica. Se abordarán las partes importantes del código, así como la gestión de errores léxicos y sintácticos.

Contenido

El código se estructura en varias funciones que cumplen diferentes roles en el proceso de análisis. A continuación, se describen las partes más relevantes:

Función analizador(cadena)

Esta función se encarga de realizar el análisis léxico del texto ingresado. Su diseño incluye:

Variables principales:

elementos: lista que almacena los tokens y sus lexemas.

estado: variable que representa el estado actual del autómata.

indice: índice para recorrer la cadena de entrada.

Bucle de procesamiento:

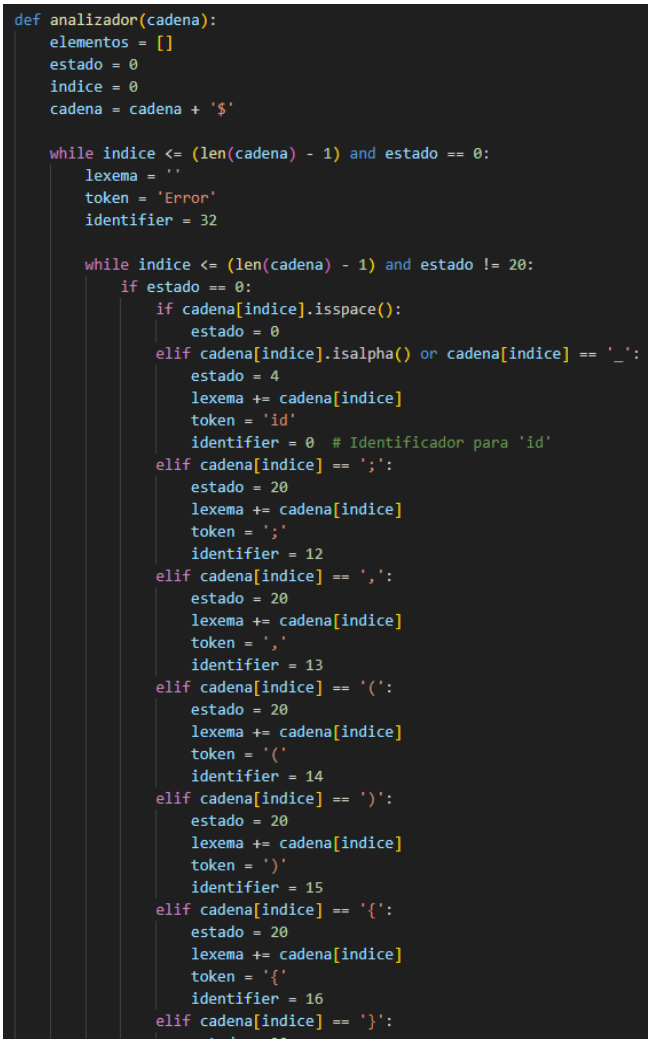
Se utiliza un bucle while que recorre la cadena hasta encontrar un símbolo de finalización ($). Dependiendo del carácter actual, se actualiza el estado y se determina el tipo de token (identificador, constante, operador, etc.).

Detección de errores:

Si un carácter no se reconoce, se clasifica como un error léxico y se agrega a los elementos con el token 'error'.

Asignación de palabras clave:

Al final del análisis léxico, se verifica si los lexemas coinciden con palabras clave predefinidas y se actualizan los tokens y los identificadores en consecuencia.



```
def analizador(cadena):
    elementos = []
    estado = 0
    indice = 0
    cadena = cadena + '$'

    while indice <= (len(cadena) - 1) and estado == 0:
        lexema = ''
        token = 'Error'
        identifier = 32

        while indice <= (len(cadena) - 1) and estado != 20:
            if estado == 0:
                if cadena[indice].isspace():
                    estado = 0
                elif cadena[indice].isalpha() or cadena[indice] == '_':
                    estado = 4
                    lexema += cadena[indice]
                    token = 'id'
                    identifier = 0  # Identificador para 'id'
                elif cadena[indice] == ';':
                    estado = 20
                    lexema += cadena[indice]
                    token = ';'
                    identifier = 12
                elif cadena[indice] == ',':
                    estado = 20
                    lexema += cadena[indice]
                    token = ','
                    identifier = 13
                elif cadena[indice] == '(':
                    estado = 20
                    lexema += cadena[indice]
                    token = '('
                    identifier = 14
                elif cadena[indice] == ')':
                    estado = 20
                    lexema += cadena[indice]
                    token = ')'
                    identifier = 15
                elif cadena[indice] == '{':
                    estado = 20
                    lexema += cadena[indice]
                    token = '{'
                    identifier = 16
                elif cadena[indice] == '}':
```

Función analizar_codigo()

Esta función se encarga de gestionar la interacción con la interfaz gráfica y coordinar el análisis. Sus componentes clave son:

Recopilación de código fuente:

Obtiene el texto ingresado en la caja de texto y lo envía a la función analizador.

Carga de reglas y tabla:

Lee archivos que contienen reglas y una tabla para el análisis sintáctico. Estos archivos se procesan y almacenan en listas para su uso posterior.

Mostrar resultados:

Inserta los resultados del análisis léxico en una caja de texto y llama a la función analizador_sintactico para llevar a cabo el análisis sintáctico.

```
def analizar_codigo():
    codigo = codigo_text.get("1.0", "end-1c")
    resultados = analizador(codigo)
    resultado_text.config(state="normal")
    resultado_text.delete("1.0", "end")

    # Cargar reglas y tabla (omitido aquí, asumiendo que ya lo tienes)
    archivo_reglas = "GR2slrRulesId.txt"
    archivo_tabla = "GR2slrRulesId.txt"
    reglas = []
    tabla = []

    with open(archivo_reglas, newline='') as file:
        csv_reader = csv.reader(file, delimiter="\t")
        for row in csv_reader:
            fila = [int(element) for element in row]
            reglas.append(fila)

    with open(archivo_tabla, newline='') as file:
        csv_reader = csv.reader(file, delimiter="\t")
        for row in csv_reader:
            fila = [int(element) for element in row]
            tabla.append(fila)

    for row in tabla:
        del row[0]

    for elemento in resultados:
        resultado_text.insert("end", f"|Token: {elemento['token']} |Lexema: {elemento['lexema']} |Identifier: {elemento['identifier']}\n")

    analisis_sintactico = analizador_sintactico(resultados, reglas, tabla)
    resultado_text.config(state="disabled")

    mostrar_popup(analisis_sintactico)
```

Función analizador_sintactico(tokens, reglas, tabla)

Esta función se encarga de realizar el análisis sintáctico utilizando una tabla de análisis y una pila. Sus elementos clave son:

Pila y cola de tokens:

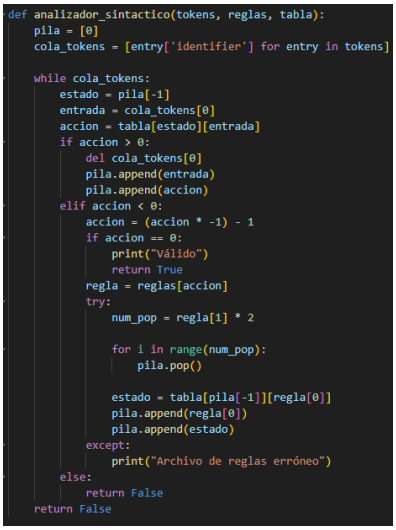
Se utiliza una pila para gestionar los estados y una cola para procesar los tokens obtenidos del análisis léxico.

Proceso de análisis:

Un bucle while que continúa mientras haya tokens por procesar. Dependiendo de la acción obtenida de la tabla, se realizan desplazamientos (shift) o reducciones (reduce).

Gestión de errores:

Si se detecta un error en la tabla de análisis, se captura la excepción y se imprime un mensaje informando sobre un archivo de reglas erróneo.



```
def analizador_sintactico(tokens, reglas, tabla):
    pila = [0]
    cola_tokens = [entry['identifier'] for entry in tokens]

    while cola_tokens:
        estado = pila[-1]
        entrada = cola_tokens[0]
        accion = tabla[estado][entrada]
        if accion > 0:
            del cola_tokens[0]
            pila.append(entrada)
            pila.append(accion)
        elif accion < 0:
            accion = (accion * -1) - 1
            if accion == 0:
                print("Válido")
                return True
            regla = reglas[accion]
            try:
                num_pop = regla[1] * 2

                for i in range(num_pop):
                    pila.pop()

                estado = tabla[pila[-1]][regla[0]]
                pila.append(regla[0])
                pila.append(estado)
            except:
                print("Archivo de reglas erróneo")
        else:
            return False
    return False
```

Función mostrar_popup(resultado)

Muestra un mensaje en pantalla que indica si el código ingresado es válido o no, utilizando cuadros de mensaje de Tkinter.

```
def mostrar_popup(resultado):
    if resultado:
        messagebox.showinfo("Resultado", "Código Válido")
    else:
        messagebox.showerror("Resultado", "Código inválido")
```

Manejo de Errores Léxicos y Sintácticos

El manejo de errores es una parte fundamental de cualquier analizador, y en este código se aborda de la siguiente manera:

Errores Léxicos

Identificación de errores:

Cuando se encuentra un carácter no reconocido durante el análisis léxico, se establece el estado en 20 (estado de error) y se clasifica el lexema como un error.

Almacenamiento de errores:

Los lexemas erróneos se añaden a la lista de elementos con el token 'error', permitiendo así que el analizador registre y notifique sobre los problemas en el código fuente ingresado.

Errores Sintácticos

Detección de errores:

Si se encuentra una acción no válida en la tabla de análisis, la función analizador_sintactico devuelve False, indicando un error en la estructura del código.

Manejo de excepciones:

En caso de que se produzca un error al intentar acceder a las reglas (por ejemplo, si el archivo de reglas está mal formateado), se captura la excepción y se imprime un mensaje que indica que el archivo de reglas es erróneo.

Conclusiones

El analizador léxico y sintáctico diseñado en este código está estructurado de manera que permite una fácil identificación de componentes léxicos y la verificación de la estructura sintáctica de una cadena de código. La implementación de una interfaz gráfica mediante Tkinter proporciona una experiencia de usuario intuitiva, y el manejo de errores léxicos y sintácticos se realiza de manera efectiva, lo que permite al usuario recibir información clara sobre la validez del código ingresado.