Integrantes:

Ariel Humberto Valle Escoto

Eduardo Quetzal Delgado Pimentel

Ricardo David López Arellano

Materia: Traductores de lenguajes II

2023b

Introducción

Un analizador sintáctico descendente comienza con el símbolo inicial de la gramática y trata de llegar a la cadena de entrada, descomponiéndola en componentes cada vez más pequeños según las reglas de la gramática.

En otras palabras, este enfoque analiza la cadena de entrada desde el símbolo inicial hacia las producciones terminales, siguiendo el camino que tomaría la derivación de la gramática.

En este contexto, una de las aplicaciones comunes de los analizadores sintácticos descendentes es la evaluación de expresiones aritméticas. Las expresiones aritméticas son fórmulas matemáticas que pueden involucrar números, operadores y paréntesis, y siguen reglas específicas de precedencia y asociatividad. Analizar y evaluar estas expresiones es esencial en áreas como las matemáticas computacionales, la ingeniería de software y la ciencia de datos.

Capturas

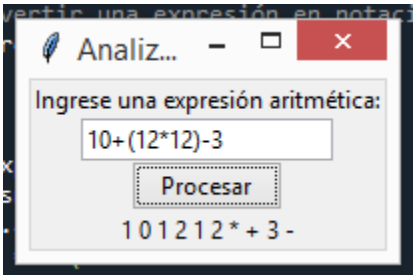

Ilustración 1

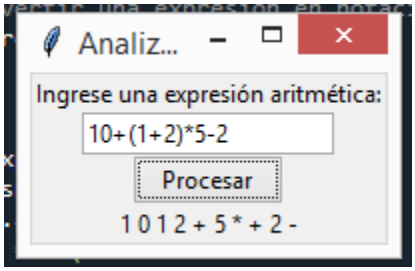

Ilustración 2

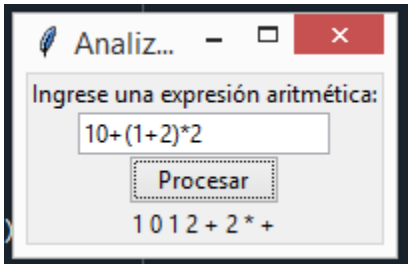

Ilustración 3

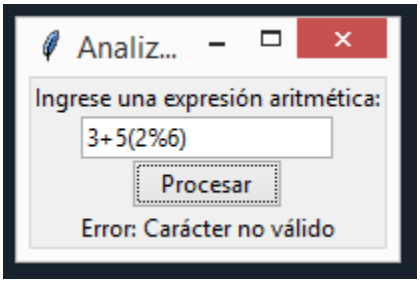

Ilustración 4

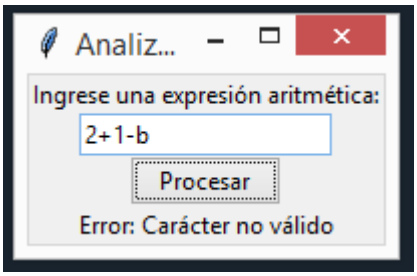

Ilustración 5

Código

```
import tkinter as tk

contador = 0
lenguaje = ''

def expr():
    t = term()
    return resto_expr(t)


def term():
    preanalisis = get_preanalisis()
    if preanalisis in ['0','1','2','3','4','5','6','7','8','9']:
        coincidir(preanalisis)
        return preanalisis
    elif preanalisis == '(':
        coincidir(preanalisis)
        resultado = expr()
        if get_preanalisis() == ')':
            coincidir(')')
            return f'{resultado}'
        else:
            raise Exception('Lenguaje no aceptado')

    else:
        raise Exception(‘Error carácter no válido’)

def resto_expr(her):
    global contador
    p = get_preanalisis()
    if p in ['+','-','*','/']:
        coincidir(p)
        term_t = term()
        ret = resto_expr(her+term_t+p)
        return ret
    elif p == ')':
        contador-=1
        return her
    else:
        return her


def coincidir(letra):
    pass

def get_preanalisis():
    global contador
```

```
    if(contador < len(lenguaje)):
        letra = lenguaje[contador]
        contador += 1
        return letra
def main():
    global lenguaje,contador
    lenguaje = entrada.get()
    contador = 0
    try:
        posfija = expr()
        resultado.set(posfija)
    except Exception as e:
        resultado.set(str(e))
ventana = tk.Tk()
ventana.title("Analizador sintactico")


etiqueta = tk.Label(ventana, text="Ingresa la expresión aritmética:")
etiqueta.pack()


entrada = tk.Entry(ventana)
entrada.pack()


calcular_button = tk.Button(ventana, text="Procesar", command=main)
calcular_button.pack()


resultado = tk.StringVar()
resultado_label = tk.Label(ventana, textvariable=resultado)
resultado_label.pack()

ventana.mainloop()
```

Conclusión

Este practica que realizamos nos ayudó a comprender cómo se pueden combinar las capacidades de procesamiento de texto de Python con las funcionalidades de las interfaces gráficas para crear herramientas interactivas. Además, que así mismo nos ayudó a comprender mejor las funciones que tiene que realizar un analizador sintáctico descendente.

En conclusión, los analizadores sintácticos descendentes son una parte esencial de los compiladores y analizadores léxicos. Estos analizadores siguen las reglas gramaticales de un lenguaje de programación desde la raíz hacia las hojas del árbol de análisis, descomponiendo la estructura sintáctica del código fuente. Su importancia radica en su capacidad para comprender y validar la sintaxis de los programas, lo que es crucial para garantizar la precisión y confiabilidad de los compiladores.