Implementación de Métodos Computacionales

Actividad Integradora 3.4 Resaltador de sintaxis

Erick Segura Sánchez

Alvaro Eduardo Lozano Medina

Jesús Guillermo Falcón Cardona

13/Mayo/2024

1. Introducción:

El análisis léxico es la primera etapa en el proceso de compilación de un programa. Consiste en escanear el código fuente y convertirlo en una secuencia de tokens, identificando las palabras reservadas, operadores, literales, comentarios, etc.

2. Implementación:

Para realizar el análisis léxico en Python, se utilizó el módulo re para definir expresiones regulares que coincidan con los distintos tokens léxicos. Se han definido categorías léxicas como palabras reservadas, operadores, literales, comentarios, variables, etc. El programa recibe un archivo fuente de Python, lo analiza y genera un documento HTML que resalta cada token con un color correspondiente a su categoría léxica.

3. Ejecución:

Asegurarse de tener Python instalado en tu sistema.

Copiar y pegar el código en un editor de texto como Sublime Text, Visual Studio Code o cualquier otro de la elección.

Guardar el archivo con el nombre "fuente.py" en la misma carpeta donde se está guardando este código.

Abrir una terminal o línea de comandos en el sistema operativo.

Navegar hasta la carpeta donde se guardó el archivo "fuente.py" usando el comando cd seguido de la ruta de la carpeta. Por ejemplo:

cd ruta/donde/guardaste/el/archivo

Una vez en la carpeta correcta, ejecutar el código Python con el siguiente comando:

python nombre_del_archivo.py

Si se está utilizando Python 3, el comando puede ser:

python3 nombre_del_archivo.py

Después de ejecutar el código, se ha creado un nuevo archivo llamado "highlighted_code.html" en la misma carpeta. Abrir este archivo en un navegador web para ver el código fuente resaltado con colores.

3. Ejemplo de Resultado:

Se adjunta un ejemplo del código fuente resaltado en HTML.

```
# Concatena caracteres si el estado no cambia o si el estado anterior es 0
if (prev_estado == 0) or (dict_estado[estado] == dict_estado[prev_estado]):
  palabra += token
  prev_estado = estado
elif estado != 9:
  # Maneja casos específicos de transición de estado
  if prev_estado == 6 and estado == 7:
    palabra += token
    prev_estado = 7
    continue
  if prev_estado == 8:
    for char in palabra:
      if char != '-':
        print(f"{char}\t{dict_simbolo[char]}")
        continue
    if palabra[-1] == '-':
      if dict_estado[estado] == 'Real' or dict_estado[estado] == 'Entero':
        palabra = '-' + token
        prev_estado = 4
      else:
```

4. Tiempo de Ejecución:

El programa tiene una complejidad de O(n) así que el tiempo de ejecución dependerá de la longitud del input. En caso de resaltar el programa de la actividad 3.2, este tiene un promedio de 0.006 segundos en tiempo de ejecución.

```
Tiempo de ejecución: 0.006005 segundos
```

5. Conclusiones:

El análisis léxico es una tarea fundamental en la compilación de programas. La implementación en Python utilizando expresiones regulares proporciona una manera eficiente de identificar los tokens léxicos. El tiempo de ejecución del programa puede variar según el tamaño del archivo fuente y la complejidad de las expresiones regulares utilizadas.