Leon Blanga Hasbani | A01784008
Marcos Dayan Mann | A01782876
Gabriel Edid Harari | A01782146

Tecnológico de Monterrey

Evidencia 1. Resaltador de sintaxis

Implementación de métodos computacionales
Gpo 602

5) Solución planteada:
Algoritmos implementados y tiempo de ejecución

Nuestra solución al resaltador de sintaxis consiste en el desarrollo de un lexer y un generador de HTML. En cuanto al lexer, este es el responsable de analizar el código fuente proveído por el usuario para categorizarlo en los diferentes tipos de tokens que lo componen; por su lado, el generador de HTML utiliza estos mismos tokens ya identificados para crear una representación visual de la clasificación del código fuente en un archivo HTML.

Algoritmos implementados en los códigos

1. Lexer

El lexer está compuesto por varias funciones específicas, cada una utilizada para identificar los diferentes tipos de tokens. Para identificar los números se utilizó la función *“num_lexer”*, para operadores *“op_lexer”*, para variables y palabras reservadas *“var_lexer”*, y finalmente para cadenas de texto (strings) *“string_lexer”*.

Cada función de lexer sigue una tabla de estados finitos (adjunta en la entrega) para determinar la categoría del token basado en los caracteres presentes en la cadena de entrada. El lexer procesa el código fuente caracter por caracter, determinando su tipo y almacenando el mismo junto con su categoría en una lista de resultados.

2. Generador de HTML

Una vez que el lexer genera la lista de resultados, este es pasado al generador de HTML para crear un archivo de este formato el cual representa de forma visual -a través de resaltado en colores- los tipos de tokens que se encuentran presentes.

Cada tipo de token tiene un color asociado específico, y el generador de HTML aplica estos colores a los tokens para crear una visualización colorida del código fuente. Se incluye una guía de colores al principio del HTML para que el usuario pueda entender la representación.

Tiempo de ejecución:

El tiempo de ejecución del lexer está directamente relacionado con la longitud del código fuente y la complejidad del contenido. Esto se debe a que el lexer analiza cada carácter individualmente y realiza comparaciones con múltiples conjuntos de caracteres (como dígitos, operadores, letras, etc.), es así que el tiempo de ejecución crece linealmente respecto a la longitud del código fuente.

Leon Blanga Hasbani | A01784008
Marcos Dayan Mann | A01782876
Gabriel Edid Harari | A01782146

Tecnológico de Monterrey

6) Complejidad del Algoritmo

La complejidad del algoritmo del lexer es O(n), donde (n) es el número de caracteres en el código fuente. Esto se debe a que el flujo del programa realiza una única pasada a través del código fuente (iteración lineal), la cual es la operación base del programa y cada carácter se procesa una sola vez.

Cálculo de Complejidad funciones:

- *Num Lexer:* Analiza cada carácter una vez, determinando su tipo en un tiempo constante **O(1)**. Recorre la cadena de longitud (n), resultando en **O(n)**.
- *Op Lexer:* Similar al num_lexer, procesa cada carácter una sola vez, también en **O(n).**
- *Var Lexer:* Recorre la cadena una sola vez, categorizando cada carácter, resultando en **O(n)**.
- *String Lexer:* Procesa cada carácter una vez, resultando en **O(n)**.

7) Reflexión

La solución que llegamos para el resaltador de sintaxis, utiliza técnicas de análisis léxico mediante *autómatas de estados finitos* para clasificar diferentes tipos de tokens en un código fuente de Python. La implementación de estas técnicas asegura que el tiempo de ejecución sea lineal respecto al tamaño del código fuente, lo cual es verdaderamente eficiente y por ende, muy adecuado para el propósito de la tarea que se nos planteó: resaltar la sintaxis del código en HTML.

Los algoritmos implementados son bastante eficientes, hablando de iteración, indexado en listas y matrices, y controles de flujo y siguen principios básicos de métodos computacionales, siendo eficientes en términos de tiempo y espacio. La complejidad **O(n)** es coherente con el tiempo de ejecución observado durante la práctica, confirmando que la solución es escalable y adecuada para archivos de código fuente de tamaño moderado.

Implicaciones Éticas

El desarrollo de herramientas de análisis y resaltado de código, como el lexer y el generador de HTML, puede tener varias implicaciones éticas, tanto positivas, como negativas en la sociedad:

1. **Facilitación del Aprendizaje y Educación:**
   Estas herramientas pueden mejorar en gran manera la comprensión del código para estudiantes y desarrolladores, haciendo más accesible el aprendizaje de lenguajes de programación, y aumentando mucho la velocidad de desarrollo, detectando errores de sintaxis y clasificando los tipos de datos y tokens de un lenguaje; es probable que los identificadores de sintaxis de los IDE's más comerciales utilicen éste método.

2. **Posible Dependencia Tecnológica:**

Leon Blanga Hasbani | A01784008
Marcos Dayan Mann | A01782876
Gabriel Edid Harari | A01782146

Existe el riesgo de que los desarrolladores se vuelvan demasiado dependientes de estas herramientas, lo que podría disminuir su habilidad para leer y comprender el código sin ayudas visuales y de clasificación.

3. **Privacidad y Seguridad:**

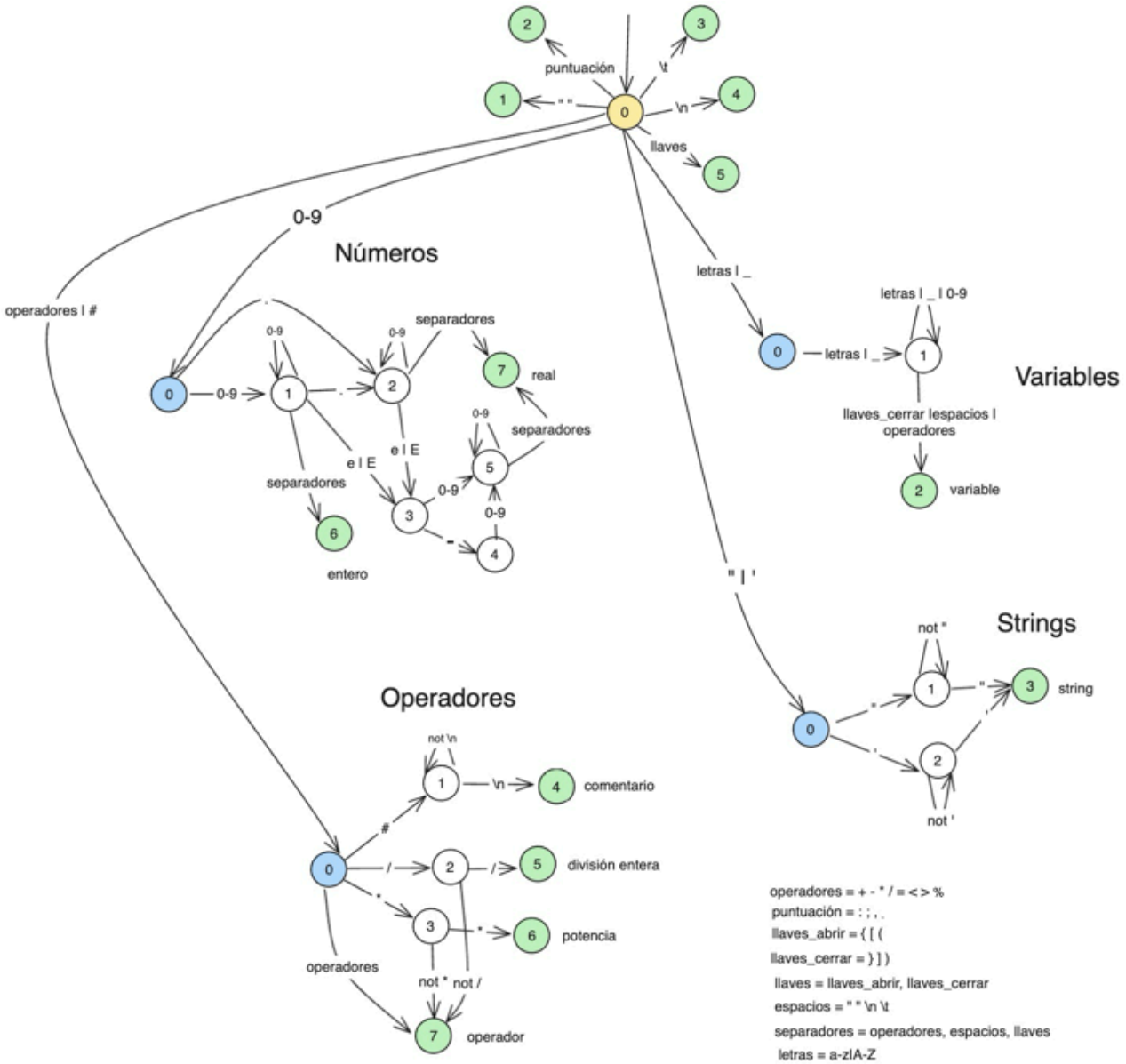
Si estas herramientas se integran en plataformas en línea, es muy importante considerar la privacidad y seguridad del código fuente que se procesa, asegurándose de que no se almacene ni se comparta sin consentimiento por la plataforma que da el servicio.

Sabemos que las herramientas de análisis léxico y resaltado de código ofrecen numerosos beneficios educativos y prácticos, sabemos esto gracias a nuestro propio aprendizaje a lo largo de la realización exitosa de la tarea. Sin embargo, también se dio a conocer algunos posibles riesgos éticos relacionados con su uso, de los cuales, como grupo el que más nos preocupa es la privacidad de los datos y los derechos de autor del código en caso de que sea mandado a un servicio en línea.

Leon Blanga Hasbani | A01784008
Marcos Dayan Mann | A01782876
Gabriel Edid Harari | A01782146

Tecnológico de Monterrey

Autómatas



Leon Blanga Hasbani | A01784008
Marcos Dayan Mann | A01782876
Gabriel Edid Harari | A01782146

Tecnológico de Monterrey

Tablas de transición

tabla de transición de números

	estado	digito	.	e | E	-	operador|" "|()	otro	
inicial	0	1	2	9	9	9	9	
registro de int	1	1	2	3	6	6	9	
registro float	2	2	9	3	7	7	9	
registro científico	3	2	9	9	4	9	9	
registro negativo cient	4	5	9	9	9	9	9	
registro digitos cient	5	5	9	9	9	8	9	
entero digito	6							
decimal digito	7							
científico digito	8							
error	9							

tabla de transición de operadores

	estado	*	/	#	operador	todo menos /n	/n	otro
inicial	0	3	2	1	7	8	8	8
registro comentario	1	1	1	1	1	1	4	1
chequeo división entera	2	8	5	8	8	8	7	8
chequeo potencia	3	6	8	8	8	8	7	8
comentario	4							
división entera	5							
potencia	6							
operador	7							
error	8							

tabla de transición de variables

	estado	letras	_	0-9	llaves_cerrar |espacios | operadores	puntuación	otro
inicial	0	1	1	3	3	3	3
registro variable	1	1	1	1	2	2	3
variable	2						
error	3						

tabla de transición de strings

	estado	"	'	\n	otro
inicial	0	1	2	4	4
registro string con "	1	3	1	1	1
registro string con '	2	2	3	2	2
string	3				
error	4				