Manual de Usuario del Compilador de Texto

Descripción General

Este compilador de texto está diseñado para analizar archivos de texto con extensión .txt. Su propósito es contar y proporcionar estadísticas sobre las palabras reservadas y otros tokens presentes en el archivo.

Funciones Esenciales

1. **Lectura de Archivos:**
    - **Descripción:** Lee el contenido de archivos .txt para analizar el texto.
    - **Uso:** Permite al compilador acceder al texto que necesita analizar. El archivo se abre, lee su contenido línea por línea o en bloques, y se almacena para su procesamiento.
2. **Tokenización:**
    - **Descripción:** Divide el texto en tokens, que son las unidades básicas de análisis (como palabras, números y símbolos).
    - **Uso:** Facilita el análisis de los componentes del texto. Identifica y clasifica cada parte del texto para su posterior análisis.
3. **Contador de Palabras Reservadas:**
    - **Descripción:** Cuenta la frecuencia de aparición de palabras reservadas específicas en el texto.
    - **Uso:** Identifica y contabiliza palabras clave o términos importantes definidos por el usuario o el sistema, proporcionando estadísticas sobre su frecuencia en el archivo.
4. **Análisis de Tokens:**
    - **Descripción:** Clasifica los tokens en diferentes categorías, como palabras reservadas, identificadores, literales, etc.
    - **Uso:** Permite al compilador comprender el tipo de cada token y su función en el contexto del texto, facilitando el análisis sintáctico y semántico.
5. **Generación de Reportes:**
    - **Descripción:** Crea un archivo de salida con estadísticas y resultados del análisis.
    - **Uso:** Proporciona al usuario un resumen del análisis realizado, incluyendo la frecuencia de cada token y otras estadísticas relevantes.

Funciones Especiales

1. **Filtrado de Comentarios:**
    - **Descripción:** Elimina o ignora comentarios en el texto que no son relevantes para el análisis léxico.

    - **Uso:** Mejora la precisión del análisis al asegurarse de que los comentarios no interfieran con la contabilización de tokens importantes.
2. **Manejo de Errores:**
    - **Descripción:** Detecta y maneja errores en el archivo de entrada o durante el proceso de análisis.
    - **Uso:** Garantiza que el compilador pueda manejar archivos mal formateados o errores inesperados sin detenerse bruscamente.
3. **Configuración Dinámica:**
    - **Descripción:** Permite ajustar las configuraciones del compilador en tiempo de ejecución, como agregar o quitar palabras reservadas.
    - **Uso:** Ofrece flexibilidad para adaptar el compilador a diferentes necesidades sin necesidad de modificar el código fuente.
4. **Optimización de Rendimiento:**
    - **Descripción:** Implementa técnicas para mejorar la eficiencia del análisis, como el uso de estructuras de datos eficientes o técnicas de procesamiento en paralelo.
    - **Uso:** Aumenta la velocidad y la eficiencia del compilador, especialmente al trabajar con archivos de gran tamaño.
5. **Soporte para Múltiples Idiomas:**
    - **Descripción:** Permite al compilador analizar texto en diferentes lenguajes de programación o formatos de texto.
    - **Uso:** Extiende la utilidad del compilador a una gama más amplia de archivos y lenguajes, adaptándose a diferentes necesidades de análisis.

Configuración Inicial

Antes de ejecutar el compilador, sigue estos pasos para prepararte:

1. **Archivos de Entrada:**
    - Asegúrate de tener los archivos .txt que deseas analizar.
    - Los archivos deben estar en el formato esperado por el compilador y ubicados en la carpeta especificada.

Ejecución del Código

1. **Preparar el Entorno:**
    - Verifica que el compilador esté correctamente instalado y que todas las dependencias necesarias estén disponibles en tu sistema.
2. **Ejecutar el Compilador:**
    - Abre una terminal o línea de comandos en tu sistema.
    - Navega al directorio donde se encuentra el compilador.
    - Ejecuta el compilador con el archivo de entrada. Por ejemplo:

```
csharp
Copiar código
[Comando de ejecución] archivo_entrada.txt
```

- Asegúrate de reemplazar [Comando de ejecución] con el comando específico para ejecutar el compilador y archivo_entrada.txt con el nombre de tu archivo de texto.

Entradas y Salidas

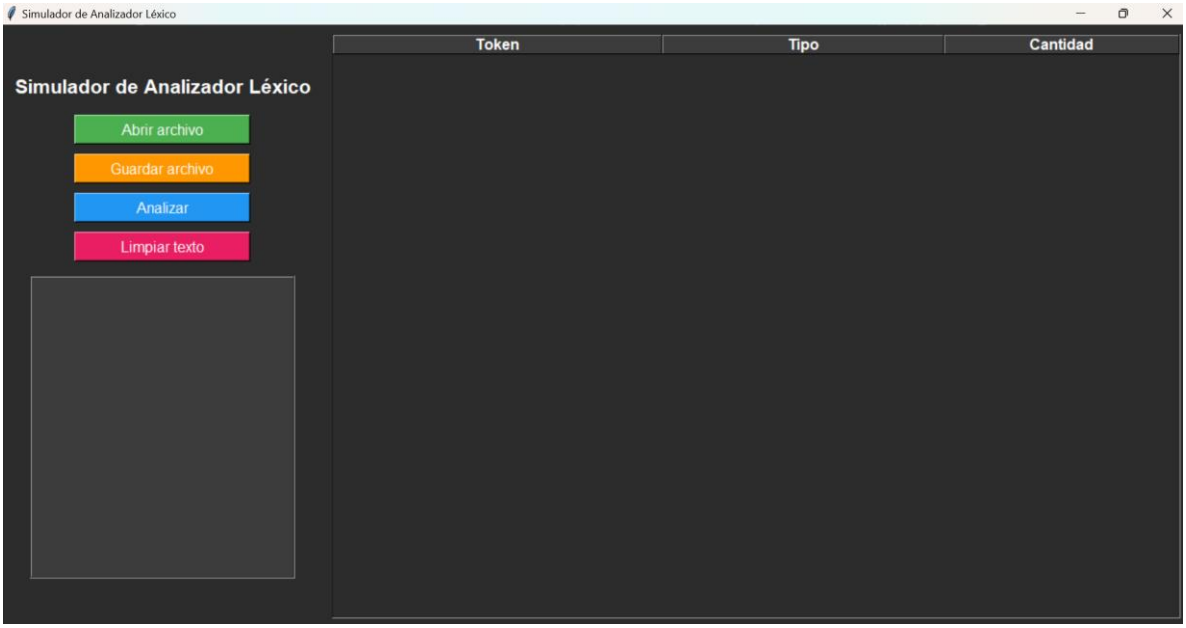
- **Entradas:**
  - **Archivo de Texto (.txt):** Debe ser el archivo que deseas analizar. Asegúrate de que esté en el formato correcto.
- **Salidas:**
  - **Archivo de Resultados:** Este archivo contendrá las estadísticas sobre los tokens analizados, incluyendo la cantidad de cada palabra reservada y otros tokens según la configuración del compilador.

Opciones y Configuraciones

El compilador puede permitirte ajustar varias opciones:

- **Palabras Reservadas:** Puedes configurar la lista de palabras reservadas que deben ser contadas por el compilador.
- **Formato de Salida:** Es posible ajustar cómo se presentan los resultados en el archivo de salida.

Revisa la configuración del compilador o su documentación para más detalles sobre cómo ajustar estas opciones.



Abrir archivo: Permite cargar un archivo de texto que contenga el código fuente de un programa. Esto es útil para analizar programas ya existentes o para probar el funcionamiento del simulador con diferentes tipos de código.

Guardar archivo: Guarda los resultados del análisis en un archivo de texto. Esto te permitirá revisar los resultados con más detalle o utilizarlos como entrada para otras herramientas.

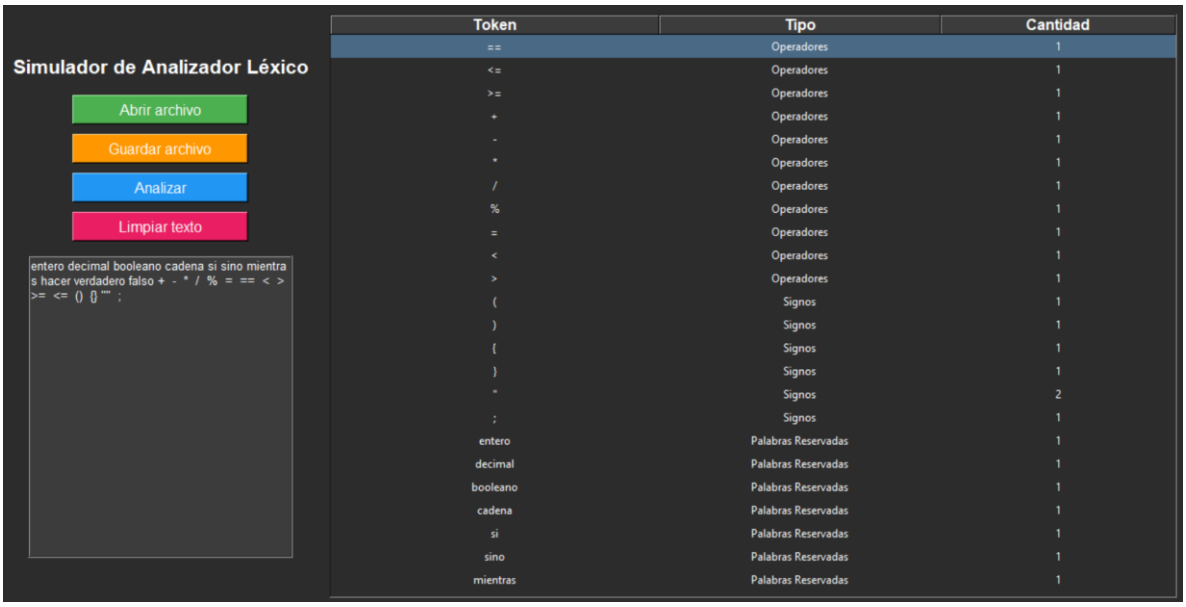
Analizar: Inicia el proceso de análisis léxico. El simulador descompondrá el texto en tokens individuales y clasificará cada token según su tipo. Los resultados se mostrarán en la tabla, donde podrás ver los tokens encontrados, su tipo y la cantidad de veces que aparecen.

Limpiar texto: Borra el texto actual y prepara el simulador para analizar un nuevo texto. Esto es útil si deseas analizar diferentes programas sin tener que cerrar y abrir el simulador cada vez.

Token: Aquí se mostrarán las palabras individuales encontradas en el texto que se analizó.

Tipo: Se indicará a qué categoría pertenece cada token (por ejemplo, palabra reservada, identificador, número, operador, etc.).

Cantidad: Se contará cuántas veces aparece cada tipo de token en el texto.



La imagen muestra los resultados obtenidos al analizar un fragmento de código. Como puedes observar, el simulador ha identificado los diferentes **tokens** que componen el código, como operadores, palabras reservadas y signos de puntuación. Cada token se clasifica en una categoría específica, lo que permite al compilador entender el significado del código.

- Los tokens ==, =, +, -, *, /, % son **operadores** que se utilizan para realizar operaciones matemáticas o lógicas.
- Los tokens entero, decimal, booleano, cadena, si, sino, mientras son **palabras reservadas** que tienen un significado especial en el lenguaje de programación.
- Los tokens ;, (,), {, } son **signos de puntuación** que se utilizan para estructurar el código."