DATA FORGE

(Manual técnico)

Introducción

El presente proyecto tiene como objetivo el desarrollo de un sistema integral capaz de llevar a cabo operaciones aritméticas y estadísticas, así como la generación de gráficos a partir de conjuntos de datos. Este sistema es diseñado en el contexto del curso de Organización de Lenguajes y Compiladores 1, impartido por la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala.

El sistema propuesto permitirá a los usuarios realizar operaciones matemáticas y estadísticas de manera eficiente y efectiva, abriendo la puerta a un conjunto diverso de aplicaciones. Desde cálculos simples hasta análisis más complejos, el sistema ofrecerá una interfaz intuitiva y funcionalidades avanzadas para satisfacer las necesidades de los usuarios.

Asimismo, la capacidad de generar gráficos a partir de conjuntos de datos brinda un componente visual que facilitará la interpretación y comprensión de los resultados obtenidos. Esta característica potencia la utilidad del sistema, convirtiéndolo en una herramienta versátil tanto para estudiantes como para profesionales que requieran realizar análisis numéricos y presentar resultados de manera gráfica.

A lo largo de este proyecto, se aplicarán los conocimientos adquiridos en el curso de Organización de Lenguajes y Compiladores 1, consolidando la comprensión teórica con la implementación práctica de analizadores léxicos y sintácticos. Este enfoque permitirá a los estudiantes no solo entender los fundamentos de la construcción de compiladores, sino también aplicar estos conocimientos en el desarrollo de soluciones tecnológicas concretas y relevantes en el ámbito de las ciencias de la computación.

Especificaciones:

1. Lenguaje utilizado:

java versión "21.0.2"

2. Editor y librerías utilizadas:

Editor: Netbeans.

Librerías:

-Jfelx: Analizador léxico

-Cup: Analizador sintáctico

-Jfreechart: Implementación de graficas

Métodos y funciones importantes:

1. **Nombre del método: run**

Ubicación: Paquete Codigo/árbol.java

Parámetros:

- arbol raiz: Un objeto que representa un árbol sintáctico.
- ArrayList<nodoSym> TS: Un ArrayList que almacena información sobre símbolos (variables, arreglos, etc.) en el programa.

Funcionalidad:

- El método recorre recursivamente el árbol sintáctico (arbol raiz) y realiza diversas acciones según la estructura y contenido de los nodos del árbol.
- Se observa que el método realiza operaciones específicas cuando encuentra nodos con ciertas etiquetas (como "DEC", "DARR", "LVAL", "LIST", "VALOR", "OPARIT", "E", "FUNC", etc.).
- Dependiendo de las condiciones especificadas, el método imprime información en la consola y agrega nodos nodoSym al ArrayList TS.

Algunas de las acciones específicas que realiza este método incluyen:

- Manejo de declaraciones de variables (DEC).
- Realización de operaciones aritméticas (OPARIT).
- Manejo de funciones (FUNC) y sus argumentos.
- Trabajo con listas y valores (LVAL, LIST, VALOR).
- Impresión en la consola (IMP).
- Creación de gráficos (DATOS, GRAPH).

2. **Nombre del método: ejecutar**

Ubicación: Paquete JavaAplication1/interfaceAnalizador.java

Parámetros**:**

- **String entrada**: Una cadena que contiene el código fuente que se va a ejecutar.

Funcionalidad:

- Inicializa algunas listas y restablece la imagen del gráfico a un estado nulo.
- Crea un ArrayList llamado **TS** para almacenar información sobre símbolos.
- Utiliza un lexer (**Lexer**) y un parser (**Parser**) para analizar el código fuente proporcionado y construir un árbol sintáctico (**arbol raiz**).
- Inicializa la consola de salida en el árbol sintáctico (**raiz.consola**) y la lista de imágenes (**raiz.Listalmagenes**).
- Ejecuta el método **run** del árbol sintáctico (**raiz.run(raiz, TS)**) para realizar operaciones semánticas.
- Imprime en la consola la tabla de símbolos (**TS**) que se ha construido durante la ejecución.

- Actualiza el contenido del área de texto en la interfaz de usuario con la salida de la consola generada durante la ejecución del programa.
- Si hay imágenes generadas durante la ejecución (**!imgs.isEmpty()**), asigna la última imagen de la lista al JLabel **LabelImgGrafica** y actualiza el índice actual.
- Captura y maneja excepciones, imprimiendo detalles en la consola en caso de que ocurran.

3. **Nombre del método: obtenerTextAreaDePestanaActual**

Ubicación: Paquete JavaAplication1/interfaceAnalizador.java

Retorno: JTextArea: El componente **JTextArea** asociado a la pestaña actualmente seleccionada, o **null** si no se encuentra un **JTextArea**.

Funcionalidad:

- Obtiene el índice de la pestaña actualmente seleccionada en el **JTabbedPane** llamado **PestañasPanel**.
- Verifica si hay alguna pestaña seleccionada (**indicePestanaActual != -1**).
- Si hay una pestaña seleccionada, obtiene el componente asociado a esa pestaña.
- Verifica si el componente asociado es un **JTextArea**. Si la pestaña está compuesta por un **JScrollPane**, entonces se extrae el componente interno.
- Devuelve el **JTextArea** si se encuentra. Si no se encuentra un **JTextArea**, el método puede devolver **null** o realizar alguna lógica adicional según las necesidades del programa.

4. **Nombre del método: agregarPestaña**

Ubicación: Paquete JavaAplication1/interfaceAnalizador.java

Parámetros:

- **String contenido**: El contenido que se desea agregar a la nueva pestaña, código fuente o texto.
- **String nombre**: El nombre que se le asignará a la nueva pestaña.

Funcionalidad:

- Crea un nuevo objeto **JTextArea** llamado **textArea**. Un **JTextArea** es un componente que permite la edición de texto multilineal.
- Establece el contenido inicial del **textArea** usando el método **append** con el contenido proporcionado como parámetro.
- Asigna un nombre a la nueva pestaña utilizando la variable **nombrePestana**.
- Agrega la nueva pestaña al **PestañasPanel**. La pestaña se crea como un **JScrollPane** que contiene el **textArea**. Un **JScrollPane** proporciona barras de desplazamiento si el contenido del **textArea** es más grande que el área visible.
- La pestaña se etiqueta con el nombre proporcionado como parámetro.

5. **Nombre del método: buscarArchivoTexto**

Ubicación: Paquete JavaAplication1/interfaceAnalizador.java

Retorno: String: El contenido del archivo de texto seleccionado o una cadena vacía si no se selecciona ningún archivo.

Funcionalidad:

- Crea un nuevo objeto **JFileChooser** llamado **fc**, que proporciona una interfaz gráfica para que el usuario seleccione un archivo.

- Define un filtro de extensión para mostrar solo archivos con extensión ".df" en el cuadro de diálogo.
- Configura el filtro en el **JFileChooser** utilizando **setFileFilter**.
- Muestra el cuadro de diálogo de selección de archivo (**showOpenDialog**) y almacena la respuesta en la variable **resp**.
- Si el usuario elige un archivo (**resp == JFileChooser.APPROVE_OPTION**), obtiene el nombre del archivo seleccionado (**this.nombreArchivo = fc.getSelectedFile().getName()**) y llama a la función **leerContenidoArchivo** para obtener el contenido del archivo seleccionado.
- Devuelve el contenido del archivo seleccionado. Si no se selecciona ningún archivo, devuelve una cadena vacía.

6. **Nombre del método: saveFile**

Parámetros:

- **String name**: El nombre del archivo que se utilizará para preseleccionar el nombre de archivo al guardar.

Funcionalidad:

- Crea un nuevo objeto **JFileChooser** llamado **fileChooser**, que proporciona una interfaz gráfica para que el usuario seleccione la ubicación y el nombre de archivo para guardar.
- Preselecciona el nombre de archivo utilizando **setSelectedFile** con el nombre proporcionado como parámetro.
- Define un filtro de extensión para mostrar solo archivos con extensión ".df" en el cuadro de diálogo de guardar.
- Configura el filtro en el **JFileChooser** utilizando **setFileFilter**.
- Muestra el cuadro de diálogo de guardar archivo (**showSaveDialog**) y almacena el resultado en la variable **result**.

- Si el usuario elige una ubicación y proporciona un nombre de archivo (**result == JFileChooser.APPROVE_OPTION**), obtiene la ruta completa del archivo seleccionado y garantiza que tenga la extensión ".df".
- Obtiene el contenido del **JTextArea** asociado a la pestaña actual mediante la función **obtenerTextAreaDePestanaActual**.
- Escribe el contenido del **JTextArea** en el archivo seleccionado utilizando un **BufferedWriter** y un **FileWriter**.
- Maneja posibles excepciones de E/S (**IOException**) imprimiendo la traza de la pila en la consola.

Conclusión:

En conclusión, este proyecto, centrado en el desarrollo de un sistema para realizar operaciones aritméticas y estadísticas con la capacidad adicional de generar gráficos a partir de conjuntos de datos, representa la aplicación práctica de los conocimientos adquiridos en el curso de Organización de Lenguajes y Compiladores 1 de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala. Al integrar conceptos teóricos con implementaciones prácticas, se busca no solo comprender la teoría detrás de los analizadores léxicos y sintácticos, sino también utilizar estas habilidades en la creación de soluciones tecnológicas relevantes en el campo de las ciencias de la computación. Este proyecto servirá como una valiosa oportunidad para consolidar la comprensión de los fundamentos de la construcción de compiladores y aplicar estos conocimientos en el desarrollo de herramientas eficientes y funcionales.