PRACTICA 1

Objetivos generales

- Familiarizar al estudiante con el lenguaje JAVA de escritorio con SWING.
- Familiarizar al estudiante con el proceso básico de un analizador léxico.
- **Elaborar la lógica para la solución del problema planteado.**

Objectivos específicos

- Implementación de ciclos, sentencias de control y vectores usando java.
- Conocer las diferencias de sintaxis entre los lenguajes que el estudiante usa.
- Implementar algoritmos para el reconocimiento de patrones y recuperación de errores.
- Aplicar buenas prácticas de programación.

Descripción de la actividad

Una de las tareas de un analizador léxico es la detección de símbolos que no forman parte de un lenguaje y también el reconocimiento de tokens válidos para el lenguaje en cuestión. Sin embargo el analizador léxico debe reaccionar correctamente ante los posibles errores que se encuentren, indicando la posición del error y una descripción del mismo.

La siguiente práctica invita al estudiante a idear algoritmos que permitan reconocer patrones en una cadena de caracteres y reportar errores que se presenten a lo largo del análisis.

Procedimiento teórico-práctico a realizar

1. Creación de la expresión regular que describa el patrón de cada token.
2. Gramática regular de cada token.
3. AFD de cada token.
4. Creación del AFD que acepte todos los tokens
    a. Diagrama de transiciones del AFD
    b. Tabla de transiciones del AFD

Carga de archivo de entrada

El usuario tiene la disponibilidad de cargar un archivo de entrada con el texto que desea analizar. El texto del archivo de texto debe ser cargado en un componente que permita su edición.

Edición de texto de entrada

El usuario tiene la disponibilidad de crear texto de entrada en un componente para su análisis.(TEXTAREA con indicador de fila a la izquierda)

Búsqueda de patrones

El usuario tiene la disponibilidad de indicar una palabra que será buscada dentro del texto de entrada y al presionar un botón una nueva área mostrará el texto de entrada resaltando el texto que coincida con la cadena buscada.

Análisis de tokens

Para esta practica, el alfabeto permitido está compuesto por los siguientes símbolos:

- Letras de la ‘a’ a la ‘z’, ya sea mayúsculas o minúsculas. No se incluye la eñe.
- Dígitos del 0 al 9
- Signos de puntuación: punto (.), coma (,), punto y coma (;), dos puntos (:)
- Operadores aritméticos: Suma (+), Resta (-), Multiplicación (*), división (/), módulo (%)
- Signos de agrupación: Paréntesis derecho ‘ (‘, Paréntesis izquierdo, ‘) ‘, Corchete izquierdo ‘ [‘, corchete derecho ‘] ‘, llave izquierda ‘ { ‘, llave derecha ‘ } ‘
- Espacio, salto de línea.

Los tokens válidos se describen a continuación:

- Identificador: Son las palabras que cumplen el iniciar con una letra y pueden estar seguidas de muchas letras o muchos dígitos.
- Número: Son palabras que cumplen con tener al menos un dígito o más, y solo puede contener dígitos.
- Decimal: Son palabras que cumplen con tener al menos un dígito o más, seguido de un punto, seguido de uno o más dígitos.
- Puntuación: Ser alguno de los signos de puntuación
- Operador: Ser alguno de los operadores aritméticos
- Agrupacion: Ser alguno de los signos de agrupación

NOTA: los espacios y los saltos de línea indican la separación de palabras pero no tienen otro significado en particular.

Dado lo anterior, el usuario también tiene la disponibilidad de ejecutar un análisis sobre el texto de entrada, esta funcionalidad genera los siguientes reportes:

Reporte de errores

Si existieran errores, se debe generar una tabla con la siguiente información

- Símbolo o cadena de error
- Posición (fila y columna dentro del text area)
-

Reporte de tokens

Este reporte se debe mostrar solo si no existen errores y genera una tabla de tokens en la cual se listan todos los lexemas con la siguiente información:

- Nombre del Token
- Lexema
- Posición (fila y columna dentro del text area)

El usuario también debe contar con la funcionalidad de recuento de lexemas, la cual se debe habilitar cuando no existan errores en el texto de entrada, y consta de una tabla con la siguiente información:

- Lexema
- Token (numero, identificador, decimal, agrupación, etc)
- Cantidad de veces que aparece en el texto de entrada.

Guardar cambios del texto de entrada

El usuario deberá tener la opción de exportar el texto de entrada en un archivo de texto.

Generación de AFD óptimo

El uso de un autómata finito determinista en su forma formal es muy importante para poder optimizar código y hacerlo escalable por lo tanto se considera obligatorio la implementación de un autómata finito determinista en su definición formal (como se ha hecho en clase), de esta manera seria muy facil el configurar una nueva gramática únicamente modificando sus reglas gramaticales o funciones de transición.

Así también es necesario indicar los movimientos que realiza este autómata por medio de su tabla de estados cuando se está analizando un token. Ejemplo: “Me moví del estado A al estado B con un carácter J ”.

Otro Ejemplo:

Con la palabra: hola

Me moví del estado 1 al estado 2 con una h,
Me moví del estado 2 al estado 3 con una o
Me moví del estado 3 al estado 4 con una l
Me moví del estado 4 al estado 5 con una a

Recuperación de errores

La forma de recuperarse de un error es enviando el token de error y a su vez reiniciando el autómata para seguir analizando. Entonces cuando exista un error se tendrá que seguir evaluando buscando un posible siguiente error o bien un token válido, ejemplo:

585f3.40 <- Error, en 585f, pero detecta token decimal = 3.40

Importante

- Práctica obligatoria para tener derecho a la siguiente práctica o proyecto
- Es válido utilizar algún IDE o cualquier editor de texto.
- **Copias obtendrán nota de cero y se notificará a coordinación.**
- **Si se va a utilizar código de internet, entender la funcionalidad para que se tome como válido.**
- **No se permite usar ninguna función propia de java o librería que tenga que ver con reconocimiento de patrones o manejo directo de cadenas. Solo manejo de chars.**
- **El funcionamiento mínimo para tener derecho a calificación de la práctica es la carga del archivo de entrada y el análisis.**

Entrega

La fecha de entrega es el dia 28 de septiembre de 2021 antes de las 11:59 p.m en la plataforma Classroom, componentes a entregar:

- Código fuente
- Manual de usuario y técnico
- Trabajo teórico-práctico (archivo PDF)

Calificación

Pendiente