Documentación

martes, 20 de mayo de 2025 11:16 a. m.

Objetivo de la Práctica
● Crear una Tabla de Manejo de Errores: Para registrar y clasificar errores léxicos (e.g., símbolos no válidos, números fuera de rango).
● Crear una Tabla de Símbolos: Para almacenar los tokens identificados (símbolos, números, "Math").
● Implementar un Analizador Léxico en Java: Que cargue y utilice estas tablas para procesar entrada de "Math".

Definición del Lenguaje "Math":
Expresión regular:
/(?<![a-zA-Z0-9\\\$])\$(?:
\\(arccos|arcsin|arctan|arg|cos|cosh|cot|coth|csc|deg|det|dim|exp|gcd|hom|inf|ker|lg|lim|liminf|limsup|ln|log|max|min|Pr|sec|sin|sinh|sup|tan|tanh|frac|cdot|to|theta|alpha|beta|gamma|delta|epsilon|pi|mu|nu|rho|sigma|phi|psi|omega)|[a-zA-Z]|[0-9]|[\^_{}=+\-*().,\s])+\$
/gm

Token	Descripción	Ejemplo
DELIMITADOR_INI	Delimitador inicial de expresión	$
DELIMITADOR_FIN	Delimitador final de expresión	$
IDENTIFICADOR	Variable de una sola letra	x, y, a
CONST_ENTERA	Números enteros	5, 42
OPERADOR_SUMA	Suma	+
OPERADOR_IGUAL	Igualdad	=
FUNCION	Función matemática válida con barra	\sin, \log
SIMBOLO	Paréntesis, corchetes, etc.	(,)
ESPACIO	Ignorado en muchos lenguajes	' '
ERROR	Carácter no reconocido	@, ?, %%

Tabla de manejo de errores
La tabla de manejo de errores registrará los errores léxicos detectados durante el análisis, incluyendo el tipo de error, la posición, y una descripción.

ID	Nombre	Descripción	Val	Pos	Solución
E1	Símbolo desconocido	Carácter no reconocido por el lenguaje	@, #	[pos]	Escriba un carácter válido
E2	Variable inválida	Una variable debe tener solo una letra	abc	[pos]	Use una sola letra
E3	Falta barra inversa	Una función debe comenzar con '\'	sin	[pos]	Agregue '\' antes de la función
E4	Función inválida	La función no existe en el lenguaje	\suma	[pos]	Verifique el nombre de la función
E5	Transición inválida	No hay transición desde el estado actual	?	[pos]	Verifique el orden o tipo de símbolo
E6	Entrada vacía	No se proporcionó entrada para analizar	(vacío)	0	Ingrese una expresión válida

Java: Se implementa mediante una clase ErrorLexico que encapsula los datos del error (ID, nombre, descripción, valor, posición y solución).
Los errores se almacenan en una lista (List<ErrorLexico>) y se van agregando durante el análisis léxico conforme se

detectan anomalías.

Tabla de símbolos
La tabla de símbolos almacenará los tokens válidos identificados, con su tipo, valor y posición en la entrada.

Nº	Lexema	Token	Posición
1	$	DELIMITADOR_INI	0
2	sin	ERROR	1
3	(	OPERADOR/SIMBOLO	4
4	x	IDENTIFICADOR	5
5	)	OPERADOR/SIMBOLO	6
6	$	DELIMITADOR_FIN	7

Java: Se implementa mediante una clase ErrorLexico que encapsula los atributos del error (ID, nombre, descripción, valor, posición y solución).
Durante el análisis léxico, los errores detectados se almacenan en una lista dinámica (ArrayList<ErrorLexico>) para ser impresos posteriormente como tabla en consola.

Código: https://github.com/Aminxcido/compilador/

Salida de Compilador:

Expresión: $x + 555$

Tabla de símbolos:
Nº | Lexema | Token | Posición
-----|------------|----------------------------------|----------
1 | $ | DELIMITADOR_INI | 0
2 | x | IDENTIFICADOR | 1
3 | | OPERADOR/SIMBOLO | 2
4 | + | OPERADOR/SIMBOLO | 3
5 | | OPERADOR/SIMBOLO | 4
6 | 5 | CONST_ENTERA | 5
7 | 5 | CONST_ENTERA | 6
8 | 5 | CONST_ENTERA | 7
9 | $ | DELIMITADOR_FIN | 8

✔ **Cadena aceptada.**

Expresión: $sin(x)$

Tabla de símbolos:
Nº | Lexema | Token | Posición
-----|--------------|----------------------------------|----------
1 | $ | DELIMITADOR_INI | 0
2 | sin | ERROR | 1
3 | (| OPERADOR/SIMBOLO | 4
4 | x | IDENTIFICADOR | 5
5 |) | OPERADOR/SIMBOLO | 6
6 | $ | DELIMITADOR_FIN | 7

✘ **Cadena rechazada.**

Tabla de errores:

ID	Nombre	Descripcion	Val	Pos	Solucion
E3	Falta barra inversa	Una función debe comenzar con '\'	sin	1	Agregue '\' antes de la función