Desarrollo de aplicaciones avanzadas de ciencias computacionales

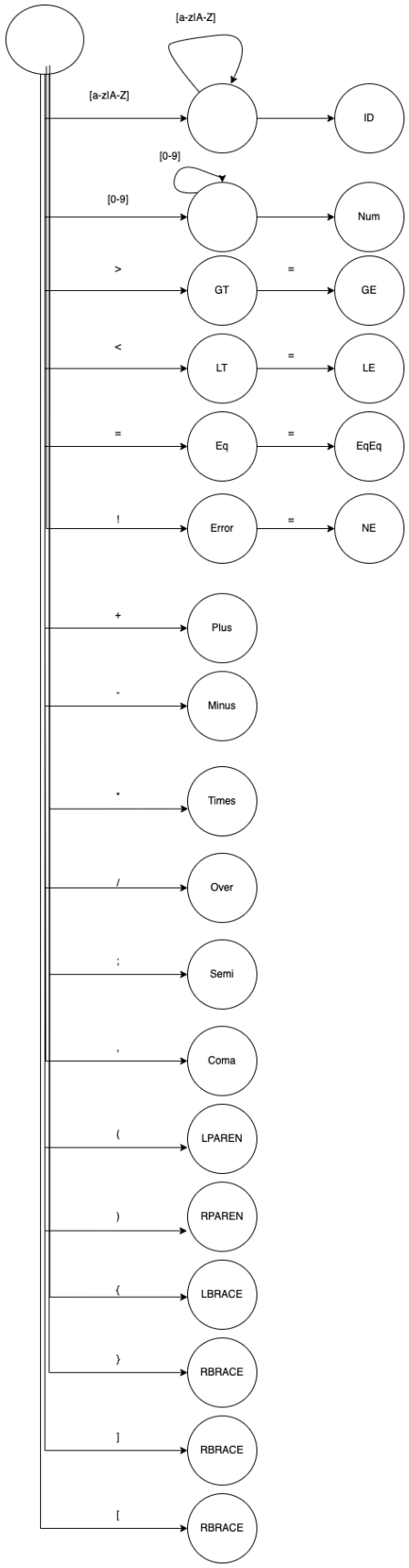
Lexer

Fecha de entrega: 09/04/2025

Alumno

Francisco Martinez Gallardo | A01782250

DFA:


[a-zlA-Z]
[a-zlA-Z]
ID
[0-9]
[0-9]
Num
>
GT
=
GE
<
LT
=
LE
=
Eq
=
EqEq
!
Error
=
NE
+
Plus
-
Minus
*
Times
/
Over
;
Semi
,
Coma
(
LPAREN
)
RPAREN
{
LBRACE
}
RBRACE
]
RBRACE
[
RBRACE

Regex:

- ^\$ → ENDFILE
- ^[0-9]+$ → NUM
- ^[a-zA-Z]+$ → ID
- ^(if|else|int|return|void|while)$ → RESERVED_WORD
- \+$ → PLUS (+)
- \-$ → MINUS (-)
- *$ → TIMES (*)
- /$ → OVER (/)
- ==$ → EQEQ (==)
- !=$ → NE (!=)
- =$ → EQ (=)
- <$ → LT (<)
- <=$ → LE (<=)
- >$ → GT (>)
- >=$ → GE (>=)
- ;$ → SEMI (;)
- ,$ → COMMA (,)
- \($ → LPAREN (()
- \)$ → RPAREN ())
- \{$ → LBRACE ({)
- \}$ → RBRACE (})
- \[$ → LBRACKET ([)
- \]$ → RBRACKET (])

Documentacion:

El lexer esta compuesto por 3 archivos de python:

main: Lee el archivo fuente (`sample2.txt`), inicializa el lexer y extrae los tokens uno por uno usando `getToken`.
lexer: Contiene toda la lógica del analizador léxico. Ya que contiene funcion que extrae el proximo token del programa.
Types: Define los tipos de tokens, estados y palabras reservadas como enums.

Este analizador léxico reconoce:
Palabras reservadas: if, else, int, return, void, while
Símbolos especiales: +, -, *, /, =, ==, !=, <, <=, >, >=, ;, ,, (,), {, }, [,]
Identificadores: Letras (a-z, A-Z), no comienzan con número
Números: Secuencias de dígitos enteros
Comentarios: // (linea) o de bloque /**/

Cómo Usar el Analizador Léxico: Tener los 3 archivos y un archivo de entrada en el directorio. Cambiar el path en main.py al de tu archivo de entrada y correr python main.py. Si se desea imprimir configure el parametro de la funcion get_token imprimir a true.