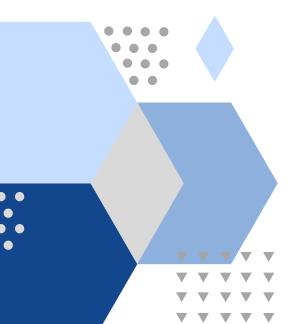


INSTITUTO TECNOLOGICO DE PACHUCA

# REQUERIMIENTO DE LA GRAMATICA

LENGUAJES Y AUTONOMAS 1



# Integrantes:

- Flores Bautista Daniel
- Gutierrez Hernandez Victor Gabriel
- Hernández Reyes Reyes
- Mendoza Gutierrez Mariana
- Torres Martínez Sergio Enrique

### Introduccion

para poder realizar este analizadors sintatico nostros utilizaremos como lenguaje de programacion principal java dentro de el utilizaremos las siguientes librerias:

- Jcup
- Jflex

# **Proposito:**

nuestro analizador sintactico tendra como proposito principal realizar analisis lexicos asi como sintatico donde podremos realizar estos analisis de manera que nos permita abrir un archivo o de manera manual ingresando nuestro codigo.

Una vez analizados nos dara como resultado por parte de lo lexico:

- token
- la linea en la que se encuentra
- simbolo que se esta utilizando

y por parte de lo sintactico nos mostrara un mensaje donde nos indicara si el analisis se realizo correctamente o en su defecto nos muestre el error de sintaxis.

### Simbolos terminales

Palabras clave:

- if
- else

Operadores:

- + o
- . /

Delimitadores:

- (
- '
- ,
- . 1
- ;

### Literales:

- Números (como num1, num2)
- Cadenas (por ejemplo, "Ingrese el primer número: ", "Ingrese el segundo número: ", "Error: No se puede dividir por cero.")

### Identificadores:

- Variables (como num1, num2, suma, resta, multiplicacion, division)
- Nombres de funciones (por ejemplo, main(), cout, cin, endl)

### Simbolos no terminales

- 1.StmtList (Lista de Declaraciones)
- 2.Stmt (Declaración)
- 3. Func Declaration (Declaración de Función)
- 4. FunctionName (Nombre de Función)
- 5. Parameters (Parámetros)
- 6. If Statement (Declaración de If)
- 7. Condition (Condición)
- 8. WhileLoop (Bucle While)
- 9. ReturnStatement (Declaración de Return)
- 10.E (Expresión)
- 11.T (Término)
- 12.F (Factor)
- 13. Literal

## **Producciones y reglas**

- 1.StmtList -> Stmt StmtList | ε
- 2.Stmt -> FuncDeclaration | IfStatement | WhileLoop | ReturnStatement
- 3.FuncDeclaration -> FunctionName ( Parameters ) { StmtList }
- 4.IfStatement -> if (Condition) { StmtList } else { StmtList }
- 5. WhileLoop -> while ( Condition ) { StmtList }
- 6. ReturnStatement -> return E;
- 7.E -> T + E | T E | T
- 8.T -> F \* T | F / T | F
- 9.F -> Literal | ( E )
- 10. Literal -> número | cadena

### Simbolos inicial

El símbolo inicial es StmtList, ya que es el punto de partida desde el cual se pueden derivar todas las listas de declaraciones y declaraciones definidas en tu código.

# Precedencia y Asociatividad (si aplica)

- 1. Operadores aritméticos:
  - Asociatividad izquierda: +, -
  - Asociatividad izquierda: \*,/
  - Mayor precedencia para \* y / que para + y -
- 2. Operadores de comparación:
  - Asociatividad no aplicable, ya que las expresiones de comparación usualmente no tienen asociatividad.
  - Igual precedencia para todos los operadores de comparación (<, >, <=, >=, !=)
- 3. Operadores lógicos:
  - Asociatividad izquierda: &&, ||
  - Mayor precedencia para && que para ||

Estas reglas de precedencia y asociatividad determinan cómo se agrupan los operadores en las expresiones y en qué orden se evalúan. Por ejemplo, en una expresión como a + b \* c, la multiplicación se realizaría antes que la suma debido a la mayor precedencia de \* sobre +.

# **Comentarios y Anotaciones**

### 1. Operadores aritméticos: 2. Asociatividad izquierda para suma y resta: +, -3.E -> E + T // La suma se evalúa de izquierda a derecha 4. E -> E - T // La resta se evalúa de izquierda a derecha 5. Asociatividad izquierda para multiplicación y división: \*,/ 6.T -> T \* F // La multiplicación se evalúa de izquierda a derecha 7.T -> T/F // La división se evalúa de izquierda a derecha 8. Mayor precedencia para \* y / que para + y -9.**E -> T** // Término 10.**T -> F** // Factor 11. Operadores de comparación: 12. Igual precedencia para todos los operadores de comparación (<, >, <=, >=, !=) 13.**F -> E < E** 14.**F** -> **E** > **E** 15.F -> E <= E 16.**F** -> **E** >= **E** 17.F -> E == E 18.**F -> E != E** 19. Operadores lógicos: 20. Asociatividad izquierda para && y || 21.F -> F && F // El operador lógico && se evalúa de izquierda a derecha 22.F -> F || F // El operador lógico || se evalúa de izquierda a derecha 23.