Trabajo teórico

Gramáticas de cada estructura:

PRINT

NT_PRINT -> PRINT NT_VALOR END
NT_VALOR -> LITERAL
NT_VALOR -> NUMERO_ENTERO
NT_VALOR -> IDENTIFICADOR
NT_VALOR -> DECIMAL

REPEAT

NT_REPEAT -> REPEAT NT_VALOR INIT NT_CONTENIDO_REPEAT END
NT_CONTENIDO_REPEAT -> NT_PRINT NT_CONTENIDO_REPEAT'
NT_CONTENIDO_REPEAT' -> NT_PRINT NT_CONTENIDO_REPEAT'
NT_CONTENIDO_REPEAT' -> ε
NT_PRINT -> PRINT NT_VALOR END
NT_VALOR -> LITERAL
NT_VALOR -> NUMERO_ENTERO
NT_VALOR -> IDENTIFICADOR
NT_VALOR -> DECIMAL

CONDICIONAL

NT_CONDICIONAL -> IF NT_BOOLEANO THEN NT_CONTENIDO_COND END
NT_BOOLEANO -> TRUE
NT_BOOLEANO -> FALSE
NT_CONTENIDO_COND -> NT_PRINT
NT_CONTENIDO_COND -> ε
NT_PRINT -> PRINT NT_VALOR END
NT_VALOR -> LITERAL
NT_VALOR -> NUMERO_ENTERO
NT_VALOR -> IDENTIFICADOR
NT_VALOR -> DECIMAL

EXPRESION

NT_EXP -> NT_T NT_EXP'
NT_EXP -> SUMA NT_T NT_EXP'
NT_EXP -> RESTA NT_T NT_EXP'
NT_EXP' -> SUMA NT_T NT_EXP'
NT_EXP' -> RESTA NT_T NT_EXP'
NT_EXP' -> ε

NT_T -> NT_F NT_T'
NT_T' -> MULTIPLICACION NT_F NT_T'
NT_T' -> DIVISION NT_F NT_T'
NT_T' -> POTENCIA NT_F NT_T'
NT_T' -> ε
NT_F -> PARENTESIS_APERTURA NT_EXP PARENTESIS_CIERRE
NT_F -> NUMERO_ENTERO
NT_F -> IDENTIFICADOR

ASIGNACION

NT_ASIGNACION -> IDENTIFICADOR ASIGNACION NT_EXP END
NT_EXP -> NT_T NT_EXP'
NT_EXP -> SUMA NT_T NT_EXP'
NT_EXP -> RESTA NT_T NT_EXP'
NT_EXP' -> SUMA NT_T NT_EXP'
NT_EXP' -> RESTA NT_T NT_EXP'
NT_EXP' -> ε
NT_T -> NT_F NT_T'
NT_T' -> MULTIPLICACION NT_F NT_T'
NT_T' -> DIVISION NT_F NT_T'
NT_T' -> POTENCIA NT_F NT_T'
NT_T' -> ε
NT_F -> PARENTESIS_APERTURA NT_EXP PARENTESIS_CIERRE
NT_F -> NUMERO_ENTERO
NT_F -> IDENTIFICADOR

Gramática general:

NT_INICIO -> NT_INSTRUCCION NT_INICIO'
NT_INICIO' -> NT_INSTRUCCION NT_INICIO'
NT_INICIO' -> ε

NT_INSTRUCCION -> NT_PRINT
NT_INSTRUCCION -> NT_REPEAT
NT_INSTRUCCION -> NT_CONDICIONAL
NT_INSTRUCCION -> NT_ASIGNACION

NT_PRINT -> PRINT NT_VALOR END
NT_VALOR -> LITERAL
NT_VALOR -> NUMERO_ENTERO
NT_VALOR -> IDENTIFICADOR

NT_REPEAT -> REPEAT NT_VALOR INIT NT_CONTENIDO_REPEAT END
NT_CONTENIDO_REPEAT -> NT_PRINT NT_CONTENIDO_REPEAT'

NT_CONTENIDO_REPEAT' -> NT_PRINT NT_CONTENIDO_REPEAT'
NT_CONTENIDO_REPEAT' -> ε

NT_CONDICIONAL -> IF NT_BOOLEANO THEN NT_CONTENIDO_COND END
NT_BOOLEANO -> TRUE
NT_BOOLEANO -> FALSE
NT_CONTENIDO_COND -> NT_PRINT
NT_CONTENIDO_COND -> ε

NT_ASIGNACION -> IDENTIFICADOR ASIGNACION NT_EXP END

NT_EXP -> NT_T NT_EXP'
NT_EXP -> SUMA NT_T NT_EXP'
NT_EXP -> RESTA NT_T NT_EXP'
NT_EXP' -> SUMA NT_T NT_EXP'
NT_EXP' -> RESTA NT_T NT_EXP'
NT_EXP' -> ε
NT_T -> NT_F NT_T'
NT_T' -> MULTIPLICACION NT_F NT_T'
NT_T' -> DIVISION NT_F NT_T'
NT_T' -> POTENCIA NT_F NT_T'
NT_T' -> ε
NT_F -> PARENTESIS_APERTURA NT_EXP PARENTESIS_CIERRE
NT_F -> NUMERO_ENTERO
NT_F -> IDENTIFICADOR

Diseño del analizador sintáctico LL

Primeros de la gramática:

First(NT_INICIO) = {PRINT, REPEAT, IF, IDENTIFICADOR}
First(NT_INICIO') = {PRINT, REPEAT, IF, IDENTIFICADOR}
First(NT_INSTRUCCION) = {PRINT, REPEAT, IF, IDENTIFICADOR}
First(NT_PRINT) = {PRINT}
First(NT_VALOR) = {LITERAL, NUMERO_ENTERO, IDENTIFICADOR}
First(NT_REPEAT) = {REPEAT}
First(NT_CONTENIDO_REPEAT) = {PRINT}
First(NT_CONTENIDO_REPEAT') = {PRINT}
First(NT_CONDICIONAL) = {IF}
First(NT_BOOLEANO) = {TRUE, FALSE}
First(NT_CONTENIDO_COND) = {PRINT}
First(NT_ASIGNACION) = {IDENTIFICADOR}
First(NT_EXP) = {SUMA, RESTA, PARENTESIS_APERTURA, NUMERO_ENTERO, IDENTIFICADOR}
First(NT_EXP') = {SUMA, RESTA}

First(NT_T) = {PARENTESIS_APERTURA, NUMERO_ENTERO, IDENTIFICADOR}
First(NT_T') = {MULTIPLICACION, DIVISION, POTENCIA}
First(NT_F) = {PARENTESIS_APERTURA, NUMERO_ENTERO, IDENTIFICADOR}

Siguientes de la gramática:

Follow(NT_INICIO) = {EOF}
Follow(NT_INICIO') = {EOF}
Follow(NT_INSTRUCCION) = {PRINT, REPEAT, IF, IDENTIFICADOR, EOF}
Follow(NT_PRINT) = {PRINT, REPEAT, IF, IDENTIFICADOR, END, EOF}
Follow(NT_VALOR) = {END, INIT}
Follow(NT_REPEAT) = {PRINT, REPEAT, IF, IDENTIFICADOR, EOF}
Follow(NT_CONTENIDO_REPEAT) = {END}
Follow(NT_CONTENIDO_REPEAT') = {END}
Follow(NT_CONDICIONAL) = {PRINT, REPEAT, IF, IDENTIFICADOR, EOF}
Follow(NT_BOOLEANO) = {THEN}
Follow(NT_CONTENIDO_COND) = {END}
Follow(NT_ASIGNACION) = {PRINT, REPEAT, IF, IDENTIFICADOR, EOF}
Follow(NT_EXP) = {END, PARENTESIS_CIERRE}
Follow(NT_EXP') = {END, PARENTESIS_CIERRE}
Follow(NT_T) = {END, SUMA, RESTA, PARENTESIS_CIERRE}
Follow(NT_T') = {END, SUMA, RESTA, PARENTESIS_CIERRE}
Follow(NT_F) = {END, SUMA, RESTA, MULTIPLICACION, DIVISION, POTENCIA, PARENTESIS_CIERRE}

Tabla de análisis sintáctico:

	EOF	PRINT	END	LITERAL	NUMERO_ENTERO	IDENTIFICADOR	REPEAT	INIT	IF	THEN
NT_INICIO		NT_INICIO -> NT_INSTRUCCION NT_INICIO'				NT_INICIO -> NT_INSTRUCCION NT_INICIO'	NT_INICIO -> NT_INSTRUCCION NT_INICIO'		NT_INICIO -> NT_INSTRUCCION NT_INICIO'	
NT_INICIO'	NT_INICIO' -> ε	NT_INICIO' -> NT_INSTRUCCION NT_INICIO'				NT_INICIO' -> NT_INSTRUCCION NT_INICIO'	NT_INICIO' -> NT_INSTRUCCION NT_INICIO'		NT_INICIO' -> NT_INSTRUCCION NT_INICIO'	
NT_INSTRUCCION		NT_INSTRUCCION -> NT_PRINT				NT_INSTRUCCION -> NT_ASIGNACION	NT_INSTRUCCION -> NT_REPEAT		NT_INSTRUCCION -> NT_CONDICIONAL	
NT_PRINT		NT_PRINT -> PRINT NT_VALOR END								
NT_VALOR				NT_VALOR -> LITERAL	NT_VALOR -> NUMERO_ENTERO	NT_VALOR -> IDENTIFICADOR				
NT_REPEAT							NT_REPEAT -> REPEAT NT_VALOR INIT NT_CONTENIDO_REPEAT END			
NT_CONTENIDO_REPEAT		NT_CONTENIDO_REPEAT -> NT_PRINT NT_CONTENIDO_REPEAT'								
NT_CONTENIDO_REPEAT'		NT_CONTENIDO_REPEAT' -> NT_PRINT NT_CONTENIDO_REPEAT'	NT_CONTENIDO_REPEAT' -> ε							
NT_CONDICIONAL									NT_CONDICIONAL -> IF NT_BOOLEANO THEN NT_CONTENIDO_COND END	
NT_BOOLEANO										
NT_CONTENIDO_COND		NT_CONTENIDO_COND -> NT_PRINT	NT_CONTENIDO_COND -> ε							
NT_ASIGNACION						NT_ASIGNACION -> IDENTIFICADOR ASIGNACION NT_EXP END				
NT_EXP					NT_EXP -> NT_T NT_EXP'	NT_EXP -> NT_T NT_EXP'				
NT_EXP'			NT_EXP' -> ε							
NT_T					NT_T -> NT_F NT_T'	NT_T -> NT_F NT_T'				
NT_T'			NT_T' -> ε							
NT_F					NT_F -> NUMERO_ENTERO	NT_F -> IDENTIFICADOR				

TRUE	FALSE	ASIGNACION	SUMA	RESTA	MULTIPLICACION	DIVISION	POTENCIA	PARENTESIS_APERTURA	PARENTESIS_CIERRE
NT_BOOLEANO -> TRUE	NT_BOOLEANO -> FALSE								
			NT_EXP -> SUMA NT_T NT_EXP'	NT_EXP -> RESTA NT_T NT_EXP'				NT_EXP -> NT_T NT_EXP'	
			NT_EXP' -> SUMA NT_T NT_EXP'	NT_EXP' -> RESTA NT_T NT_EXP'					NT_EXP' -> ε
								NT_T -> NT_F NT_T'	
			NT_T' -> ε	NT_T' -> ε	NT_T' -> MULTIPLICACION NT_F NT_T'	NT_T' -> DIVISION NT_F NT_T'	NT_T' -> POTENCIA NT_F NT_T'		NT_T' -> ε
								NT_F -> PARENTESIS_APERTURA NT_EXP PARENTESIS_CIERRE	

Algoritmo:

INICIO Evaluar

REPETIR
    SI la pila de nodos está vacía ENTONCES salir del ciclo

    P = cima de la pila de símbolos
    E = tipo del token actual en la lista de tokens

    nodoActual = cima de la pila de nodos

    SI modoPanico ES verdadero ENTONCES
    Crear nodoError con valor "ERROR"
    Agregar nodoError como hijo de nodoActual
    Marcar nodoActual como inválido

    MIENTRAS aún queden tokens
    tipoActual = tipo del token actual

    SI tipoActual ES "PRINT" o "IF" o "REPEAT" o "IDENTIFICADOR" ENTONCES
        SALIR del ciclo

```
        avanzar apuntador

        SI apuntador está fuera del límite de la lista de tokens ENTONCES
        SALIR del ciclo

        modoPanico = falso
        CONTINUAR con la siguiente iteración

        SI P es un terminal o es "EOF" ENTONCES
        SI P es igual a E ENTONCES
        desapilar símbolo de pila
        desapilar nodo de pila
        asignar valor del token actual al nodoActual
        avanzar apuntador
        SINO
        registrar error de entrada
        modoPanico = verdadero
        desapilar símbolo y nodo
        SINO
        SI existe una producción para (P, E) en la tabla de análisis ENTONCES
        desapilar símbolo y nodo
        produccion = obtener producción de la tabla para (P, E)
        hijos = lista vacía

        PARA CADA símbolo en produccion
                SI símbolo no es "ε" ENTONCES
                crear nodo hijo con ese símbolo
                agregar hijo a lista hijos

        PARA CADA hijo en hijos
                agregar hijo al nodoActual

        invertir hijos
        PARA CADA hijo en hijos
                apilar hijo en pila de nodos

        invertir produccion
        PARA CADA símbolo p en produccion
                SI p no es "ε" ENTONCES
                apilar p en la pila de símbolos
        SINO
        registrar error de entrada
        modoPanico = verdadero
        desapilar símbolo y nodo
HASTA que la pila de símbolos esté vacía

RETORNAR árbol sintáctico construido con la raíz
FIN
```