PL Grammar

entrada → instruccion entrada | Epsilon
instruccion → funcion | asignacion ';' | condicional | inicializacion ';'

funcion → TIPO ID '(' inicializacion ')' '{' contenido '}'
condicional → IF '(' operacionLog ')' '{' contenido '}' | IF '(' operacionLog ')' '{' contenido '}' ELSE '{' contenido '}' | WHILE '(' operacionLog ')' '{' contenido '}'
contenido → asignacion ';' contenido | inicializacion ';' contenido | condicional contenido | retorno ';' contenido | Epsilon
retorno → RETU devuelve
devuelve → operacion | Epsilon
asignacion → ID ASSIGN operacion

operacion → operacionAri | operacionLog
operacionLog → exprLog
exprLog → factor AND exprLog | factor EQ exprLog | factor (EQ, LT, LE, GT, GE, NE, AND, OR, NOT) exprLog | factor | Epsilon
operacionAri → sum PLUS operacionAri | sum MINUS operacionAri | sum
sum → exprAri TIMES exprAri | exprAri DIVIDE exprAri | factor
exprAri → '(' exprAri ')' | factor PLUS factor | factor DIVIDE factor | factor TIMES factor | factor MINUS factor | factor
factor → MINUS NUMBER | NUMBER | ID

Creamos otra "inicializaci�n" que se adec�e a las funciones

inicializacion_cabecera → TIPO ID Resto_cabecera | Epsilon
resto_cabecera → ',' TIPO ID Resto_cabecera | Epsilon

inicializacion → TIPO ID resto | TIPO asignacion resto
TIPO → INT
resto → ',' ID resto | ',' ID asignacion resto | Epsilon