Parse -> Program\_Code $

Program\_Code -> Lpp\_Program Program\_Code  
 | Html Program\_Code  
 |ϵ

Lpp\_Program -> Program\_Header   
 Statement\_List

Program\_Header -> Types\_List  
 | Declare  
 |Methods\_List  
 | ϵ

Types\_List -> Types\_Structure Types\_List   
 |ϵ

Types\_Structure -> TIPO ID ES Type   
 |REGISTRO ID Declare FIN REGISTRO

Type -> entero  
 |real  
 |booleano  
 |caracter  
 |cadena [ const\_entero ]  
 | ARREGLO [ Array\_Size ] DE Type  
 | ARCHIVO Arch\_Type   
 |ID

Arch\_Type-> DE Type  
 | Secuencial

Declare-> Declarar Declare\_Variable Declare  
 | ϵ

Declare\_Variables -> Variables\_Group

Variables\_Group -> Type ID ID\_List

ID\_List-> , ID ID\_List  
 | ϵ

Array\_Size -> const\_entero Integer\_List

Integer\_List -> , const\_entero Integer\_List  
 | ϵ

Methods\_List -> procedimiento ID Params\_List  
 Declare   
 Method\_Body  
 Methods\_List  
 | funcion ID Params\_List : Type  
 Declare   
 Method \_Body  
 Methods\_List  
 | ϵ

Method \_Body -> inicio  
 Statement\_List   
 fin

Function \_Body -> inicio  
 Statement\_List  
 retorne Expression  
 fin

Params\_List -> ( Declare\_Params )  
 | ϵ

Declare\_Params -> Param Param\_Group

Param\_Group -> , Param Param\_Group  
 | ϵ

Param -> var Type ID  
 |Type ID

Statement\_List -> Statement Statement\_List  
 | ϵ

Statement -> Statement\_Si  
 |Statement\_Mientras   
 |Statement\_Llamar   
 |Statement\_Case  
 |Statement\_Abrir\_Archivo   
 |Statement\_Escribir\_Archivo   
 |Statement\_Leer\_Archivo   
 |Statement\_Para   
 |Statement\_Repita   
 |Statement\_ Assignment  
 |Statement\_Escriba  
 |Statement\_Cerrar\_Archivo   
 |Statement\_Return

Statement\_Return-> retorne Expression

Statement\_Si -> si Expression entonces  
 Statement\_List  
 Statement\_Sino  
 fin si

Statement\_Sino-> sino Statement\_SinoP  
 | ϵ

Statement\_SinoP-> Statement\_Si  
 | Statement\_List

Statement\_Para -> para Variable <- Expression hasta Expression haga  
 Statement\_List  
 fin para

Variable -> Simple\_Variable Compuest\_Variable

Simple\_Variable -> ID Array\_Variable

Compuest\_Variable -> . Variable  
 | ϵ

Array \_Variable -> [ Expression Expression\_List ]  
 | ϵ

Expression\_List -> Expression Expression\_Group  
 | ϵ

Expression\_ Group -> , Expression Expression\_Group   
 | ϵ

Statement\_Mientras -> mientras Expression haga  
 Statement\_List  
 fin mientras

Statement\_Repita -> repita   
 Statement\_List  
 hasta Expression

Statement\_Llamar -> llamar ID Expression\_List

Statement\_ Assignment -> Variable <- Expression

Statement\_Case -> caso Variable  
 Case\_List  
 Sino\_Case  
 fin caso

Case\_List -> Define\_Case Case\_Group

Define\_Case -> Literal\_List : Statements\_List

Case\_Group -> Define\_Case Case\_Group   
 | ϵ

Literal\_List -> Literal Literal\_Group

Literal\_Group -> , Literal Literal\_Group  
 | ϵ

Literal-> const\_entero  
 |const\_real  
 |const\_caracter  
 |const\_cadena

Sino\_Case-> sino : Statement\_List  
 | ϵ

Statement\_Escriba -> escribe Expression\_List

Statement\_Abrir\_Archivo -> abrir Expression como Variable Operation\_List

Operation\_List -> para Operation Operation\_Group  
 | ϵ

Operation\_Group ->, Operation  
 | ϵ

Operation-> lectura   
 |escritura

Statement\_Cerrar\_Archivo -> cerrar Variable

Statement\_Escribir\_Archivo -> escribir Expression , Expression\_List

Statement\_Leer\_Archivo -> leer Variable , Variable Variable\_List

Variable\_List -> , Variable Variable\_List   
 | ϵ

Expression -> Expression\_Bool ExpressionP

ExpressionP -> y Expression\_Bool ExpressionP  
 |o Expression\_Bool ExpressionP  
 | ϵ

Expression\_Bool -> Expression\_Basic Expression\_BoolP

Expression\_BoolP -> < Expression\_Basic Expression\_BoolP  
 |> Expression\_Basic Expression\_BoolP  
 |<= Expression\_Basic Expression\_BoolP  
 |>= Expression\_Basic Expression\_BoolP  
 |= Expression\_Basic Expression\_BoolP  
 |<> Expression\_Basic Expression\_BoolP  
 | ϵ

Expression\_Basic -> Factor Expression\_BasicP

Expression\_BasicP-> + Factor Expression\_BasicP  
 |- Factor Expression\_BasicP  
 | ϵ

Factor -> Exp\_Op FactorP

FactorP -> \* Exp\_Op FactorP  
 |/ Exp\_Op FactorP  
 |div Exp\_Op FactorP   
 |mod Exp\_Op FactorP  
 | ϵ

Exp\_Op -> LogicalNot Exp\_OpP

Exp\_OpP-> ^ LogicalNot Exp\_OpP  
 | ϵ

LogicalNot -> no Term  
 | Term

Term -> ID Id\_Term  
 |const\_entero   
 | - Const\_Negative   
 |const\_cadena  
 |const\_caracter   
 |const\_real   
 |verdadero  
 |falso   
 |(Expression\_List)

Id\_term-> (Expression\_ListFunctions)  
 |Variable\_Factor   
 |ϵ

Variable\_Factor -> Array\_Variable Compuest\_Variable

Const\_Negative -> const\_entero  
 | const\_real  
 | (Expression\_List)  
 | Id