**Gramática tipo 2 análisis sintáctico ascendente**

Terminales = {{, }, : , ; , - >, ~, ., |, \*, +, ?, CONJ, ID, Le, LeSinEspacio, Dig, SChars, PorcentajeDoble, Cadena, \_, Especiales }

Nota: aquí los terminales son todos mis tokens.

No Terminales = {<inicio>, ...<>}

Inicio = <inicio>

**Producciones**

<Inicio> ::= {<Instrucciones>}

<instrucciones> ::= <instrucciones> <instrucción>| <instrucción>

<Instrucción> ::= <DeclaracionCONJ> | <DeclaracionesExps> <DeclaracionPorcentajes> <DefinicionEntrada>

<DeclaracionCONJ> ::= CONJ : ID -> <DefinicionCONJ>;

<DefinicionCONJ> ::= <DefinicionVirgulilla> | <DefinicionConComas>

<DefinicionVirgulilla> ::= LeSinEspacio~LeSinEspacio | Dig~Dig | {\_}~SChars | SChars~SChars

<DefinicionConComas> ::= <DefinicionConComas>,<ComaElemento> | <ComaElemento>

<ComaElemento> ::= LeSinEspacio | {\_} | Dig

<DeclaracionesExps> ::= <DeclaracionesExps> <DeclaracionExp> | <DeclaracionExp>

<DeclaracionExp> ::= ID -> (<DefinicionExp> | {ID} | Cadena) ;

<DefinicionExp> ::= <DefinicionExp><Operador> | <Operador>

<0perador> ::= . | {|} | \* | + | ? | {ID} | Cadena | Especiales

<DeclaracionPorcentajes> ::= <DeclaracionPorcentajes> PorcentajeDoble | PorcentajeDoble

<DefinicionEntrada> ::= ID : Cadena ;

EXTRA POR SI ACASO

<DefinicionExp> ::= <DefinicionExp><Operacion> | <Operación>

<Operación> ::= ((? | \* | +)({ID}| Cadena | <Operación> ))

| (({|} | .) ({ID}| Cadena | <Operación>) ({ID}| Cadena))