Imagen que contiene dibujo, plato, taza

Descripción generada automáticamenteUniversidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 2

Guillermo Alfredo Peitzner Estrada – 201504468.

# Gramática Ascendente

|  |  |
| --- | --- |
| PRODUCCION | REGLA |
| INIT -> CUERPO\_GLOBAL | INIT.VAL = CUERPO\_GLOBAL.VAL; |
| CUERPO\_GLOBAL -> LISTA\_GLOBAL | CUERPO\_GLOBAL.VAL = LISTA\_GLOBAL.VAL; |
| CUERPO\_GLOBAL -> | CUERPO\_GLOBAL.VAL = NONE; |
| LISTA\_GLOBAL -> LISTA\_GLOBAL1 INSTRUCCION\_GLOBAL | LISTA\_GLOBAL1.ADD(INSTRUCCION\_GLOBAL); LISTA\_GLOBAL.VAL = LISTA\_GLOBAL1.VAL; |
| LISTA\_GLOBAL -> INSTRUCCION\_GLOBAL | LISTA\_GLOBAL.VAL = Lista(INSTRUCCION\_GLOBAL.VAL); |
| INSTRUCCION\_GLOBAL -> DECLARACION; | ESTRUCTURA ; | FUNCION | INSTRUCCION\_GLOBAL.VAL = DECLARACION.VAL | ESTRUCTURA.VAL | FUNCION.VAL; |
| ESTRUCTURA -> struct identificador { CARACTERISTICAS } | ESTRUCTURA.VAL = Estructura(identificador, CARACTERISTICAS.VAL, lineno); |
| CARACTERISTICAS -> LISTA\_CARACTERISTICAS | CARACTERTISCIAS.VAL = LISTA\_CARACTERISTICAS.VAL; |
| CARACTERISTICAS -> | CARACTERTISICAS.VAL = NONE; |
| LISTA\_CARACTERISTICAS -> LISTA\_CARACTERISTICAS CARACTERISTICA | LISTA\_CARACTERISTICAS1.APPEND(CARACTERISTICA.VAL); LISTA\_CARACTERISTICAS.VAL = LISTA\_CARACTERISTICAS1.VAL; |
| LISTA\_CARACTERITISCAS -> CARACTERISTICA | LISTA\_CARACTERISTICAS.VAL = Lista(CARACTERISTICA.VAL); |
| CARACTERISTICA -> DECLARACION; | CARACTERISTICA.VAL = DECLARACION.VAL; |
| FUNCION -> TIPO identificador ( PARAMETROS ) { CUERPO\_LOCAL } | FUNCION.VAL = Funcion(TIPO.VAL, identificador, PARAMETROS.VAL, CUERPO.VAL); |
| PARAMETROS -> LISTA\_PARAMETROS | PARAMETROS.VAL = LISTA\_PARAMETROS.VAL; |
| PARAMETROS -> | PARAMETROS.VAL = NONE; |
| LISTA\_PARAMETROS -> LISTA\_PARAMETROS1 , PARAMETRO | LISTA\_PARAMETROS1.ADD(PARAMETRO.VAL); LISTA\_PARAMETROS.VAL = LISTA\_PARAMETROS1.VAL; |
| LISTA\_PARAMETROS -> PARAMETRO | LISTA\_PARAMETROS.VAL = Lista(PARAMETRO.VAL); |
| PARAMETRO -> TIPO identificador | PARAMETRO.VAL = Parametro(TIPO.VAL, false, identificador); |
| PARAMETRO -> TIPO & identificador | PARAMETRO.VAL = Parametro(TIPO.VAL, true, identificador); |
| CUERPO\_LOCAL -> LISTA\_LOCAL | CUERPO\_LOCAL.VAL = LISTA\_LOCAL.VAL; |
| CUERPO\_LOCAL -> | CUERPO\_LOCAL.VAL = NONE; |
| LISTA\_LOCAL -> LISTA\_LOCAL1 INSTRUCCION\_LOCAL | LISTA\_LOCAL1.ADD(INSTRUCCION\_LOCAL.VAL); LISTA\_LOCAL.VAL = LISTA\_LOCAL1.VAL; |
| LISTA\_LOCAL -> INSTRUCCION\_LOCAL | LISTA\_LOCAL.VAL = Lista(INSTRUCCION\_LOCAL.VAL); |
| INSTRUCCION\_LOCAL -> ETIQUETA | SALTO | DECLARACION ; | ASIGNACION ; | IF | SWITCH | WHILE | DO | FOR | PRINT ; | METODO ; | INSTRUCCION\_LOCAL.VAL = ETIQUETA.VAL | SALTO.VAL | DECLARACION.VAL | ASIGNACION.VAL | IF.VAL | SWITCH.VAL | WHILE.VAL | DO.VAL | FOR.VAL | PRINT.VAL | METODO.VAL; |
| PRINT -> printf ( LISTA\_EXPRESIONES ) | PRINT.VAL = PrintF(LISTA\_EXPRESIONES.VAL, lineno); |
| INSTRUCCION\_LOCAL -> continue; | INSTRUCCION\_LOCAL.VAL = Continue(lineno); |
| INSTRUCCION\_LOCAL -> break; | INSTRUCCION\_LOCAL.VAL = Break(lineno); |
| INSTRUCCION\_LOCAL -> return EXPRESION; | INSTRUCCION\_LOCAL.VAL = Return(EXPRESION.val, lineno); |
| INSTRUCCION\_LOCAL -> return; | INSTRUCCION\_LOCAL.VAL = Return(NONE, lineno); |
| METODO -> identificador ( EXPRESIONES ) | METODO.VAL = Metodo(identidicador, EXPRESIONES.VAL, lineno); |
| ETIQUETA -> identificador: | ETIQUETA.VAL = Etiqueta(identificador, lineno); |
| SALTO -> goto identificador; | SALTO.VAL = Salto(identificador, lineno); |
| DECLARACION -> TIPO LISTA\_DECLARACION | DECLARACION.VAL = Declaracion(TIPO.VAL, LISTA\_DECLARACION.VAL); |
| LISTA\_DECLARACION -> LISTA\_DECLARACION1 , DECLARACION\_FINAL | LISTA\_DECLARACION1.ADD(DECLARACION\_FINAL.VAL); LISTA\_DECLARACION.VAL = LISTA\_DECLARACION1.VAL; |
| LISTA\_DECLARACION -> DECLARACION\_FINAL | LISTA\_DECLARACION.VAL = Lista(DECLARACION\_FINAL.VAL); |
| DECLARACION\_FINAL -> identificador INDICES | DECLARACION\_FINAL.VAL = DeclaracionFinal(identificador, INDICES.VAL, NONE, lineno); |
| DECLARACION\_FINAL -> identificador INDICES = EXPRESION | DECLARACION\_FINAL.VAL = DeclaracionFinal(identificador, INDICES.VAL, EXPRESIONES.VAL, lineno); |
| INDICES -> ACCESOS | INDICES.VAL = ACCESOS.VAL |
| INDICES -> | INDICES.VAL = NONE; |
| ACCESOS -> ACCESOS1 ACCESO | ACCESOS1.ADD(ACCESO.VAL); ACCESOS.VAL = ACCESOS1.VAL; |
| ACCESOS -> ACCESO | ACCESOS.VAL = Lista(ACCESO.VAL); |
| ACCESO -> [ EXPRESION ] | ACCESO.VAL = EXPRESION.VAL; |
| ACCESO -> [ ] | ACCESO.VAL = Lista(); |
| ASIGNACION -> identificador INDICES COMPUESTO EXPRESION | ASIGNACION.VAL = AsignacionNormal(identificador, INDICES.VAL, COMPUESTO.VAL, EXPRESION.VAL, lineno); |
| ASIGNACION -> identificador INDICES . identificador INDICES COMPUESTO EXPRESION | ASIGNACION.VAL = AsignacionEstructura(identificador, INDICES.VAL, identificador, INDICES.val, COMPUESTO.VAL, EXPRESION.VAL, lineno); |
| ASIGNACION -> identificador ++ | ASIGNACION.VAL = AsignacionAumento(identificador, lineno); |
| ASIGNACION -> ++ identificador | ASIGNACION.VAL = AsignacionAumento(identificador, lineno); |
| ASIGNACION -> identificador -- | ASIGNACION.VAL = AsignacionDecremento(identificador, lineno); |
| ASIGNACION -> -- identificador | ASIGNACION.VAL = AsignacionDecremento(identificador, lineno); |
| COMPUESTO -> = | += | -= | \*= | /= | %= | <<= | >>= | & | |= | ^= | COMPUESTO.VAL = = | += | -= | \*= | /= | %= | <<= | >>= | & | |= | ^=; |
| IF -> if ( EXPRESION ) { CUERPO\_LOCAL } | IF.VAL = If(EXPRESION.VAL, CUERPO\_LOCAL.VAL, NONE, NONE, lineno); |
| IF -> if ( EXPRESION ) { CUERPO\_LOCAL } ELSE | IF.VAL = If(EXPRESION.VAL, CUERPO\_LOCAL.VAL, NONE, ELSE.VAL, lineno); |
| IF -> if ( EXPRESION ) { CUERPO\_LOCAL } ELSEIF IF\_FINAL | IF.VAL = If(EXPRESION.VAL, CUERPO\_LOCAL.VAL, ELSEIF.VAL, IF\_FINAL.VAL, lineno); |
| IF\_FINAL -> ELSEIF IF\_FINAL | IF\_FINAL.VAL = Lista(ELSEIF) + IF\_FINAL.VAL; |
| IF\_FINAL -> ELSE | IF\_FINAL.VAL = Lista(ELSE.VAL); |
| IF\_FINAL -> | IF\_FINAL.VAL = NONE; |
| ELSEIF -> else if ( EXPRESION ) { CUERPO\_LOCAL } | ELSEIF.VAL -> ElseIf(EXPRESION.VAL, CUERPO\_LOCAL.VAL); |
| ELSE -> else { CUERPO\_LOCAL } | ELSE.VAL = Else(CUERPO\_LOCAL.VAL); |
| SWITCH -> switch ( EXPRESION ) { CASES DEFAULT\_CASE } | SWITCH.VAL = Switch(EXPRESION.VAL, CASES.VAL, DEFAULT\_CASE.VAL, lineno); |
| CASES -> LISTA\_CASE | CASES.VAL = LISTA\_CASE.VAL |
| CASES -> | CASES.VAL = NONE; |
| LISTA\_CASE = LISTA\_CASE1 CASE | LISTA\_CASE1.ADD(CASE.VAL); LISTA\_CASE.VAL = LISTA\_CASE1.VAL; |
| LISTA\_CASE -> CASE | LISTA\_CASE.VAL = Lista(CASE.VAL); |
| CASE -> case EXPRESION : CUERPO\_LOCAL | CASE.VAL = Case(EXPRESION.VAL, CUERPO\_LOCAL.VAL); |
| DEFAULT\_CASE -> default : CUERPO\_LOCAL | DEFAULT\_CASE.VAL = DefaultCase(CUERPO\_LOCAL.VAL); |
| WHILE -> while ( EXPRESION ) { CUERPO\_LOCAL } | WHILE.VAL = While(EXPRESION.VAL, CUERPO\_LOCAL.val, lineno); |
| DO -> do { CUERPO\_LOCAL } while ( EXPRESION ) ; | DO.VAL = Do(CUERPO\_LOCAL.VAL, EXPRESION.VAL, lineno); |
| FOR -> for ( INICIO\_FOR ; EXPRESION ; ASIGNACION ) { CUERPO\_LOCAL } | FOR.VAL = For(INICIO\_FOR.VAL, EXPRESION.VAL; ASIGNACION.VAL, CUERPO\_LOCAL.VAL, lineno); |
| INICIO\_FOR -> DECLARACION | ASIGNACION | INICIO\_FOR.VAL = DECLARACION.VAL | ASIGNACION.VAL; |
| EXPRESION -> EXPRESION1 [+|-|\*|/|%] EXPRESION2 | EXPRESION.VAL -> ExpresionAritmetica(EXPRESION1.VAL, [+|-|\*|/|%], EXPRESION2.VAL); |
| EXPRESION -> EXPRESION1 [==|!=|>|<|>=|<=] EXPRESION2 | EXPRESION.VAL = ExpresionRelacional(EXPRESION1.VAL, [==|!=|>|<|>=|<=], EXPRESION2.VAL); |
| EXPRESION -> EXPRESION1 [and|or] EXPRESION2 | EXPRESION.VAL = ExpresionLogica(EXPRESION1.VAL, [and|or], EXPRESION2.VAL); |
| EXPRESION -> EXPRESION1 [<<|>>|&|"|"|^] EXPRESION2 | EXPRESION.VAL = ExpresionBit(EXPRESION1, [<<|>>|&|"|"|^], EXPRESION2); |
| EXPRESION -> EXPRESION1 ? EXPRESION2 : EXPRESION3 | EXPRESION.VAL = ExpresionTernaria(EXPRESION1, EXPRESION2, EXPRESION3); |
| EXPRESION -> [-|!|~] EXPRESION1 | EXPRESION.VAL = ExpresionUnaria([-|!|~], EXPRESION1); |
| EXPRESION -> & identificador | EXPRESION.VAL = ExpresionReferencia(identificador); |
| EXPRESION -> METODO | EXPRESION.VAL = METODO.VAL; |
| EXPRESION -> ( EXPRESION1 ) | EXPRESION.VAL = EXPRESION1.VAL; |
| EXPRESION -> identificador INDICES . identificador INDICES | EXPRESION.VAL = ExpresionEstructura(identificador, INDICES.VAL, identificador, INDICES.VAL); |
| EXPRESION -> identificador ACCESOS | EXPRESION.VAL = ExpresionIdentificadorArreglo(identificador, ACCESOS); |
| EXPRESION -> { EXPRESIONES } | EXPRESION.VAL = ExpresionElementos(EXPRESIONES.VAL); |
| EXPRESION -> sizeof ( TIPO ) | EXPRESION.VAL = SizeOf(TIPO.VAL); |
| EXPRESION -> identificador ++ | EXPRESION.VAL = ExpresionAumentoDecremento(identificador, ++, post); |
| EXPRESION -> ++ identificador | EXPRESION.VAL = ExpresionAumentoDecremento(identificador, ++, pre); |
| EXPRESION -> identificador -- | EXPRESION.VAL = ExpresionAumentoDecremento(identificador, --, post); |
| EXPRESION -> -- identificador | EXPRESION.VAL = ExpresionAumentoDecremento(identificador, --, pre); |
| EXPRESION -> scanf ( ) | EXPRESION.VAL = ExpresionScan(); |
| EXPRESION -> ( TIPO ) EXPRESION | EXPRESION.VAL = ExpresionCasteo(TIPO.VAL, EXPRESION.VAL); |
| EXPRESION -> carácter | EXPRESION.VAL = caracter; |
| EXPRESION -> cadena | EXPRESION.VAL = cadena; |
| EXPRESION -> entero | EXPRESION.VAL = entero; |
| EXPRESION -> decimal | EXPRESION.VAL = decimal; |
| EXPRESION -> identificador | EXPRESION.VAL = identificador; |
| EXPRESIONES -> LISTA\_EXPRESIONES | EXPRESIONES.VAL = LISTA\_EXPRESIONES.VAL |
| EXPRESIONES -> | EXPRESIONES.VAL = NONE; |
| LISTA\_EXPRESIONES -> LISTA\_EXPRESIONES1 , EXPRESION | LISTA\_EXPRESIONES1.ADD(EXPRESION.VAL); LISTA\_EXPRESIONES.VAL = LISTA\_EXPRESIONES1.VAL; |
| LISTA\_EXPRESIONES -> EXPRESION | LISTA\_EXPRESIONES.VAL = Lista(EXPRESION.VAL); |
| TIPO -> int | char | double | float | void | TIPO.VAL = Tipo([int|char|double|float|void], None); |
| TIPO -> struct identificador | TIPO.VAL = Tipo(struct, identificador); |