Universidad 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);