DEFINICION DIRIGIDA POR SINTAXIS

INTEGRANTES:

- *Aguilar Castro Carlos Alfonso*
- *Bustamante Piza Karla Mireli*
- *Ugalde Vivo José Francisco*

Facultad de Ingeniería, UNAM

Grupo: 01

Compiladores

Producción	Reglas semánticas
PROGRAMA -> DECLARACIONES FUNCIONES	TS_pila = nuevaPilaTabSimb() TT_pila = nuevaPilaTabTip() TS_push(TS_pila, nuevaTS(“Global”)) TT_push(TT_pila, nuevaTT(“Global”)) PROGRAMA.code = FUNCIONES.code PilaTSpop(TS_pila) PilaTTpop(TT_pila)
DECLARACIONES -> TIPO LISTA_VAR DECLARACIONES	Tipo = Tipo.tipo
DECLARACIONES -> TIPO_REGISTRO LISTA_VAR SCOLON DECLARACIONES	Tipo = TIPO_REGISTRO.tipo
TIPO_REGISTRO -> STRUCT START DECLARACIONES END	TS_push(TS_pila, nuevaTS(“STRUCT”)) TT_push(TT_pila, nuevaTT(“STRUCT”)) TTS = PilaTSpop(TS_pila) TTP = PilaTTpop(TT_pila) TTS.tt = T.tt TIPO_REGISTRO.tipo = TT_nuevo(getCima(TT_pila),nuevoTipo(“STRUCT”,TTS.dirMax,-1,TTS))
TIPO -> BASE	Base = BASE.base
TIPO -> TIPO_ARREGLO	TIPO.tipo = TIPO_ARREGLO.tipo
BASE -> INT	BASE.tipo = ent
BASE -> FLOAT	BASE.tipo = real
BASE -> DOUBLE	BASE.tipo = dreal
BASE -> CHAR	BASE.tipo = car
BASE -> SIN	BASE.tipo = sin

TIPO_ARREGLO -> LPAR NUM RPAR TIPO_ARREGLO	Si NUM.tipo = 0 Entonces
	  Si NUM.dir > 0 Entonces
	  TIPO_ARREGLO.tipo = TT_nuevo(getCima(TT_pila),nuevoTipo(“Arreglo”,NUM.dir,TIPO_ARREGLO.tipo,NULL))
	  Sino
	  Error(“El tamaño no es valido para el arreglo indicado”)
	  Fin Si
	Sino
	  Error(“El tamaño indicado para el arreglo no es entero”)
	Fin Si

TIPO_ARREGLO -> Epsilon	TIPO_ARREGLO.tipo = base

LISTA_VAR -> ID LISTA_VAR_P	Si IDexists(getCima(TS_pila),ID.dir) == -1 Entonces
	  TS_nuevo(getCima(TT_pila), getCima(TS_pila), nuevoSimb(ID.dir, Tipo, ”variable”, NULL))
	Sino
	  Error(“El ID ya existe”)
	Fin

LISTA_VAR_P : COMMA ID LISTA_VAR_P	Si IDexists(getCima(TS_pila),ID.dir) == -1 Entonces

	TS_nuevo(getCima(TT_pila), getCima(TS_pila), nuevoSimb(ID.dir, Tipo, ”variable”, NULL)) Sino Error(“El ID ya existe”) Fin
FUNCIONES -> DEF TIPO ID LPAR ARGUMENTOS RPAR START DECLARACIONES SENTENCIAS END FUNCIONES1	Si ¡IDexists(getCima(TS_pila),ID.dir) Entonces TS_push(TS_pila, nuevaTS(ID.dir)) Lista_ret =nuevaLista() Si cRet(Lista_ret, TIPO.tipo) Entonces L=nuevaEtiqueta() Backpatch(SENTENCIAS.nextlist, L) FUNCIONES.code = etiqueta(ID.dir) || SENTENCIAS.code || etiqueta(L) || FUNCIONES1.code Sino Error(“El tipo de la función no coincide con los valores”) Fin Si Sino Error(“El ID ya existe”) Fin Si
FUNCIONES -> Epsilon	FUNCIONES.code = genCode(“”)
ARGUMENTOS -> LISTA_ARG	ARGUMENTOS.lista = LISTA_ARG.lista ARGUMENTOS.num = LISTA_ARG.num
ARGUMENTOS -> SIN	ARGUMENTOS.lista = NULL ARGUMENTOS.num = 0
LISTA_ARG -> ARG LISTA_ARG_P	LISTA_ARG.lista = nuevaListaArg() nuevoArg(LISTA_ARG.lista, ARG.tipo)
LISTA_ARG_P -> COMMA ARG LISTA_ARG_P	nuevoArg(LISTA_ARG.lista, ARG.tipo)
ARG -> TIPO_ARG ID	ARG.tipo = TIPO_ARG.tipo nuevaTS(getCima(TT_pila),getCima(TS_pila), nuevoSimb(ID.dir, TIPO_ARG.tipo, “variable”, NULL))
TIPO_ARG -> BASE PARAM_ARR	Base = BASE.base TIPO_ARG.tipo = PARAM_ARR.tipo
PARAM_ARR -> LPAR RPAR PARAM_ARR1	PARAM_ARR.tipo = nuevoTT(getCima(TT_pila),nuevoTip(“Arreglo”, 0, PARAM_ARR1.tipo,NULL))
PARAM_ARR -> Epsilon	PARAM_ARR.tipo = base
SENTENCIAS -> SENTENCIA SENTENCIAS_P	L = nuevaEtiqueta() Backpatch(SENTENCIAS_P.nextList,L) SENTENCIAS_P.nextList = SENTENCIA.nextList SENTENCIAS.code = SENTENCIAS_P.code || etiqueta(L) || SENTENCIA.code
SENTENCIAS -> Epsilon	SENTENCIAS.nextList = NULL FUNCIONES.code = genCode(“”)
SENTENCIAS_P -> SENTENCIA SENTENCIAS_P	L = nuevaEtiqueta() Backpatch(SENTENCIAS_P.nextList,L) SENTENCIAS_P.nextList = SENTENCIA.nextList SENTENCIAS_P.code = SENTENCIAS_P.code || etiqueta(L) || SENTENCIA.code

SENTENCIA -> IF E_BOOL THEN SENTENCIA END	L = nuevaEtiqueta() Backpatch(E_BOOL.trueList,L) SENTENCIA.nextList = comb(E_BOOL.falseList, SENTENCIA.nextList) SENETENCIA.code = E_BOOL.code || etiqueta(L) || SENTENCIA.code
SENTENCIA -> IF E_BOOL THEN SENTENCIA1 ELSE SENTENCIA2 END	L = nuevaEtiqueta() L2 = nuevaEtiqueta() Backpatch(E_BOOL.trueList,L) Backpatch(E_BOOL.falseList,L2) SENTENCIA.nextList = comb(SENTENCIA1.nextList, SENTENCIA2.nextList) SENTENCIA.code = E_BOOL.code || etiqueta(L) || SENTENCIA1.code || genCode(“goto”) || genCode(getFisrt(SENTENCIA1.nextList)) || etqueta(L2) || SENTENCIA2.code
SENTENCIA -> WHILE E_BOOL DO SENTENCIA END	L = nuevaEtiqueta() L2 = nuevaEtiqueta() Backpatch(SENTENCIA.nextList,L) Backpatch(E_BOOL.trueList,L2) SENTENCIA.nextList = E_BOOL.falseList SENTENCIA.code = etiqueta(L) || E_BOOL.code || etiqueta(L2) || SENTENCIA.code || genCode(“goto”) || genCode(getFirst(SENTENCIA.nextList))
SENTENCIA -> DO SENTENCIA WHILE E_BOOL SCOLON	L = nuevaEtiqueta() Backpatch(E_BOOL.trueList,L) SENTENCIA.nextList = E_BOOL.falseList SENTENCIA.code = etiqueta(L) || SENTENCIA.code || E_BOOL.code
SENTENCIA -> SWITCH LPAR VARIABLE RPAR DO CASES PREDETERMINADO END	SWITCH = variable.dir Backpatch(E_BOOL.trueList,L) SENTENCIA.nextList = CASES.nextList Comb(SENTENCIA.nextList, PREDETERMINADO.nextList) SENETNCIA.code = CASES.code || PREDETERMINADO.code
SENTENCIA -> VARIABLE ASIGN2 EXPRESION SCOLON	SENTENCIA.nextList = NULL Temp = red(EXPRESION.dir,EXPRESION.tipo, VARIABLE.tipo) SENTENCIA.code = VARIABLE.dir “=” EXPRESION.dir
SENTENCIA -> WRITE EXPRESION SCOLON	SENTENCIA.code = EXPRESION.code || genCode(“ESCRIBIR”) || EXPRESION.dir SENTENCIA.nextList = NULL
SENTENCIA -> READ VARIABLE SCOLON	SENTENCIA.code = genCode(“LEER”) || VARIABLE.dir SENTENCIA.nextList = NULL
SENTENCIA -> RETURN SCOLON	SENTENCIA.nextList = NULL SENTENCIA.code = genCode(“return”)
SENTENCIA -> RETURN EXPRESION SCOLON	SENTENCIA.nextList = NULL agregarArg(Lista_retorno,EXPRESION.tipo)

	SENTENCIA.code = EXPRESION.code || genCode(“return”) || EXPRESION.dir
SENTENCIA -> TERM SCOLON	T = nuevaTemporal() SENTENCIA.nextList = nuevaLista() Backpatch(SENTENCIA.nextList,L) SENTENCIA.code = genCode(“goto”) || T
SENTENCIA -> START SENTENCIAS END	SENTENCIA.nextList = SENTENCIAS.nextList SENTENCIA.code = SENTENCIAS.code
CASES -> CASE NUM DCOLON SENTENCIA CASES1	L = nuevaEtiqueta() L2 = nuevaEtiqueta() Backpatch(SENTENCIA.nextList,L) CASES.nextList = CASES1.nextList Comb(CASES.nextList,SENTENCIA.nextList) CASES.code = genCode(“if”) || switch || genCode(“=”) || NUM.dir || genCode(“goto”) || genCode(getFirst(SENTENCIA.nextList)) || genCode(“goto”) || genCode(L2) || etiqueta(L) || SENTENCIA.code || eiqueta(L2) || CASES.code
CASES -> CASE NUM DCOLON SENTENCIA	L = nuevaEtiqueta() Backpatch(SENTENCIA.nextList,L) CASES.nextList = SENTENCIA.nextList CASES.code = genCode(“if”) || switch || genCode(“=”) || NUM.dir || genCode(“goto”) || genCode(L) || genCode(getFirst(SENTENCIA.nextList)) || SENTENCIA.code
PREDETERMINADO -> PRED DCOLON SENTENCIA	PREDETERMINADO.nextList = SENTENCIA.nextList PREDETERMINADO.code = SENTENCIA.code
PRDETERMINADO -> Epsilon	PREDETERMINADO.nextList = NULL PREDETERMINADO.code = genCode(“”)
E_BOOL -> E_BOOL1 AND E_BOOL2	L = nuevaEtiqueta() Backpatch(E_BOOL1.trueList,L) E_BOOL.trueList = E_BOOL2.trueList E_BOOL.falseList = Comb(E_BOOL1.falseList, E_BOOL2.falseList) E_BOOL.code = E_BOOL1.code || etqieta(L) || E_BOOL2.code
E_BOOL -> E_BOOL1 OR E_BOOL2	L = nuevaEtiqueta() Backpatch(E_BOOL1.falseList,L) E_BOOL.trueList = Comb(E_BOOL1.trueList ,E_BOOL2.trueList) E_BOOL.falseList = E_BOOL2.falseList E_BOOL.code = E_BOOL1.code || etqieta(L) || E_BOOL2.code
E_BOOL -> NOT E_BOOL1	E_BOOL.trueList = E_BOOL1.falseList E_BOOL.falseList = E_BOOL1.trueList E_BOOL.code = E_BOOL1.code
E_BOOL -> RELACIONAL	E_BOOL.trueList = RELACIONAL.trueList E_BOOL.falseList = RELACIONAL.falseList
E_BOOL -> TRUE	Temp0 = nuevoIndx() E_BOOL.trueList = nuevaLista()

	agregarLista(E_BOOL.trueList,Temp0) E_BOOL.code = genCode(“goto”) || Temp0
E_BOOL -> FALSE	Temp0 = nuevoIndx() E_BOOL.falseList = nuevaLista(Temp0) E_BOOL.code = genCode(“goto”) || Temp0
RELACIONAL -> EXPRESION	RELACIONAL.dir = EXPRESION.dir RELACIONAL.tipo = EXPRESION.tipo RELACIONAL.trueList = NULL RELACIONAL.falseList = NULL
RELACIONAL -> RELACIONAL1 MORE RELACIONAL2	Temp0 = nuevoIndx() Temp1 = nuevoIndx() RELACIONAL.trueList = nuevaLista(Temp0) RELACIONAL.falseList = nuevaLista(Temp1) RELACIONAL.code = genCode(“if”) || RELACIONAL1.dir || genCode(“>”) || RELACIONAL2.dir || genCode(“goto”) || Temp0 || genCode(“goto”) || Temp1
RELACIONAL -> RELACIONAL1 LESS RELACIONAL2	Temp0 = nuevoIndx() Temp1 = nuevoIndx() RELACIONAL.trueList = nuevaLista(Temp0) RELACIONAL.falseList = nuevaLista(Temp1) RELACIONAL.code = genCode(“if”) || RELACIONAL1.dir || genCode(“<”) || RELACIONAL2.dir || genCode(“goto”) || Temp0 || genCode(“goto”) || Temp1
RELACIONAL -> RELACIONAL1 MOREEQ RELACIONAL2	Temp0 = nuevoIndx() Temp1 = nuevoIndx() RELACIONAL.trueList = nuevaLista(Temp0) RELACIONAL.falseList = nuevaLista(Temp1) RELACIONAL.code = genCode(“if”) || RELACIONAL1.dir || genCode(“>=”) || RELACIONAL2.dir || genCode(“goto”) || Temp0 || genCode(“goto”) || Temp1
RELACIONAL -> RELACIONAL1 LESSEQ RELACIONAL2	Temp0 = nuevoIndx() Temp1 = nuevoIndx() RELACIONAL.trueList = nuevaLista(Temp0) RELACIONAL.falseList = nuevaLista(Temp1) RELACIONAL.code = genCode(“if”) || RELACIONAL1.dir || genCode(“<=”) || RELACIONAL2.dir || genCode(“goto”) || Temp0 || genCode(“goto”) || Temp1
RELACIONAL -> RELACIONAL1 NOTEQ RELACIONAL2	Temp0 = nuevoIndx() Temp1 = nuevoIndx() RELACIONAL.trueList = nuevaLista(Temp0) RELACIONAL.falseList = nuevaLista(Temp1) RELACIONAL.code = genCode(“if”) || RELACIONAL1.dir || genCode(“<>”) || RELACIONAL2.dir || genCode(“goto”) || Temp0 || genCode(“goto”) || Temp1
RELACIONAL -> RELACIONAL1 ASIGN RELACIONAL2	Temp0 = nuevoIndx() Temp1 = nuevoIndx()

	RELACIONAL.trueList = nuevaLista(Temp0) RELACIONAL.falseList = nuevaLista(Temp1) RELACIONAL.code = genCode(“if”) || RELACIONAL1.dir || genCode(“=”) || RELACIONAL2.dir || genCode(“goto”) || Temp0 || genCode(“goto”) || Temp1
EXPRESION -> LPAR EXPRESION1 RPAR	EXPRESION.dir = EXPRESION1.dir EXPRESION.tipo = EXPRESION1.tipo
EXPRESION -> VARIABLE	EXPRESION.dir = VARIABLE.dir EXPRESION.tipo = VARIABLE.tipo
EXPRESION -> NUM	EXPRESION.dir = NUM.dir EXPRESION.tipo = ent
EXPRESION -> CARAC	EXPRESION.dir = CARAC.dir EXPRESION.tipo = carac
EXPRESION -> EXPRESION1 PLUS EXPRESION2	EXPRESION.tipo = max(EXPRESION1.tipo,EXPRESION2.tipo) Temp0 = amp(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) Temp1 = amp(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) EXPRESION.dir = nuevaTemp() EXPRESION.code = EXPRESION.dir || genCode(“=”) || Temp0 || genCode(“+”) || Temp1
EXPRESION -> EXPRESION1 MINUS EXPRESION2	EXPRESION.tipo = max(EXPRESION1.tipo,EXPRESION2.tipo) Temp0 = amp(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) Temp1 = amp(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) EXPRESION.dir = nuevaTemp() EXPRESION.code = EXPRESION.dir || genCode(“=”) || Temp0 || genCode(“-”) || Temp1
EXPRESION -> EXPRESION1 MUL EXPRESION2	EXPRESION.tipo = max(EXPRESION1.tipo,EXPRESION2.tipo) Temp0 = amp(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) Temp1 = amp(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) EXPRESION.dir = nuevaTemp() EXPRESION.code = EXPRESION.dir || genCode(“=”) || Temp0 || genCode(“*”) || Temp1
EXPRESION -> EXPRESION 1 DIV EXPRESION2	EXPRESION.tipo = max(EXPRESION1.tipo,EXPRESION2.tipo) Temp0 = amp(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) Temp1 = amp(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) EXPRESION.dir = nuevaTemp() EXPRESION.code = EXPRESION.dir || genCode(“=”) || Temp0 || genCode(“/”) || Temp1

EXPRESION -> EXPRESION 1 MOD EXPRESION2	EXPRESION.tipo = max(EXPRESION1.tipo,EXPRESION2.tipo) Temp0 = amp(EXPRESION1.dir, EXPRESION1.tipo, EXPRESION.tipo) Temp1 = amp(EXPRESION2.dir, EXPRESION2.tipo, EXPRESION.tipo) EXPRESION.dir = nuevaTemp() EXPRESION.code = EXPRESION.dir || genCode("=") || Temp0 || genCode("%") || Temp1
VARIABLE -> ID VARIABLE_COMP	Si existsID(getCima(TS_pila), ID.dir) Entonces ID_pilaPush(ID_pila,ID.dir) VARIABLE.tipo = VARIABLE_COMP.tipo Si getTipo(getCima(TS_pila),ID.dir) = VARIABLE_COMP.tipo Entonces nombre = getVar(getCima(TS_pila),ID.dir) Si nombre = "funcion" Entonces VARIABLE.dir = VARIABLE_COMP.dir VARIABLE.code = VARIABLE_COMP.code Sino VARIABLE.dir = ID.dir VARIABLE.code = genCode("") Fin Si ID_pilaPop(ID_pila) Sino Errr("El ID no existe") Fin Si
VARIABLE_COMP -> DATO_EST_SIM	VARIABLE_COMP.tipo = DATO_EST_SIM.base VARIABLE_COMP.code = DATO_EST_SIM.code
VARIABLE_COMP -> ARREGLO	VARIABLE_COMP.dir = ARRGELO.dir VARIABL_COMP.tipo = ARREGLO.base
VARIABLE_COMP ->LPAR PARAMETROS RPAR	Templista = getArgs(getCima(TS_pila), getCima(ID_pila)) Si compararListas(Templista, PARAMETROS.lista) ¡= -1 Entonces Temp = getTipo(getCima(TS_pila),getCima(IS_pila)) VARIABLE_COMP.tipo = Temp VARIABLE_COMP.dir = nuevaTemp() VARIABLE_COMP.code PARAMETROS.code || VARIABLE.dir || genCode("= CALL") || getCima(ID_pila) || genCode(",") || getSize(Templista)
DATO_EST_SIM -> Epsilon	Temp = getTipo(getCima(TS_pila), getCima(ID_pila)) Nombre = getNombre(getCima(TT_pila),Temp) Si nombre = "STRUCT" Entonces Temp0 = nuevaTemp() Dirección = getDir(getCima(TS_pila),getCima(ID_pila)) DATO_EST_SIM.code = Temp0 || genCode("=") || direccion Fin Si DATO_EST_SIM.base = getTipo(getCima(TS_pila),getCima(ID_pila))

DATO_EST_SIM -> DATO_EST_SIM COLON ID	nombre = getNombre(getCima(TT_pila),DATO_EST_SIM.base)
	Si nombre = “STRUCT” Entonces
	  Temp0 = nuevaTemp()
	  Temp1 = nuevaTemp()
	
	  Temp = getCima(TT_pila)
	  TempBase = DATO_EST_SIM.base
	  TS_pilaPush(TS_pila,getSymTab(Temp,TempBase))
	  TT_pilaPush(TT_pila,getTypeTab(getCima(TS_pila)))
	
	  Direccion = getDir(getCima(TS_pila),ID.dir)
	
	  DATO_EST_SIM.code = Temp0 || genCode(“=”) || Direccion || Temp1 || genCode(“=”) || DATO_EST_SIM.dir || genCode(“+”) || Temp0
	  DATO_EST_SIM.base = getTipo(getCima(TS_pila),ID.dir)
	  DATO_EST_SIM.dir = Temp1
	  TS_pilaPop()
	  TT_pilaPop()
	Sino
	  Error(“La variable no corresponde a una estructura”)
	Fin

ARREGLO -> LPAR EXPRESION RPAR	Temp = nuevaTemp() ARREGLO.dir = Temp TempTipo = getTipo(getCima(TS_pila),getCima(ID_pila)) ARREGLO.base = getBase(getCima(TT_pila),TempTipo) ARREGLO.code = EXPRESION.code || Temp || genCode(“=”) || EXPRESION.dir || genCode(“*”) || getTam(getCima(TT_pila),TempTipo)
ARREGLO -> LPAR EXPRESION RPAR ARREGLO1	Temp0 = nuevaTemp() Temp1 = nuevaTemp() ARREGLO.dir = Temp1 TempTipo = getTipo(getCima(TS_pila),getCima(ID_pila)) ARREGLO.base = getBase(getCima(TT_pila),TempTipo) ARREGLO.code = Temp0 || genCode(“=”) || EXPRESION.dir || genCode(“*”) || getTam(getCima(TT_pila),ARREGLO1.base) || Temp1 || genCode(“=”) || ARREGLO1.dir || genCode(“+”) || Temp0
PARAMETROS -> LISTA_PARAM	PARAMETROS.lista = LISTA_PARAM.lista PARAMETROS.code = LISTA_PARAM.code
PARAMETROS -> Epsilon	PARAMETROS.lista = NULL
LISTA_PARAM -> EXPRESION	LISTA_PARAM.lista =crearLitsaArg() AgregarArg(LISTA_PARAM.lista, EXPRESION.tipo) LISTA_PARAM.code = EXPRESION.code

LISTA_PARAM -> LISTA_PARAM1 COMMA EXPRESION	LISTA_PARAM.lista = LISTA_PARAM1.lista AgregarArg(LISTA_PARAM.lista, EXPRESION.tipo) LISTA_PARAM.code = LISTA_PARAM1.code || EXPRESION.code