Universidad Nacional Autónoma de México

Facultad de Ingeniería

Compiladores

Grupo: 01

Esquema de traducción

Tapia Álvarez Jorge Saúl

Amalfi Figueroa Issac

Alvarez Solís Iván

Escamilla Jaime Neftalí

Fecha de Entrega: 19 de junio de 2020

Profesor: ING. Mercado Martínez Adrián Ulises

programa → declaraciones { dir = 0 StackTT = newStackTT() StackTS = newStackTS() ts = newSymTab() tt = newTypeTab() StackTT.push(tt) StackTS.push(ts) TablaDeCadenas = newTablaCadenas() } **n**funciones
declaraciones → tipo lista_var { type = tipo.tipo } **n**declaraciones1
declaraciones → **registro \n inicio**declaraciones1 **\n fin \n**declaraciones2 { base = base.tipo tipo.tipo = tipo arreglo.tipo }
declaraciones → ε
tipo → base tipo_arreglo { tipo.tipo=tipo; }
base → **ent** { base.tipo = ent }
base → **real** { base.tipo = real }
base → dreal { base.tipo = dreal }
base → car { base.tipo = car }
base → sin { base.tipo = sin }
tipo_arreglo → [**num**] { si num.tipo = ent y num.val > 0 entonces tipo arreglo.tipo = StackTT.getCima().addTipo(”array”, num.val, tipo arreglo1.tipo) en otro caso Error(”El ındice tiene que ser entero y mayor que cero”) fin } tipo_arreglo1
tipo_arreglo → ε {tipo arreglo.tipo = base }
lista_var → lista_var1 **, id**{ si StackTS.getCima().getId(id.lexval) = -1 entonces StackT S.getCima().addSym(id.lexval, tipo, dir, ”var”) dir = dir + StackT T.getCima().getT am(tipo) en otro caso Error(”El identificador ya fue declarado”) fin }
lista_var → **id** { si StackTS.getCima().getId(id.lexval) = -1 entonces StackT S.getCima().addSym(id.lexval, tipo, dir, ”var”) dir = dir + StackT T.getCima().getT am(tipo) en otro caso

Error(”El identificador ya fue declarado”) fin }
funciones → **func**tipo **id** {si StackTS.getFondo().getId(id.lexval) 6= -1 entonces StackT S.getF ondo().addSym(id.lexval, tipo, −−, ”func”) StackDir.push(dir) FuncT ype = tipo.tipo FuncReturn = f alse dir = 0 StackT T.push(tt) StackT S.push(ts) dir = StackDir.pop() add quad(code,’label’, −, −, id.lexval) L = newLabel() backpatch(code, sentencias.next, L) add quad(code,’label’, −, −, L) StackT T.pop() StackT S.pop() dir = StackDir.pop() StackT S.getCima().addArgs(id.lexval, argumentos.lista) si (tipo.tipo 6= sin) y (FuncReturn = false) entonces Error(la funci on no tiene valor de retorno) endif en otro caso Error(”El identificador ya fue declarado”) fin } (argumentos)**inicio \n**declaraciones sentencias**\n fin \n** funciones
funciones → ε { printf(“Fin del programa”); }
argumentos → listar_arg { argumentos.lista = lista arg.lista }
argumentos →**sin** { argumentos.lista = nulo }
lista_arg → lista_arg { lista arg.lista = lista arg1.lista lista arg.lista.add(arg.tipo) } arg
arg → tipo **id**{ si StackTS.getCima().getId(id.lexval) = -1 entonces StackT S.getCima().addSym(id.lexval, tipo, dir, ”var”) dir = dir + StackT T.getCima().getT am(tipo) en otro caso Error(”El identificador ya fue declarado”) fin arg.tipo = tipo arg.tipo }
sentencias → sentencias**\n**{ L = newLabel() backpatch(code, sentecias.listnext, L) sentencias.listnext = sentencia.listnext

} sentencia
sentencias → sentencia {sentencias.listnext = sentencia.listnext}
sentencia → **si** expresion_booleana { L = newLabel() sentencias \n fin backpatch(code, expresion booleana.listtrue, L) sentencia.listnext = combinar(expresion booleana.listfalse, sentencias.listnext) } **entonces \n**sentencias**\n fin**
sentencia → **si** expresion_booleana **\n**{L = newLabel() L1 = newLabel() backpatch(code, expresion boolean.listtrue, L) backpatch(code, expresion boolean.listfalse, L1) sentencia.listnext = combinar(sentencias1.listnext, sentencias2.listnext) } sentencias **\n sino \n**sentencias**\n fin**
sentencia → **mientras \n**expresion_booleana { L = newLabel() L1 = newLabel() backpatch(code, sentencias.listnext, L) backpatch(code, expresion boolean.listtrue, L1) sentencia.listnext = expresion booleana.listfalse add quad(code, ”goto”, -, -, L) L = newLabel() backpatch(code, expresion boolean.listtrue, L) backpatch(code, sentencias.listnext, L1) sentencia.listnext = expresion booleana.listfalse add quad(code, ”label”, -, -, L) } **hacer \n**sentencias **\n fin**
sentencia → **hacer \n**sentencia **\n**{backpatch(verdadero): sentencia.dir=sentencias.dir if expresion_booleana.val = true goto verdadero;} **mientras que**expresion_booleana

sentencia → **id :=**expresion {
si StackTS.getCima().getId(id.lexval) 6= -1 entonces t = StackT
S.getCima().getT ipo(id.lexval)
d = StackT S.getCima().getDir(id.lexval)
α = reducir(expresion.dir, expresion.tipo, t) add
quad(code, ” = ”, α, −, ”Id” + d)
en otro caso
Error(”El identificador no ha sido declarado”)
fin
sentecia.listnext= nulo
}

sentencia → **escribir**expresion { add quad(code, ”print”, expresion.dir, -,-) sentecia.listnext= nulo }
sentencia → **leer**variable { add quad(code, ”scan”, -, -,variable.dir) sentecia.listnext= nulo }
sentencia → **devolver**{ si FuncType = sin entonces add quad(code, ”return”, −, −, −) en otro caso Error(”La funci on debe retornar alg un valor de tipo ” + FuncType) fin sentencia.listnext= nulo }
sentencia → **devolver**expresion { si FuncType 6= sin entonces α = reducir(expresion.dir, expresion.tipo,FuncType) add quad(code, ”return”, expresion.dir, −, −) FuncReturn = true en otro caso Error(”La funci on no puede retornar alg un valor de tipo ”) fin sentencia.listnext= nulo }
sentencia → **segun**(expresion) **n**casos predeterminado **n** {backpatch(casos) setencia.dir=caso.dir+setencias.dir}**fin**
sentencia → **terminar** { I = newIndex() add quad(code, ”goto”, -, -, I) sentencia.listnext = newList() sentencia.listnext.add(I) }
casos →**caso num:** **n**sentencias **n**{backpatch(caso) casos.dir=sentencia.dir Casos.val=setencia.val}
casos →casos **n caso num:** **n**sentencias **n** {backpatch(casos) casos.dir=casos1.dir+setencias.dir}
casos →casos **n caso num:** **n**sentencias **n** {backpatch(casos) casos.dir=casos1.dir+setencias.dir}
predeterminado → predet:\nsentencias {predeterminado.dir=sentencias.dir}
predeterminado → ε {predeterminado.dir=null}
expresion_booleana → expresion_booleana **oo**expresion_booleana { L = newLabel() expresion booleana2 backpatch(code, expresion booleana1.listfalse, L) expresion booleana.listtrue = combinar(expresion booleana1.listtrue, expresion booleana2.lissttrue) expresion booleana.listfalse= expresion booleana2.listfalse add quad(code, ”label”, -, -, L) }

expresion_booleana → expresion_booleana **yy**expresion_booleana { L = newLabel() expresion booleana2 backpatch(code, expresion booleana1.listtrue, L) expresion booleana.listtrue =expresion booleana2.listtrue expresion booleana.listfalse =combinar(expresion booleana1.listfalse, expresion booleana2.lisstfalse) add quad(code, ”label”, -, -, L) }
expresion_booleana → **no**expresion_booleana { expresion booleana.listtrue = expresion booleana1.listfalse expresion booleana.listfalse = expresion booleana1.listtrue } expresion_booleana → relacional { expresion booleana.listtrue = relacional.listtrue expresion booleana.listfalse = relacional.listfalse }
expresion_booleana → **verdadero**{ I = newIndex() expresion booleana.listtrue = newList() expresion booleana.listtrue.add(I) add quad(code, ”goto”, -, -, I) expresion booleana.listfalse = nulo }
expresion_booleana → **falso** { I = newIndex() expresion booleana.listtrue = nulo expresion booleana.listfalse = newList() expresion booleana.listfalse.add(I) add quad(code, ”goto”, -, -, I) }
relacional → relacional <relacional { relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional2.tipo) α1 = ampliar(relacional1.dir, relacional1.tipo, relacional.tipo) α2 = ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) add quad(code, ”<”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1) }
relacional → relacional >relacional { relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I)

relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional2.tipo) α1 = ampliar(relacional1.dir, relacional1.tipo, relacional.tipo) α2 = ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) add quad(code, ”>”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1) }
relacional → relacional <=relacional { relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional2.tipo) α1 = ampliar(relacional1.dir, relacional1.tipo, relacional.tipo) α2 = ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) add quad(code, ”<=”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1) }
relacional → relacional >=relacional { relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional2.tipo) α1 = ampliar(relacional1.dir, relacional1.tipo, relacional.tipo) α2 = ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) add quad(code, ”>=”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1) }
relacional → relacional ==relacional { relacional.listtrue = newList() relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional2.tipo) α1 = ampliar(relacional1.dir, relacional1.tipo, relacional.tipo) α2 = ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) add quad(code, ”==”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1) }
relacional → relacional <>relacional { relacional.listtrue = newList()

relacional.listfalse = newList() I= newIndex(), I1 = newIndex() relacional.listtrue.add(I) relacional.listfalse.add(I1) relacional.tipo = max(relacional1.tipo, relacional2.tipo) α1 = ampliar(relacional1.dir, relacional1.tipo, relacional.tipo) α2 = ampliar(relacional2.dir, relacional2.tipo, relacional.tipo) add quad(code, ”<>”,α1 ,α2, I) add quad(code, ”goto”, -, -, I1) }
relacional → expresion { relacional.tipo = expresion.tipo relacional.dir = expresion.dir }
expresion → expresion **+**expresion { expresion.tipo = max(expresion1.tipo, expresion2.tipo) expresion.dir = newTemp() α1 = ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) α2 = ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) add quad(code, ”+”,α1 ,α2, expresion.dir) }
expresion → expresion **-**expresion { expresion.tipo = max(expresion1.tipo, expresion2.tipo) expresion.dir = newTemp() α1 = ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) α2 = ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) add quad(code, ”–”,α1 ,α2, expresion.dir) }
expresion → expresion ***** expresion { expresion.tipo = max(expresion1.tipo, expresion2.tipo) expresion.dir = newTemp() α1 = ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) α2 = ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) add quad(code, ”*”,α1 ,α2, expresion.dir) }
expresion → expresion /expresion { expresion.tipo = max(expresion1.tipo, expresion2.tipo) expresion.dir = newTemp() α1 = ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) α2 = ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) add quad(code, ”/”,α1 ,α2, expresion.dir) }
expresion → expresion **%**expresion { expresion.tipo = max(expresion1.tipo, expresion2.tipo) expresion.dir = newTemp()

α1 = ampliar(expresion1.dir, expresion1.tipo, expresion.tipo) α2 = ampliar(expresion2.dir, expresion2.tipo, expresion.tipo) add quad(code, ”%”,α1 ,α2, expresion.dir) }
expresion → (expresion) { expresion.dir = expresion1.dir expresion.tipo = expresion1.tipo }
expresion → variable { expresion.dir = newTemp() expresion.tipo = variable.tipo add quad(code, ”*”, variable.base[variable.dir] , -, expresion.dir) }
expresion → **num** {expresoin.tipo = num.tipo expresion.dir = num.val }
expresion → **cadena**{expresion.tipo = cadena expresion.dir = TablaDeCadenas.add(cadena) }
expresion → **caracter**{expresion.val=caracter.val;}
expresion → **id**(parametros) {expresion.val=id.val}
param_arr → **id[ ]**{if cimaPTS.get(id) != -1 then param_arr.id=id param_arr .tipo=TT.gettipo(id) genCode (param_arr)}
param_arr → param_arr **[ ]**{if cimaPTS.get(param_arr1) != -1 then param_arr.id=param_arr1.id param_arr .tipo=TT.gettipo(param_arr1) genCode (param_arr)}
variable → **id**parte_arreglo { if TS.getval(id,parte_arreglo.dimension)!=-1 Variable=TS.getval(id,parte_arreglo.dimension) }
variable → **id.id1** { variable.dir=id1.dir variable.val=id1.val}
parte_arreglo → [expresion] parte_arreglo {
parte_arreglo.dimension=expresion.val,parte_arreglo.val }
parte_arreglo.dimension=expresion.val,parte_arreglo.val }
parte_arreglo → ε {parte_arreglo.val=null}
parte_arreglo → ε {parte_arreglo.val=null}
parametros → lista_param {parametros.tipos=lista_parametros.tipos;}
parametros → ε { parametros.tipos=5 }
lista param → lista_param , expresion {lista_parametros.tipos=list_parametros1.tipos lista_parametros.tipos.add(expresion.tipos) Lista_parametros.dir=expresion.dir }

lista param → expresion {lista_parametros.tipos=expresion.tipos
Lista_parametros.dir=expresion.dir
}