Proyecto Final Compiladores

Definición Dirigida por Sintaxis

Producción	Regla Semántica
Programa → declaraciones funciones P → D F	dir = 0 STS.push(newTS ()) STT.push(newTT ()) P.codigo = F.codigo
Declaraciones → |tipo lista_var; declaraciones |tipo_registro lista_var; declaraciones |ϵ D →T Lv; D D → Tr Lv; D D → ϵ	 Lv.tipo = T.tipo Lv.tipo = Tr.tipo
Tipo_registro → estructura inicio declaraciones fin Tr → struct {D}	STS.push(newTS ()) STT.push(newTT ()) SDir.push(dir) dir = 0 TS = STS.pop() TT = STT.pop() tam = TS dir = SDir.pop() T.tipo=STT.getTop().insert(”struct”,tam,TS)
Tipo → base tipo_arreglo T → B Ta	Tipo.type = tipo_arreglo.type Base = B.base
Base → ent | real | dreal | car | sin B →ent B →real B →dreal B →car B →sin	 B.base = int B.base = float B.base =double B.base = char B.base = void
Tipo_arreglo → |(num) tipo_arreglo |ϵ Ta→ (num) Ta_1 Ta→ϵ	Si num.type = ent Entonces Si num.dir > 0 Entonces tipo arreglo.type = STT.getTop().insert(’array’,num,Ta_1.tipo) Sino error(’...’) Fin Si Sino error(’...’) Fin Si

Lista _var →
  | Lista_var$_{1,}$ id
  | id
 	
Lv→Lv,id
Lv→ id
	Si STS.getTop().existe(id) Entonces
	  STS.getTop().insert(id, typeGBL, dir, 'var', null, null)
	  dir dir + STT.getTop().getTam(typeGBL)
	Sino
	  error('el id no esta definido')
	Fin Si

Funciones →
  | def tipo id(argumentos) inicio declaraciones sentencias fin funciones
 	
  | ϵ
 	
F→ define T id (A) {D Ss}
F→ ϵ
	Si no STS.getTop().existe(id) Entonces
	  STS.push(newTS())
	  STT.push(newTT())
	  SDir.push(dir)
	  dir = 0
	 listaRET = newListRet()
	 Si cmpRet(lista retorno, tipo.type) Entonces
	  L = newLabel()
	  backpatch(sentencias.nextlist, L)
	  genCode(label L)
	  STS.pop()
	  STT.pop()
	  Sino
	  error('Los tipos no coinciden')
	  Fin Si
	Sino
	  error('El id ya existe')
	Fin Si

Argumentos →
  | lista_arg
  |sin
 	
A→La
A→sin	

Lista_arg →
  |Lista_arg , arg
  |arg
 	
La→ La , Arg
La → Arg	

Arg → | tipo_arg id Arg→ Targ id	
Tipo_arg → |base param_arr Targ→ B Pa	

Param_arr→ | () param_arr | ϵ Pa→ () Pa Pa → ϵ	
Sentencias → | sentencias sentencia |sentencia Ss → Ss S Ss → S	
Sentencia → | si e_bool entonces sentencia fin | si e_bool entonces sentencia sino sentencia fin | mientras e_bool hacer sentencia fin |hacer sentencia mientras e_bool; |según (variable) hacer casos predeterminado fin | variable := expresión; |escribir expreseion; |leer variable; |devolver; |devolver expresión; |terminar; |inicio sentencias fin Ss→ if e_bool then Ss_1 Ss→ if e_bool then Ss_1 else Ss_2 Ss→ while e_bool do Ss Ss→ do Ss_1 while e_bool Ss→ swith (V) Ss→ Ss→ Ss→ Ss→ Ss→	
Casos →	
Predeterminado →	
E_bool →	
Relacional →	
Expresion →	
Variable →	
Variable_comp →	
Dato_est_sim →	

Arreglo →	
Parametros →	
Lista_param →	

Nombre	Símbolo	Nombre	Símbolo
Programa	P	Sentencias	Ss
Declaraciones	D	Sentencia	S
Tipo registro	Tr	Casos	C
Tipo	T	Predeterminado	Pre
Base	B	E_bool	Eb
Tipo_arreglo	Ta	Relacional	R
Lista _var	Lv	Expresion	E
Funciones	F	Variable	V
Argumentos	A	Variable_comp	Vc
Lista_arg	La	Dato_est_sim	Des
Arg	Arg	Arreglo	Ar
Tipo_arg	Targ	Parametros	Par
Param_arr	Pa	Lista_param	Lp