

Gramática de Atributos acondicionada para la comprobación de las restricciones contextuales (2º Cuatrimestre)

Prog ::= Cabecera Decs Bloque
Cabecera.err = Cabecera.err v Decs.err v Bloque.err

Cabecera ::= PROGRAM id PYCOMA

Sección de declaraciones

Decs ::= DTipos RDecs
Decs.err = DTipos.err v RDecs.err

RDecs ::= λ
RDecs.err = false

RDecs ::= Procs
RDecs.err = Procs.err

RDecs ::= Vars RDecs2
RDecs.err = Vars.err v RDecs2.err

RDecs2 ::= Procs
RDecs2.err = Procs.err

RDecs2 ::= λ
RDecs2.err = false

Decs ::= Vars RDecs3
Decs.err = Vars.err v RDecs3.err

RDecs3 ::= Procs
RDecs3.err = Procs.err

RDecs3 ::= λ
RDecs3.err = false

Decs ::= λ
Decs.err = false

Declaración de tipos

DTipos ::= SECTIPOS RTipos
DTipos.err = RTipos.err

RTipos ::= RTipos2 RTipos
RTipos0.err = RTipos2.err ∨ RTipos1.err

RTipos ::= λ
RTipos.err = false

RTipos2 ::= id IGUAL Tipo PYCOMA
RTipos2.err = (existeID(RTipos2.tsp, id.lex) ^ (RTipos2.tsp[id.lex].nivel == RTipos2.nivelh)) ∨ Tipo.err

Tipo ::= TIPENT
Tipo.err = false

Tipo ::= TIPBOOL
Tipo.err = false

Tipo ::= id
***Tipo.err = if existeID(Tipo.tsp, id.lex) then
 (Tipo.tsp[id.lex].clase < > Tipo)
 else
 True***

Tipo ::= TPUNTERO Tipo
Tipo0.err = Tipo1.err

Tipo ::= TARRAY [0..numero] of Tipo
Tipo0.err = Tipo1.err ∨ Referencia(Tipo1.tipo, Tipo0.tsph)

Declaración de variables

Vars ::= VAR Tvar2

Vars.err = Tvar2.err

Tvar2 ::= λ

Tvar2.err = false

Tvar2 ::= id RTvar2

***Tvar2.err = (existeID(RTvar2.tsp, id.lex) ^ Tvar2.tsp[id.lex].nivel == Tvar2.nivelh) v
(existeID(RTvar2.tsp, id.lex) v RTvar2.err)***

RTvar2 ::= COMA RTvar2

RTvar2o.err = RTvar21.err

RTvar2 ::= 2PUNTOS Tipos PYCOMA

RTvar2.err = Tipos.err

Tipos ::= TIPENT

Tipos.err = false

Tipos ::= TIPBOOL

Tipos.err = false

Tipos ::= id

***Tipo.err = if existeID(Tipo.tsp, id.lex) then
 (Tipo.tsp[id.lex].clase < > Tipo)
 else
 True***

Declaración de procedimientos

Procs ::= TProc Procs

Procs0.err = TProc.err v Procs1.err

Procs ::= λ

Procs.err = false

TProc ::= PROC id Params PYCOMA BloqProc

TProc.err = Params.err v BloqProc.err v (existeID(TProc.tsp, id.lex) ^ (TProc.tsp[id.lex].nivel == TProc.nivelh))

Params ::= PA ListaParams PC

Params.err = ListaParams.err

Params ::= λ

Params.err = false

ListaParams ::= Params2 RListaParams

ListaParams.err = Params2.err v RListaparams.err

RListaParams ::= PYCOMA ListaParams

RListaParams.err = Listaparams.err

RListaParams ::= λ

RListaParams.err = false

ListaParams ::= TVAR RListaParams2

ListaParams.err = RListaParams2.err

RListaParams2 ::= Params2

RListaParams2.err = Params2.err

RListaParams2 ::= Params2 PYCOMA ListaParams

RListaParams2.err = Params2.err v ListaParams.err

Params2 ::= id RParams2

Params2.err = (existeID(Params2.tsph, id.lex) ^ (Params2.tsp[id.lex].nivel == Params2.nivelh)) v existeID(RParams2.tsp, id.lex) v RParams2.err

RParams2 ::= COMA Params2

RParams2.err = Params2.err

RParams2 ::= 2PUNTOS Tipos

RParams2.err = Tipos.err

BloqProc ::= Decs2 Bloque

BloqProc.err = Decs2.err v Bloque.err

Decs2 ::= Vars
Decs2.err = Vars.err

Decs2 ::= λ
Decs2.err = false

Cuerpo del programa

Bloque ::= INICIO TBloque2 FIN
Bloque.err = TBloque2.err

TBloque2 ::= TSentencia TBloque2
TBloque2₀.err = TSentencia.err \vee TBloque2₁.err

TBloque2 ::= λ
TBloque2.err = false

TSentencia ::= TAsig
TSentencia.err = TAsig.err

TSentencia ::= TRead
TSentencia.err = TRead.err

TSentencia ::= TWrite
TSentencia.err = TWrite.err

TSentencia ::= TNPunt
TSentencia.err = TNPunt.err

TSentencia ::= TLiberar
TSentencia.err = TLiberar.err

TSentencia ::= TLLamadaProc
TSentencia.err = TLLamadaProc.err

TSentencia ::= TIf
TSentencia.err = TIf.err

TSentencia ::= TWhile
TSentencia.err = TWhile.err
TBloque2.nivelh = TWhile.nivelh

TIf ::= SI PA Exp PC ENTONCES INICIO TBloque2 FIN RTif
TIf.err = Exp.err \vee TBloque2.err \vee (Exp.tipo.t \neq Boolean)

RTif ::= λ
RTif.err = false

RTif ::= SINO INICIO TBloque2 FIN

RTif.err = TBloque2.err

TWhile ::= MIENTRAS PA Exp PC HACER INICIO TBloque2 FIN

TWhile.err = Exp.err v TBloque2.err v (Exp.tipo.t < > Boolean)

TLlamadaProc ::= id PA Params3 PC PYCOMA

Params3.paramsh = TLlamadaProc.tsph[id.lex].params

TLlamadaProc.err = ¬existeID(TLlamadaProc.tsph, id.lex) v

(TLlamadaProc.tsph[id.lex].clase < > proc) v Params3.err v

(longitud(TLlamadaProc.tsph[id.lex].params) != Params3.nparams) v

(TLlamadaProc.tsph[id.lex].nivelh < > TLlamadaProc.nivelh)

Params3 ::= ListaParams3

ListaParams3.paramsh = Params3.paramsh

Params3.err = ListaParams3.err

Params3 ::= λ

ListaParams3.err = false

ListaParams3 ::= Exp RListaParams3

ListaParams3.err = (ListaParams3.nparams < > RListaParams3.nparams + 1) v

(ListaParams3.paramsh[1].modo == variable ^ Exp.modo= valor) v

¬compatibles(ListaParams3.paramsh[1].tipo, Exp.tipo, ListaParams3.tsph) v Exp.err
v RListaParams3.err

RListaParams3 ::= COMA ListaParams3

RListaParams3.err = ListaParams2.err

RListaParams3 ::= λ

RListaParams3.err = false

TRead ::= LEER PA id PC PYCOMA

TRead.err = ¬existeID(TRead.tsph, id.lex) v (TRead.tsph[id.lex].clase < > variable)

TWrite ::= ESCRIBIR PA id PC PYCOMA

TWrite.err = ¬existeID(TWrite.tsph, id.lex) v ((TWrite.tsph[id.lex].clase < >

variable) ^ ((TWrite.tsph[id.lex].tipo.t == Entero) v (TWrite.tsph[id.lex].tipo.t ==

boolean))) v (TWrite.tsph[id.lex].nivel != TWrite.nivelh)

TNPunt ::= NUEVO PA id PC PYCOMA

TNPunt.err = ¬existeID(TNPunt.tsph, id.lex) v (TNPunt.tsph[id.lex].tipo.t < >

puntero) v (TNPunt.tsph[id.lex].nivel != TNPunt.nivelh)

TLiberar ::= LIBERAR PA id PC PYCOMA

TLiberar.err = ¬existeID(TLiberar.tsph, id.lex) v (TLiberar.tsph[id.lex].tipo.t < >

puntero) v (TLiberar.tsph[id.lex].nivel != TLiberar.nivelh)

TAsig ::= Descriptor ASIG Exp
TAsig.err* = \neg compatibles(*Descriptor.tipo*, *Exp.tipo*, *TAsig.tsph*) v *Descriptor.err* v *Exp.err

Descriptor ::= Descriptor2
Descriptor.err* = *Descriptor2.err
Descriptor.tipo* = *Descriptor2.tipo

Descriptor2 ::= id
***Descriptor2.error* = \neg existeID(*Descriptor2.tsph*, *id.lex*) v**
(*Descriptor2.tsph*[*id.lex*].*nivel* != *Descriptor2.nivelh*)
***Descriptor2.tipo* =**
if (existeID(*Descriptor2.tsph*, *id.lex*) ^ *Descriptor2.tsph*[*id.lex*].*clase* = *variable*) then
***referencia*(*Descriptor2.tsph*[*id.lex*].*tipo*, *Descriptor2.tsph*)**
else
<t: tipoError>

Descriptor2 ::= Descriptor2 CA Exp CC
Descriptor.err* = *Descriptor2.err* v *Exp.err
***Descriptor.tipo* = if (*Descriptor2.tipo.t* = *Array* ^ *Exp.tipo.t* = *Entero*) then**
***referencia*(*Descriptor2.tipo.tBase*, *Descriptor.tsph*)**
else
<t: tipoError>

Descriptor2 ::= ^Descriptor2
***Descriptor0.err* = (*Descriptor1.tipo.t* != *puntero*)**
***Descriptor0.tipo* = if (*Descriptor1.tipo.t* = *puntero*) then**
***referencia*(*Descriptor21.tipo.tBase*, *Descriptor0.tsph*)**
else
<t: tipoError>

Exp ::= ExpSum RExp
***Exp.tipo* = if compatibles(*ExpSum.tipo*, *RExp.tipo*, *Exp.tsph*) then**
***asigna_tipo*(*ExpSum.tipo*, *RExp.tipo*, *Exp.tsph*)**
else
<t: tipoError>
RExp.tipoh* = *ExpSum.tipo
Exp.err* = *ExpSum.err* v *RExp.err

RExp ::= OpRel Exp
***RExp.tipo* = if tipoRelacion(*RExp.tipoh.t*, *Exp.tipo.t*, *OpRelacion*) then**
<t: Boolean>
else
<t: tipoError>
RExp.err* = *Exp.err

RExp ::= λ
RExp.err* = *false

ExpSum ::= ExpProd RExpSum
ExpSum.tipo = if compatibles(*ExpProd.tipo*, *RExpSum.tipo*, *ExpSum.tsph*) then
 asigna_tipo(*ExpProd.tipo*, *RExpSum.tipo*, *ExpSum.tsph*)
 else
 <*t*: tipoError>
RExpSum.tipoh = *ExpProd.tipo*
ExpSum.err = *ExpProd.err* v *RExpSum.err*

RExpSum ::= OpAd ExpSum
RExpSum.err = *ExpSum.err* v (\neg compatibles(*RExpSum.tipoh*, *ExpSum.tipo*,
RExpSum.tsph)
RExpSum.tipo = tipoNum(*RExpSum.tipo.t*, *ExpSum.tsph*)

RExpSum ::= OR ExpSum
RExpSum.err = *ExpSum.err* v (\neg compatibles(*RExpSum.tipoh*, *ExpSum.tipo*,
RExpSum.tsph)
RExpSum.tipo = tipoBoolean(*RExpSum.tipo.t*, *ExpSum.tsph*)

RExpSum ::= λ
RExpSum.err = false

ExpProd ::= ExpFact RExpProd
ExpProd.tipo = if compatibles(*ExpFact.tipo*, *RExpProd.tipo*, *ExpProd.tsph*) then
 asigna_tipo(*ExpFact.tipo*, *RExpProd.tipo*, *ExpProd.tsph*)
 else
 <*t*: tipoError>
RExpProd.tipoh = *ExpFact.tipo*
ExpProd.err = *ExpFact.err* v *RExpProd.err*

RExpProd ::= OpProd ExpProd
RExpProd.err = *ExpProd.err* v (\neg compatibles(*RExpProd.tipoh*, *ExpProd.tipo*,
RExpProd.tsph)
RExpProd.tipo = tipoNum(*RExpProd.tipo.t*, *ExpProd.tsph*)

RExpProd ::= AND ExpProd
RExpProd.err = *ExpProd.err* v (\neg compatibles(*RExpProd.tipoh*, *ExpProd.tipo*,
RExpProd.tsph)
RExpProd.tipo = tipoBoolean(*RExpProd.tipo.t*, *ExpProd.tsph*)

RExpProd ::= λ
RExpProd.err = false

ExpFact ::= PA Exp PC
ExpFact.err = *Exp.err*
ExpFact.tipo = *Exp.tipo*

ExpFact ::= OpAd numero
ExpFact.err = false
ExpFact.tipo = Entero

ExpFact ::= numero
ExpFact.err = false
ExpFact.tipo = Entero

ExpFact ::= True
ExpFact.err = false
ExpFact.tipo = Boolean

ExpFact ::= False
ExpFact.err = false
ExpFact.tipo = Boolean

ExpFact ::= Not ExpFact
ExpFact0.err = false
ExpFact0.tipo = if (ExpFact1.tipo.t = Boolean) then
 ExpFact0.tipo.t
 else
 <t: tipoError>

ExpFact ::= Descriptor
ExpFact.err = Descriptor.err
ExpFact.tipo = if (Descriptor.tipo.t < > tipoError) then
 tipoDeID(ExpFact.tsph, Descriptor.tipo)
 else
 <t: tipoError>