

Gramática de Atributos acondicionada para la Traducción (2º Cuatrimestre)

Prog ::= Cabecera Decs Bloque
Prog.cod = inicio(Decs.nivel, Decs.dir) // ir_a (Decs.etq) // Decs.cod // Bloque.cod //
stop
Decs.etqh = longInicio +1;
Bloque.etqh = Decs.etq

Cabecera ::= PROGRAM id PYCOMA

Sección de declaraciones

Decs ::= DTipos RDecs
RDecs.etqh = Decs.etqh
Decs.etq = RDecs.etq
Decs.cod = RDecs.cod

RDecs ::= λ
RDecs.etq = RDecs.etqh
RDecs.cod = λ

RDecs ::= Procs
Procs.etqh = RDecs.etqh
RDecs.etq = Procs.etq
RDecs.cod = Procs.cod

RDecs ::= Vars RDecs2
RDecs2.etqh = RDecs.etqh
RDecs.etq = RDecs2.etq
RDecs.cod = RDecs2.cod

RDecs2 ::= Procs
Procs.etqh = RDecs2.eth
RDecs2.etq = Procs.etq
RDecs2.cod = Procs.cod

RDecs2 ::= λ
RDecs2.etq = RDecs2.etqh
RDecs2.cod = λ

Decs ::= Vars RDecs3
RDecs3.etqh = Decs.etqh
Decs.etq = RDecs3.etq
Decs.cod = RDecs3.cod

$RDecs3 ::= Procs$
 $Procs.etqh = RDecs3.eth$
 $RDecs3.etq = Procs.etq$
 $RDecs3.cod = Procs.cod$

$RDecs3 ::= \lambda$
 $RDecs3.etq = RDecs3.etqh$
 $RDecs3.cod = \lambda$

$Decs ::= \lambda$
 $Decs.etq = Decs.etqh$
 $Decs.cod = \lambda$

Declaración de procedimientos

$Procs ::= TDProc\ Procs$
 $TDProc.etqh = Procs_0.etqh$
 $Procs_1.etqh = TDProc.etq$
 $Procs_0.etq = Procs_1.etq$
 $Procs_0.cod = TDProc.cod \parallel Procs_1.cod$

$Procs ::= \lambda$
 $Procs.etq = Procs.etq$
 $Procs.cod = \lambda$

$TDProc ::= PROC\ id\ Params\ PYCOMA\ BloqProc$
 $BloqProc.etqh = TDProc.etqh$
 $TDProc.etq = BloqProc.etq$
 $TDProc.cod = BloqProc.cod$

$BloqProc ::= Decs2\ Bloque$
 $BloqProc.inicio = Bloque.etq$
 $Bloque.etqh = BloqProc.etqh + longPrologo$
 $BloqProc.etq = Bloque.etq + longEpilogo + 1$
 $BloqProc.cod = prologo(BloqProc.nivelh, Decs2.dir) \parallel Bloque.cod$
 $epilogo(BloqProc.nivelh) \parallel ir-indice$

Cuerpo del programa

Bloque ::= INICIO TBloque2 FIN

Bloque.cod = TBloque2.cod

TBloque2.etqh = Bloque.etqh

Bloque.etq = TBloque2.etq

TBloque2 ::= TSentencia TBloque2

TBloque2₀.cod = TSentencia.cod // TBloque2₁.cod

TSentencia.etqh = TBloque2₀.etqh

TBloque2₁.etqh = TSentencia.etq

TBloque2₀.etq = TBloque2₁.etq

TBloque2 ::= λ

Tbloque2.cod = λ

Tbloque2.etq = Cuerpo.etqh

TSentencia ::= TAsig

TSentencia.cod = TAsig.cod

TAsig.etqh = TSentencia.etqh

TSentencia.etq = TAsig.etq

TSentencia ::= TRead

TSentencia.cod = TRead.cod

TRead.etqh = TSentencia.etqh

TSentencia.etq = TRead.etq

TSentencia ::= TWrite

TSentencia.cod = TWrite.cod

TWrite.etqh = TSentencia.etqh

TSentencia.etq = TWrite.etq

TSentencia ::= TNPunt

TSentencia.cod = TNPunt.cod

TNPunt.etqh = TSentencia.etqh

TSentencia.etq = TNPunt.etq

TSentencia ::= TLiberar

TSentencia.cod = TLiberar.cod

TLiberar.etqh = TSentencia.etqh

TSentencia.etq = TLiberar.etq

TSentencia ::= TLLamadaProc

TSentencia.cod = TLLamadaProc.cod

TLLamadProc.etqh = TSentencia.etqh

TSentencia.etq = TLLamadaProc.etq

TSentencia ::= TIf
TSentencia.cod = *TIf.cod*
TIf.etqh = *TSentencia.etqh*
TSentencia.etq = *TIf.etq*

TSentencia ::= TWhile
TSentencia.cod = *TWhile.cod*
TWhile.etqh = *TSentencia.etqh*
TSentencia.etq = *TWhile.etq*

TIf ::= SI PA Exp PC ENTONCES INICIO TBloque2 FIN RTif
TIf.cod = *Exp.cod* // *ir_f(TBloque2.etq+1)* // *TBloque2.cod* // *ir_a(RTif.etq)* // *RTif.cod*
Exp.etqh = *TIf.etqh*
TBloque2.etqh = *Exp.etq+1*
RTif.etqh = *TBloque2.etq*
TIf.etq = *RTif.etq*
Exp.parh = *false*

RTif ::= λ
RTif.etq = *RTif.etqh*
Rtif.cod = λ

RTif ::= SINO INICIO TBloque2 FIN
TBloque2.etqh = *Rtif.etqh+1*
Rtif.etq = *TBloque2.etq*
Rtif.cod = *TBloque2.cod*

TWhile ::= MIENTRAS PA Exp PC HACER INICIO TBloque2 FIN
TWhile.cod = *Exp.cod* // *ir_f(TBloque2.etq+1)* // *TBloque2.cod* // *ir_a(TWhile.etqh)*
Exp.etqh = *TWhile.etqh*
TBloque2.etqh = *Exp.etq +1*
TWhile.etq = *TBloque2.etq +1*
Exp.parh = *false*

TLLamadaProc ::= id PA Params3 PC PYCOMA
TLLamadaProc.cod = *apilaDirRetorno(TLLamadaProc.etq)* // *Params3.cod* // *ir_a(TLLamadaProc.tsph[id.lex].inicio)*
Params3.etqh = *TLLamadaProc.etqh + longApilaRetorno*
TLLamadaProc.etq = *Params3.etq +1*

Params3 ::= ListaParams3
Params3.cod = *inicioPaso* // *ListaParams3.cod* // *finPaso*
ListaParams3.etqh = *Params3.etqh + longInicioPaso*
Params3.etq = *ListaParams3.etq + longFinPaso*

Params3 ::= λ
Params3.cod = λ
Params3.etq = *Params3.etqh*

ListaParams3 ::= Exp **RListaParams3**
RListaParams3.codh = copia //Exp.cod //pasoParametro(Exp.moda, ListaParams3.paramsh
 [1])
LisRealParams.cod = **RListaParams3.cod**
Exp.etqh = **ListaParams3.etqh** + 1
RListaParams3.etqh = **Exp.etq** + longPasoParametro
Exp.parh = **ListaParams3.paramsh[1].moda** == variable

RListaParams3 ::= COMA Exp **RListaParams3**
RListaParams3₁.codh = **RListaParams3₀.codh** // copia// direccionPFormal
 (**RListaParams3₀.paramsh[RListaParams3₀.nparams]**)
 // Exp.cod // pasoParametro(Exp.moda,
 RListaParams3₀.paramsh[RListaParams3₀.nparams])
 RListaParams3₀.cod = **RListaParams3₁.cod**
 Exp.etqh = **RListaParams3₀.etqh** + 1 + longPFormal
 RListaParams3₁.etqh = **Exp.etq** + longPasoParametro
 RListaParams3₀.etq = **RListaParams3₁.etq**
Exp.parh = **RListaParams3₀.paramsh[RListaParams3₀.nparams].moda** == variable

RListaParams3 ::= λ
RListaParams3.etq = **RListaParams3.etqh**
RListaParams3.cod = **RListaParams3.codh**

TRead ::= LEER PA id PC PYCOMA
TRead.cod = lecturaPantalla(**TRead.tsph[id.lex].dir**)
TRead.etq = **TRead.etqh**

TWrite ::= ESCRIBIR PA id PC PYCOMA
TWrite.cod = escrituraPantalla(cimaPila())
TWrite.etq = **TWrite.etqh**

TNPunt ::= NUEVO PA id PC PYCOMA
TNPunt.cod = reservar(**TNPunt.tsph[id.lex].tam**) // desapila_ind
TNPunt.etq = **TNPunt.etqh**

TLiberar ::= LIBERAR PA id PC PYCOMA
TLiberar.cod = liberar(**TLiberart.tsph[id.lex].tam**)
TLiberar.etq = **TLiberar.etqh**

TAsig ::= Descriptor ASIG Exp
TAsig.cod = if compatible(Descriptor.tipo, <t: Entero>, Exp.tsph) v
 compatible(Descriptor.tipo, <t: Boolean>, Exp.tsph) then Descriptor.cod
 // Exp.cod // desapilaIndice() else Descriptor.cod // Exp.cod //
 mueve(Descriptor.tipo.tam)
Descriptor.etqh = **TAsig.etqh**
Exp.etqh = **Descriptor.etq**
TAsig.etq = **Exp.etq** + 1
Exp.parh = false

Descriptor ::= Descriptor2
Descriptor.cod = Descriptor2.cod
Descriptor2.etqh = Descriptor.etqh
Descriptor.etq = Descriptor2.etq

Descriptor2 ::= id
Descriptor2.cod = accesoVar(Descriptor2.tsph[id.lex])
Descriptor2.etq = Descriptor2.etqh + longAccesoVar (Descriptor2.tsph[id.lex])

Descriptor2 ::= Descriptor2[Exp]
Descriptor2₀.cod = Descriptor2₁.cod // Exp.cod //apila(Descriptor2₁.tipo.tBase.tam)
// multiplica // suma
Descriptor2₁.etqh = Descriptor2₀.etqh
Exp.etqh = Descriptor2₁.etq
Descriptor2₀.etq = Exp.etq + 3

Descriptor2 ::= ^Descriptor2
Descriptor2₀.cod = Descriptor2₁.cod // apilaInd
Descriptor2₁.etqh = Descriptor2₀.etqh
Descriptor2₀.etq = Descriptor2₁.etq +1

Exp ::= ExpSum RExp
RExp.codh= ExpSum.cod
Exp.cod= RExp.cod
ExpSum.etqh= Exp.etqh
RExp.etqh= ExpSum.etq
Exp.etq= RExp.etq
ExpSum.parh= Exp.parh

RExp ::= OpRel ExpSum RExp
RExp₁.codh= RExp₀.cod//ExpSum.cod//OpRel.op
RExp₀.cod= RExp₁.cod
ExpSum.etqh= RExp₀.etqh
RExp₁.etqh= ExpSum.etq +1
RExp₀.etq= RExp₁.etq
ExpSum.parh= false

RExp ::= λ
RExpresion.cod= RExpresion.codh
RExpresion.etq= RExpresion.etqh

ExpSum ::= ExpProd RExpSum
RExpSum.codh= ExpProd.cod
ExpSum.cod= RExpSum.cod
ExpProd.etqh= ExpSum.etqh
RExpSum.etqh= ExpProd.etq
ExpSum.etq= RExpSum.etq
ExpProd.parh= ExpSum.parh

RExpSum ::= OpAd ExpProd RExpSum
RExpSum₁.codh = RExpSum₀.cod // ExpProd.cod // OpAd.op
RExpSum₀.cod = RExpSum₁.cod
ExpProd.etqh = RExpSum₀.ethq
RExpSum₁.etqh = ExpProd.etq + 1
RExpSum₀.etq = RExpSum₁.etq
ExpProd.parh = false

RExpSum ::= OR ExpProd RExpSum
RExpSum₁.codh = RExpSum₀.cod // ExpProd.cod // or
RExpSum₀.cod = RExpSum₁.cod
ExpProd.etqh = RExpSum₀.ethq
RExpSum₁.etqh = ExpProd.etq + 1
RExpSum₀.etq = RExpSum₁.etq
ExpProd.parh = false

RExpSum ::= λ
RExpSum.cod = RExpSum.codh
RExpSum.etq = RExpSum.etqh

ExpProd ::= ExpFact RExpProd
RExpProd.codh = ExpFact.cod
ExpProd.cod = RExpProd.cod
ExpFact.etqh = ExpProd.etqh
RExpProd.etqh = ExpFact.etq
ExpProd.etq = RExpProd.etq
ExpFact.parh = ExpProd.parh

RExpProd ::= OpProd ExpFact RExpProd
RExpProd₁.codh = RExpProd₀.cod // ExpFact.cod // OpProd.op
RExpProd₀.cod = RExpProd₁.cod
ExpFact.etqh = RExpProd₀.etqh
RExpProd₁.etqh = ExpFact.etq + 1
RExpProd.etq = RExpProd.etq
ExpFact.parh = false

RExpProd ::= AND ExpFact RExpProd
RExpProd₁.codh = RExpProd₀.cod // ExpFact.cod // and
RExpProd₀.cod = RExpProd₁.cod
ExpFact.etqh = RExpProd₀.etqh
RExpProd₁.etqh = ExpFact.etq + 1
RExpProd₀.etq = RExpProd₁.etq
ExpFact.parh = false

RExpProd ::= λ
RExpProd.cod = RExpProd.codh
RExpProd.etq = RExpProd.etqh

ExpFact ::= (Exp)
ExpFact.cod = *Exp.cod*
Exp.etqh = *ExpFact.etqh*
ExpFact.etq = *Exp.etq*
Exp.parh = *ExpFact.parh*

ExpFact ::= OpAd ExpFact
ExpFact₀.cod = if (*OpAd.op* == *suma*) then *ExpFact₁.cod*
 else *ExpFact₁.cod*) // *resta*
ExpFact.etq = *ExpFact.etqh* + 1

ExpFact ::= numero
ExpFact.cod = *apila(valorDe(numero.lex))*
ExpFact.etq = *ExpFact.etqh* + 1

ExpFact ::= True
ExpFact.cod = *apila(True)*
ExpFact.etq = *ExpFact.etqh* + 1

ExpFact ::= False
ExpFact.cod = *apila(False)*
ExpFact.etq = *ExpFact.etqh* + 1

ExpFact ::= Not ExpFact
ExpFact₀.cod = *ExpFact₁.cod* // *negacion*
ExpFact₁.etqh = *ExpFact₀.etqh*
ExpFact₀.etq = *ExpFact₁.etq* + 1
ExpFact₁.parh = *false*

ExpFact ::= Descriptor
ExpFact.cod = if (*compatible(Descriptor.tipo, <t:Integer>, ExpFact.tsph)* v
compatible(Descriptor.tipo, <t:Bool>, ExpFact.tsph)) ^ ¬*ExpFact.parh* then
Descriptor.cod // *apila_ind* else *Descriptor.cod*
Descriptor.etqh = *ExpFact.etqh*
ExpFact.etq = *Descriptor.etq* + (if (*compatible(Descriptor.tipo, <t:Integer>, ExpFact.tsph)* v
compatible(Descriptor.tipo, <t:Bool>, ExpFact.tsph)) ^ ¬*ExpFact.parh* then 1 else 0)