Fase 3: Desarrollo de constructores de ASTs

Grupo 6

Javier García Viana

Ibon Malles Altolaguirre

David Peromingo Peromingo

Francisco Prieto Gallego

1. Tiny

1.1. Sintaxis abstracta

programa ➔ bloque

bloque ➔ **{** declaraciones instrucciones **}**

- ***Eliminación de terminales sin carga semántica***
  programa ➔ bloque
  bloque ➔ declaraciones instrucciones

declaraciones ➔ lista-declaraciones **&&**
declaraciones ➔ ε

lista-declaraciones ➔ lista-declaraciones **;** declaración
lista-declaraciones ➔ declaración

- ***Eliminación de terminales sin carga semántica***
  declaraciones ➔ lista-declaraciones
  declaraciones ➔ ε

  lista-declaraciones ➔ lista-declaraciones declaración
  lista-declaraciones ➔ declaración

declaración ➔ **proc identificador** parámetros-formales bloque
declaración ➔ **type** tipo **identificador**
declaración ➔ tipo **identificador**

- ***Eliminación de terminales sin carga semántica***
  declaración ➔ **identificador** parámetros-formales bloque
  declaración ➔ **type** tipo **identificador**
  declaración ➔ tipo **identificador**

parámetros-formales ➔ **(** lista-parámetros-formales-e **)**

lista-parámetros-formales-e ➔ lista-parámetros-formales
lista-parámetros-formales-e ➔ ε

lista-parámetros-formales ➔ lista-parámetros-formales **,** parámetro-formal
lista-parámetros-formales ➔ parámetro-formal

- ***Eliminación de terminales sin carga semántica***
  parámetros-formales ➔ lista-parámetros-formales-e

  lista-parámetros-formales-e ➔ lista-parámetros-formales
  lista-parámetros-formales-e ➔ ε

  lista-parámetros-formales ➔ lista-parámetros-formales parámetro-formal
  lista-parámetros-formales ➔ parámetro-formal

- ***Simplificación de géneros envoltorio***
  parámetros-formales ➔ lista-parámetros-formales
  parámetros-formales ➔ ε

lista-parámetros-formales ➔ lista-parámetros-formales parámetro-formal
lista-parámetros-formales ➔ parámetro-formal

parámetro-formal ➔ tipo **& identificador**
parámetro-formal ➔ tipo **identificador**

tipo ➔ T0
T0 ➔ T0 **[literalEntero]**
T0 ➔ T1

T1 ➔ **^** T1
T1 ➔ T2

T2 ➔ **int**
T2 ➔ **real**
T2 ➔ **bool**
T2 ➔ **string**
T2 ➔ **identificador**
T2 ➔ **struct {** lista-campos **}**

- ***Eliminación de terminales sin carga semántica***
  tipo ➔ T0
  T0 ➔ T0 **literalEntero**
  T0 ➔ T1

  T1 ➔ **^** T1
  T1 ➔ T2

  T2 ➔ **int**
  T2 ➔ **real**
  T2 ➔ **bool**
  T2 ➔ **string**
  T2 ➔ **identificador**
  T2 ➔ **struct** lista-campos

- ***Fusión de no terminales del patrón expresión***
  tipo ➔ T
  T ➔ T **literalEntero**
  T ➔ T

  T ➔ **^** T
  T ➔ T

  T ➔ **int**
  T ➔ **real**
  T ➔ **bool**
  T ➔ **string**
  T ➔ **identificador**
  T ➔ **struct** lista-campo

- ***Eliminación de géneros sinónimos***
  tipo ➔ T
  T ➔ T **literalEntero**
  T ➔ **^** T
  T ➔ **int**
  T ➔ **real**
  T ➔ **bool**

T ➔ **string**
T ➔ **identificador**
T ➔ **struct** lista-campo

- ***Simplificación de géneros envoltorio***
  tipo ➔ tipo **literalEntero**
  tipo ➔ **^** tipo
  tipo ➔ **int**
  tipo ➔ **real**
  tipo ➔ **bool**
  tipo ➔ **string**
  tipo ➔ **identificador**
  tipo ➔ **struct** lista-campos

lista-campos ➔ lista-campos **,** campo
lista-campos ➔ campo

campo ➔ tipo **identificador**

- ***Eliminación de terminales sin carga semántica***
  lista-campos ➔ lista-campos campo
  lista-campos ➔ campo

  campo ➔ tipo **identificador**

instrucciones ➔ lista-instrucciones
instrucciones ➔ ε

lista-instrucciones ➔ lista-instrucciones **;** instrucción
lista-instrucciones ➔ instrucción

- ***Eliminación de terminales sin carga semántica***
  instrucciones ➔ lista-instrucciones
  instrucciones ➔ ε

  lista-instrucciones ➔ lista-instrucciones instrucción
  lista-instrucciones ➔ instrucción

instrucción ➔ **@** expresión
instrucción ➔ **call identificador** parámetros-reales
instrucción ➔ **nl**
instrucción ➔ **new** expresión
instrucción ➔ **read** expresión
instrucción ➔ **write** expresión
instrucción ➔ **delete** expresión
instrucción ➔ **while** expresión bloque
instrucción ➔ **if** expresión bloque **else** bloque
instrucción ➔ **if** expresión bloque
instrucción ➔ bloque

- ***Eliminación de terminales sin carga semántica***
  instrucción ➔ **@** expresión
  instrucción ➔ **identificador** parámetros-reales
  instrucción ➔ **nl**
  instrucción ➔ **new** expresión
  instrucción ➔ **read** expresión

instrucción ➔ **write** expresión
instrucción ➔ **delete** expresión
instrucción ➔ **while** expresión bloque
instrucción ➔ **if** expresión bloque **else** bloque
instrucción ➔ **if** expresión bloque
instrucción ➔ bloque

parámetros-reales ➔ **(** lista-expresiones-e **)**

lista-expresiones-e ➔ lista-expresiones
lista-expresiones-e ➔ ε

lista-expresiones ➔ lista-expresiones **,** expresión
lista-expresiones ➔ expresión

- ***Eliminación de terminales sin carga semántica***
  parámetros-reales ➔ lista-expresiones-e

  lista-expresiones-e ➔ lista-expresiones
  lista-expresiones-e ➔ ε

  lista-expresiones ➔ lista-expresiones expresión
  lista-expresiones ➔ expresión

- ***Simplificación de géneros envoltorio***
  parámetros-reales ➔ lista-expresiones
  parámetros-reales ➔ ε

  lista-expresiones ➔ lista-expresiones expresión
  lista-expresiones ➔ expresión

expresión ➔ E0

E0 ➔ E1 **=** E0
E0 ➔ E1

E1 ➔ E1 OP1 E2
E1 ➔ E2

E2 ➔ E2 **+** E3
E2 ➔ E3 **-** E3
E2 ➔ E3

E3 ➔ E4 **and** E3
E3 ➔ E4 **or** E4
E3 ➔ E4

E4 ➔ E4 OP4 E5
E4 ➔ E5

E5 ➔ OP5 E5
E5 ➔ E6

E6 ➔ E6 **[** E0 **]**
E6 ➔ E6 **. identificador**
E6 ➔ E6 **^**
E6 ➔ E7

E7 ➔ **literalEntero**
E7 ➔ **literalReal**
E7 ➔ **identificador**
E7 ➔ **true**
E7 ➔ **false**
E7 ➔ **literalCadena**
E7 ➔ **null**
E7 ➔ **(** E0 **)**

- ***Eliminación de terminales sin carga semántica***
  expresión ➔ E0

  E0 ➔ E1 **=** E0
  E0 ➔ E1

  E1 ➔ E1 OP1 E2
  E1 ➔ E2

  E2 ➔ E2 **+** E3
  E2 ➔ E3 **-** E3
  E2 ➔ E3

  E3 ➔ E4 **and** E3
  E3 ➔ E4 **or** E4
  E3 ➔ E4

  E4 ➔ E4 OP4 E5
  E4 ➔ E5

  E5 ➔ OP5 E5
  E5 ➔ E6

  E6 ➔ E6 E0
  E6 ➔ E6 **. identificador**
  E6 ➔ E6 **^**
  E6 ➔ E7

  E7 ➔ **literalEntero**
  E7 ➔ **literalReal**
  E7 ➔ **identificador**
  E7 ➔ **true**
  E7 ➔ **false**
  E7 ➔ **literalCadena**
  E7 ➔ **null**
  E7 ➔ E0

- ***Fusión de no terminales equivalentes***
  expresión ➔ E

  E ➔ E **=** E
  E ➔ E

  E ➔ E OP1 E
  E ➔ E

  E ➔ E **+** E
  E ➔ E **-** E

E ➔ E

E ➔ E **and** E
E ➔ E **or** E

E ➔ E OP4 E
E ➔ E

E ➔ OP5 E
E ➔ E

E ➔ E E
E ➔ E **. identificador**
E ➔ E **^**
E ➔ E

E ➔ **literalEntero**
E ➔ **literalReal**
E ➔ **identificador**
E ➔ **true**
E ➔ **false**
E ➔ **literalCadena**
E ➔ **null**
E ➔ E

- ***Eliminación de géneros sinónimos***
  expresión ➔ E

  E ➔ E **=** E
  E ➔ E OP1 E
  E ➔ E **+** E
  E ➔ E **-** E
  E ➔ E **and** E
  E ➔ E **or** E
  E ➔ E OP4 E
  E ➔ OP5 E
  E ➔ E E
  E ➔ E **. identificador**
  E ➔ E **^**

  E ➔ **literalEntero**
  E ➔ **literalReal**
  E ➔ **identificador**
  E ➔ **true**
  E ➔ **false**
  E ➔ **literalCadena**
  E ➔ **null**

- ***Eliminación de géneros discriminativos***
  expresión ➔ E

  E ➔ E **=** E
  E ➔ E **+** E
  E ➔ E **-** E
  E ➔ E **and** E
  E ➔ E **or** E
  E ➔ E **<** E

E ➔ E **>** E
E ➔ E **<=** E
E ➔ E **>=** E
E ➔ E **==** E
E ➔ E **!=** E
E ➔ E ***** E
E ➔ E **/** E
E ➔ E **%** E
E ➔ **-** E
E ➔ **not** E
E ➔ E E
E ➔ E **identificador**
E ➔ E **^**

E ➔ **literalEntero**
E ➔ **literalReal**
E ➔ **identificador**
E ➔ **true**
E ➔ **false**
E ➔ **literalCadena**
E ➔ **null**

- ***Simplificación de géneros envoltorio***
  expresión ➔ expresión **=** expresión
  expresión ➔ expresión **+** expresión
  expresión ➔ expresión **-** expresión
  expresión ➔ expresión **and** expresión
  expresión ➔ expresión **or** expresión
  expresión ➔ expresión **<** expresión
  expresión ➔ expresión **>** expresión
  expresión ➔ expresión **<=** expresión
  expresión ➔ expresión **>=** expresión
  expresión ➔ expresión **==** expresión
  expresión ➔ expresión **!=** expresión
  expresión ➔ expresión ***** expresión
  expresión ➔ expresión **/** expresión
  expresión ➔ expresión **%** expresión
  expresión ➔ **-** expresión
  expresión ➔ **not** expresión
  expresión ➔ expresión expresión
  expresión ➔ expresión **. identificador**
  expresión ➔ expresión **^**

  expresión ➔ **literalEntero**
  expresión ➔ **literalReal**
  expresión ➔ **identificador**
  expresión ➔ **true**
  expresión ➔ **false**
  expresión ➔ **literalCadena**
  expresión ➔ **null**

OP1 ➔ **<**
OP1 ➔ **>**
OP1 ➔ **<=**
OP1 ➔ **>=**

OP1 ➔ **==**
OP1 ➔ **!=**

OP4 ➔ *****
OP4 ➔ **/**
OP4 ➔ **%**

OP5 ➔ **-**
OP5 ➔ **not**

1.2. Constructoras de ASTs

Gramática abstracta	Constructor
programa ➔ bloque	prog: Bloq ➔ Prog
bloque ➔ declaraciones instrucciones	bloq: Decs ✖ Instrs ➔ Bloq
declaraciones ➔ lista-declaraciones	siDecs: LDecs ➔ Decs
declaraciones ➔ ε	noDecs: ➔ Decs
lista-declaraciones ➔ lista-declaraciones declaración	muchasDecs: LDecs ✖ Dec ➔ Ldecs
lista-declaraciones ➔ declaración	unaDec: Dec ➔ LDecs
declaración ➔ **identificador** parámetros-formales bloque	decProc: **string** ✖ ParamForms ✖ Bloq ➔ Dec
declaración ➔ **type** tipo **identificador**	decType: Tipo ✖ **string** ➔ Dec
declaración ➔ tipo **identificador**	decVar: Tipo ✖ **string** ➔ Dec
parámetros-formales ➔ lista-parámetros-formales	siParam: LParams ➔ ParamForms
parámetros-formales ➔ ε	noParam: ➔ ParamForms
lista-parámetros-formales ➔ lista-parámetros-formales parámetro-formal	muchosParams: LParams ✖ ParamForm➔ LParams
lista-parámetros-formales ➔ parámetro-formal	unParam: ParamForm ➔ LParams
parámetro-formal ➔ tipo **& identificador**	paramFormRef: Tipo ✖ **string** ➔ ParamForm

parámetro-formal ➔ tipo **identificador**	paramForm: Tipo ✖ **string** ➔ ParamForm
tipo ➔ tipo **literalEntero**	tArray: Tipo ✖ **string** ➔ Tipo
tipo ➔ **^** tipo	tPunt: Tipo ➔ Tipo
tipo➔ **int**	tInt: ➔ Tipo
tipo➔ **real**	tReal: ➔ Tipo
tipo➔ **bool**	tBool: ➔ Tipo
tipo➔ **string**	tString: ➔ Tipo
tipo➔ **identificador**	tIden: **string** ➔ Tipo
tipo➔ **struct** lista-campos	tStruct: LCampos ➔ Tipo
lista-campos ➔ lista-campos campo	muchosCamps: LCampos ✖ Campo ➔ LCampos
lista-campos ➔ campo	unCamp: Campo ➔ LCampos
campo ➔ tipo **identificador**	campo: Tipo ✖ **string** ➔ Campo
instrucciones ➔ lista-instrucciones	siInstrs: LInstrs ➔ Instrs
instrucciones ➔ ε	noInstrs: ➔ Instrs
lista-instrucciones ➔ lista-instrucciones instrucción	muchasInstrs: LInstrs ✖ Instr ➔ Linstrs
lista-instrucciones ➔ instrucción	unaInstr: Instr ➔ Linstrs
instrucción ➔ **@** expresión	arrobaInstr : Exp ➔ Instr
instrucción ➔ **identificador** parámetros-reales	procInstr : **string** ✖ ParamReales ➔ Instr
instrucción ➔ **nl**	nlInstr : ➔ Instr
instrucción ➔ **new** expresión	newInstr : Exp ➔ Instr
instrucción ➔ **read** expresión	readInstr : Exp ➔ Instr
instrucción ➔ **write** expresión	writeInstr : Exp ➔ Instr
instrucción ➔ **delete** expresión	deleteInstr : Exp ➔ Instr

1.3. Gramática s-atribuída

programa ➔ bloque
$programa\,.\,a \;=\; prog(bloque.\,a)$

bloque ➔ **{** declaraciones instrucciones **}**
$bloque\,.\,a \;=\; bloq(declaraciones.\,a,\; instrucciones.\,a)$

declaraciones ➔ lista-declaraciones **&&**
$declaraciones.\,a \;=\; siDecs(listaDeclaraciones.\,a)$

declaraciones ➔ ε
$declaraciones.\,a \;=\; noDecs()$

lista-declaraciones ➔ lista-declaraciones **;** declaración
$listaDeclaraciones_0.\,a \;=\; muchasDecs(listaDeclaraciones_1.\,a,\; declaracion.\,a)$

lista-declaraciones ➔ declaración
$listaDeclaraciones.\,a \;=\; unaDec(declaracion.\,a)$

declaración ➔ **proc identificador** parámetros-formales bloque
$declaracion.\,a \;=\; decProc(identificador.\,lex,\; parametrosFormales.\,a,\; bloque.\,a)$

declaración ➔ **type** tipo **identificador**
$declaracion.\,a \;=\; decType(tipo.\,a,\; identificador.\,lex)$

declaración ➔ tipo **identificador**
$declaracion.\,a \;=\; decVar(tipo.\,a,\; identificador.\,lex)$

parámetros-formales ➔ **(** lista-parámetros-formales-e **)**
$parametrosFormales.\,a \;=\; listaParametrosFormalesE.\,a$

lista-parámetros-formales-e ➔ lista-parámetros-formales
$listaParametrosFormalesE.\,a \;=\; siParam(listaParametrosFormales.\,a)$

lista-parámetros-formales-e ➔ ε
$listaParametrosFormalesE.\,a \;=\; noParam()$

lista-parámetros-formales ➔ lista-parámetros-formales **,** parámetro-formal

$listaParametrosFormales_0.a = muchosParams(listaParametrosFormales_1.a, parametroFormal.a)$

lista-parámetros-formales ➔ parámetro-formal

$listaParametrosFormales.a = unParam(parametroFormal.a)$

parámetro-formal ➔ tipo **& identificador**

$parametroFormal.a = paramFormRef(tipo.a, identificador.lex)$

parámetro-formal ➔ tipo **identificador**

$parametroFormal.a = paramForm(tipo.a, identificador.lex)$

tipo ➔ T0

$tipo.a = T0.a$

T0 ➔ T0 **[literalEntero]**

$T0_0.a = tArray(T0_1.a, literalEntero.lex)$

T0 ➔ T1

$T0.a = T1.a$

T1 ➔ **^** T1

$T1_0.a = tPunt(T1_1.a)$

T1 ➔ T2

$T1.a = T2.a$

T2 ➔ **int**

$T2.a = tInt()$

T2 ➔ **real**

$T2.a = tReal()$

T2 ➔ **bool**

$T2.a = tBool()$

T2 ➔ **string**

$T2.a = tString()$

T2 ➔ **identificador**

$T2.a = tIden(identificador.lex)$

T2 ➔ **struct {** lista-campos **}**

$T2.a = tStruct(listaCampos.a)$

lista-campos ➔ lista-campos **,** campo

$listaCampos_0.a = muchosCamps(listaCampos_1.a, campo.a)$

lista-campos ➔ campo

$listaCampos.a = unCamp(campo.a)$

campo ➔ tipo **identificador**

$campo.tipo = tipo.a$

$campo.id = identificador.lex$

instrucciones ➔ lista-instrucciones

$intrucciones.a = siInstrs(listaInstrucciones.a)$

instrucciones ➔ ε

$intrucciones.a = noInstrs()$

lista-instrucciones ➔ lista-instrucciones **;** instrucción

$listaInstrucciones.a = muchasInstrs(listaInstrucciones.a, instrucción.a)$

lista-instrucciones ➔ instrucción

$listaInstrucciones.a = unaInstr(instrucción.a)$

instrucción ➔ **@** expresión
$intruccion.a = arrobaInstr\ (expresion.a)$

instrucción ➔ **call** identificador parámetros-reales
$intruccion.a = procInstr\ (identificador.lex,\ parametrosReales.a)$

instrucción ➔ **nl**
$intruccion.a = nlInstr\ ()$

instrucción ➔ **new** expresión
$intruccion.a = newInstr(expresion.a)$

instrucción ➔ **read** expresión
$intruccion.a = readInstr(expresion.a)$

instrucción ➔ **write** expresión
$intruccion.a = writeInstr(expresion.a)$

instrucción ➔ **delete** expresión
$intruccion.a = deleteInstr(expresion.a)$

instrucción ➔ **while** expresión bloque
$intruccion.a = whileInstr(expresion.a,\ bloque.a)$

instrucción ➔ **if** expresión bloque **else** bloque
$intruccion.a = ifElseInstr(expresion.a,\ bloque_0.a,\ bloque_1.a)$

instrucción ➔ **if** expresión bloque
$intruccion.a = ifInstr(expresion.a,\ bloque.a)$

instrucción ➔ bloque
$intruccion.a = bloqueInstr(bloque.a)$

parámetros-reales ➔ **(** lista-expresiones-e **)**
$parametrosReales.a = listaExpresionesE.a$

lista-expresiones-e ➔ lista-expresiones
$listaExpresionesE.a = siExp(listaExpresiones.a)$

lista-expresiones-e ➔ ε
$listaExpresionesE.a = noExp()$

lista-expresiones ➔ lista-expresiones **,** expresión
$listaExpresiones_0.a = muchasExp(listaExpresiones_1.a,\ expresion.a)$

lista-expresiones ➔ expresión
$listaExpresiones.a = unaExp(expresion.a)$

expresión ➔ E0
$expresion.a = E0.a$

E0 ➔ E1 **=** E0
$E0_0.a = asignación(E1.a,\ E0_1.a)$

E0 ➔ E1
$E0.a = E1.a$

E1 ➔ E1 OP1 E2

$E1.a = mkop1(OP1.op,\ E1.a,\ E2.a)$

E1 ➔ E2

$E1.a = E2.a$

E2 ➔ E2 **+** E3

$E2_0.a = suma(E2_1.a,\ E3.a)$

E2 ➔ E3 **-** E3

$E2.a = resta(E3_0.a,\ E3_1.a)$

E2 ➔ E3

$E2.a = E3.a$

E3 ➔ E4 **and** E3

$E3.a = and(E4.a, E3.a)$

E3 ➔ E4 **or** E4

$E3.a = or(E4_0.a, E4_1.a)$

E3 ➔ E4

$E3.a = E4.a$

E4 ➔ E4 OP4 E5

$E4_0.a = mkop4(OP4.op,\ E4_1.a,\ E5.a)$

E4 ➔ E5

$E4.a = E5.a$

E5 ➔ OP5 E5

$E5.a = mkop5(OP5.op,\ E5.a)$

E5 ➔ E6

$E5.a = E6.a$

E6 ➔ E6 **[** E0 **]**

$E6_0.a = array(E6_1.a,\ E0.a)$

E6 ➔ E6 **. identificador**

$E6_0.a = expCampo(E6_1.a,\ identificador.lex)$

E6 ➔ E6 **^**

$E6_0.a = punt(E6_1.a)$

E6 ➔ E7

$E6.a = E7.a$

E7 ➔ **literalEntero**

$E7.a = litEnt(literalEntero.lex)$

E7 ➔ **literalReal**

$E7.a = litReal(litReal.lex)$

E7 ➔ **identificador**

$E7.a = iden(identificador.lex)$

E7 ➔ **true**

$E7.a = true()$

E7 ➔ **false**

$E7.a = false()$

E7 ➔ **literalCadena**

$E7.a = litCad(literalCadena.lex)$

E7 ➔ **null**
$E7.a = null()$
E7 ➔ **(** E0 **)**
$E7.a = E0.a$

OP1 ➔ **<**
$OP1.op = " < "$
OP1 ➔ **>**
$OP1.op = " > "$
OP1 ➔ **<=**
$OP1.op = " <= "$
OP1 ➔ **>=**
$OP1.op = " >= "$
OP1 ➔ **==**
$OP1.op = " == "$
OP1 ➔ **!=**
$OP1.op = "! = "$

OP4 ➔ *****
$OP4.op = " * "$
OP4 ➔ **/**
$OP4.op = "/"$
OP4 ➔ **%**
$OP4.op = "\%"$

OP5 ➔ **-**
$OP5.op = " - "$
OP5 ➔ **not**
$OP5.op =$ "not"

fun mkop1(op, a1, a2):
op = "**<**" ➔ return **menor**(a1,a2)
op = "**>**" ➔ return **mayor**(a1,a2)
op = "**<=**" ➔ return **menorIgual**(a1,a2)
op = "**>=**" ➔ return **mayorIgual**(a1,a2)
op = "**==**" ➔ return **igual**(a1,a2)
op = "**!=**" ➔ return **desigual**(a1,a2)

fun mkop5(op, a):
op = "**-**" ➔ return **neg**(a)
op = "**not**" ➔ return **not**(a)

fun mkop4(op, a1, a2):
op = "*****" ➔ return **mul**(a1,a2)
op = "*/*" ➔ return **div**(a1,a2)
op = "**%**" ➔ return **mod**(a1,a2)

1.4. Acondicionamiento

programa ➔ bloque

$programa.a = prog(bloque.a)$

bloque ➔ **{** declaraciones instrucciones **}**
$bloque.a = bloq(declaraciones.a, instrucciones.a)$

declaraciones ➔ lista-declaraciones **&&**
$declaraciones.a = siDecs(listaDeclaraciones.a)$
declaraciones ➔ ε
$declaraciones.a = noDecs()$

lista-declaraciones ➔ declaración r-lista-declaraciones
$rListaDecs.ah = unaDec(declaracion.a)$
$listaDeclaraciones.a = rListaDecs.a$
r-lista-declaraciones ➔ **;** declaración r-lista-declaraciones
$rListaDecs_1.ah = muchasDecs(rListaDecs_0.ah, declaracion.a)$
$rListaDecs_0.a = rListaDecs_1.a$
r-lista-declaraciones ➔ ε
$rListaDecs.a = rListaDecs.ah$

declaración ➔ **proc identificador** parámetros-formales bloque
$declaracion.a = decProc(identificador.lex, parametrosFormales.a, bloque.a)$
declaración ➔ **type** tipo **identificador**
$declaracion.a = decType(tipo.a, identificador.lex)$
declaración ➔ tipo **identificador**
$declaracion.a = decVar(tipo.a, identificador.lex)$

parámetros-formales ➔ **(** lista-parámetros-formales-e **)**
$parametrosFormales.a = listaParametrosFormalesE.a$

lista-parámetros-formales-e ➔ lista-parámetros-formales
$listaParametrosFormalesE.a = siParam(listaParametrosFormales.a)$
lista-parámetros-formales-e ➔ ε
$listaParametrosFormalesE.a = noParam()$

lista-parámetros-formales ➔ parámetro-formal r-lista-parámetros-formales
$rListaParamsForm.ah = unParam(parametroFormal.a)$
$listaParametrosFormales.a = rListaParamsForm.a$
r-lista-parámetros-formales ➔ **,** parámetro-formal r-lista-parámetros-formales
$rListaParamsForm_1.ah = muchosParam(rListaParamsForm_0.ah, parametroFormal.a)$
$rListaParamsForm_0.a = rListaParamsForm_1.a$
r-lista-parámetros-formales ➔ ε
$rListaParamsForm.a = rListaParamsForm.ah$

parámetro-formal ➔ tipo r-parámetro-formal
$rParamForm.ah = tipo.a$
$parametroFormal.a = rParamForm.a$
r-parámetro-formal ➔ **& identificador**
$rParamForm.a = paramFormRef(rParamForm.ah, identificador.lex)$
r-parámetro-formal ➔ **identificador**

$rParamForm.a = paramForm(rParamForm.ah, identificador.lex)$

tipo ➔ T0

$tipo.a = T0.a$

T0 ➔ T1 RT0

$RT0.ah = T1.a$

$T0.a = RT0.a$

RT0 ➔ **[literalEntero]** RT0

$RT0_1.ah = tArray(RT0_0.ah, literalEntero.lex)$

$RT0_0.a = RT0_1.a$

RT0 ➔ ε

$RT0.a = RT0.ah$

T1 ➔ **^** T1

$T1_0.a = tPunt(T1_1.a)$

T1 ➔ T2

$T1.a = T2.a$

T2 ➔ **int**

$T2.a = tInt()$

T2 ➔ **real**

$T2.a = tReal()$

T2 ➔ **bool**

$T2.a = tBool()$

T2 ➔ **string**

$T2.a = tString()$

T2 ➔ **identificador**

$T2.a = tIden(identificador.lex)$

T2 ➔ **struct {** lista-campos **}**

$T2.a = tStruct(listaCampos.a)$

lista-campos ➔ campo r-lista-campos

$rListaCampos.ah = unCamp(campo.a)$

$listaCampos.a = rListaCampos.a$

r-lista-campos ➔ **,** campo r-lista-campos

$rListaCampos_1.ah = muchosCamps(rListaCampos_0.ah, campo.a)$

$rListaCampos_0.a = rListaCampos_1.a$

r-lista-campos ➔ ε

$rListaCampos.a = rListaCampos.ah$

campo ➔ tipo **identificador**

$campo.a = campo(tipo.a, identificador.lex)$

instrucciones ➔ lista-instrucciones

$intrucciones.a = siInstrs(listaInstrucciones.a)$

instrucciones ➔ ε

$intrucciones.a = noInstrs()$

lista-instrucciones ➔ instrucción r-lista-instrucciones

$rListaInstrs.ah = unaInstr(instrucción.a)$

$listaInstrucciones.a = rListaInstrs.a$

r-lista-instrucciones ➔ **;** instrucción r-lista-instrucciones

$rListaInstrs_1.ah = muchasInstrs(rListaInstrs_0.ah, instruccion.a)$

$rListaInstrs_0.a = rListaInstrs_1.a$

r-lista-instrucciones ➔ ε

$rListaInstrs.a = rListaInstrs.ah$

instrucción ➔ **@** expresión

$intruccion.a = arrobaInstr\ (expresion.a)$

instrucción ➔ **call** identificador parámetros-reales

$intruccion.a = procInstr\ (identificador.lex, parametrosReales.a)$

instrucción ➔ **nl**

$intruccion.a = nlInstr\ ()$

instrucción ➔ **new** expresión

$intruccion.a = newInstr\ (expresion.a)$

instrucción ➔ **read** expresión

$intruccion.a = readInstr\ (expresion.a)$

instrucción ➔ **write** expresión

$intruccion.a = writeInstr\ (expresion.a)$

instrucción ➔ **delete** expresión

$intruccion.a = deleteInstr\ (expresion.a)$

instrucción ➔ **while** expresión bloque

$intruccion.a = whileInstr\ (expresion.a, bloque.a)$

instrucción ➔ **if** expresión bloque r-instrucción-if

$rIntruccionIf.exp = expresion.a$

$rIntruccionIf.blq = bloque.a$

$instruccion.a = rIntruccionIf.a$

r-instrucción-if ➔ **else** bloque

$rIntruccionIf.a = ifElseInstr(rIntruccionIf.exp, rIntruccionIf.blq, bloque.a)$

r-instrucción-if ➔ ε

$rIntruccionIf.a = ifInstr(rIntruccionIf.exp, rIntruccionIf.blq)$

instrucción ➔ bloque

$intruccion.a = bloqueInstr(bloque.a)$

parámetros-reales ➔ **(** lista-expresiones-e **)**

$parametrosReales.a = listaExpresionesE.a$

lista-expresiones-e ➔ lista-expresiones

$listaExpresionesE.a = siExp(listaExpresiones.a)$

lista-expresiones-e ➔ ε

$listaExpresionesE.a = noExp()$

lista-expresiones ➔ expresión r-lista-expresiones

$rListaExp.ah = unaExp(expresion.a)$

$listaExps.a = rListaExp.a$

r-lista-expresiones ➔ **,** expresión r-lista-expresiones

$rListaExp_1.ah = muchasExp(rListaExp_0.ah, expresion.a)$

$rListaExp_0.a = rListaExp_1.a$

r-lista-expresiones → ε

$rListaExp.a = rListaExp.ah$

expresión ➔ E0

$expresion.a = E0.a$

E0 ➔ E1 RE0

$RE0.ah = E1.a$

$E0.a = RE0.a$

RE0 ➔ **=** E0

$RE0.a = asignación(RE0.ah, E0.a)$

RE0 ➔ ε

$RE0.a = RE0.ah$

E1 ➔ E2 RE1

$RE1.ah = E2.a$

$E1.a = RE1.a$

RE1 ➔ OP1 E2 RE1

$RE1_1.ah = mkop1(OP1.op, RE1_0.ah, E2.a)$

$RE1_0.a = RE1_1.a$

RE1 ➔ ε

$RE1.a = RE1.ah$

E2 ➔ E3 RE2 REC2

$RE2.ah = E3.a$

$REC2.ah = RE2.a$

$E2.a = REC2.a$

RE2 ➔ **-** E3

$RE2.a = resta(RE2.ah, E3.a)$

RE2 ➔ ε

$RE2.a = RE2.ah$

REC2 ➔ **+** E3 REC2

$REC2_1.ah = suma(REC2_0.ah, E3.a)$

$REC2_0.a = REC2_1.a$

REC2 ➔ ε

$REC2.a = REC2.ah$

E3 ➔ E4 RE3

$RE3.ah = E4.a$

$E3.a = RE3.a$

RE3 ➔ **and** E3

$RE3.a = and(RE3.ah, E3.a)$

RE3 ➔ **or** E4

$RE3.a = or(RE3.ah, E4.a)$

RE3 ➔ ε

$RE3.a = RE3.ah$

E4 ➔ E5 RE4

$RE4.ah = E5.a$

$E4.a = RE4.a$

RE4 ➔ OP4 E5 RE4

$RE4_1.ah = mkop4(OP4.op, RE4_0.ah, E5.a)$

$RE4_0.a = RE4_1.a$

RE4 ➔ ε

$RE4.a = RE4.ah$

E5 ➔ OP5 E5

$E5.a = mkop5(OP5.op, E5.a)$

E5 ➔ E6

$E5.a = E6.a$

E6 ➔ E7 RE6

$RE6.ah = E7.a$

$E6.a = RE6.a$

RE6 ➔ **[** E0 **]** RE6

$RE6_1.ah = array(RE6_0.ah, E0.a)$

$RE6_0.a = RE6_1.a$

RE6 ➔ **. identificador** RE6

$RE6_1.ah = expCampo(RE6_0.ah, identificador.lex)$

$RE6_0.a = RE6_1.a$

RE6 ➔ **^** RE6

$RE6_1.ah = punt(RE6_0.ah)$

$RE6_0.a = RE6_1.a$

RE6 ➔ ε

$RE6.a = RE6.ah$

E7 ➔ **literalEntero**

$E7.a = litEnt(literalEntero.lex)$

E7 ➔ **literalReal**

$E7.a = litReal(litReal.lex)$

E7 ➔ **identificador**

$E7.a = iden(identificador.lex)$

E7 ➔ **true**

$E7.a = true()$

E7 ➔ **false**

$E7.a = false()$

E7 ➔ **literalCadena**

$E7.a = litCad(literalCadena.lex)$

E7 ➔ **null**

$E7.a = null()$

E7 ➔ **(** E0 **)**

$E7.a = E0.a$

OP1 → **<**
$OP1.op = " < "$
OP1 → **>**
$OP1.op = " > "$
OP1 → **<=**
$OP1.op = " <= "$
OP1 → **>=**
$OP1.op = " >= "$
OP1 → **==**
$OP1.op = " == "$
OP1 → **!=**
$OP1.op = "! = "$

OP4 → ***
$OP4.op = " * "$
OP4 → **/**
$OP4.op = "/"$
OP4 → **%**
$OP4.op = "\%"$

OP5 → **-**
$OP5.op = " - "$
OP5 → **not**
$OP5.op =$ "not"

fun mkop5(op, a):
op = "**-**" → return **neg**(a)
op = "**not**" → return **not**(a)

fun mkop1(op, a1, a2):
op = "**<**" → return **menor**(a1,a2)
op = "**>**" → return **mayor**(a1,a2)
op = "**<=**" → return **menorIgual**(a1,a2)
op = "**>=**" → return **mayorIgual**(a1,a2)
op = "**==**" → return **igual**(a1,a2)
op = "**!=**" → return **desigual**(a1,a2)
fun mkop4(op, a1, a2):
op = "***" → return **mul**(a1,a2)
op = "*/*" → return **div**(a1,a2)
op = "**%**" → return **mod**(a1,a2)

1.5. Procesamiento

imprime(**prog**(Bloq)):
imprime(Bloq)

imprime(**bloq**(Decs, Instrs)):
print "{"
nl
imprime(Decs)

```
        imprime(Instrs)
        nl
        print "}"

imprime(siDecs(LDecs)):
        imprime(LDecs)
        nl
        print "&&"
        nl

imprime(noDecs()):
        noop

imprime(muchasDecs(LDecs,Dec)):
        imprime(LDecs)
        print ";"
        nl
        imprime(Dec)

imprime(unaDec(Dec)):
        imprime(Dec)

imprime(decProc(id, ParamForms, Bloq)):
        print "<proc> "
        print id
        imprime (ParamForms)
        print " "
        imprime(Bloq)

imprime(decType(Tipo, id)):
        print "<type> "
        imprime (Tipo)
        print " "
        print id

imprime(decVar(Tipo, id)):
        imprime (Tipo)
        print " "
        print id

imprime(siParam(LParams)):
        print "("
        imprime(LParams)
        print ")"

imprime(noParam()):
        print "()"
```

```
imprime(muchosParams(LParams, ParamForm)):
        imprime(LParams)
        print ", "
        imprime(ParamForm)

imprime(unParam(ParamForm)):
        imprime(ParamForm)

imprime(paramFormRef(Tipo, id)):
        imprime(Tipo)
        print " & "
        print id

imprime(paramFormal(Tipo, id)):
        imprime(Tipo)
        print " "++id

imprime(tArray(Tipo, id)):
        imprime (Tipo)
        print "["
        print id
        print "]"

imprime(tPunt(Tipo)):
        print "^"
        imprime (Tipo)

imprime(tInt()):
        print "<int>"

imprime(tReal()):
        print "<real>"

imprime(tBool()):
        print "<bool>"

imprime(tString()):
        print "<string>"

imprime(tIden(id)):
        print id

imprime(tStruct(LCampos)):
        print "<struct> {"
        nl
        imprime(LCampos)
        nl
        print "}"
```

```
imprime(muchosCamps(LCampos, Campo)):
        imprime(LCampos)
        print “, ”
        nl
        imprime(Campo)

imprime(unCamp(Campo)):
        imprime(Campo)

imprime(campo(Tipo, id)):
        imprime(Tipo)
        print “ “++id

imprime(siInstrs(LInstrs)):
        imprime(LInstrs)

imprime(noInstrs(LInstrs)):
        noop

imprime(muchasInstrs(LInstrs,Instr)):
        imprime(LInstrs)
        print ";"
        nl
        imprime(Instr)

imprime(unaInstr(Instr)):
        imprime(Instr)

imprime(arrobaInstr(Exp)):
        print "@ "
        imprime(Exp)

imprime(procInstr(id, ParamReales)):
        print "<call> "
        print id
        imprime(ParamReales)

imprime(nlInstr()):
        print "<nl>"

imprime(newInstr(Exp)):
        print "<new> "
        imprime (Exp)

imprime(readInstr(Exp)):
        print "<read> "
        imprime (Exp)
```

```
imprime(writeInstr(Exp)):
        print "<write> "
        imprime (Exp)

imprime(deleteInstr(Exp)):
        print "<delete> "
        imprime (Exp)

imprime(whileInstr(Exp, Bloq)):
        print "<while> "
        imprime(Exp)
        print " "
        imprime(Bloq)

imprime(ifElseInstr(Exp, Bloq, Bloq)):
        print "<if> "
        imprime (Exp)
        print " "
        imprime (Bloq)
        nl
        print "<else> "
        imprime (Bloq)

imprime(ifInstr(Exp, Bloq)):
        print "<if> "
        imprime(Exp)
        print " "
        imprime(Bloq)

imprime(bloqueInstr(Bloq)):
        imprime(Bloq)

imprime(siExp(LExps)):
        print “(”
        imprime(LExps)
        print “)”

imprime(noExp()):
        print “()”

imprime(muchasExp(LExps, Exp)):
        imprime(LExps)
        print “, ”
        imprime(Exp)

imprime(unaExp(Exp)):
        imprime(Exp)
```

```
imprime(litEnt(ent)):
        print ent

imprime(litReal(real)):
        print real

imprime(iden(iden)):
        print iden

imprime(true()):
        print “<true>”

imprime(false()):
        print “<false>”

imprime(litCad(cad)):
        print cad

imprime(null()):
        print “<null>”

imprime(asignacion(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"=",Opnd1,1,0)

imprime(menor(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"<",Opnd1,1,2)

imprime(mayor(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,">",Opnd1,1,2)

imprime(menorIgual(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"<=",Opnd1,1,2)

imprime(mayorIgual(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,">=",Opnd1,1,2)

imprime(igual(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"==",Opnd1,1,2)

imprime(desigual(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"!=",Opnd1,1,2)

imprime(suma(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"+",Opnd1,2,3)

imprime(resta(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"-",Opnd1,3,3)
```

```
imprime(and(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"and",Opnd1,4,3)

imprime(or(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,"or",Opnd1,4,4)

imprime(mul(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0," * ",Opnd1,4,5)

imprime(div(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,“/",Opnd1,4,5)

imprime(mod(Opnd0,Opnd1)):
        imprimeExpBin(Opnd0,“%",Opnd1,4,5)

imprime(neg(Opnd)):
        print “ -”
        imprimeOpnd(Opnd, 5)

imprime(not(Opnd)):
        print “ not ”
        imprimeOpnd(Opnd, 5)

imprime(array(Opnd1, Opnd2)):
        imprimeOpnd(Opnd1, 6)
        print “[”
        imprimeOpnd(Opnd2, 0)
        print “]”

imprime(expCampo(Opnd, id)):
        imprimeOpnd(Opnd, 6)
        print “.”
        print id

imprime(punt(Opnd)):
        imprimeOpnd(Opnd, 6)
        print “^”

imprimeExpBin(Opnd0,Op,Opnd1,np0,np1):
        imprimeOpnd(Opnd0,np0)
        print " "++Op++" "
        imprimeOpnd(Opnd1,np1)

imprimeOpnd(Opnd,MinPrior):
        if prioridad(Opnd) < MinPrior
                print “(“
        end if
```

```
        imprime(Opnd)
        if prioridad(Opnd) < MinPrior
                print ")"
        end if

prioridad(asignacion(_,_)): return 0
prioridad(menor(_,_)): return 1
prioridad(mayor(_,_)): return 1
prioridad(menorIgual(_,_)): return 1
prioridad(mayorIgual(_,_)): return 1
prioridad(igual(_,_)): return 1
prioridad(desigual(_,_)): return 1
prioridad(suma(_,_)): return 2
prioridad(resta(_,_)): return 2
prioridad(and(_,_)): return 3
prioridad(or(_,_)): return 3
prioridad(mul(_,_)): return 4
prioridad(div(_,_)): return 4
prioridad(mod(_,_)): return 4
prioridad(neg(_)): return 5
prioridad(not(_)): return 5
prioridad(array(_,_)): return 6
prioridad(expCampo(_,_)): return 6
prioridad(punt(_)): return 6
```