Restricciones contextules de nuestra práctica.

Las restricciones contextuales que introducimos de forma informal son las siguientes:

- Un identificador debe haber sido declarado antes de usarse.

- Un mismo identificador no puede ser declarado más de una vez.

- Al declarar un identificador, se le debe asociar un tipo reconocible por el programa: bool, int, real.

- Comprobación de tipos con las siguientes restricciones:
  (Reglas de la forma: Tipo1 x Tipo2: Tipo3
  Con Tipo1 y Tipo2 los tipos de los valores de entrada de una función, y Tipo3 el tipo del valor de salida).

  bool x bool : bool
  int x int : int
  int x real o real x int : real
  real x real : real

- Se debería comprobar que un identificador esté inicializado, pero en una primera versión hemos decidido inicializar todos los identificadores a cero al declararlos, esta inicialización es tanto para valores reales y enteros como para boléanos, siendo el valor cero equivalente a falso.

Los atributos que se definen para la gramática de atributos son:

Producción	Atributos Sintetizados	Atributos Heredados
Prog	**ts,err**	
Ident		
Iden		
Bloque	**ts**	
Tvar	**ts,err**	
Tvar2	**ts,dir,err**	**dirh**
Tipo	**tipo**	
TBloque	**err**	tsh
TBloque2	**err**	tsh
TRead	**err**	tsh
TWrite	**err**	tsh
Text	**err**	tsh

TAsig	**err**	tsh
Exp	**err,tipo**	tsh
ExpSimple	**err,tipo**	tsh
Term	**err,tipo**	tsh
Fact	**err,tipo**	tsh
OpMul	**tipo**	
OpAd	**tipo**	
OpUn	**tipo**	
Comp	**Id**	

Prog ::= program Ident PYCOMA Bloque PUNTO
      ***Prog*.err = *Bloque*.err**

Ident ::= id PA Iden PC

Iden ::= id

Iden::= Iden COMA id

Bloque ::= TBloque
      TBloque.tsh = creaTS()
      ***Bloque*.err = *TBloque*.err**

Bloque ::= Tvar TBloque
      TBloque.tsh = *Tvar*.ts
      ***Bloque*.err = *TBloque*.err v *Tvar*.err**

Tvar ::= var Tvar2
      Tvar.ts = *Tvar2*.ts
      ***Tvar*.err = *Tvar2*.err**

Tvar2 ::= id 2PUNTOS Tipo PYCOMA
      Tvar2.tsh = creaTS()
      Tvar2.dirh = DIR_BASE
      Tvar2.ts = añadeID (*Tvar2*.tsh, *id*.lex, *Tipo*.tipo, *Tvar2*.dirh)
      *Tvar2.*dir = *Tvar2.*dirh + *Tipo.*tam
      ***Tvar2*.err = false**

Tvar2 ::= id 2PUNTOS Tipo PYCOMA Tvar2
      $Tvar2_0$.ts = añadeID ($Tvar2_1$.ts, *id*.lex, *Tipo*.tipo, $Tvar2_1$.dir)
      $Tvar2_0$.dir = $Tvar2_1$.dir + *Tipo.*tam
      $Tvar2_0$.err = $Tvar2_1$.err v existeID($Tvar2_1$.ts, *id*.lex)

Tipo ::= integer
Tipo.tipo = integer
Tipo.tam = TAM_INT

Tipo ::= boolean
Tipo.tipo = boolean
Tipo.tam = TAM_BOOL

TBloque ::= begin TBloque2 end
TBloque2.tsh = *TBloque*.tsh
TBloque.err = _Tbloque2_.err

TBloque2 ::= λ

TBloque2 ::= TAsig TBloque2
$TBloque2_1$.tsh = $TBloque2_0$.tsh
TAsig.tsh = $TBloque2_0$.tsh
$TBloque2_0$.err = $TBloque2_1$.err v _TAsig_.err

TBloque2 ::= TRead TBloque2
$TBloque2_1$.tsh = $TBloque2_0$.tsh
TRead.tsh = $TBloque2_0$.tsh
$TBloque2_0$.err = $TBloque2_1$.err v _TRead_.err

TBloque2 ::= TWrite TBloque2
$TBloque2_1$.tsh = $TBloque2_0$.tsh
TWrite.tsh = $TBloque2_0$.tsh
$TBloque2_0$.err = $TBloque2_1$.err v _TWrite_.err

TRead ::= read TA id TC PYCOMA
TRead.err = false

TWrite ::= write TA Text TC PYCOMA
Text.tsh = *TWrite*.tsh
TWrite.err = _Text_.err

Text ::= texto
Text.err = false

Text ::= id
Text.err = false

TAsig ::= id ASIG Exp
*Exp.*tsh = *TAsig.*tsh
TAsig.err = ¬ existeID(_TAsig_.tsh, id.lex) v _Exp_.err v
(dameTipo(_TAsig_.tsh, id.lex) != _Exp_.tipo)

Exp ::= ExpSimple
ExpSimple.tsh = *Exp*.tsh
Exp.err = _ExpSimple_.err
Exp.tipo = _ExpSimple_.tipo

Exp ::= ExpSimple Comp ExpSimple

```
            ExpSimple0.tsh = Exp.tsh
            ExpSimple1.tsh = Exp.tsh
            Exp.err = ExpSimple0.err v ExpSimple1.err v
            ( if (Comp.id == '=') v (Comp.id == '!=') then
            ExpSimple0.tipo != ExpSimple1.tipo
                  else
                  (ExpSimple0.tipo != integer) v (ExpSimple1.tipo !=
      integer) )
            Exp.tipo = boolean

      ExpSimple ::= ExpSimple OpAd Term
            ExpSimple1.tsh = ExpSimple0.tsh
            Term.tsh = ExpSimple.tsh
            ExpSimple0.err = ExpSimple1.err v Term.err v
                        (ExpSimple1.tipo != OpAd.tipo) v
                        (Term.tipo != OpAd.tipo)
            ExpSimple0.tipo = OpAd.tipo


      ExpSimple ::= Term
            Term.tsh = ExpSimple.tsh
            ExpSimple.err = Term.err
            ExpSimple.tipo = Term.tipo

      Term ::= Term OpMul Fact
            Term1.tsh = Term0.tsh
            Fact.tsh = Term0.tsh
            Term0.err = Term1.err v Fact.err  v
                     (OpMul.tipo != Fact.tipo) v
                              (OpMul.tipo != Term1.tipo)
            Term0.tipo = OpMul.tipo

      Term ::= Fact
                  Fact.tsh = Term.tsh
            Term.err = Fact.err
            Term.tipo = Fact.tipo

      Fact ::= numero
            Fact.err = false
            Fact.tipo = integer

      Fact ::= true | false
            Fact.err = false
      Fact.tipo = boolean

      Fact ::= id
            Fact.err = ¬ existeID(Fact.tsh, id.lex)
            Fact.tipo = DameTipo(Fact.tsh, id.lex)

      Fact ::= OpUn Fact
            Fact1.tsh = Fact0.tsh
            Fact0.err = Fact1.err v Fact1.tipo != OpUn.tipo
            Fact0.tipo = OpUn.tipo
```

```
Fact ::= (Exp)
      Exp.tsh = Fact.tsh
      Fact.err = Exp.err
      Fact.tipo = Exp.tipo

OpAd ::= +
      OpAd.tipo = integer

OpAd ::= -
      OpAd.tipo = integer

OpAd ::= or
      OpAd.tipo = boolean

OpMul ::= *
      OpMul.tipo = integer

OpMul ::= /
      OpMul.tipo = integer

OpMul ::= and
      OpMul.tipo = boolean

OpUn ::= +
      OpMul.tipo = integer

OpUn ::= -
      OpMul.tipo = integer

OpUn ::= not
      OpMul.tipo = boolean

Comp ::= <=
      Comp.id = <=.lex

Comp ::= >=
      Comp.id = >=.lex

Comp ::= <
      Comp.id = <.lex

Comp ::= >
      Comp.id = >.lex

Comp ::= =
      Comp.id = =.lex

Comp ::= ≠
      Comp.id = ≠.lex
```