Restricciones contextuales 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: boolean, integer.

- 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

- 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 booleanos, siendo el valor cero equivalente a falso.

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

Producción	Atributos Sintetizados	Atributos Heredados
Prog	**err**	
Ident		
Iden		
Bloque		
Tvar	**err**	
Tvar2	**err**	
Tipo	**tipo**	
TBloque	**err**	
TBloque2	**err**	
TRead	**err**	
TWrite	**err**	
Text	**err**	
TAsig	**err**	
Exp	**err,tipo**	
ExpSimple	**err,tipo**	
Term	**err,tipo**	

Fact	**err,tipo**	
OpMul	**tipo**	
OpAd	**tipo**	
OpUn	**tipo**	
Comp	**Id**	

La gramática queda de la siguiente manera:

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

Ident ::= id PA Iden PC

Iden ::= id

Iden::= Iden COMA id

Bloque ::= TBloque
Bloque.err = TBloque.err

Bloque ::= Tvar TBloque
Bloque.err = TBloque.err v Tvar.err

Tvar ::= var Tvar2
Tvar.err = Tvar2.err

Tvar2 ::= id 2PUNTOS Tipo PYCOMA
Tvar2.err = false

Tvar2 ::= id 2PUNTOS Tipo PYCOMA Tvar2
$Tvar2_0$.err = $Tvar2_1$.err v existeID($Tvar2_1$.ts, id.lex)

Tipo ::= integer
Tipo.tipo = integer

Tipo ::= boolean
Tipo.tipo = boolean

TBloque ::= begin TBloque2 end
TBloque.err = Tbloque2.err

TBloque2 ::= λ

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

```
TBloque2 ::= TRead TBloque2
      TBloque2₀.err = TBloque2₁.err v TRead.err

TBloque2 ::= TWrite TBloque2
      TBloque2₀.err = TBloque2₁.err v TWrite.err

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

TWrite ::= write TA Text TC PYCOMA
      TWrite.err = Text.err

Text ::=  texto
      Text.err = false

Text ::=  id
      Text.err = false

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

Exp ::= ExpSimple
      Exp.err = ExpSimple.err

Exp ::= ExpSimple Comp ExpSimple
      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) )

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

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

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

Term ::= Fact
      Term.err = Fact.err


Fact ::= numero
      Fact.err = false
```

```
Fact ::= true | false
      Fact.err = false

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

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

Fact ::= (Exp)
      Fact.err = Exp.err

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
```