

### 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
<i>Prog</i>	<b>err</b>	
<i>Ident</i>		
<i>Iden</i>		
<i>Bloque</i>		
<i>Tvar</i>	<b>err</b>	
<i>Tvar2</i>	<b>err</b>	
<i>Tipo</i>		
<i>TBloque</i>	<b>err</b>	
<i>TBloque2</i>	<b>err</b>	
<i>TRead</i>	<b>err</b>	
<i>TWrite</i>	<b>err</b>	
<i>Text</i>	<b>err</b>	
<i>TAsig</i>	<b>err</b>	
<i>Exp</i>	<b>err, tipo</b>	
<i>ExpSimple</i>	<b>err, tipo</b>	
<i>Term</i>	<b>err, tipo</b>	

<i>Fact</i>	<b>err, tipo</b>	
<i>OpMul</i>	<b>tipo</b>	
<i>OpAd</i>	<b>tipo</b>	
<i>OpUn</i>	<b>tipo</b>	
<i>Comp</i>	<b>Id</b>	

La gramática queda de la siguiente manera:

```

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

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
        Tvar20.err = Tvar21.err v existeID(Tvar21.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
          TBloque20.err = TBloque21.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 =  $\neg$  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 =  $\neg$  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
```