Definición de la tabla de Símbolos

Los campos que componen la tabla de símbolos serían los siguientes (para la definición que llevamos hecha hasta el momento, ya que con toda seguridad en la práctica del segundo cuatrimestre tendremos que añadir alguno más):

- **Lexema**: contiene el identificador. Es equivalente a las entradas de la tabla, que son los identificadores y palabras clave de nuestro lenguaje.

- **Tipo**: especifica si se trata de un número, un identificador o una palabra reservada del lenguaje…

- **Dirección**: Dirección de la variable en memoria

Para el manejo de la tabla, utilizaremos las siguientes **operaciones**:

- **CreaTS(): TS** : Crea una tabla de símbolos inicializada con las palabras reservadas.

- **AñadeID(TS, id, tipo, dirección): TS** : Añade a la tabla de símbolos el nuevo identificador id con su tipo y devuelve la nueva tabla modificada.

- **ExisteID(TS, id): bool** : Indica si el identificador id está incluido en la tabla de símbolos.

- **DameTipo (ts, id)**: **bool** (¿??) Devuelve un valor (TRUE si es bolean o FALSE si es integer), que es el Tipo del identificador "id" pasado como parámetro.

Una vez explicadas las operaciones soportadas por la tabla de símbolos, hemos de introducir la gramática de atributos que define nuestra Tabla. La gramática de atributos formalizamos la relación entre el sub-lenguaje empleado para la declaración de los símbolos del lenguaje y la información de los mismos que es necesaria almacenar.

Una primera versión, a falta de depurar quedaría de la siguiente manera:

```
Prog ::= program  Ident PYCOMA Bloque PUNTO

Ident ::= id PA Iden PC

Iden ::= id

Iden::= Iden COMA id

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

Bloque ::=  Tvar TBloque
      TBloque.tsh = Tvar.ts
```

```
Tvar ::= var Tvar2
      Tvar.ts = Tvar2.ts

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 ::=  id 2PUNTOS Tipo PYCOMA Tvar2
      Tvar2₀.ts = añadeID (Tvar2₁.ts, id.lex, Tipo.tipo,
Tvar2₁.dir)
      Tvar2₀.dir = Tvar2₁.dir + Tipo.tam


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

TBloque2 ::= λ

TBloque2 ::= TAsig TBloque2
TBloque2₁.tsh = TBloque2₀.tsh
TAsig.tsh = TBloque2₀.tsh

TBloque2 ::= TRead TBloque2
TBloque2₁.tsh = TBloque2₀.tsh
TRead.tsh = TBloque2₀.tsh

TBloque2 ::= TWrite TBloque2
TBloque2₁.tsh = TBloque2₀.tsh
TWrite.tsh = TBloque2₀.tsh

TRead ::= read TA id TC PYCOMA

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

Text ::=  texto

Text ::= id

TAsig ::= id ASIG  Exp
      Exp.tsh = TAsig.tsh

Exp ::= ExpSimple
ExpSimple.tsh = Exp.tsh
```

Exp ::= ExpSimple Comp ExpSimple
***ExpSimple$_0$*.tsh = *Exp*.tsh**
***ExpSimple$_1$*.tsh = *Exp*.tsh**

ExpSimple ::= ExpSimple OpAd Term
***ExpSimple$_1$*.tsh = *ExpSimple$_0$*.tsh**
***Term*.tsh = *ExpSimple*.tsh**

ExpSimple ::= Term
***Term*.tsh = *ExpSimple*.tsh**

Term ::= Term OpMul Fact
***Term$_1$*.tsh = *Term$_0$*.tsh**
***Fact*.tsh = *Term$_0$*.tsh**

Term ::= Fact
***Fact*.tsh = *Term*.tsh**

Fact ::= numero | true | false | id

Fact ::= *OpUn Fact*
***Fact$_1$*.tsh = *Fact$_0$*.tsh**

Fact ::= (*Exp*)
***Exp*.tsh = *Fact*.tsh**

OpAd ::= + | - | or

OpMul ::= * | / | and

OpUn ::= +| - | not

Comp ::= <= | >= | < | > | = | ≠