# Lenguaje TYNY

(Lauden, 2004)

- Un programa en TYNY tiene una estructura muy simple: es una secuencia de sentencias separadas mediante signos de punto y coma en una sintaxis semejante a la de Ada o Pascal.
- La última sentencia en cualquier sección (e.g. al final del programa, al final de un THEN, al final de un repeat) NO lleva punto y coma (**;**)
- No hay funciones ni declaraciones.
- Todas las variables son enteras y se declaran al asignarle (**:=**) valores a las mismas (de modo parecido a FORTAN, BASIC o Python).
- Existen solamente dos secuencias de control:
    - Una sentencia **`if`**
    - Una sentencia **`repeat`**
- Ambas sentencias de control pueden contener secuencias de sentencias.
- La sentencia **if** tiene una parte opcional **`else`** y debe terminarse mediante la palabra reservada **end**.
- Existen sentencias **`read`**/**`write`** de lectura/escritura que realizan entrada/salida
- Los comentarios se colocan dentro de llaves **{ }**. NO se permiten comentarios anidados.
- Las expresiones se encuentran limitadas a aritméticas enteras y booleanas:
    - Una expresión booleana se compone de una comparación de dos expresiones aritméticas que utilizan cualesquiera de los dos operadores de comparación **<** y **=**.
    - Una expresión aritmética puede involucrar constantes enteras, variables, paréntesis y cualquiera de los cuatro operadores enteros: **+**, **-**, ***** y **/** (división entera), con las propiedades matemáticas habituales.
    - Las expresiones booleanas pueden aparecer solamente como pruebas en sentencias de control: no hay variables booleanas, asignación o E/S.

**NOTA:** Aunque TINY carece de muchas de las características necesarias para los lenguajes de programación reales (e.g. funciones, arreglos y valores de punto flotante, como algunas de las omisiones más serias), todavía es lo suficientemente extenso para ejemplificar la mayoría de las características esenciales de un compilador en el salón de clase.

## Ejemplo de programa en TINY

```
{ Programa de muestra
  en lenguaje TINY –
  calcula el factorial
}
read x; { introducir un entero }
if x > 0 then { no calcule si x <= 0 }
      fact := 1;
      repeat
            fact := fact * x;
            x := x – 1;
      until x = 0;
      write fact { salida del factorial de x }
end
```

**Gramática BNF para TINY**

*programa* → *secuencia-sent*

*secuencia-sent* → *secuencia-sent* **;** *sentencia* | *sentencia*

*sentencia* → *sent-if* | *sent-repeat* | *sent-assign* | *sent-read* | *sent-write*

*sent-if* → **if** *exp* **then** *secuencia-sent* **end**

| **if** *exp* **then** *secuencia-sent* **else** *secuencia-sent* **end**

*sent-repeat* → **repeat** *secuencia-sent* **until** *exp*

*sent-assign* → **identificador :=** *exp*

*sent-read* → **read identificador**

*sent-write* → **write** *exp*

*exp* → *exp-simple op-comparación exp-simple* | *exp-simple*

*op-comparación* → **<** | **=**

*exp-simple* → *exp-simple opsuma term* | *term*

*opsuma* → **+** | **-**

*term* → *term opmult factor* | *factor*

*opmult* → ***** | **/**

*factor* → ( *exp* ) | **número** | **identificador**

**Gramática EBNF para TINY**

*programa* → *secuencia-sent*

*secuencia-sent* → *sentencia* { **;** *sentencia* }

*sentencia* → *sent-if* | *sent-repeat* | *sent-assign* | *sent-read* | *sent-write*

*sent-if* → **if** *exp* **then** *secuencia-sent* [ **else** *secuencia-sent* ] **end**

*sent-repeat* → **repeat** *secuencia-sent* **until** *exp*

*sent-assign* → **identificador :=** *exp*

*sent-read* → **read identificador**

*sent-write* → **write** *exp*

*exp* → *exp-simple* [ *op-comparación exp-simple* ]

*op-comparación* → **<** | **=**

*exp-simple* → *term* { *opsuma term* }

*opsuma* → **+** | **-**

*term* → *factor* { *opmult factor* }

*opmult* → ***** | **/**

*factor* → ( *exp* ) | **número** | **identificador**