LÉXICO

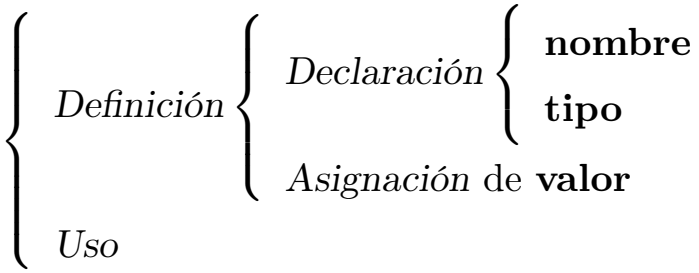
- palabras reservadas: `array` etc.
- combinaciones reservadas: `+`, `:=` etc.
- comentarios `(* cualquier cosa *)`
- identificadores:
  <identificador>::= <letra>{<letra>|<dígito>}
- *identificadores predefinidos:* `integer` *etc.*
- *números:* `123`, `1234.56E-2` *etc.*
- *cadenas literales:* `'cualquier cosa dentro de una línea'`
- *separadores: espacio, tabulador, comentario etc.*


```
and       array   begin   case    const      div     do    downto
else      end     file    for     function   goto    if    in
label     mod     nil     not     of         or      packed
procedure         program         record     repeat  set   then
to        type    until   var     while      with
+    -    *    /    :=    .    ,    ;    :    '    =    <>
<    <=   >=   >    (    )    [    ]    {    }    ^    ..
```

`integer, real, boolean, read, readln, sin, cos, TRUE ...`

<identificador>::= <letra>{<letra>|<dígito>}

<fracción opt.>::= `.`*<dígitos>* |
<signOpt>::= `+` | `-` |
<parteExpOpt>::= `e` *<signOpt><dígitos>*|
`E` *<signOpt><dígitos>*
<número sin signo>::= <dígitos><fracción opt.><parteExpOpt>




```
program PrimerEjemplo (output);
(* Área de un círculo de radio 2'5 *)
const PI = 3.141592;
var radio, area : real ;
BEGIN
    writeln ('Considerando el valor de PI como ', PI);
    radio := 2.5 ;
    area := PI * radio * radio;
    writeln ('área de círculo de radio', radio, '=', area)
END.
```

<*programa*>::=<*cabecera*><*bloque*>
<*cabecera*>::=`program` <*identificador*> `(` <*ids.archivos*> `);`
<*bloque*>::=<*decl rótulos*><*def de constantes*>
<*def de tipos*><*decl de variables*>
<*def de subprogramas*><*sent. compuesta*>.
<*sent. compuesta*>::= `begin` <*sentencia*>{ `;` <*sentencia*> } `end`

VARIABLES

- *Definición*
  - *Declaración*: `var`
  - *Asignación* de **valor**: asignación, lectura ...
- *Uso*: expresiones; significa su **valor**

<*decl de variables*>::= `var` <*lista de identificadores*>: <*tipo*>
{ ; <*lista de identificadores*>: <*tipo*>};
<*asignación*>::= <*identificador*>:= <*expresión*>
<*s. de lectura*>::= `readln` (<*lista de ids. de variable*>)

CONSTANTES

- *Definición*
  - *Declaración*
    - **nombre**: pueden no tenerlo
    - **tipo**: se deduce del valor
  - *Asignación* de **valor** constante
- *Uso*: en expresiones; significa su valor

<*def de constantes*>::= `const` <*identificador*>= <*valor*>
{`;` <*identificador*>= <*valor*>}

No admiten otro tipo de asignación.

EXPRESIONES

- *Definición*
  - *Declaración*
    - **nombre**: no tienen
    - **tipo** se deduce de operandos y operadores
  - *Asignación* de **valor**: se calcula
- *Uso*: significa su valor


```
program SegundoEjemplo (input, output);
(* Área de un círculo de radio dado por el usuario*)
const PI = 3.141592;
var radio, area : real ;
BEGIN
  writeln ('Escriba un valor para el radio de un círculo');
  readln (radio);
  writeln ('Considerando el valor de PI como ', PI);
  area := PI * radio * radio;
  writeln ('área de círculo de radio', radio, '=', area)
END.
```

TIPO DE DATOS

- un conjunto de valores (posibles)
- un conjunto de operaciones que se pueden realizar con ellos

<tipo>::=<tipo simple>| <tipo estructurado>| <tipo puntero>
<tipo simple>::=<tipo escalar>| <tipo subrango>|
<tipo identificador>

Tipos simples *identificador* predefinidos :
`integer`, `real`, `boolean`, `char`

LECTURA

Asignación de los *valores proporcionados por el usuario* en el momento de la *ejecución* **a** las variables que se indican, por la vía que se indica.

<sentencia de lectura>::=
`read` (*<fichero>*, *<lista de identificadores de variable>*) |
`readln` (*<fichero>*, *<lista de identificadores de variable>*) |
`readln` (*<fichero>*) |
`read` (*<lista de identificadores de variable>*) |
`readln` (*<lista de identificadores de variable>*) |
`readln`

ESCRITURA

Evaluación de los *valores de las expresiones* que se indican en el momento de la *ejecución* **escribiéndolos** por la vía que se indica.

<*sentencia de escritura*>::=
`write` (<*fichero*>, <*lista de expresiones*>) |
`writeln` (<*fichero*>, <*lista de expresiones*>) |
`writeln` (<*fichero*>) |
`write` (<*lista de expresiones*>) |
`writeln` (<*lista de expresiones*>) |
`writeln`

<*expresión para imprimir*>::= <*expresión*> |
<*expresión*>:<*m*> |
<*expresión*>:<*m*>:<*n*>

TIPOS DE DATOS

- un conjunto de valores posibles
- un conjunto de operaciones y funciones asociadas

implantados / no
predefinidos / definidos por el programador / semidefinidos
simples / estructurados
ordenados / no
ordinales / no
estáticos / dinámicos
permanentes / no
estándar / no

- *Simples (estáticos)*
  - *Predefinidos*
    - *entero (ordinal)* : **integer**
    - *real (ordenado)* : **real**
    - *carácter (ordinal)* : **char**
    - *lógico (ordinal)* : **boolean**
    - ...
  - *Semidef. prog.*
    - *enumerado (ordinal, def. programador)*
    - *subrango(ordinal)*
    - *puntero*
- *Estructurados*
  - *Estáticos (semidef. programador)*
    - *vectoriales* : **array**
    - *registro* : **record**
    - *conjunto* : **set**
    - *cadena (no estándar)* : **string**
    - *archivo* : **file**
  - *Dinámicos (def.programador)*
    - *pila*
    - *cola*
    - *lista doblemente enlazada*
    - *árbol*
    - *grafo*
    - ...

TIPO ENTERO: **integer**

valores: enteros de $[-\text{MAXINT}-1, \text{MAXINT}]$ (32767)

operaciones: (con las precedencias y asociatividades habituales)

cambio de signo:	- (unario)	suma:	+ (binario)
diferencia:	- (binario)	producto:	* (binario)
cociente:	**div** (binario)	resto:	**mod** (binario)

funciones:

valor absoluto	**abs**	cuadrado	**sqr**
predecesor	**pred**	sucesor	**succ**

Sólo se garantiza que el resultado de a op b es correcto si

$$\mathbf{abs}(a) \leq \mathbf{MAXINT}$$
$$\wedge\, \mathbf{abs}(b) \leq \mathbf{MAXINT}$$
$$\wedge\, \mathbf{abs}(a \text{ op } b) \leq \mathbf{MAXINT}$$

TIPO REAL: **real**

valores: algunos valores reales $\supset$ **integer**
(Mayor $\approx 1'7 \times 10^{38}$; Menor $\approx 2'9 \times 10^{-38}$; Cifras ≈ 11)

operaciones: (con las precedencias y asociatividades habituales)

cambio de signo:	- (unario)	suma:	+ (binario)
diferencia:	- (binario)	producto:	* (binario)
cociente:	/ (binario)		

funciones:

valor absoluto:	**abs**	cuadrado:	**sqr**
raíz cuadrada:	**sqrt**	exponencial:	**exp**
logaritmo:	**ln**	seno:	**sin**
coseno:	**cos**	arco tangente:	**arctan**
truncado a entero:	**trunc**	redondeo a entero:	**round**

TIPO CARÁCTER: **char**

valores: al menos

mayúsculas del alfabeto inglés, ordenadas

minúsculas del alfabeto inglés, ordenadas

dígitos del '0' al '9', ordenados

signos de puntuación y otros

funciones:

predecesor:	**pred**
sucesor:	**succ**
número de carácter:	**ord** (devuelve entero $\in [0, 255]$)
carácter de número:	**chr** (devuelve el carácter)

TIPO LÓGICO: **boolean**

valores: **FALSE** y **TRUE**

operaciones:
y: **and** (binario) ó: **or** (binario)
no: **not** (unario)

operadores relacionales: resultado lógico sobre ordenados:

menor que: < menor o igual: <= igual: =
mayor que: > mayor o igual: >= distinto: <>

operaciones derivadas de los relacionales:

implicación: <= equivalencia: = o exclusivo (*xor*): <>

funciones:
predecesor: **pred** sucesor: **succ**
impar: **odd**
fin de línea: **eoln** fin de fichero: **eof**

TIPO ENUMERADO

Tipo simple, ordinal, definido por el programador.

valores: los que el programador defina:
`(` <*identificador*> `{ ,` <*identificador*>`})`

operaciones: operadores relacionales (resultado **boolean**):

menor que: < menor o igual: <= igual: =
mayor que: > mayor o igual: >= distinto: <>

funciones:
predecesor: **pred** sucesor: **succ**
número de orden : **ord**

No están contempladas lectura de teclado ni escritura en pantalla.

Los identificadores deben ser únicos.

```
var mes1, mes2 : (ene, feb, mar, abr, may, jun,
                  jul, ago, sep, oct, nov, dic);
   estado      : (soltero, casado, viudo, separado,
                  emparejadodehecho, clerigo);
   color       : (rojo, amarillo, azul, naranja,
                  verde, violeta);
   diasem      : (lunes, martes, miercoles, jueves,
                  viernes, sabado, domingo);
   nota        : (ut, re, mi, fa, sol, la, si);
Begin
  if (dia=13) and (diasem = martes) then estado := casado;
  if   mes1 = dic   then mes2 := ene
  else mes2 := succ(mes1);
  if diasem > viernes then (*** fin de semana ***)
  ...
```


TIPO SUBRANGO

Tipo simple, ordinal, definido por el programador.

valores: subconjunto elementos consecutivos de un tipo ordinal:
<primer valor> `..` *<último valor>*
siendo *<primer valor>*$\leq$ *<último valor>*

operaciones y funciones: las que el tipo del que se extrae permita.

Se detectará la permanencia en el rango (quizá mediante una opción de compilación). (Opción `-Cr` ó `{$R+}`)

TIPOS CON NOMBRE

type <*identificador*>= <*tipo*> { ; <*identificador*>= <*tipo*>}

```
type entero = integer;
     tDiasDeUnMes = 1 ..31;
     tDiasDeSemana = (lunes, martes, miercoles, jueves,
                      viernes, sábado, domingo);
     tDiaLectivo = lunes .. viernes;
     tMinuscula = 'a'..'z';
     tDigito = 0 ..9;
     tCharDigito = '0' ..'9';
     tPositivo = 0 .. maxint;
var  dia1, dia2 : tDiasDeUnMes; (* entero *)
     dia        : tDiaLectivo;  (* enumerado *)
```


RESUMEN DE OPERADORES PASCAL

En orden decreciente de precedencia:

unarios:	- not
binarios producto:	* div mod / and
binarios aditivos:	+ - or
relacionales:	= < > <= >= <> in

Los asociativos, a la izquierda. Los paréntesis especifican un orden de evaluación preferente a todos ellos.

+, -, *, =, <>, <= y >= también se usarán en tipo conjunto (**set**).

in está asociado exclusivamente al tipo conjunto.

Los relacionales (salvo **in**) también tienen significado entre cadenas de caracteres.

= y <>también tienen significado entre tipos puntero.

función	*significado*	*argumento*	*resultado*
abs(x)	valor absoluto	entero o real	del mismo tipo
sqr(x)	cuadrado del argumento	entero o real	del mismo tipo
sin(x)	seno trigonométrico	real o entero	real
cos(x)	coseno trigonométrico	real o entero	real
exp(x)	e^x	real o entero	real
ln(x)	logaritmo neperiano	real o entero	real
sqrt(x)	raíz cuadrada	real o entero	real
arctan(x)	arco tangente	real o entero (en radianes)	real
odd(x)	imparidad	entero	lógico
eof(x)	fin de fichero	fichero	lógico
eoln(x)	fin de línea	texto	lógico
trunc(x)	parte entera	real	entero
round(x)	redondeo	real	entero
ord(x)	número de orden	carácter	entero
chr(x)	carácter cuyo número es x	entero	carácter
succ(x)	sucesor	ordinal	del mismo tipo
pred(x)	predecesor	ordinal	del mismo tipo