Nomenclátor de notación algorítmica

Leyenda:

Negrita	se utiliza para las palabras clave del lenguaje.
Cursiva	se tiene que sustituir por el identificador definido por el usuario (nombre de la variable, del tipo, etc.); no significa que los identificadores vayan en *cursiva* en los algoritmos.
Castellano	se utiliza para describir instrucciones.

Tipos básicos

```
integer
char
real
boolean
string
```

Declaración de constantes

```
const
   CONST_NAME: type = valor;
end const
```

Declaración de tuplas

```
type
   typeName = record
      fieldName: tipo del campo;
   end record
end type
```

Condicional

```
if expresión condicional then
   bloque de instrucciones
end if
```

```
if expresión condicional then
   bloque de instrucciones
else
   bloque de instrucciones
end if
```

Declaración de tipos

```
type
   typeName = declaración del tipo
end type
```

Declaración de variables

```
var
   varName: tipo de la variable;
end var
```

Declaración de vectores

```
var
   vectorName: vector[longitud] of tipo del campo;
end var
```

Iteración

```
while expresión condicional do
   bloque de instrucciones
end while
```

```
for varName := valor inicial to valor final [step valor de incremento/decremento] do
   bloque de instrucciones
end for
```

Clase de parámetros

```
in
out
inout
```

Declaración de algoritmo

```
algorithm
   bloque de instrucciones
end algorithm
```

Declaración de punteros

```
var
  varName: pointer to tipo del campo
end var
```

Operadores lógicos

```
and
or
not
```

Operador de asignación

```
:=
```

Operadores de comparación

```
=
≠
<
>
≤
≥
```

Constantes booleanas

```
true
false
```

Comentario

```
{ comentario }
```

Función

```
function functionName (parName1: tipo del parámetro, ... , parNameN: tipo del parámetro): tipo del retorno
   bloque de instrucciones
end function
```

Acción

```
action actionName (clase del parámetro parName1: tipo del parámetro, ... , clase del parámetro parNameN: tipo del parámetro)
   bloque de instrucciones
end action
```

Funciones de cambio de tipo

```
function integerToReal(varName: integer): real;
function realToInteger(varName: real): integer;
function charToCode(varName: char): integer;
function codeToChar(varName: integer): char;
```

Funciones y acciones de entrada/salida

```
function readInteger(): integer;
function readReal(): real;
function readChar(): char;
function readString(): string;
function readBoolean(): boolean;
action writeInteger(in varName: integer);
action writeReal(in varName: real);
action writeChar(in varName: char);
action writeString(in varName: string);
action writeBoolean(in varName: boolean);
```