EL LENGUAJE VHDL

Sección 2 : Elementos Básicos del Lenguaje

Contenido

- Elementos léxicos
- Objetos
- Tipos de datos
- Operadores y expresiones
- Atributos

Elementos léxicos

Palabras reservadas

abs	access	after	alias	all	and	architecture
array	assert	attribute	begin	block	body	buffer
bus	case	component	else	elsif	end	entity
exit	file	for	function	generate	generic	guarded
if	in	inout	is	label	library	nand
new	next	nor	not	null	of	on
open	or	others	out	.....		

Identificadores

```
COUNT
aBc
X
f123
VHDL
VH_DL
ABC
```

Enteros

```
2
3E4
1000_000
```

Reales

```
2.0
3.0E4
1.110_001
```

Comentario

```
bla bla -- esto es un comentario
esto no -- esto sí
```

Literales

Base	**Carácter**	**Físicos**
2#110_1010#	'a'	10 ns
16#CA#	'A'	2.2 V
16#f.ff#e+2	'@'	50 pF

Decimales	**String**	**Bit String**
12	"Tiempo"	X"F0f"
0	"A"	B"111_100"
1E6	"@#4%+"	o"7417"
3.141_593	"110101"	

Elementos léxicos
Palabras reservadas

- Elementos léxicos:
  - Palabras reservadas, identificadores, símbolos, literales
- Palabras reservadas:
  - Palabras que tienen un significado en VHDL. Se utilizan para construir sentencias.

abs	architecture	body	elsif
access	array	buffer	end
after	assert	bus	entity
alias	attribute	case	exit
all	begin	component	file
and	block	else	...

Elementos léxicos
Identificadores

- Identificadores:
  - Sirven para dar nombre a elementos VHDL
  - Reglas:
    - No tienen longitud máxima. Se recomienda utilizar un tamaño que permita formar identificadores significativos
    - Pueden contener caracteres de ‘a’ a ‘z’, de ‘A’ a ‘Z’, de ‘0’ a ‘9’ o ‘_’. VHDL no distingue mayúsculas de minúsculas
    - Deben empezar con un carácter alfabético, no pueden terminar en ‘_’ ni contener dos ‘_’ seguidos
    - No puede usarse una palabra reservada como identificador

Elementos léxicos
Identificadores

- Ejemplos de identificadores:

Correctos

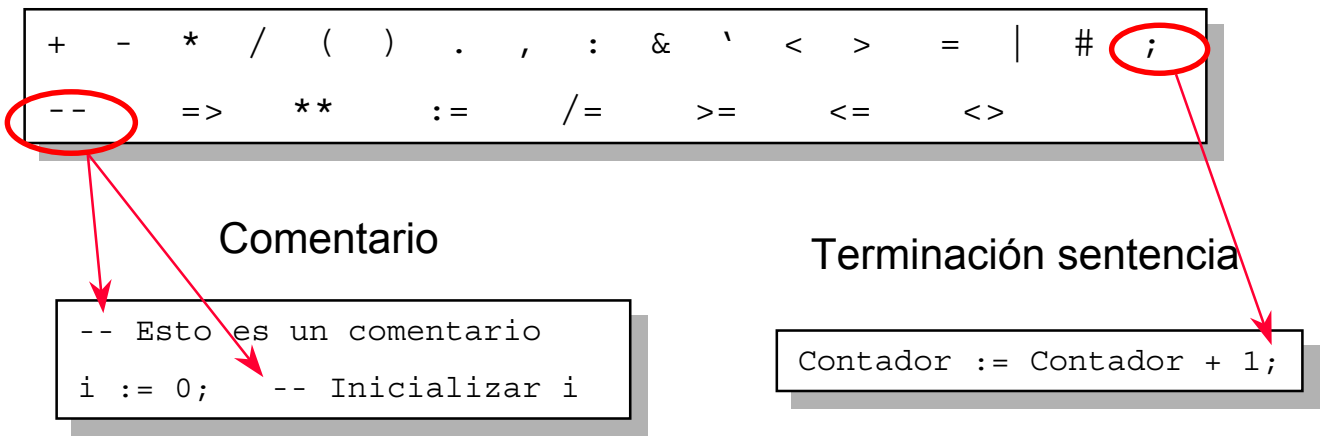
```
i
i_2
RelojADC
AcT_iVaPrOc_EsO
Interrupcion_3_del_micro
```

Incorrectos

```
3
i2_
_i2
Reloj$ADC
AcT__ivaPrOc_EsO
Interrupción_3_del_micro
```

Elementos léxicos
Símbolos

- Símbolos:
  - Diferentes funciones: operadores, parte de sentencias, símbolos de puntuación, ...
  - Formados por 1 ó 2 caracteres



Elementos léxicos
Literales

- Literal:
  - Símbolo cuyo valor se obtiene directamente de su representación. Existen literales numéricos, carácter, cadena de caracteres y cadena de bits
  - Literal numérico: entero, real o físico
  - Entero, real: Valores enteros y reales

```
3   3.0   23   23.03   -43   -43.344
```

  - Físico: Valor numérico más una unidad

```
5.34 mm    4 ms    5.25 Kg    200 Ohms
```

Elementos léxicos
Literales

- Opciones de representación:

```
3e2   3.0e2   3.23E-2   4e-1 mg
33_231_342.2   3.14_15_92 Km
2#110#e4   10#42.21#e-2   16#ab.c5
```

Notación exponencial

'_' para mejorar legibilidad

Base#Valor#Exponente

- Literal carácter:
  - Carácter encerrado entre comillas simples

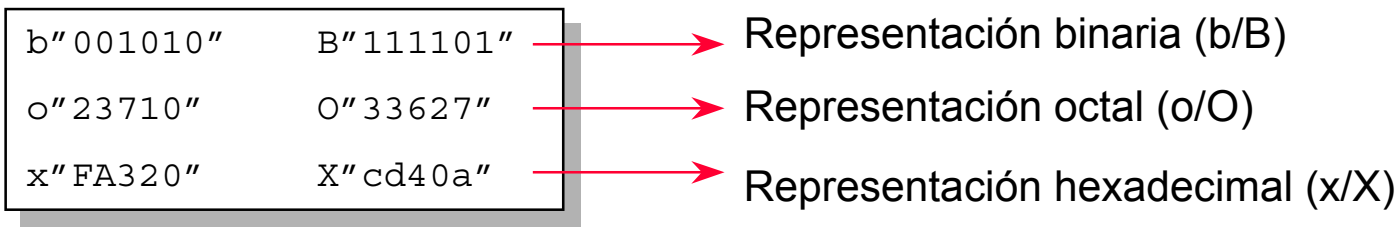
```
'a'     '$'     'P'     'ç'
```

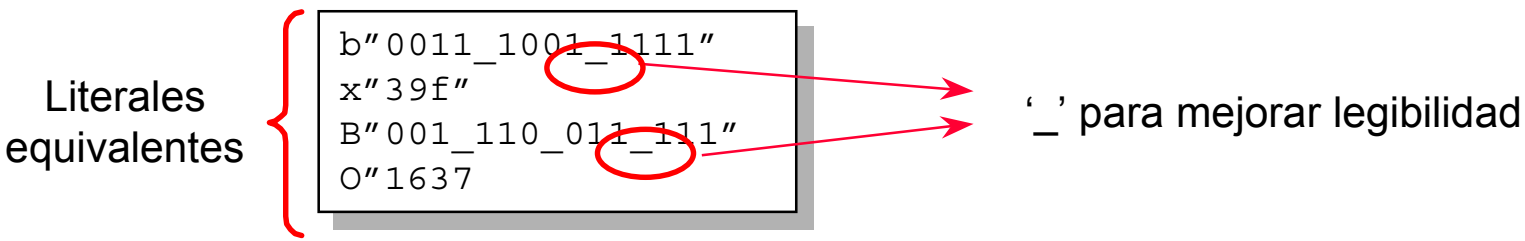
- Literal cadena de caracteres:
  - Conjunto de caracteres entre comillas dobles

```
"Esto es una cadena"     "&&& $$$ ///"     " "
```

Elementos léxicos
Literales

- Literal cadena de bits:
  - Conjunto de bits entre comillas dobles precedidos de un carácter que indica la base de representación.





Objetos VHDL

- Objeto VHDL
  - Elemento del lenguaje que tiene asignado un valor
- Tipos de objetos VHDL
  - Constante
  - Variable
  - Señal
  - Fichero
- Declaración de un objeto

```
<Clase objeto> <identificador> : <tipo datos> [:= Valor inicial];
```

Objetos VHDL
Constante

- Constante:
  - Objeto cuyo valor no puede modificarse
  - Normalmente su valor se fija en la declaración
  - Mejora legibilidad y actualización del código

```
constant PI : real := 3.1415927;
constant BitsPalabra : natural := 8;
constant NumPalabras : natural := 1024;
constant TotBits : natural := BitsPalabra * NumPalabras;
constant NumeroIteraciones : integer;
```

Declaración de constantes

Objetos VHDL
Variable

- Variable:
  - Objeto cuyo valor puede modificarse mediante sentencias de asignación a variable

    <Nombre> := <Expresión>;
  - Existen en el dominio secuencial (dentro de un proceso)

```
variable Contador : integer :=0;
variable Incremento : integer;
```

Declaración de variables

```
Incremento := 2;
Contador := Contador + Incremento;
```

Asignación a variables

Objetos VHDL
Señal

- Señal:
  - Objeto cuyo valor se puede modificar mediante sentencias de asignación a señal

    <Nombre> <= <Expresión>;
  - Características:
    - Abstracción de conexión *hardware* o bus
    - Se usan para la sincronización de procesos.
    - Tienen un valor actual y una lista de asignaciones pendientes

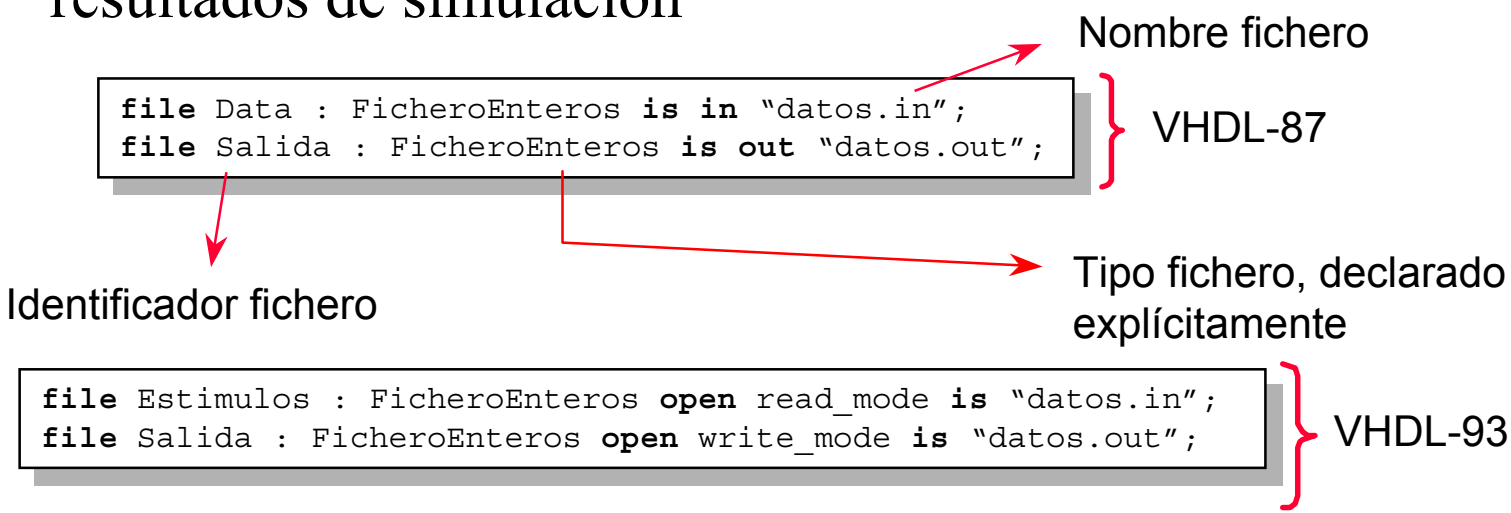
```
signal Reloj : bit := '0';
...
Reloj <= '1';
```

Declaración de señal

Asignación a señal

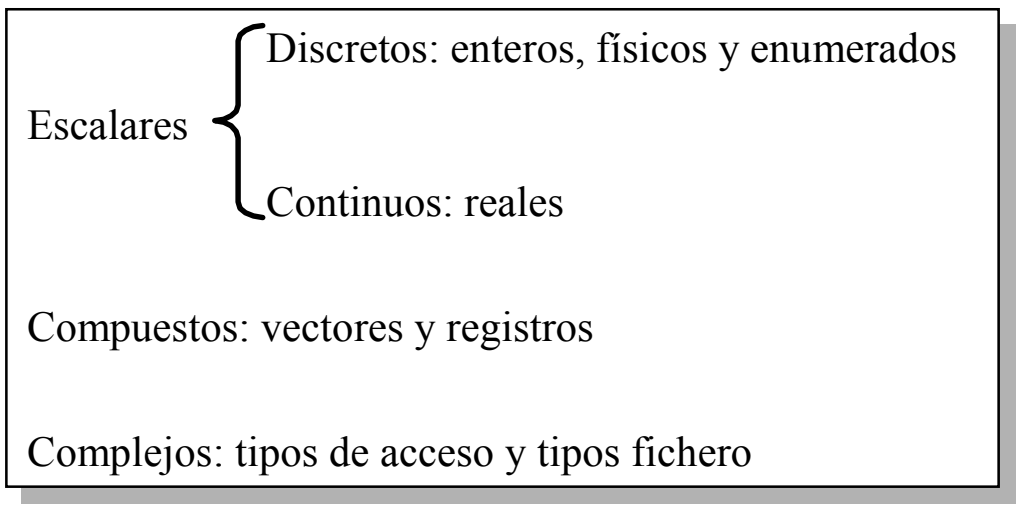
Objetos VHDL
Fichero

- Fichero
  - Permite leer y escribir datos que persisten cuando termina la simulación
  - Normalmente utilizados para almacenar estímulos y resultados de simulación



Tipos de datos

- Tipo de datos:
  - Define el conjunto de posibles valores de un objeto. Concepto estático, no varia durante la simulación
- Clasificación de tipos de datos



Tipos de datos

- Tipo Escalar:
  - Contiene valores formados por unidades indivisibles
  - Tipo discreto:
    - Representa un número finito de valores
  - Tipo continuo:
    - Modela todos los valores dentro de un intervalo
      - Contiene un conjunto finito de valores representables
      - Un valor se redondea al valor representable más próximo
- Tipos compuestos:
  - Contienen valores divisibles en unidades más pequeñas

Tipos de datos
Tipo entero

- Tipo entero:
  - integer: Tipo predefinido del lenguaje.
  - Valores enteros entre -2.147.483.647 y 2.147.483.647

```
type Precio is integer;
type Mes is range 1 to 12;
```

Declaración de tipos enteros

Especificación de rango

Tipos de datos
Tipos físicos

- Tipo físico:
  - Representa medidas del mundo real.
  - Valores = literal numérico + unidad (literal físico)
  - Define unidad primaria y permite unidades secundarias
  - Valores múltiplos de la unidad primaria
  - Declaración:

```
type <identificador> is range <literal> to | downto <literal>
 units
  <identificador unidad primaria>;
  {<identificador unidad secundaria> = <literal físico>;}
 end units [<identificador>];
```

Tipos de datos
Tipos físicos

Declaración del tipo predefinido time

```
type time is range 0 to 1e20
 units
  fs;
  ps = 1000 fs;
  ns = 1000 ps;
  us = 1000 ns;
  ms = 1000 us;
  sec = 1000 ms;
  min = 60 sec;
  hr = 60 min;
 end units time;
```

Unidad primaria

Unidades secundarias

A, B y C tienen el mismo valor. 4.3211 se redondea

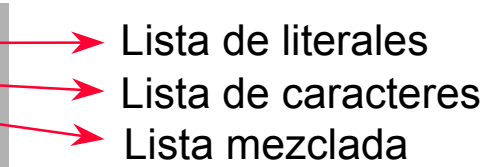
```
variable A, B, C : time;
...
A := 4321 fs;
B := 4.321 ps;
C := 4.3211 ps;
```

Tipos de datos
Tipos enumerados

- Tipo enumerado:
  - Se define el conjunto de valores mediante una lista
  - Declaración:

```
type <identificador> is (<ident.> | <carácter> {, <id.> | <car.>});
```

```
type Cardinal is (Norte, Sur, Este, Oeste);
type Teclas is ('N', 'S', 'E', 'O');
type Mezclado is ('N', Sur, Este, 'O');
...
variable Direccion : Cardinal := Este;
variable Accion : Teclas := 'O';
variable  Var : Mezclado := Sur;
...
Var := 'N';
```



Tipos de datos
Tipos enumerados

```
type boolean is (false, true);
type bit is ('0', '1');
type severity_level is (note, warning, error, failure);
type character is (NUL, SOH, STX, ETX, EOT, ENQ, ACK, BEL, BS,
HT, LF, VT, FF, CR, SO, SI, DEL, DC1, DC2, DC3, DC4, DC5, NAK,
SYN, ETB, CAN, EM, SUB, ESC, FSP, GSP, RSP, USP, ' ', '!', '"',
'#', '$', '%', '&', ''', '(', ')', '*', '+', ',', '-', '.', '/',
'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '<',
'=', '>', '?', '@', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
'W', 'X', 'Y', 'Z', '[', '\', ']', '^', '_', '`', 'a', 'b', 'c',
'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '{', '|', '}');
```

Tipos predefinidos del lenguaje

256 caracteres ASCII

```
type std_ulogic is (  'U',   -- No inicializado
                      'X',   -- Forzando desconocido
                      '0',   -- Forzando cero
                      '1',   -- Forzando uno
                      'Z',   -- Alta impedancia
                      'W',   -- Desconocido débil
                      'L',   -- Cero débil
                      'H',   -- Uno débil
                      '-');  -- Redundante
```

Definido en el paquete std_logic_1164 de IEEE. Adecuado para modelar las propiedades de las conexiones físicas

Tipos de datos
Tipo real

- Tipo Real:
  - real: Tipo predefinido del lenguaje.
  - Valores enteros entre -1.0e38 y 1.0e38.
  - Mínimo 6 decimales de precisión

```
type Resultado is real;
type Puntuacion is range 10.0 downto 0.0;
```

Declaración de tipos reales

Especificación de rango

Tipos de datos
Inicialización de tipos escalares

- Inicialización de objetos de tipo escalar
  - Durante su declaración se les asigna un valor:
    - Especificado en la sentencia

```
signal Reloj is bit := '1';
variable Contador : integer := 20;
```

    - Por defecto el valor más a la izquierda de los del tipo

```
variable Resultado is boolean;
type Mes is range 12 to 1;
variable MesActual : Mes;
type Cardinal is (Norte, Sur, Este, Oeste);
variable Direccion : Cardinal;
signal Conexión : std_ulogic := 'U';
```

Resultado = false

MesActual = 12

Direccion = Norte

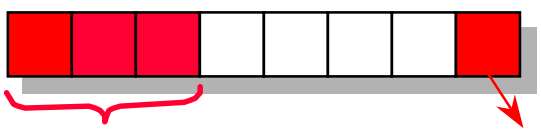
Redundante!!

Tipos de datos
Vectores

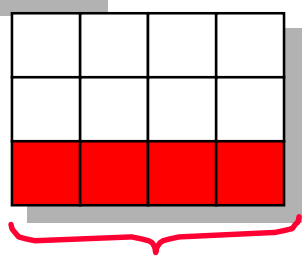
- Vectores
  - Valores compuestos de objetos del mismo tipo
  - Objetos ordenados mediante 1 o más índices

```
type byte is array (character 'h' downto 'a') of
bit;
type tabla is array (0 to 2, 0 to 3) of bit;
...
variable Op1 : byte := "01010011";
variable Op2 : tabla;
```

Vector unidimen-
sional o vector



Vector multidimen-
sional o matriz



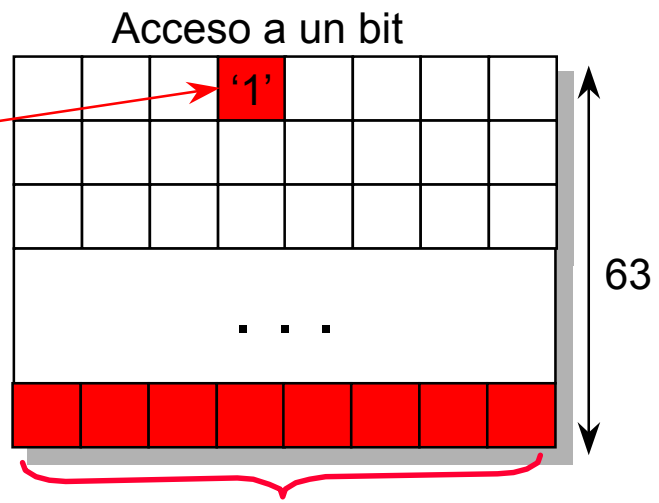
Op1('h' **downto** 'f') = "010"; Op1('a') := '1';

Op2(2, 0 **to** 3) := x"c";

Tipos de datos
Vectores

- Ejemplo: Modelado de una memoria RAM

```
type Memoria is array (0 to 63, 0 to 7) of bit;
...
variable RamA, RamB : Memoria;
...
RamA(0,3) := '1';
```




```
type Direccion is range 0 to 63;
type Palabra is array (0 to 7) of bit;
...
variable DireccionRamA : Direccion;
variable PalabraRamA : Palabra;
...
DireccionRamA := 63;
PalabraRamA := RamA(DireccionRamA, 0 to 7);
```

```
RamB := RamA;
```

Acceso a toda la Ram

Tipos de datos
Vectores no restringidos

- Vectores no restringidos:
  - No se especifica el rango en la definición del tipo sino en el momento de declarar un objeto del tipo.

```
type <identificador> is array (<tipo índice> range <> {, ...} of tipo_objetos;
```

```
type bit_vector is array (positive range <>) of bit;
type string is array (positive range <>) of character;
type std_ulogic_vector is array (positive range <>) of std_ulogic;
```

Tipos predefinidos

std_logic_1164

```
variable Operador1: bit_vector(7 downto 0) := "01101011";
variable Mensaje : string(1 to 50) := (others => ' ');
variable BusDatos : std_ulogic_vector(31 downto 0) := (others => 'Z');
```

Agregado

Tipos de datos
Registros

- Registros
  - Valores compuestos de objetos de tipos distintos. Estos objetos reciben el nombre de campos.
  - Ninguna relación con un registro *hardware*
  - Declaración

```
type <identificador> is record
 <identificador campo> {, ...} : tipo;
 {...}
end record [<identificador>];
```

Tipos de datos
Registros

- Ejemplo: Registro para almacenar fechas

```
type TipoMes is (Ene, Feb, Mar, Abr, May, Jun, Jul, Ago, Sep, Oct, Nov, Dic);
type Fecha is record
Dia  : integer range 1 to 31;
Mes  : TipoMes;
Anyo : integer;
end record Fecha;
```

Declaración de campos

Acceso a un campo

```
variable Hoy, Ayer : Fecha;
...
Hoy.Dia := 15; Hoy.Mes := Enero; Hoy.Anyo := 1998;
Ayer := Hoy;
```

Acceso al registro entero

Tipos de datos
Agregados

- Agregado:
  - Lista de valores útil para actualizar objetos compuestos

```
( [ <posición> | <campo>{, ...} => ] <literal> {, ...})
```

```
type Punto is array (0 to 2) of real;
...
variable Origen : Punto := (0.0, 0.0, 0.0);
variable Bits : bit_vector (0 to 9) := (others => '1');
...
Origen := (0 => 2.5, 1 => -7.2, 2 => 2.5);
Origen := (1 => -7.2, 2 => 2.5, 0 => 2.5);
Origen := (1 => -7.2, others => 2.5);
```

Notación posicional

Notación nominal

Asigna los restantes

Asignaciones equivalentes

```
Hoy := (15,  Ene, 1998);
Hoy := (Anyo => 1998, Dia => 15, Mes => Ene);
```

Tipos de datos
Subtipos de datos

- Subtipo de datos
  - Se asocia una restricción a un tipo base para obtener un subconjunto de valores de este tipo.

```
subtype <identificador> is <id_tipo> [<rango>];
```

Atributo

```
subtype natural is integer range 0 to integer'high;
subtype positive is integer range 1 to integer'high;
```

Subtipos predefinidos

```
type Decimal is ('0', '1', '2', '3', '5', '6', '7', '8', '9');
subype Octal is Decimal range '0' to '7';
sybtype Abecedario is character range 'a' to 'z';
subtype DiaMes is integer range 1 to 31;
...
variable MesActual is range 1 to 12;
```

Declaración de subtipo implícita

Tipos de datos
Tipos de acceso

- Tipos de acceso:
  - Tipos de datos dinámicos : apuntadores

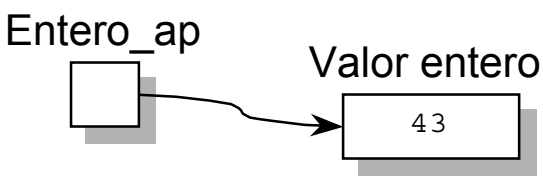
```
type ApuntaEntero is access integer;
...
variable Entero_ap : ApuntaEntero;
...
Entero_ap := new integer;
Entero_ap.all := 43;
...
deallocate(Entero_ap);
```

Declaración de un tipo apuntador a objetos enteros

Creación de un objeto integer

Acceso al objeto apuntado

Función implícita para liberar la memoria del objeto



Tipos de datos
Tipos fichero

- Tipos fichero:
  - Tipo para almacenar un objeto fichero

```
type <identificador> is file of <tipo_objetos>;
```

```
type FicheroEnteros is file of integer;
file Estimulos : FicheroEnteros is in "datos.in";
```

datos.in

```
12
23
-4
...
```

```
type line is access string;
type text is file of string;
file Resultado : text is in
"datos.out";
```

Tipo fichero de texto (paquete textio, biblioteca std)

datos.out

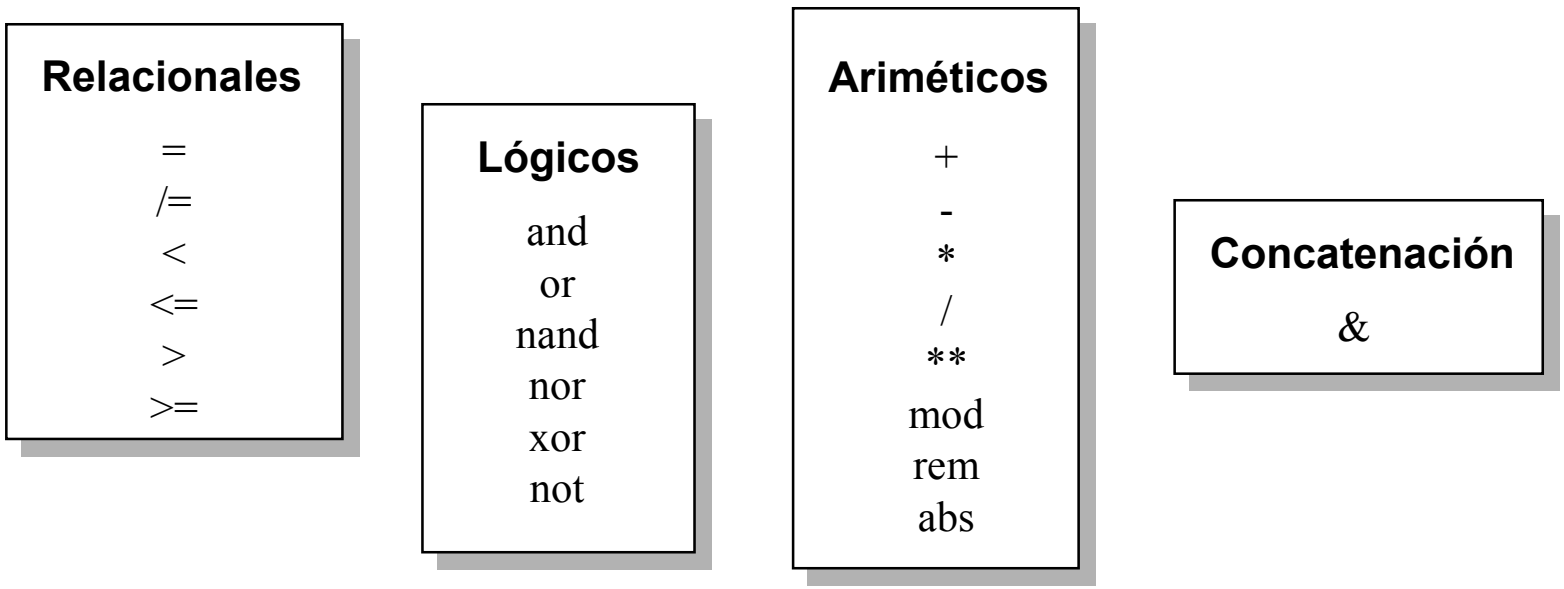
```
RESULTADO

1)  In = 12; Out = X;
2)  In = 23; Out = 1;

...
```

Operadores y expresiones
Operadores

- Operador:
  - Símbolo que indica una operación
- Tipos de operadores:



Operadores y expresiones
Expresiones

- Expresión:
  - Fórmula que indica cómo debe calcularse un valor
  - Formada por operadores y operandos
  - Los operandos son literales, identificadores, atributos, calificaciones de tipo, conversiones de tipo y paréntesis

Aritmética
```
(-b + sqrt(b**2 - 4.0*a*c))/(2.0*a)
```

Relacional
```
retardo <= 20 ns
nombre > "Smith"
```

Construcción
```
BitSigno & VectorValor
```

Lógica
```
(a xor b) and not c;
```

Operadores y expresiones

- Multiplicación y división con literales físicos:

físico * entero → físico	3.4 m * 2 = 6.8 m
físico * real → físico	3.4 m * 2.0 = 6.8 m
entero * físico → físico	2 * 3.4 m = 6.8 m
real * físico → físico	2.0 * 3.4 m = 6.8 m
físico / entero → físico	3.4 m / 2 = 1.7 m
físico / real → físico	3.4 m / 2.0 = 1.7 m
físico / físico → entero	3.4 m / 2.0 cm = 170

físico * físico → no soportado
entero / físico → no soportado
real / físico → no soportado

Deben definirse nuevos tipos físicos y usar funciones explícitas de conversión. Ej.:
Longitud y Area (L * L → A)
Periodo y frecuencia (1 / T → F)

Atributos

- Atributo:
  - Característica que se asocia a un elemento de un modelo VHDL
  - No confundir el valor de un objeto con un atributo de un objeto. Un objeto tiene un único valor y puede tener varios atributos
  - VHDL tiene atributos predefinidos y también permite la definición de nuevos atributos
  - Sintaxis para hacer referencia a un atributo

```
<identificador_elemento>'<identificador_atributo>
```

Atributos sobre rangos de vectores

- Atributos sobre rangos de vectores

A'left(n)	Valor izquierdo del índice n de A
A'right(n)	Valor derecho del índice n de A
A'low(n)	Valor mínimo del índice n de A
A'high(n)	Valor máximo del índice n de A
A'ascending(n)	Verdadero si el rango del índice n de A es ascendente
A'range(n)	Rango del índice n de A
A'reverse_range(n)	Rango del índice n de A invertido
A'length(n)	Número de valores del rango n de A

"Array"

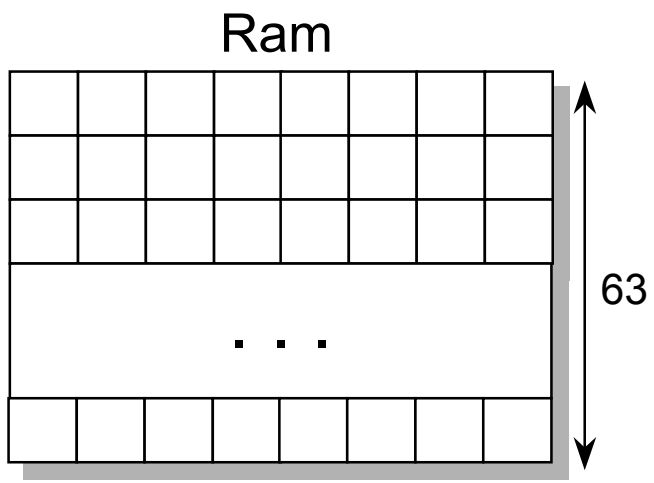
Dimensión sobre la que se aplica el atributo.
Si no se indica, la primera por defecto

Atributos

Atributos sobre rangos de vectores

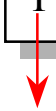
```
type Memoria is array (0 to 63, 7 downto 0) of bit;
...
variable Ram : Memoria;
```

Ram'left = 0
Ram'left(2) = 7
Ram'right(2) = 0
Ram'right(1) = 63
Ram'low(2) = 0
Ram'high(1) = 63
Ram'ascending = true
Ram'ascending(2) = false
Ram'range = 0 to 63
Ram'reverse_range(2) = 0 to 7
Ram'length(2) = 8
Ram'length(1) = 64



- Atributos sobre tipos de datos

T'base	Tipo base de T
T'left	Valor más a la izquierda de T
T'right	Valor más a la derecha de T
T'low	Valor mínimo de T
T'high	Valor máximo de T
T'pos(x)	Posición ocupada por x en T
T'val(x)	Valor de la posición x en T
T'pred(x)	Valor de la posición anterior a x en T
T'succ(x)	Valor de la posición siguiente a x en T
T'leftof(x)	Valor de la posición a la izquierda de x en T
T'rightof(x)	Valor de la posición a la derecha de x en T



Tipo de datos

Atributos

Atributos sobre tipos de datos

```
type PuntosCardinales is (Norte, Sur, Este, Oeste);
```

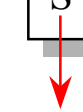
PuntosCardinales'left = Norte
PuntosCardinales'right = Oeste
PuntosCardinales'low = Norte
PuntosCardinales'high = Oeste
PuntosCardinales'pos(Sur) = 1
PuntosCardinales'val(3) = Oeste
PuntosCardinales'pred(Sur) = Norte
PuntosCardinales'succ(Sur) = Este
PuntosCardinales'leftof(Sur) = Norte
PuntosCardinales'rightof(Sur) = Este

```
type Num is range 8 downto 3;
```

Num'base = integer
Num'left = 8
Num'right = 3
Num'low = 3
Num'high = 8
Num'pos(6) = 6 (por ser entero)
Num'val(8) = 8
Num'val(0): error
(0 no está en el rango "8 downto 3")
Num'pred(4) = 3
Num'succ(4) = 5
Num'leftof(4) = 5
Num'rightof(4) = 3

- Atributos sobre señales

S'delayed(t)	Señal S retrasada t unidades de tiempo
S'stable(t)	Señal que toma el valor verdadero si la señal S es estable hace t unidades de tiempo
S'quiet(t)	Verdadero si no ha habido asignación a S en t unidades de tiempo
S'transaction	Señal tipo bit: cambia de valor cuando hay asignación a S
S'event	Verdadero si ocurre un evento en S
S'active	Verdadero si hay una asignación en S
S'last_event	Unidades de tiempo desde el último evento de S
S'last_active	Unidades de tiempo desde la última asignación a S
S'last_value	Valor anterior de S



Señal

Atributos
Definición de atributos

- Atributos definidos por el usuario
  - VHDL permite asociar atributos a cualquier elemento
    - Declaración: Determina su tipo de datos
    - Especificación: Lo asigna a un elemento y le da un valor
    - Concepto estático, el valor de un atributo no varía

```
attribute <identificador> :  <tipo_datos>;
```

→ Declaración de atributo

```
attribute <identificador> of  <id_elemento> : <clase_elemento> is <expresión>;
```

↓

Especificación de atributo

Atributos
Definición de atributos

```
signal Reloj : bit;
attribute NumeroPin : natural;
attribute NumeroPin of Reloj : signal is 5;
```

Atributo sobre señal

```
type PuntosCardinales is (Norte, Sur, Este, Oeste);
attribute CodigoEstados : bit_vector;
attribute CodigoEstados of Norte : literal is "10";
attribute CodigoEstados of Sur : literal is "00";
attribute CodigoEstados of Este : literal is "11";
attribute CodigoEstados of Oeste : literal is "01";
```

Atributo sobre valores de un tipo enumerado

```
entity INV is
  port ( Entrada : in bit;
         Salida : out bit);
attribute Retardo : time;
attribute Retardo of INV : entity is 1.32 ns;
```

Atributo sobre una entidad