

Primeiro contato com a linguagem

Tópicos

- **Entidade de projeto**

Declaração da entidade e declaração da arquitetura

- **Classes de objetos:** constante, variável e sinal

- **Tipos**

Tipos escalares

Tipos compostos

Definição de novos tipos

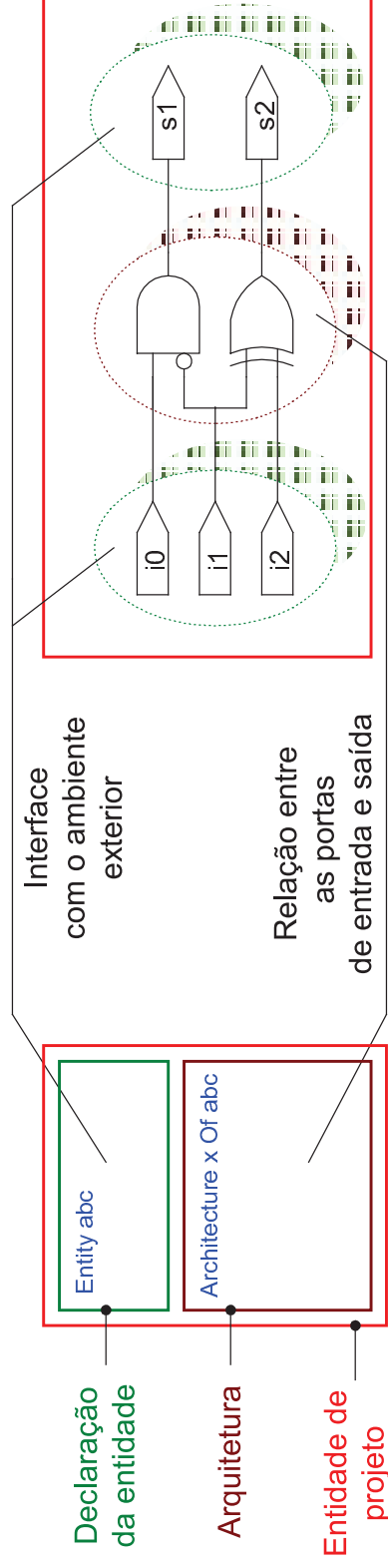
- **Operadores**

- **Exemplos de utilização**



Entidade de projeto

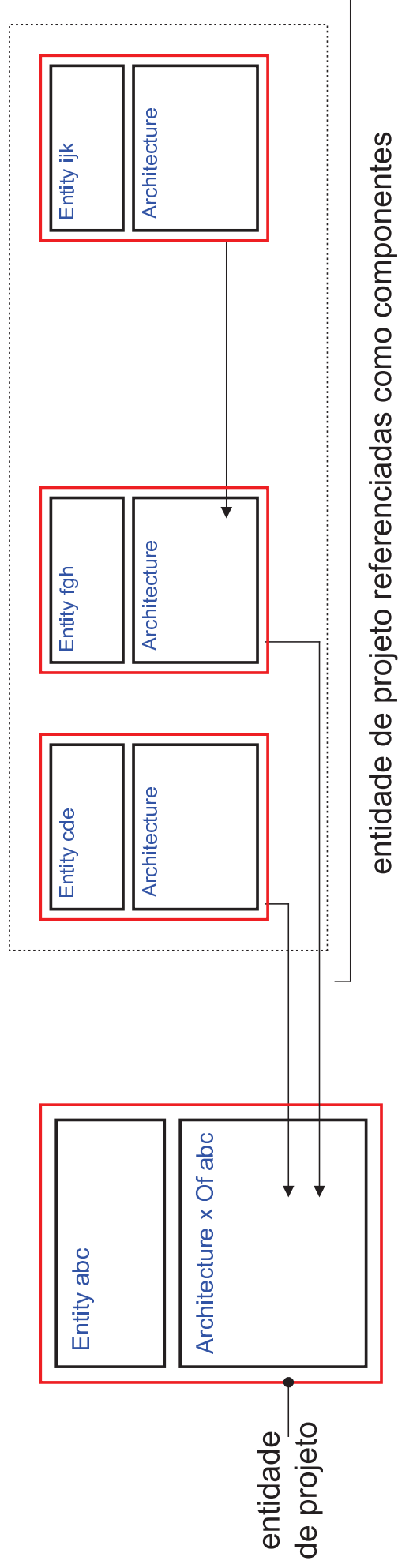
- **A entidade de projeto** pode representar:
 - de uma simples porta lógica a um sistema completo
- Composta de duas partes:
 - **Declaração da entidade**
 - define portas de entrada e de saída da descrição
 - equivalente ao símbolo de um bloco em captura esquemática
 - **Arquitetura**
 - descreve as relações entre as portas
 - equivalente ao esquema contido no bloco em captura esquemática



Entidade de projeto

- **Entidade de projeto**

- pode ser descrita na forma: interligação de outras entidades
- estabelece um projeto hierárquico
- não existe limite para o nível de hierarquia
- diferentes estilos de descrição podem ser empregados



Declaração da entidade

- **ENTITY**: inicia a declaração
- **GENERIC**: permite a passagem de parâmetros para a entidade
- **PORT**: define modo e tipo das portas
 - modo **IN** : entrada
 - modo **OUT** : saída
 - modo **BUFFER** : saída – pode ser referenciada internamente
 - modo **INOUT** : bidirecional
- **END**: termina a declaração

```
ENTITY entidade_abc IS

GENERIC (n      : INTEGER := 5);

PORT (x0, x1    : IN      tipo_a;      -- entradas
      y0, y1    : OUT      tipo_b;      -- saídas
      y2        : BUFFER   tipo_c;      -- saída
      z0, z1    : INOUT    tipo_d);     -- entrada / saída

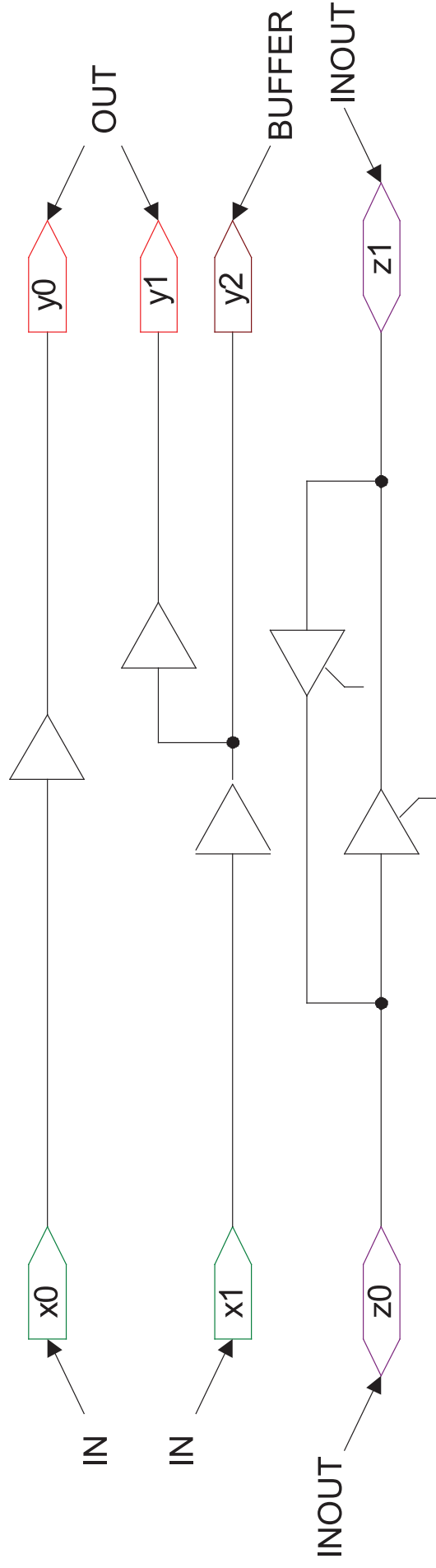
END entidade_abc;
```



Declaração da entidade

- Exemplo: declaração de uma entidade e esquema da arquitetura

```
ENTITY entidade_abc IS
    PORT (x0, x1 : IN      tipo_a;      -- entradas
          y0, y1 : OUT      tipo_b;      -- saídas
          y2     : BUFFER   tipo_c;      -- saída
          z0, z1 : INOUT    tipo_d);     -- entrada / saída
END entidade_abc;
```



Declaração da arquitetura

- **ARCHITECTURE**: inicia a declaração
- As linhas que seguem podem conter:
 - declaração de sinais e constantes
 - declaração de componentes referenciados
 - descrição de subprogramas locais
 - definição de novos tipos
- **BEGIN**: inicia a descrição
- **END**: termina a descrição

```
ARCHITECTURE estilo_abc OF entidade_abc IS  
  -- declaracoes de sinais e constantes  
  -- declaracoes de componentes referenciados  
  -- descricao de sub-programas locais  
  -- definicao de novos tipos de dados locais  
  --  
BEGIN  
  --  
  -- comandos concorrentes  
  --  
END estilo_abc;
```

Classe de objetos: constante, variável e sinal

- **Objetos:** são elementos que contêm um valor armazenado
- Quatro tipos:
 - **CONSTANT**: valor estático
 - **VARIABLE**: valor imposto pode ser alterado
regiões de código sequencial
 - **SIGNAL**: valor imposto pode ser alterado
regiões de código sequencial e concorrente
 - **FILE**: associado à criação de arquivos
- Exemplos de atribuição de valores entre objetos

```
signal_2    <= sinal_1;           -- atribuicao valor para sinal
signal_3    <= variavel_1;        -- atribuicao valor para sinal
signal_4    <= constante_1;       -- atribuicao valor para sinal

variavel_2  := sinal_1;           -- atribuicao valor para variavel
variavel_3  := variavel_1;        -- atribuicao valor para variavel
variavel_4  := constante_1;       -- atribuicao valor para variavel
```

Exemplo de uma descrição completa

- São definidas três portas: uma de entrada e duas de saída
- Tipo das portas: **INTEGER**
- Operações: transferência de valores
- Declarados: **s1** sinal interno e **c1** constante – linhas 7 e 8

```
1 ENTITY atrib_1 IS
2   PORT (x1 : IN  INTEGER;
3         y1,z1 : OUT INTEGER);
4 END;
5
6 ARCHITECTURE teste OF atrib_1 IS
7   SIGNAL  s1 : INTEGER;
8   CONSTANT c1 : INTEGER := 7;
9 BEGIN
10  y1 <= s1;
11  s1 <= x1;
12
13  z1 <= c1;
14 END teste;
```

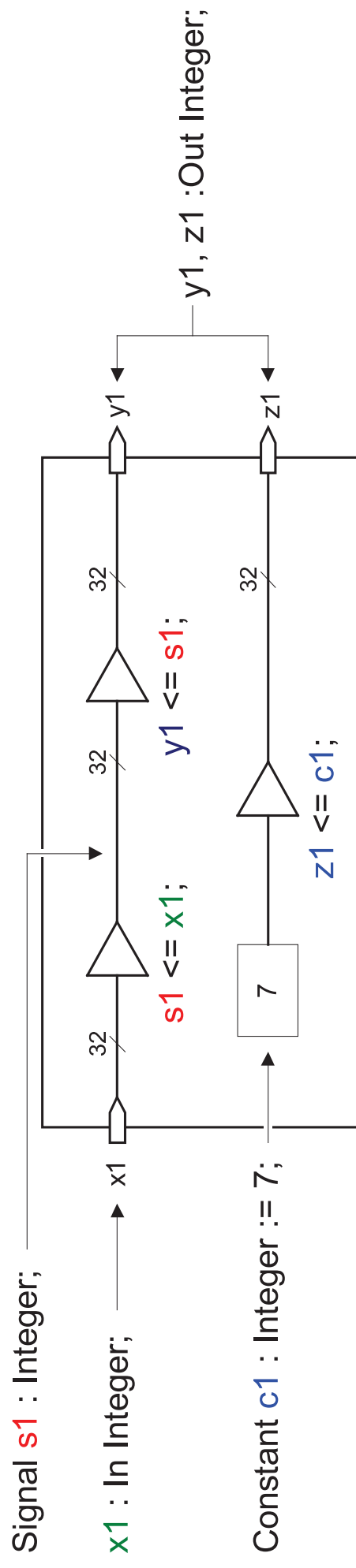
Exemplo de uma descrição completa (continuação)

- Observar concorrência no código:

linha 11: **s1** <= **x1** – **s1** recebe o valor da porta de entrada **x1**

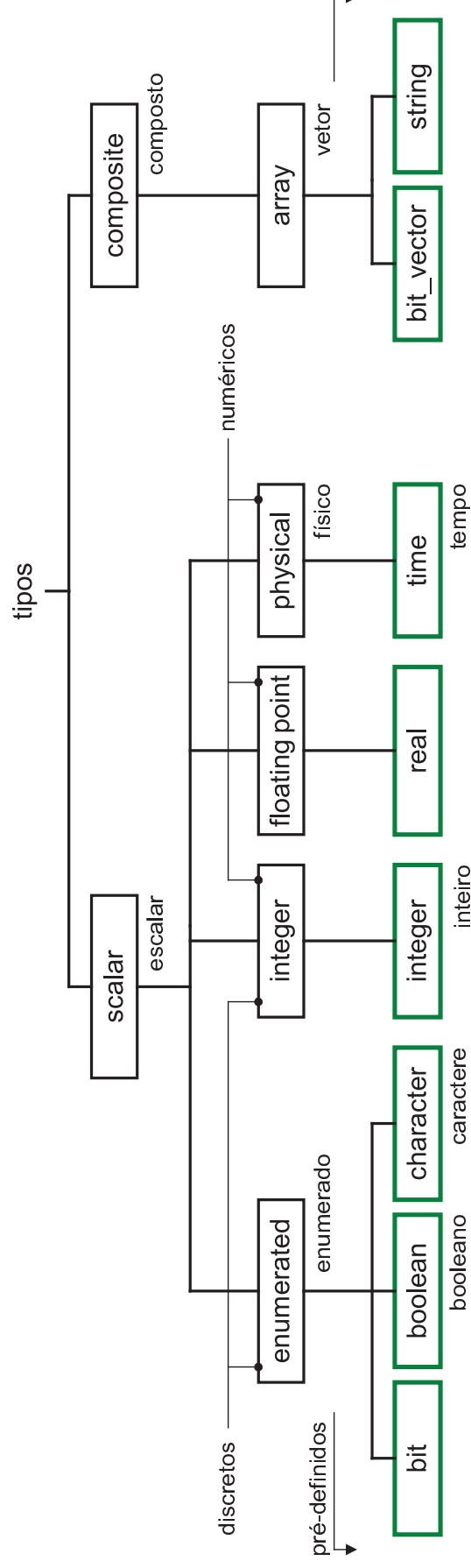
linha 10: **y1** <= **s1** – valor de **s1** transferido para a porta de saída **y1**

```
6 ARCHITECTURE teste OF atrib_1 IS
7   SIGNAL s1 : INTEGER;
8   CONSTANT c1 : INTEGER := 7;
9 BEGIN
10    y1 <= s1;
11    s1 <= x1;
12
13    z1 <= c1;
```



Tipos

- **Objetos:** devem ser declarados segundo uma especificação de tipo
- **Objetos de tipos diferentes:** não é permitida a transferência de valores
- **Tipos definidos no pacote padrão VHDL:**
 - bit, boolean, character, integer, real, time, bit_vector e string.

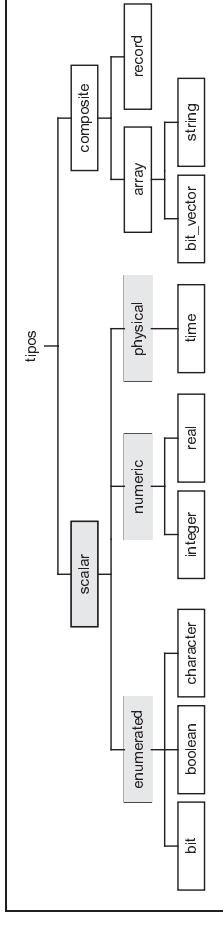


Tipos escalares

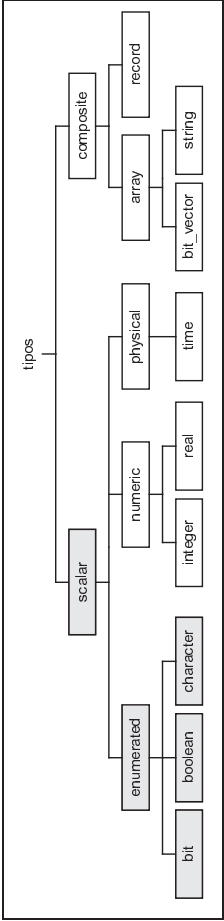
- **Classes:**

Enumerado, Inteiro, Ponto Flutuante e Físico

- **São ordenados:** podem ser aplicados operadores relacionais (maior, menor)



Tipos escalares



- **Classe:** enumerado
 - **BIT**: empregado para representar níveis lógicos alto e baixo
 - **BOOLEAN**: empregado em comandos que executam uma decisão
 - **CHARACTER**: qualquer caractere ASCII padrão (estendido versão VHDL 1993)

classe	tipo	valor	exemplos
enumerado	BIT	um, zero	1, 0
	BOOLEAN	verdadeiro, falso	TRUE, FALSE
	CHARACTER	caracteres ASCII	a, b, c, A, B, C, ?, (



Tipos escalares

- **Classe:** numérico

- **INTEGER:** representa um número inteiro entre $-2^{31} \leq x \leq 2^{31}-1$

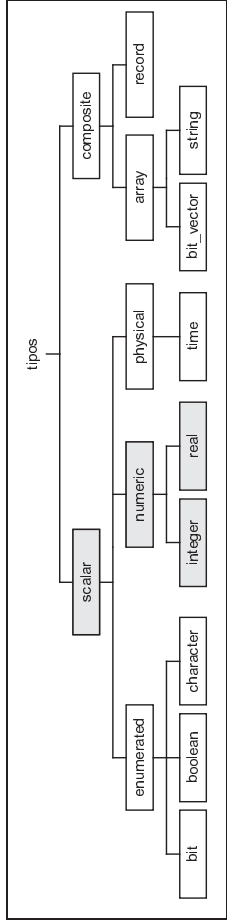
- Subtipos de **INTEGER**

- **NATURAL:** valores entre $0 \leq x \leq 2^{31}-1$

- **POSITIVE:** valores entre $1 \leq x \leq 2^{31}-1$

- Nota: a troca de valores entre tipo ↔ subtipo é permitida

- **REAL:** ponto flutuante



classe	tipo	valor	exemplos
numérico	INTEGER	$-2^{31} \leq x \leq 2^{31}-1$	123, 8#173#, 16#7B#
	NATURAL	$0 \leq x \leq 2^{31}-1$	2#11_11_011#
	POSITIVE	$1 \leq x \leq 2^{31}-1$	
	REAL	$-3.65 \times 10^{47} \leq x \leq +3.65 \times 10^{47}$	1.23, 1.23E+2, 16#7.B#E+1



Tipos escalares – observações quanto à síntese

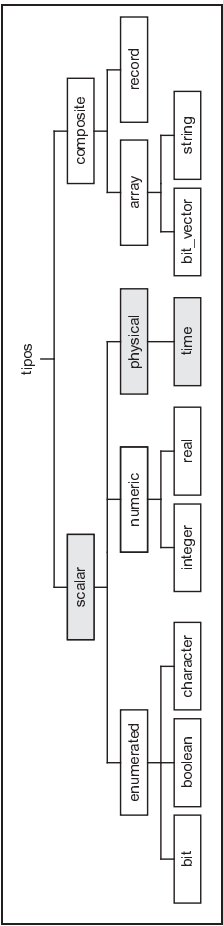
- **Classe:** numérico
 - **INTEGER:** representa um número inteiro entre $-2^{31} \leq x \leq 2^{31}-1$
é necessária uma linha de 32 bits para representação!
é conveniente limitar a faixa de valores na declaração
 - **REAL:** ponto flutuante
- não suportado pelas ferramentas de síntese

classe	tipo	valor	exemplos
numérico	INTEGER	$-2^{31} \leq x \leq 2^{31}-1$	123, 8#173#, 16#7B#
	NATURAL	$0 \leq x \leq 2^{31}-1$	2#11_11_011#
	POSITIVE	$1 \leq x \leq 2^{31}-1$	
	REAL	$-3.65 \times 10^{47} \leq x \leq +3.65 \times 10^{47}$	1.23, 1.23E+2, 16#7.B#E+1

Tipos escalares

- **Classe:** físico

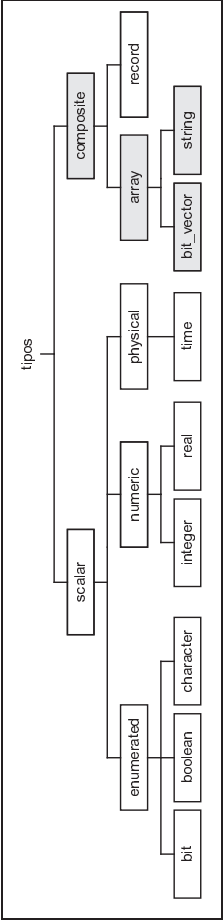
- representa quantidades físicas
- estabelecidos: uma unidade básica e múltiplos desta unidade
- **TIME:** tipo físico definido no pacote padrão
 - quantidade física: tempo
 - unidade básica: $fs = 10^{-15}s$
 - múltiplos: ps, ns, us, ms, us, sec, min, hr
 - síntese: tipo time é ignorado pelas ferramentas de síntese



classe	tipo	valor	exemplos
físico	TIME	ps =10 ³ fs ns =10 ³ ps us =10 ³ ns ms =10 ³ us min =60 sec	1 us, 100 ps, 1 fs



Tipos compostos



- **Classes:**

- vetor (*array*): agrupa elementos de um mesmo tipo

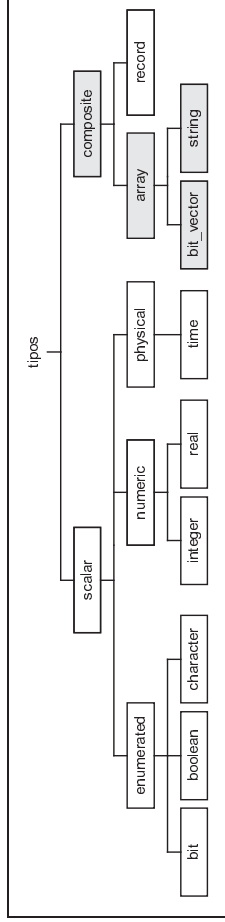
- Classes definidas no pacote padrão:

- **BIT_VECTOR**: vetor contendo elementos tipo **bit**
 - **STRING**: vetor contendo elementos tipo **character**

Classe	tipo	valor	exemplos
vetor	BIT_VECTOR	1 , 0	"1010", B"10_10", O"12", X"A"
	STRING	tipo "character"	"texto", ""incluindo_aspas""



Tipos compostos

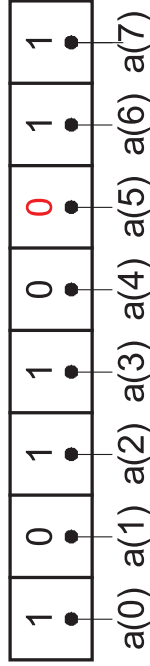


• Declarações:

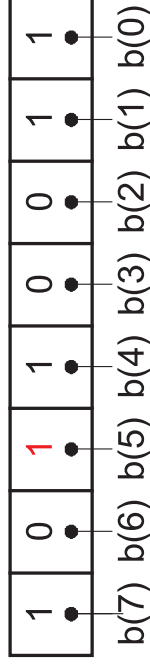
- limites definidos por **TO** e **DOWNTO**

• Exemplos de declarações de tipos: **bit_vector** e **string**:

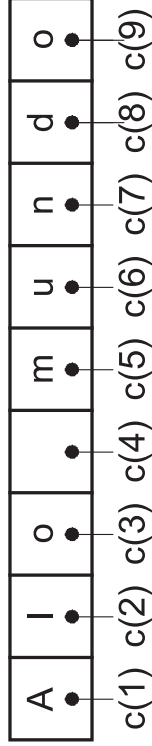
SIGNAL a: BIT_VECTOR(0 TO 7) := "10110011"



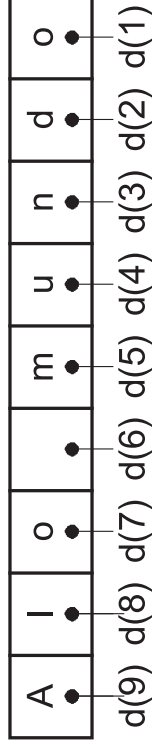
SIGNAL b: BIT_VECTOR(7 DOWNTO 0) := "10110011"



SIGNAL c: STRING(1 TO 9) := "Alo mundo"



SIGNAL d: STRING(9 DOWNTO 1) := "Alo mundo"



Agregados

- Uma expressão indicando o valor de um tipo composto
- Cada elemento do tipo composto tem o seu valor definido

• OTHERS

- identifica todos os elementos não especificados
- deve ser a última associação na lista de associações

```
-- valor 00010 agregado notacao posicional
-- posicao:
--      4      3      2      1      0
s2 <= ('0','0','0','0','1','1','0');      -- equivalente: s2 <= "00010";

-- valor 00011 agregado associacao por nomes
--
s3 <= (1=>'1', 0=>'1', OTHERS=>'0');      -- equivalente: s3 <= "00011";
```



Definição de novos tipos (visão preliminar)

- A linguagem permite a criação de novos tipos

- Aplicações:

- facilitar a leitura do código: estados de uma máquina
- definir novos tipos físicos: resistência, capacitância etc.
- novos tipos compostos: definição de memórias

- Declaração: palavra reservada **TYPE**

- Exemplo:

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3);  
SIGNAL abc : estado := parado;
```

- é definido um novo tipo **estado**

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3);
```

- declaração de um sinal do tipo **estado**

```
SIGNAL abc : estado := parado;
```

- valores possíveis para o sinal **abc**: parado, inicio, caso_1, caso_2, caso_3

