Instituto Politécnico Nacional
Escuela Superior de Cómputo
Academia de Sistemas Digitales

Funciones y Procedimientos

Autor: Victor Hugo García Ortega

INTRODUCCIÓN

Colectivamente las funciones y procedimientos son llamados subprogramas.

En las funciones y procedimientos se ejecuta el código secuencialmente de la misma forma que en un proceso.

Las funciones y procedimientos generalmente se colocan dentro de un paquete.

FUNCIONES

Las funciones pueden usar las mismas sentencias que un proceso (IF, FOR, case-when).

La sentencia que no puede ser usada en la función es WAIT.

Tampoco se pueden usar señales (SIGNAL) e instancias de componentes (PORT MAP).

La construcción de la función necesita la declaración, la definición y la llamada a la función.

FUNCIONES

```
[ Pure | Impure ] FUNCTION function_name [<parameter list>] RETURN data_type IS
                      [declarations]
                  BEGIN
                      (sequential statements)
                  END function_name;
```

Los parámetros solo pueden ser CONSTANTES (default) SEÑALES y ARCHIVOS (FILE), las variables NO son permitidas.

El único modo que se permite en los parámetros de las funciones es IN.

No se deben especificar rangos en los tipos de datos como en los enteros y vectores.

FUNCIONES

Las funciones puras regresan el mismo valor cada vez que son llamadas con los mismos valores de parámetros.

Las funciones impuras pueden regresar diferentes valores cuando son llamadas varias veces con los mismos valores de parámetros. Además las funciones impuras pueden actualizar objetos fuera de su alcance.

FUNCIONES

EJEMPLO:

```
FUNCTION f1 (a, b: INTEGER; SIGNAL c: STD_LOGIC_VECTOR)
    RETURN BOOLEAN IS
BEGIN
    (sequential statements)
END f1;
```

Llamadas a funciones:

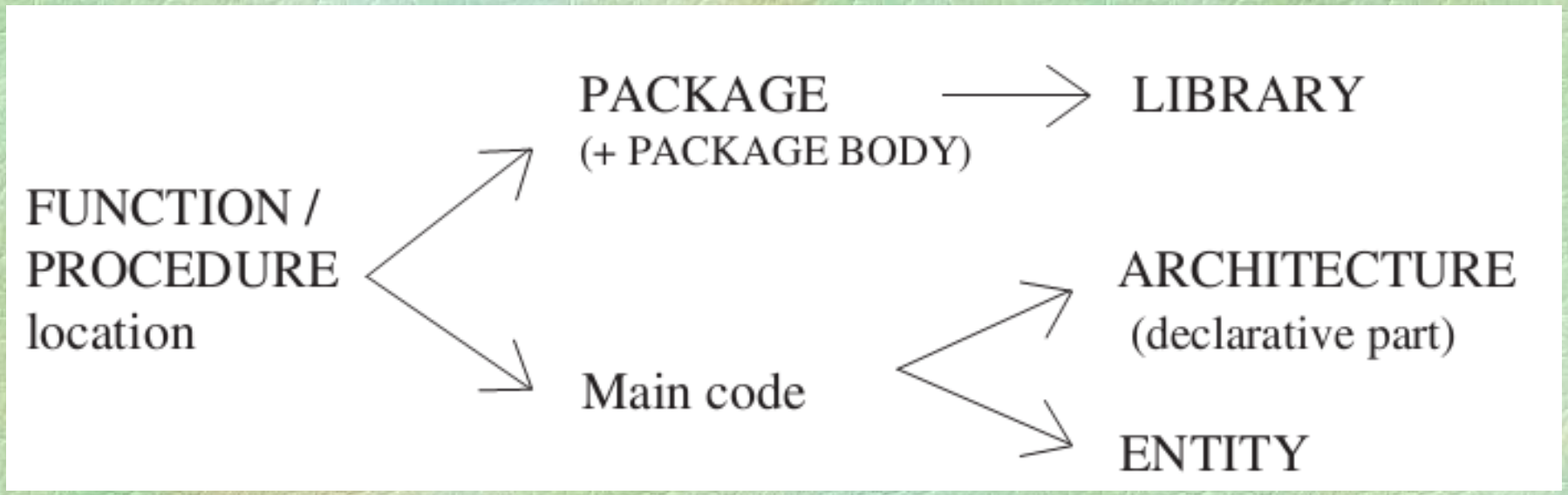
```
x <= conv_integer(a);          -- converts a to an integer
                               -- (expression appears by itself)
y <= maximum(a, b);            -- returns the largest of a and b
                               -- (expression appears by itself)
IF x > maximum(a, b) ...       -- compares x to the largest of a, b
                               -- (expression associated to a
                               -- statement)
```

FUNCIONES

EJEMPLO:

```
------ Function body: -------------------------------
FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN IS
BEGIN
   RETURN (s'EVENT AND s='1');
END positive_edge;
------ Function call: -------------------------------
...
IF positive_edge(clk) THEN...
...
-----------------------------------------------------
```

FUNCIONES



Usualmente la función se coloca en un paquete, sin embargo, también se puede colocar en el código principal dentro de la arquitectura o la entidad.

FUNCIONES

Cuando se coloca en un paquete, el cuerpo del paquete contiene el cuerpo o la definición de la función. La función tiene que ser declarada en la sección de declaraciones del paquete.

FUNCIONES

Ejemplo: Función dentro del código principal.

```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY dff IS
6     PORT ( d, clk, rst: IN STD_LOGIC;
7            q: OUT STD_LOGIC);
8  END dff;
9  ----------------------------------------
```

FUNCIONES

```
10 ARCHITECTURE my_arch OF dff IS
11 ----------------------------------------
12    FUNCTION positive_edge(SIGNAL s: STD_LOGIC)
13       RETURN BOOLEAN IS
14    BEGIN
15       RETURN s'EVENT AND s='1';
16    END positive_edge;
17 ----------------------------------------
18 BEGIN
19    PROCESS (clk, rst)
20    BEGIN
21       IF (rst='1') THEN q <= '0';
22       ELSIF positive_edge(clk) THEN q <= d;
23       END IF;
24    END PROCESS;
25 END my_arch;
```

FUNCIONES

Ejemplo: Función dentro de un paquete.

```
1  ------- Package: ------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ---------------------------------------------
5  PACKAGE my_package IS
6     FUNCTION positive_edge(SIGNAL s: STD_LOGIC) RETURN BOOLEAN;
7  END my_package;
8  ---------------------------------------------
9  PACKAGE BODY my_package IS
10    FUNCTION positive_edge(SIGNAL s: STD_LOGIC)
11       RETURN BOOLEAN IS
12    BEGIN
13       RETURN s'EVENT AND s='1';
14    END positive_edge;
15 END my_package;
16 ---------------------------------------------
```

FUNCIONES

```
1  ------ Main code: ----------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE work.my_package.all;
5  ----------------------------------------------
6  ENTITY dff IS
7     PORT ( d, clk, rst: IN STD_LOGIC;
8            q: OUT STD_LOGIC);
9  END dff;
10 ----------------------------------------------
11 ARCHITECTURE my_arch OF dff IS
12 BEGIN
13    PROCESS (clk, rst)
14    BEGIN
15       IF (rst='1') THEN q <= '0';
16       ELSIF positive_edge(clk) THEN  q <= d;
17       END IF;
18    END PROCESS;
19 END my_arch;
```

FUNCIONES

Ejemplo: Función para llenar una memoria ROM.

```
 1  LIBRARY ieee;
 2  USE ieee.std_logic_1164.ALL;
 3  USE ieee.std_logic_UNSIGNED.ALL;
 4  USE ieee.std_logic_ARITH.ALL;
 5  USE WORK.PAQUETE.ALL;
 6
 7  entity ROM is
 8     PORT( ADDRESS : IN  STD_LOGIC_VECTOR( ADDR_N-1 DOWNTO 0);
 9           DATA : OUT STD_LOGIC_VECTOR( DATA_N-1 DOWNTO 0)
10           );
11  end ROM;
12
13  ARCHITECTURE PROGRAMA OF ROM IS
14  signal MROM : MEMORIA := LLENAR_ROM("DATOS.TXT");
15  BEGIN
16     MEMP : PROCESS( ADDRESS )
17     BEGIN
18        DATA <= MROM( CONV_INTEGER(ADDRESS) );
19     END PROCESS MEMP;
20  END PROGRAMA;
```

FUNCIONES

Ejemplo: Función para llenar una memoria ROM.

```
1  LIBRARY STD;
2  library IEEE;
3
4  USE STD.TEXTIO.ALL;
5  USE ieee.std_logic_TEXTIO.ALL;    --PERMITE USAR STD_LOGIC
6
7  use IEEE.STD_LOGIC_1164.all;
8  use IEEE.STD_LOGIC_UNSIGNED.all;
9
10 package PAQUETE is
11    CONSTANT ADDR_N : INTEGER := 4;
12    CONSTANT DATA_N : INTEGER := 8;
13    TYPE MEMORIA IS ARRAY(0 TO 2**ADDR_N-1) OF STD_LOGIC_VECTOR(DATA_N-1 DOWNTO 0);
14
15 -- function <function_name>  (signal <signal_name> : in <type_declaration>)
16 -- return <type_declaration>;
17    impure function LLENAR_ROM(ARCHIVO : in string) return MEMORIA;
18 end PAQUETE;
```

FUNCIONES

Ejemplo: Función para llenar una memoria ROM.

```
19
20  package body PAQUETE is
21
22     impure function LLENAR_ROM(ARCHIVO : in string) return MEMORIA is
23        FILE ARCH_DATOS       : text is in ARCHIVO;
24        variable LINEA_DATOS : line;
25        variable MEM_ROM      : MEMORIA;
26        VARIABLE I : INTEGER;
27     begin
28      for I in MEMORIA'range loop
29          readline(ARCH_DATOS, LINEA_DATOS);
30          Hread (LINEA_DATOS, MEM_ROM(I));
31      end loop;
32      return MEM_ROM;
33     end function;
34
35  end PAQUETE;
```

PROCEDIMIENTOS

Un procedimiento es muy similar a una función y tiene los mismos propósitos. Sin embargo, un procedimiento puede regresar más de un valor.

El cuerpo de un procedimiento se define como:

```
PROCEDURE procedure_name [<parameter list>] IS
    [declarations]
BEGIN
    (sequential statements)
END procedure_name;
```

GRACIAS POR SU ATENCIÓN

vgarciao@ipn.mx