

Database Programming with PL/SQL

8-3: Passing Parameters

# Practice Activities **Vocabulary**

Identify the vocabulary word for each definition below:



|  |  |
| --- | --- |
| **OUT Parameter** | Returns a value to the caller |
| **IN Parameter** | Provides values for a subprogram to process |
| **Named Notation** | Lists the actual parameters in arbitrary order and uses the association operator ( ‘=>' which is an equal and an arrow together) to associate a named formal parameter with its actual parameter |
| **Combination Notation** | Lists some of the actual parameters as positional (no special operator) and some as named (with the => operator) |
| **Positional notation** | Lists the actual parameters in the same order as the formal parameters |
| **IN OUT Parameter** | Supplies an input value, which may be returned as a modified value |

**Try It / Solve It**

1. Name the three modes for parameters and indicate which mode is the default mode. Which mode cannot be modified inside the procedure?

IN is the default mode, OUT cannot be modified in inside of the procedure and we have also OUT .

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

1. Procedures:
   1. Create a procedure that receives a country\_id as an IN parameter and returns the name and population of that country as OUT parameters. Include an exception handler to trap the NO\_DATA\_FOUND exception if the country does not exist. The procedure should not display the returned values; this will be done in the next step. Name your procedure find\_area\_pop. Save your code.

CREATE OR REPLACE PROCEDURE find\_area\_pop

(p\_country\_id IN countries.country\_id%TYPE,

p\_name OUT countries.country\_name%TYPE,

p\_population OUT countries.population%TYPE)

IS

BEGIN

SELECT country\_name, population INTO p\_name, p\_population

FROM countries

WHERE country\_id = p\_country\_id;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Exception No data found');

END;

* 1. Test your procedure by creating and executing an anonymous block which invokes the procedure and displays the returned OUT values. Save your code. Run the block twice, with country\_ids 2 (Canada) and 10 (does not exist).

DECLARE

p\_country\_name countries.country\_name%TYPE;

p\_population countries.population%TYPE;

BEGIN

find\_area\_pop(2, p\_country\_name, p\_population);

DBMS\_OUTPUT.PUT\_LINE('Name:' || p\_country\_name || ' population: ' || p\_population);

END;

* 1. Retrieve your procedure code and modify it to add a third OUT parameter which is the population density of the country, using the formula: density = (population / area). You will need to modify your SELECT statement to fetch the area column value into a local variable. Save your modified code.

CREATE OR REPLACE PROCEDURE find\_area\_pop

(p\_country\_id IN countries.country\_id%TYPE,

p\_name OUT countries.country\_name%TYPE,

p\_population OUT countries.population%TYPE,

p\_density OUT countries.area%TYPE)

IS

v\_area countries.area%TYPE;

BEGIN

SELECT country\_name, population, area INTO p\_name, p\_population, v\_area

FROM countries

WHERE country\_id = p\_country\_id;

p\_density := (p\_population / v\_area);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Exception No data found');

END;

* 1. Test your modifed procedure using country\_id 2. You will need to modify your calling anonymous block to declare and pass a fourth actual parameter to receive the population density from the procedure. Save your code.

DECLARE

p\_country\_name countries.country\_name%TYPE;

p\_population countries.population%TYPE;

p\_density NUMBER;

BEGIN

find\_area\_pop(2, p\_country\_name, p\_population, p\_density);

DBMS\_OUTPUT.PUT\_LINE('Name:' || p\_country\_name || ' population: ' || p\_population|| ' density:' || p\_density);

END;

1. Create a procedure which accepts an integer as an IN OUT parameter and returns the square of that integer, for example the square of 4 is 16. Save your code. Test your procedure from an anonymous block three times, using integer values 4, 7, and –20 (negative 20).

CREATE OR REPLACE PROCEDURE square

(p\_integer IN OUT INTEGER)

IS

BEGIN

p\_integer := p\_integer \*\*2;

END;

DECLARE v\_integer INTEGER;

BEGIN v\_integer := 4;

square(v\_integer);

DBMS\_OUTPUT.PUT\_LINE('Square is: ' || v\_integer);

END;

1. List the three methods of passing parameters to a procedure.
   1. Retrieve your anonymous block from question 2D and modify its call to find\_area\_pop to pass the four parameters using named notation. Test your block, again using country\_id 2

(Canada). If you have forgotten the p\_ names of the procedure’s formal parameters, how can you refresh your memory?

DECLARE v\_country\_id countries.country\_id%TYPE;

v\_country\_name countries.country\_name%TYPE;

v\_population countries.population%TYPE;

v\_pop\_density NUMBER;

BEGIN v\_country\_id := 2;

find\_area\_pop2(p\_country\_id => v\_country\_id, p\_name => v\_country\_name, p\_population => v\_population, p\_density => v\_pop\_density);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_country\_name ||' Population: ' || v\_population ||' Density: '|| v\_pop\_density);

END;

* 1. Modify the anonymous block from the previous step to pass the FIRST two parameters using named notation and the LAST two using positional notation. Test the block again. What happens?

DECLARE v\_country\_id countries.country\_id%TYPE;

v\_country\_name countries.country\_name%TYPE;

v\_population countries.population%TYPE;

v\_pop\_density NUMBER;

BEGIN v\_country\_id := 2;

find\_area\_pop2(p\_country\_id => v\_country\_id, p\_name => v\_country\_name,v\_population, v\_pop\_density);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_country\_name ||' Population: ' || v\_population ||' Density: '|| v\_pop\_density);

END;

Avem o eroare pt ca parametrii pozitionali trebuie sa se afla intotdeauna inaintea celor prin notatie.

* 1. Correct the problem in the previous step by modifying the anonymous block again to pass the first two parameters using positional notation and the last two using named notation. Test the block again.

DECLARE v\_country\_id countries.country\_id%TYPE;

v\_country\_name countries.country\_name%TYPE;

v\_population countries.population%TYPE;

v\_pop\_density NUMBER;

BEGIN v\_country\_id := 2;

find\_area\_pop2(v\_country\_id, v\_country\_name,p\_population => v\_population,p\_density => v\_pop\_density);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_country\_name ||' Population: ' || v\_population ||' Density: '|| v\_pop\_density);

END;

1. In your own words, describe the purpose of the DEFAULT option for parameters and state the two syntax options for providing the default value in the procedure header.

Scopul optiunii DEFAULT este de a da automat o valaore parametrilor.

1. Find the country\_id of your own country by executing a suitable SELECT…FROM countries…. Then retrieve your find\_area\_pop procedure from question 2C. Modify the code to use your country\_id as a default value for the country\_id IN parameter. Save your code. Then retrieve your anonymous block from question 2D and modify it so that it does NOT pass the country\_id to the procedure. Test the block and check that your country’s details are returned and displayed. If your modified anonymous block does not work, correct it so it will.

SELECT country\_id

FROM countries

WHERE country\_name = 'Romania';

CREATE OR REPLACE PROCEDURE find\_area\_pop

(p\_country\_id IN countries.country\_id%TYPE DEFAULT 40,

p\_name OUT countries.country\_name%TYPE,

p\_population OUT countries.population%TYPE,

p\_density OUT countries.area%TYPE)

IS

v\_area countries.area%TYPE;

BEGIN

SELECT country\_name, population, area INTO p\_name, p\_population, v\_area

FROM countries

WHERE country\_id = p\_country\_id;

p\_density := (p\_population / v\_area);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Exception No data found');

END;

DECLARE v\_country\_id countries.country\_id%TYPE;

v\_country\_name countries.country\_name%TYPE;

v\_population countries.population%TYPE;

v\_pop\_density NUMBER;

BEGIN

v\_country\_id := 2;

find\_area\_pop(p\_name => v\_country\_name, p\_population => v\_population, p\_density => v\_pop\_density);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || v\_country\_name ||' Population: ' || v\_population ||' Density: '|| v\_pop\_density);

END;