Parameter Passing

There are different ways in which parameter data can be passed in and out of procedures and functions. Let us assume that a function Y() is called from another function X(). In this case X is called the “caller function” and Y is called the “called function”. Also, the arguments which X sends to Y are called actual arguments and the parameters of Y are called formal arguments.

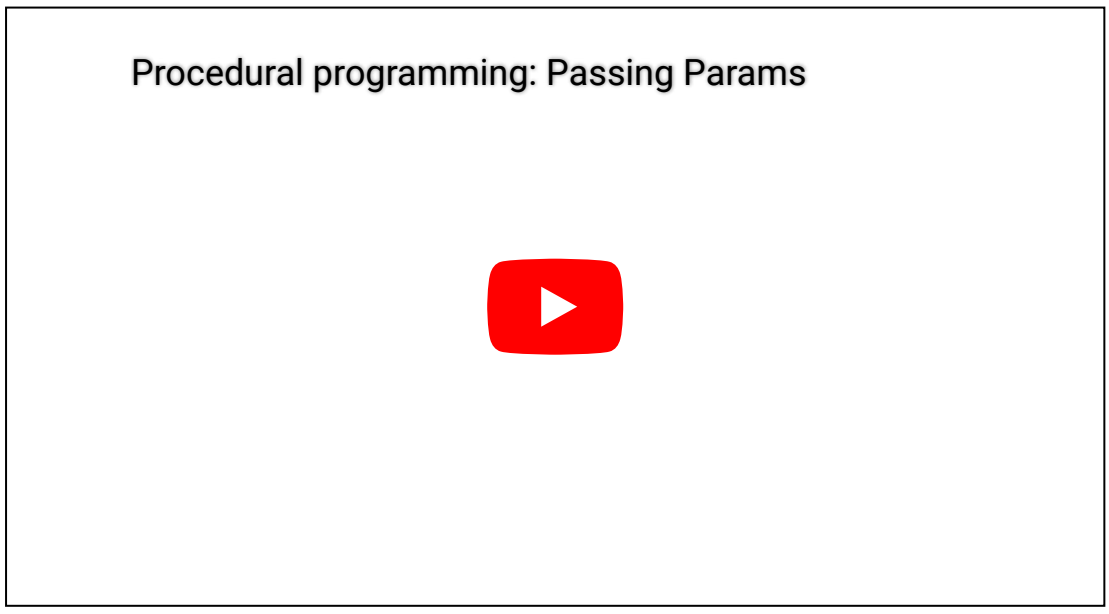
- Formal Parameter : A variable and its type as they appear in the prototype of the function or method.
- Actual Parameter : The variable or expression corresponding to a formal parameter that appears in the function or method call in the calling environment.
- Modes:
    1. IN: Passes info from the caller to called.
    2. OUT: Called writes values in caller.
    3. IN/OUT: Caller tells called value of variable, which may be updated by called.

Parameter Passing

- Important methods of Parameter Passing
- Pass By Value : This method uses in-mode semantics. Changes made to formal parameter do not get transmitted back to the caller. In other words, if you pass a parameter from your

caller function to the called function by value, that means that you are actually taking a copy of the parameter first, and then passing that copy to the function. If you make any changes to the copy of the parameter inside the function, it doesn't change the data held in the original parameter. This method is also referred to as "call by value".

- Pass by reference (aliasing) : This technique uses in/out-mode semantics. Changes made to formal parameter do get transmitted back to the caller through parameters passing. If you pass a parameter from your caller function to a called function by reference, that means that you are actually passing a pointer to the function that points to the original parameter, the original piece of data. If you make any changes to the parameter inside the function, then the original parameter will be changed as well. This method is also referred to as "call by reference".



Parameter Passing examples

Let's use what we've learned !

In our example, we are going to create a procedure that takes a number as a parameter passed by value and it will increment it:

```
PROCEDURE add_num(num:INTEGER) // num is the local variable for the procedure


BEgIN
   num := num+1
   Write(num) // will display  21
END
ALGORITHM test_by_value
VAR
   myData : INTEGER := 20;


BEGIN
   // display the variable
   Write(myData); // will display 20
   // call the procedure add_num
   add_num(myData); // Calling the proc add_num with myData as a parameter
   // display the variable again
   Write(myData) // will display 20
END
```

Now if we manually execute this algorithm, the result will be: 20 21 20

as we can see that the variable myData doesn't change after executing the procedure add_num, that's because the parameter was passed by value

The next type of passing parameters is the passing by reference.

As its name explains, the variable num will point at the same memory allocation of the variable myData so any changes to num variable will affect the myData variable.

Here how it's going to be done:

```
PROCEDURE add_num(VAR num:INTEGER)
BEGIN
   num := num+1
```

```
   Write(num) // will display  21
END
ALGORITHM test_by_value
VAR
   myData : INTEGER := 20;
BEGIN
   // display the variable
   Write(myData); // will display 20
   // call the procedure add_num
   add_num(myData); // now the variable num will point at
   //the same memory allocation of myData variable so any changes of num will affect
   // display the variable again
   Write(myData) // will display 21
END
```

The keyword to differentiate between passing by value and passing by reference is the VAR in the parameter declaration.

‹ Previous next ›