Contents

1 Types of Passing Arguments

1.1 Call-by-value

In the call-by-value evaluation strategy, when a function is called the arguments passed to it are all evaluated beforehand, and copies of the values are placed into newly allocated memory. During execution of the function body, the formal parameters now refer to their associated copied values in this new memory. At the function's conclusion this memory may simply be deallocated.

1.2 Call-by-reference

In the call-by-reference evaluation strategy, when a function is called it receives the memory locations of the arguments passed to it. Therefore, during execution of the function, its formal parameters may refer to and modify local variables of the caller of the function. In this case, no extra memory need be allocated: the formal parameters of the function simply use the same memory used by the passed arguments.

1.3 Call-by-Name

A call-by-name mechanism passes a code block to the function call and the code block is complied, executed and calculated the value.

1.4 Advantages of using Call by value method

- The method doesn't change the original variable, so it is preserving data.
- Whenever a function is called it, never affect the actual contents of the actual arguments.
- Value of actual arguments passed to the formal arguments, so any changes made in the formal argument does not affect the real cases.

1.5 Advantages of using Call by reference method

- The function can change the value of the argument, which is quite useful.
- It does not create duplicate data for holding only one value which helps you to save memory space.
- In this method, there is no copy of the argument made. Therefore it is processed very fast.

1.6 Disadvantages of using Call by value method

- You can't directly change a variable in a function body.

- Sometime argument can be complex expressions
- There are two copies created for the same variable which is not memory efficient.

1.7 Disadvantages of using Call by reference method

- A function taking in a reference need to make sure that the input is non-null.
- A lifetime guarantee is a big issue with references. This is specifically dangerous when working with lambdas and multi-threaded programs.

2 Parameter Passing Modes in C

- Call by value parameter passing only
- Objects can be modified in a function by passing pointers to the object to the function
- Arrays and pointers are exchangeble in C: an array is automatically passed as a pointer to the array

3 Parameter Passing Modes in Fortran

- Call by reference parameter passing only

4 Parameter Passing Modes in Pascal

- Call by value and call by reference parameter passing
- Call by value is similar to C
- Call by reference is indicated by varparameters

```
1 procedureswap(vara:integer, varb:integer)
2 var t;
3 begin
4 t := a; a := b; b := t
5 end
```

5 Parameter Passing Modes in C++

- Call by value and call by reference parameter passing
- Call by value is similar to C
- Call by reference is indicated by using & for formal parameters

```
swap(int &a, int &b)
{
   int t = a;
   a = b;
   b = t;
}
```

- Large objects should be passed by reference instead of by value to increase efficiency
- Arrays are automatically passed by reference (like in C)
- To avoid objects to be modified when passed by reference, const parameters can be used

```
store_record_in_file(const huge_record &r)
{
   ...
}
```

6 Parameter Passing Modes in Java

- Call by value and call by reference/sharing parameter passing
- Java adopts both value and reference models of variables
  - Variables of built-in types are passed by value
  - Class instances are passed by sharing

7 Parameter Passing Modes in Ada

- Call by value, call by result, and call by value/result parameter passing
- Indicated by Ada's in(by value), out(by result), and in out(by value/result) modes for formal parameters

```
procedure shift(a:out integer, b:in out integer,
c:in integer) is
begin
   a := b; b := c;
end shift;
```

- in mode parameters can be read but not written in the subroutine
- out mode parameters can be written but not read in the subroutine
- in out mode parameters can be read and written in the subroutine

8 Parameter Passing Modes in Python

Python uses a mechanism, which is known as "Call-by-Object", sometimes also called "Call by Object Reference" or "Call by Sharing".

9 Parameter Passing Modes in COBOL

The parameters can be passed between programs in two ways :

- By Reference
- By Content

9.1 Call By Reference

If the values of variables in the called program are modified, then their new values will reflect in the calling program. If "BY" clause is not specified, then variables are always passed by reference.

```
CALL sub-prog-name USING variable-1, variable-2.
```

Following example is the MAIN calling program and UTIL is the called program .

Main Program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-STUDENT-ID PIC 9(4) VALUE 1000.
   01 WS-STUDENT-NAME PIC A(15) VALUE 'Tim'.

PROCEDURE DIVISION.
   CALL 'UTIL' USING WS-STUDENT-ID, WS-STUDENT-NAME.
   DISPLAY 'Student Id : ' WS-STUDENT-ID
   DISPLAY 'Student Name : ' WS-STUDENT-NAME
STOP RUN.
```

Call [Util] Program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. UTIL.

DATA DIVISION.
   LINKAGE SECTION.
   01 LS-STUDENT-ID PIC 9(4).
   01 LS-STUDENT-NAME PIC A(15).

PROCEDURE DIVISION USING LS-STUDENT-ID, LS-STUDENT-NAME.
   DISPLAY 'In Called Program'.
   MOVE 1111 TO LS-STUDENT-ID.
EXIT PROGRAM.
```

9.2 Call By Content

If the values of variables in the called program are modified, then their new values will not reflect in the calling program.

```
CALL sub-prog-name USING
BY CONTENT variable-1, BY CONTENT variable-2.
```

Following example is the MAIN calling program and UTIL is the called program :

Main Program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-STUDENT-ID PIC 9(4) VALUE 1000.
   01 WS-STUDENT-NAME PIC A(15) VALUE 'Tim'.

PROCEDURE DIVISION.
   CALL 'UTIL' USING BY CONTENT WS-STUDENT-ID, BY CONTENT
   WS-STUDENT-NAME.
   DISPLAY 'Student Id : ' WS-STUDENT-ID
   DISPLAY 'Student Name : ' WS-STUDENT-NAME
STOP RUN.
```

Call [Util] Program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. UTIL.

DATA DIVISION.
   LINKAGE SECTION.
   01 LS-STUDENT-ID PIC 9(4).
   01 LS-STUDENT-NAME PIC A(15).

PROCEDURE DIVISION USING LS-STUDENT-ID, LS-STUDENT-NAME.
   DISPLAY 'In Called Program'.
   MOVE 1111 TO LS-STUDENT-ID.
EXIT PROGRAM.
```