# PARAMETER PASSING

There are two types of parameters:

1. Formal parameters
2. Actual parameters

Based on these parameters there are various parameter passing methods.
The most common methods are:

## 1. Call by value:

1. This is the simplest method of parameter passing.
2. The actual parameters are evaluated and their r-values are passed to called procedure.
3. The operations on formal parameters do not changes the values of actual parameter.

Example: Languages like C, C++ use actual parameter passing method.

## 2. Call by reference:

1. This method is also called as call by address or call by location.
2. The L-value, the address of actual parameter is passed to the called routines activation record.
3. The value of actual parameters can be changed.
4. The actual parameter should have an L-value.

Example: Reference parameter in C++,PASCAL'S var parameters.

## 3. Copy restore:

1. This method is a hybrid between call by value and call by reference.
2. This is also known as copy-in-copy-out or values result.
3. The calling procedure calculates the value of actual parameters and it is then copied to activation record for the called procedure.
4. During execution of called procedure, the actual parameters value is not affected.
5. If the actual parameters have L-value then at return the value of formal parameter is copied to actual parameter.

Example: In Ada this parameter passing method is used.

## 4. Call by name:

1. This is less popular method of parameter passing.
2. Procedure is treated like macro.
3. The procedure body is substituted for call in caller with actual parameters substituted for formals.
4. The actual parameters can be surrounded by parenthesis to preserve their integrity.
5. The locals name of called procedure and names of calling procedure are distinct.

Example: ALGOL uses call by name method.

## Related Posts:

1. Introduction to Compiler
2. Analysis and synthesis model of compilation
3. Bootstrapping and Porting
4. Lexical Analyzer: Input Buffering

5. Storage Allocation Strategies
6. Type Checking
7. Specification & Recognition of Tokens
8. Front end and back end of the compiler
9. LEX
10. Analysis synthesis model of compilation
11. Data structure in CD
12. Register allocation and assignment
13. Loops in flow graphs
14. Dead code elimination
15. Syntax analysis CFGs
16. L-attribute definition
17. Operator precedence parsing
18. Analysis of syntax directed definition
19. Recursive descent parser
20. Function and operator overloading
21. Storage allocation strategies
22. Equivalence of expression in type checking
23. Storage organization
24. Run time environment
25. Type checking
26. Code generation issue in design of code generator
27. Boolean expression
28. Declaration and assignment in intermediate code generation
29. Code optimization
30. Sources of optimization of basic blocks
31. Loop optimization

32. Global data flow analysis
33. Data flow analysis of structure flow graph (SFG)