

PARAMETER PASSING METHODS

Parameter-passing methods are the ways in which parameters are transmitted to and/or from called subprograms.

Parameter passing depends on model of subprogram. There are two models for parameter passing-

1. Semantics Models of Parameter Passing
2. Implementation Models of Parameter Passing

1. Semantics Models of Parameter Passing: Formal parameters are characterized by one of three distinct semantics models:

1. They can receive data from the corresponding actual parameter, called **in mode**.
2. They can transmit data to the actual parameter, called **out mode**.
3. They can do both, called **inout mode**.

2. Implementation Models of Parameter Passing: This model consists of the following ways of parameter passing.

1. Pass by value
2. Pass by reference
3. Pass-by-Result
4. Pass-by-Value-Result
5. Pass-by-Name

1. Pass by value: Value of actual parameter in read only mode is transmitted to formal

parameters.

2. Pass by reference: Reference/address of actual parameter is transmitted to formal parameters.

3. Pass-by-Result: When a parameter is passed by result, no value is transmitted to the subprogram.

4. Pass-by-Value-Result: Pass-by-value-result is an implementation model for inout-mode parameters. Pass-by-value-result is sometimes called pass-by-copy, because the actual parameter is copied to the formal parameter at subprogram entry and then copied back at subprogram termination.

5. Pass-by-Name: Pass-by-name is an inout-mode parameter transmission method. In it parameters are passed by name. Implementing a pass-by-name parameter requires a subprogram to be passed to the called subprogram to evaluate the address or value of the formal parameter.

Program examples:

Call by value:

```
#include <iostream>
using namespace std;
```

```
void show(int x)
{
    cout<<x<<endl;
}

int main()
{
    int age = 20;
    show(age);
    show(10);

    return 0;
}
```

Call by reference:

```
#include <iostream>
using namespace std;

void show(int *x)
{
    cout<<*x<<endl;
}

int main()
{
    int age = 20;
    show(&age);
}
```

```
    return 0;  
}
```

References:

1. Sebesta, "Concept of programming Language", Pearson Edu
2. Louden, "Programming Languages: Principles & Practices", Cengage Learning
3. Tucker, "Programming Languages: Principles and paradigms", Tata McGraw -Hill.
4. E Horowitz, "Programming Languages", 2nd Edition, Addison Wesley

Related posts:

1. Sequence Control & Expression | PPL
2. PPL:Named Constants
3. Parse Tree | PPL | Prof. Jayesh Umre
4. Basic elements of Prolog
5. Loops | PPL | Prof. Jayesh Umre
6. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
7. Programming Paradigms | PPL | Prof. Jayesh Umre
8. Subprograms Introduction | PPL | Prof. Jayesh Umre
9. Phases of Compiler | PPL | Prof. Jayesh Umre
10. Parse Tree | PPL
11. Influences on Language design | PPL | Prof. Jayesh Umre
12. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
13. Programming Paradigm
14. Influences on Language Design
15. Language Evaluation Criteria

16. OOP in C++ | PPL
17. OOP in C# | PPL
18. OOP in Java | PPL
19. PPL: Abstraction & Encapsulation
20. PPL: Semaphores
21. PPL: Introduction to 4GL
22. PPL: Variable Initialization
23. PPL: Conditional Statements
24. PPL: Array
25. PPL: Strong Typing
26. PPL: Coroutines
27. PPL: Exception Handler in C++
28. PPL: OOP in PHP
29. PPL: Character Data Type
30. PPL: Exceptions
31. PPL: Heap based storage management
32. PPL: Primitive Data Type
33. PPL: Data types
34. Programming Environments | PPL
35. Virtual Machine | PPL
36. PPL: Local referencing environments
37. Generic Subprograms
38. Local referencing environments | PPL | Prof. Jayesh Umre
39. Generic Subprograms | PPL | Prof. Jayesh Umre
40. PPL: Java Threads
41. PPL: Loops
42. PPL: Exception Handling

43. PPL: C# Threads
44. Pointer & Reference Type | PPL
45. Scope and lifetime of variable
46. Design issues for functions
47. Fundamentals of sub-programs
48. Subprograms
49. Design issues of subprogram
50. Garbage Collection
51. Issues in Language Translation
52. PPL Previous years solved papers
53. Type Checking | PPL | Prof. Jayesh Umre
54. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
55. PPL Viva Voce
56. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre
57. Concurrency
58. Basic elements of Prolog
59. Introduction and overview of Logic programming
60. Application of Logic programming
61. PPL: Influences on Language Design
62. Language Evaluation Criteria PPL
63. PPL: Sequence Control & Expression
64. PPL: Programming Environments
65. PPL: Virtual Machine
66. PPL: Programming Paradigm
67. PPL: Pointer & Reference Type
68. try-catch block in C++