FUNDAMENTALS OF SUBPROGRAMS

General characteristics for all subprograms except co-routines:

- Each subprogram has a single entry point.
- The calling program unit is suspended during the execution of the called subprogram.
- Only one subprogram remain in execution at any given time.
- Control always returns to the caller subprogram when the called subprogram execution terminates.

Basic definitions:

- A subprogram call is the explicit request that a specific subprogram be executed.
- A active subprogram is which is in execution.
- A subprogram header, which is the first part of the definition, provides name, type and parameters for subprogram. For example, int show(int a){ }.
- The parameter profile of a subprogram contains the number, order, and types of its formal parameters.
- The protocol of a subprogram is its parameter plus, its return type (if function).
- The parameters in the subprogram header are called formal parameters.
- Values passed in called subprograms parameters is called actual parameters.
- The first actual parameter is bound to the first formal parameter and so forth. Such parameters are called positional parameters.

Principles of Programming Languages:

EasyExamNotes.com covered following topics in these notes.

- Language Evaluation Criteria

- Influences on Language Design
- Language Categories
- Programming Paradigms
- Compilation
- Virtual Machines
- Programming Environments
- Issues in Language Translation
- Parse Tree
- Pointer and Reference type
- Concept of Binding
- Type Checking
- Strong typing
- Sequence control with Expression
- Exception Handling
- Subprograms
- Fundamentals of sub-programs
- Scope and lifetime of variable
- Static and dynamic scope
- Design issues of subprogram and operations
- Local referencing environments
- Parameter passing methods
- Overloaded sub-programs
- Generic sub-programs
- Design issues for functions
- Co routines
- Abstract Data types
- Abstraction and encapsulation
- Static and Stack-Based Storage management

- Garbage Collection
- OOP in C++
- OOP in Java
- OOP in C#
- OOP in PHP
- Concurrency
- Semaphores
- Monitors
- Message passing
- Java threads
- C# threads
- Exception handling
- Exceptions
- Exception Propagation
- Exception handler in C++
- Exception handler in Java
- Introduction and overview of Logic programming
- Basic elements of Prolog
- Application of Logic programming
- Functional programming languages
- Introduction to 4GL

Practicals:

- Memory Implementation of 2D Array.
- Memory Implementation of 3D Array.
- Implementation of pointers in C++.
- Write a program in Java to implement exception handling.
- Write a program in C++ to implement call by value parameter

passing Method.

- Write a program in C++ to implement call by reference parameter passing Method.
- Write a program in Java to implement concurrent execution of a job using threads.
- Implement Inheritance in C#.
- Implement Encapsulation in C#.
- Implement static/compiletime Polymorphism in C#.
- Implement dynamic/runtime Polymorphism in C#.

Previous years solved papers:

- PPL|RGPV|May 2018
- PPL|RGPV|June 2017

A list of Video lectures

- Click here

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. Parameter passing methods
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++