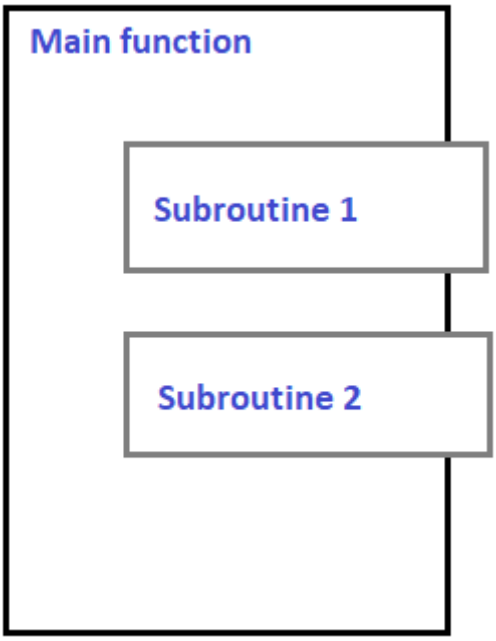
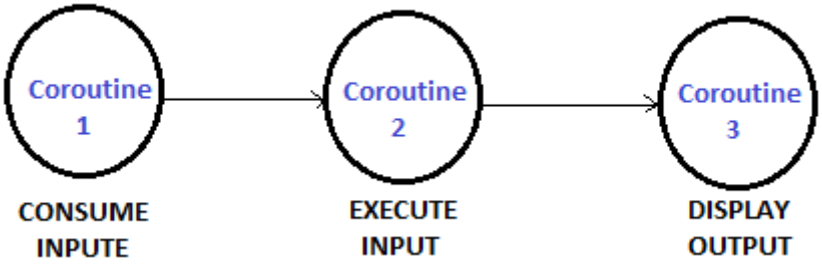
COROUTINES

To understand coroutines first we should know about subroutines.
In computer programming, a subroutine is a sequence of program instructions that perform a specific task. For example a program for addition, subtraction. Subroutines is also known as function.



Coroutines are generalizations of the subroutines.
A subroutine has the same starting point and the same endpoint all the time, while a coroutine has multiple entry points for suspending and resuming execution. Coroutines are cooperative, that means if a coroutine consume input data, another coroutine can consume it, and another coroutine can be used to display the output.
Coroutines are nothing but cooperative functions.



Viva Vice on Coroutines

Q1. Explain what is meant by a recursive subroutine.

Answer =A recursive subroutine is simply one that calls itself either directly or through a chain of calls involving other subroutines.

Q2. Coroutine is just another name for a subroutine. True/False.

Answer = True

Q3. A two pass assembler uses its machine opcode table in the first pass of assembly. True/False.

Answer = True

Q4. Explain what is meant by a recursive subroutine.

Answer =A recursive subroutine is simply one that calls itself either directly or through a chain of calls involving other subroutines.

Q5. How many coroutines can run at a given time?

Answer – Only one coroutine can run at a given time.

Q6. What is coroutine?

Answer = Coroutine is a function that allows pausing its own execution and resuming from the exact same point after a condition is met.

Q7. How to Start Coroutine?

Answer = Coroutine can be start by using the StartCoroutine() function.

Q8. How to Stop Coroutine?

Answer = Coroutine can be stop by using the StopCoroutine() function.

Q9. Which type of method is used to start and stop coroutine?

Answer = It use IEnumerator based method to Start and Stop Coroutine.

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: Exception Handler in C++
27. PPL: OOP in PHP
28. PPL: Character Data Type
29. PPL: Exceptions
30. PPL: Heap based storage management
31. PPL: Primitive Data Type
32. PPL: Data types
33. Programming Environments | PPL
34. Virtual Machine | PPL
35. PPL: Local referencing environments
36. Generic Subprograms
37. Local referencing environments | PPL | Prof. Jayesh Umre
38. Generic Subprograms | PPL | Prof. Jayesh Umre
39. PPL: Java Threads
40. PPL: Loops
41. PPL: Exception Handling
42. PPL: C# Threads
43. Pointer & Reference Type | PPL

44. Scope and lifetime of variable
45. Design issues for functions
46. Parameter passing methods
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++