There are basically four programming paradigms in programming languages:

- 1) Imperative programming languages
- 2) Functional programming languages
- 3) Logic programming languages
- 4) Object-oriented programming languages

Others are:

- 5) Event-Driven programming languages
- 6) Concurrent / parallel programming languages
- 7) Special purpose programming languages

## 1) Imperative programming languages:

It is also known as procedural languages.

An imperative language uses a sequence of statements to determine how to reach a certain goal. Means here you will get the answer how to do a task not what to do.

For example in C,

```
int a = 4;
int b = 5;
int sum = 0;
sum = a + b;
```

From assigning values to each variable to the final addition of those values, each statement

changes the state of the program. Using a sequence of five statements the program shows how to add the numbers 4 and 15.

Example of Imperative language is: C, C++

## 2) Functional programming languages:

It is also known as applicative languages.

Functional programming is a form of declarative programming (mathematical function).

Programming consists of building the function that computes the answer.

There are two types:

- 1. Pure Functional languages: It supports only functional paradigm. Ex. Haskell.
- 2. Impure Functional languages: It supports both functional and imperative paradigm. Ex. Lisp.

Example of Functional language is: Lisp, Python, Haskell.

## 3) Logic programming languages:

This type of languages concentrate on what is the expected outcome for the program instead of how the outcome is achieved.

Logical programming is something like math .Logic program statements express facts and rules about problems.

To understand the rules, lets take an example like, "A is true if B and C is true".

And to understand facts we can say that "A is true".

Example of Logic language is: Prolog.

### 4) Object-oriented programming languages:

Object oriented programming language based on object instead of just functions and procedures. It involves concepts of oops programming languages.

For example: C++, Java.

# 5) Event Driven programming languages:

These languages are execute various operations based on user activities like mouse click and other events.

For example: Visual Basic, Java.

## 6) Concurrent or Parallel programming languages:

These are used to build various distributed programs.

For example: SR, Linda.

## 7) Special purpose programming languages:

These programming languages are used for special task.

For example: SQL, HTML.

#### **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. Influences on Language Design
- 14. Language Evaluation Criteria
- 15. OOP in C++ | PPL
- 16. OOP in C# | PPL
- 17. OOP in Java | PPL
- 18. PPL: Abstraction & Encapsulation
- 19. PPL: Semaphores
- 20. PPL: Introduction to 4GL
- 21. PPL: Variable Initialization
- 22. PPL: Conditional Statements
- 23. PPL: Array
- 24. PPL: Strong Typing
- 25. PPL: Coroutines
- 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++