Prolog is a logic programming language that is primarily used for artificial intelligence and computational linguistics applications. It is based on a formal system called first-order logic.

The basic elements of Prolog include:

1. Facts:

In Prolog, facts are statements about the world that are assumed to be true. Facts consist of predicates and arguments. Predicates represent relationships or properties, and arguments provide the specific details.

Facts are declared using the syntax:

```
predicate(argument1, argument2, ...).
```

For example:

```
parent(sanjay, maya).
```

2. Rules:

Rules in Prolog define relationships or conditions based on other facts or rules. They are used to derive new information or make logical inferences. Rules consist of a head (the conclusion) and a body (the condition). The head specifies the result or conclusion, while the body contains the conditions or requirements for the rule to be true.

Rules are declared using the syntax:

```
head :- condition1, condition2, ...
```

For example:

```
ancestor(X, Y) :- parent(X, Y).
```

3. Queries:

In Prolog, you can ask queries to the system to retrieve information or test relationships. Queries are entered in the form of goals, which are logical statements that you want to prove or find solutions for. You can ask queries using the syntax:

```
?- goal.
```

For example:

```
?- ancestor(sanjay, maya).
```

4. Variables:

Variables are used in Prolog to represent unknown values or placeholders. They start with an

uppercase letter or an underscore (_) and can be used to instantiate values during the execution of a query or rule. Variables allow Prolog to find solutions and perform unification, which is the process of matching values. For example:

```
?- parent(X, maya).
```

Here, X is a variable, and Prolog will attempt to find a value for X that satisfies the query.

5. Logical Operators:

Prolog provides logical operators to combine conditions and goals.

The main logical operators in Prolog are:

- Conjunction (,): Represents logical AND. It requires both conditions to be true.
- Disjunction (;): Represents logical OR. It satisfies the goal if either condition is true.
- Negation (not or +): Represents logical NOT. It negates a condition, making it false.

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. Fundamentals of sub-programs
49. Subprograms
50. Design issues of subprogram
51. Garbage Collection
52. Issues in Language Translation
53. PPL Previous years solved papers
54. Type Checking | PPL | Prof. Jayesh Umre
55. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
56. PPL Viva Voce
57. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre
58. Concurrency
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++