

POINTER AND REFERENCE TYPE

Pointer: A pointer is a variable which is used to store the address of another variable.

It is used to allocate memory dynamically at run time.

The pointer variable might be int, float, char, double, short etc.

Pointer syntax:

```
int *a;
```

Some points to remember about pointer:

- Pointer variable stores the address of the variable.
- The content of the pointer always be a whole number i.e. address.
- Pointer is initialized to null, i.e. `int *a = null`.
- The value of null pointer is 0.
- & symbol is used to get the address of the variable.
- * symbol is used to get the value of the variable whose address pointer is holding.
- Two pointers can be subtracted to know number of elements between these two pointers.
- Pointer addition, multiplication, division are not allowed.

Pointer example:

```
int main()  
{  
    int *p, q;  
    q = 10;  
    p = &q;  
    return *p;  
}
```

What are the design issues with pointer type?

The design issues for pointer types are

- what the scope and lifetime of a pointer variable are,
- what the lifetime of a heap-dynamic variable (the value a pointer references) is,
- if pointers are restricted as to the type of value to which they can point,
- if pointers are used for dynamic storage management, indirect addressing, or both, and
- if the language should support pointer types, reference type, or both.

MCQs on Pointer and Reference type

Q1. Pointer is special kind of variable which is used to store _____ of the variable.

- a. Value
- b. Address
- c. Variable Name

Q2. Pointer variable is declared using preceding _____ sign.

- a. *
- b. %
- c. &

Q3. Address stored in the pointer variable is of type _____.

- a. Integer
- b. Character
- c. Array

Q4. In order to fetch the address of the variable we write preceding _____ sign before variable name.

- a. Comma
- b. Ampersand
- c. Asterisk

Q5. "&" is called as _____ in pointer concept.

- a. Address Operator
- b. None of these
- c. Conditional Operator

Q6. What do the following declaration signify?

`char *arr[4];`

- a. arr is a array of 4 character pointers.
- b. arr is a array of function pointer.
- c. arr is a array of characters.

MCQs Answers

- Q1. (b)
- Q2. (a)
- Q3. (a)
- Q4. (b)
- Q5. (a)
- Q6. (a)

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. Loudon, "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. 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. Basic elements of Prolog
- 60. Introduction and overview of Logic programming
- 61. Application of Logic programming
- 62. PPL: Influences on Language Design
- 63. Language Evaluation Criteria PPL
- 64. PPL: Sequence Control & Expression
- 65. PPL: Programming Environments
- 66. PPL: Virtual Machine
- 67. PPL: Programming Paradigm
- 68. try-catch block in C++