

Object Oriented programming in C++:

Object Oriented programming is a programming style which is associated with the concepts like class, object, Inheritance, Encapsulation, Abstraction, Polymorphism.

Class: A class is a collection of method and variables.

We can define a class using the class keyword and the class body enclosed by a pair of curly braces, as shown in the following example:

```
public class A
{
}
```

Inheritance: Inheritance feature allows code reusability when a class includes property of another class. In C++ ":" is used to represent inheritance, as shown in the following example:

```
public class Papa {
}
```

```
class Child : public Papa {
}
```

Objects: Object is a component of a program that knows how to perform certain actions and how to interact with other elements of the program, as shown in the following example:

```
Rectangle Rect;
```

Here, Rectangle is a class, Rect is an object of class Rectangle.

Abstraction: Abstraction can be achieved using abstract classes in C++. Abstract classes contain abstract methods, which are implemented by the derived class.

Encapsulation: Encapsulation means bundling of data and methods within one unit, e.g., a class in C++. Encapsulation enables a programmer to implement the desired level of abstraction.

Polymorphism: Polymorphism means overloading and overriding, as shown in the following example:

Polymorphism overloading example:

```
public class OverLoading
{
    void sum(int a, int b)
    {
        cout <<a + b;
    }
    void sum(int a, int b, int c)
```

```
{  
    cout << a + b + c ;  
}
```

Polymorphism overriding example:

```
public class Parent  
{  
    void Show()  
    {  
        cout << "Welcome";  
    }  
}  
  
public class Child : Parent  
{  
    void Show()  
    {  
        cout << "Swagatam";  
    }  
}
```

MCQs on OOPs in C++

Q1. Which of the following type of class allows only one object of it to be created?

- A. Virtual class
- B. Abstract class
- C. Singleton class

Q2. Which of the following is not a type of constructor?

- A. Copy constructor

- B. Friend constructor
- C. Default constructor

Q3. Which of the following is not the member of class?

- A. Static function
- B. Friend function
- C. Const function

Q4. Which of the following concepts means determining at runtime what method to invoke?

- A. Data hiding
- B. Dynamic Typing
- C. Dynamic binding

MCQs Answers

- Q1. (c)
- Q2. (b)
- Q3. (b)
- Q4. (c)

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 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++