Introduction to Object Oriented Programming in C++

Complete Notes for BCA / UG Level Students
Hindi + English Mix Explanation with Examples

1. Object Oriented Programming क्या है?

Object Oriented Programming (OOP) एक programming approach है जिसमें program को **classes** और **objects** के रूप में design किया जाता है। इसमें data और functions को एक unit में bind किया जाता है।

Simple meaning: Real world चीजों जैसे Student, Employee, Book, Account आदि को program में object की तरह represent करना OOP कहलाता है।

```
class Student {
public:
    int rollNo;
    string name;

    void display() {
        cout << rollNo << " " << name;
    }
};
```

यहाँ `Student` एक class है, `rollNo` और `name` data members हैं, और `display()` member function है।

2. Procedural Programming vs Object Oriented Programming

Procedural Programming में program mainly functions/procedures पर based होता है। Data और functions अलग-अलग रखे जाते हैं। Example: C language.

Object Oriented Programming में data और functions को एक class में रखा जाता है। Objects के through program operate होता है। Example: C++.

Basis	Procedural Programming	Object Oriented Programming
Main Focus	Functions / procedures	Objects and classes
Approach	Top-down approach	Bottom-up approach
Data Security	कम data security	Access specifiers की वजह से अधिक security
Code Reuse	Comparatively difficult	Inheritance and classes से easy
Suitable For	Small programs	Large and complex projects
Examples	C, Pascal	C++, Java, Python

3. Features of C++

- **Simple and Powerful:** C++ का syntax C language से मिलता है, इसलिए C learners के लिए आसान है।
- **Object Oriented:** Class, Object, Inheritance, Polymorphism, Encapsulation और Abstraction support करता है।
- **Middle Level Language:** High-level features और low-level memory control दोनों provide करता है।
- **Fast Execution:** C++ compiled language है, इसलिए performance अच्छी होती है।
- **Rich Library:** Standard Template Library (STL) में vector, stack, queue, map आदि मिलते हैं।
- **Dynamic Memory Management:** `new` और `delete` operators use होते हैं।
- **Function and Operator Overloading:** Same function/operator को different meanings दे सकते हैं।
- **Platform Dependent:** Compiled executable generally particular system/environment पर depend करता है।

```
int *p = new int;   // dynamic memory allocation
*p = 50;
delete p;          // memory release
```

4. Structure of C++ Program

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello C++";
    return 0;
}
```

- `#include <iostream>`: Input-output के लिए header file.
- `using namespace std;`: `std::cout` की जगह direct `cout` लिखने के लिए.
- `int main()`: Program execution यहीं से start होता है.
- `cout`: Output display करने के लिए.
- `return 0;`: Successful program termination show करता है।

5. Data Types in C++

Data type बताता है कि variable में किस प्रकार का data store होगा।

Data Type	Meaning	Example
int	Integer number	10, 25, -5
float	Decimal number	75.5
double	Large decimal value	123.45678
char	Single character	'A'
bool	True/False value	true, false
string	Text data	"Rahul"

```
#include <iostream>
using namespace std;

int main() {
    int age = 20;
    float marks = 75.5;
    char grade = 'A';
    bool pass = true;
    string name = "Amit";

    cout << name << endl;
    cout << age << endl;
    cout << marks << endl;
    cout << grade << endl;
    cout << pass << endl;

    return 0;
}
```

6. Variables in C++

Variable memory location का name होता है जहाँ value store होती है।

```
int age = 20;
```

यहाँ `int` data type है, `age` variable name है और `20` value है।

Rules for Variable Names

- Variable name letter या underscore से start होना चाहिए।
- Space allowed नहीं है।
- C++ keywords जैसे `class`, `int`, `return` variable name नहीं हो सकते।
- C++ case-sensitive है: `Age` और `age` अलग variables हैं।

7. Constants in C++

Constant ऐसी value है जो program execution के दौरान change नहीं होती।

```
const float PI = 3.14;
```

```
#include <iostream>
using namespace std;

int main() {
    const float PI = 3.14;
    float radius = 5;
    float area = PI * radius * radius;

    cout << "Area = " << area;
    return 0;
}
```

8. Input and Output in C++

C++ में input लेने के लिए `cin` और output देने के लिए `cout` use होता है।

```
#include <iostream>
using namespace std;

int main() {
    int age;
    cout << "Enter your age: ";
    cin >> age;
    cout << "Your age is: " << age;
    return 0;
}
```

9. Concept of Class

Class एक blueprint/template है जिससे objects बनाए जाते हैं। Class में data members और member functions होते हैं।

```
class Student {
public:
    int rollNo;
    string name;

    void display() {
        cout << rollNo << " " << name;
    }
};
```

- `Student` : class name

- `rollNo`, `name`: data members
- `display()`: member function

10. Concept of Object

Object class का real instance होता है। Object के through class के public members को access किया जाता है।

```
Student s1;   // s1 is object of Student class
```

```
#include <iostream>
using namespace std;

class Student {
public:
    int rollNo;
    string name;

    void display() {
        cout << "Roll No: " << rollNo << endl;
        cout << "Name: " << name << endl;
    }
};

int main() {
    Student s1;
    s1.rollNo = 101;
    s1.name = "Rahul";
    s1.display();
    return 0;
}
```

Output:
`Roll No: 101`
`Name: Rahul`

11. Structure and Class in C++

C++ में structure और class दोनों data और functions को group कर सकते हैं, लेकिन default access अलग होता है।

```
struct Student {
    int rollNo;
    string name;
};

class StudentClass {
    int rollNo;
    string name;
};
```

Basis	Structure	Class
Default Access	Public	Private
Main Use	Data grouping	Data + functions + security
OOP Support	Limited use	Full OOP implementation
Encapsulation	Less strict	Strong

12. Access Specifiers in C++

Access specifiers decide करते हैं कि class के members कहाँ से access हो सकते हैं। C++ में तीन main access specifiers होते हैं: **private, public, protected**.

12.1 Private Access Specifier

Private members केवल same class के अंदर access किए जा सकते हैं। Class के बाहर direct access नहीं होता।

```
#include <iostream>
using namespace std;

class Student {
private:
    int rollNo;

public:
    void setRollNo(int r) {
        rollNo = r;
    }

    void display() {
        cout << "Roll No: " << rollNo;
    }
};

int main() {
    Student s1;
    s1.setRollNo(101);
    s1.display();
    return 0;
}
```

`s1.rollNo = 101;` लिखने पर error आएगी क्योंकि `rollNo` private है।

12.2 Public Access Specifier

Public members class के बाहर भी object के through access किए जा सकते हैं।

```
#include <iostream>
using namespace std;

class Student {
public:
    int rollNo;
    string name;
};

int main() {
    Student s1;
    s1.rollNo = 101;
    s1.name = "Amit";
    cout << s1.rollNo << " " << s1.name;
    return 0;
}
```

12.3 Protected Access Specifier

Protected members same class और derived class में access हो सकते हैं। Outside class से direct access नहीं होता।

```
#include <iostream>
using namespace std;

class Person {
protected:
    string name;
};

class Student : public Person {
public:
    void setName(string n) {
        name = n;
    }

    void display() {
        cout << "Name: " << name;
    }
};

int main() {
    Student s1;
    s1.setName("Neha");
    s1.display();
    return 0;
}
```

Access Specifier	Same Class	Derived Class	Outside Class
private	Yes	No	No
protected	Yes	Yes	No
public	Yes	Yes	Yes

13. Member Functions

Class के अंदर defined functions को **member functions** कहते हैं। Member function object के through call होता है।

```
class Student {
public:
    void display() {
        cout << "Hello";
    }
};

int main() {
    Student s1;
    s1.display();
}
```

14. Inline Functions

Inline function ऐसा function है जिसका code function call की जगह directly expand हो जाता है। इससे function call overhead कम हो सकता है।

```
inline return_type function_name(parameters) {
    // code
}
```

```
#include <iostream>
using namespace std;

inline int square(int n) {
    return n * n;
}

int main() {
    cout << square(5);
    return 0;
}
```

Output: `25`

Inline Function inside Class

Class के अंदर defined small function generally inline माना जाता है।

```
class Calculator {
public:
    int square(int n) {
        return n * n;
    }
};
```

Advantages of Inline Function

- Function call overhead कम होता है।
- Small functions के लिए execution fast हो सकता है।
- Program readability बनी रहती है।

Disadvantages of Inline Function

- Large function inline करने से program size बढ़ सकता है।
- Compiler inline request को ignore कर सकता है।
- Recursive functions generally inline नहीं होते।

15. Static Data Member

Static data member class का ऐसा member होता है जिसकी only one copy पूरी class के लिए shared होती है। Normal data member हर object के लिए अलग copy बनाता है, लेकिन static data member की single copy होती है।

```
class ClassName {
public:
    static int count;
};

int ClassName::count = 0;
```

```
#include <iostream>
using namespace std;

class Student {
public:
    static int count;

    Student() {
        count++;
    }

    void displayCount() {
        cout << "Total Objects: " << count << endl;
    }
};

int Student::count = 0;

int main() {
    Student s1;
    Student s2;
    Student s3;

    s1.displayCount();
    return 0;
}
```

Output: `Total Objects: 3`

16. Static Member Function

Static member function object के बिना class name से call किया जा सकता है। यह directly केवल static data members को access कर सकता है।

```
class ClassName {
public:
    static void functionName() {
        // code
    }
};

ClassName::functionName();
```

```
#include <iostream>
using namespace std;

class Student {
private:
    static int count;

public:
    Student() {
        count++;
    }

    static void showCount() {
        cout << "Total Students: " << count;
    }
};

int Student::count = 0;

int main() {
    Student s1, s2, s3;
    Student::showCount();
    return 0;
}
```

Output: `Total Students: 3`

Basis	Normal Data Member	Static Data Member
Memory	हर object के लिए अलग copy	पूरी class के लिए one copy
Access	Object से	Object या class name से
Initialization	Object create होने पर	Class के बाहर define/initialize
Use	Individual object data	Common shared data

17. Complete Example: Class, Object, Access Specifier, Inline and Static

```
#include <iostream>
using namespace std;

class Student {
private:
    int rollNo;
    string name;
    static int count;

public:
    void setData(int r, string n) {
        rollNo = r;
        name = n;
        count++;
    }

    inline void display() {
        cout << "Roll No: " << rollNo << endl;
        cout << "Name: " << name << endl;
    }

    static void showCount() {
        cout << "Total Students: " << count << endl;
    }
};

int Student::count = 0;

int main() {
    Student s1, s2;

    s1.setData(101, "Rahul");
    s2.setData(102, "Neha");

    s1.display();
    s2.display();
    Student::showCount();

    return 0;
}
```

18. Important Exam Definitions

Term	English Definition	Hindi Meaning
OOP	Programming approach based on classes and objects.	Class और object based programming technique.
Class	Blueprint for creating objects.	Object बनाने का template/blueprint.
Object	Instance of a class.	Class का real example/instance.
Access Specifier	Controls visibility of class members.	Members कहाँ से access होंगे यह बताता है।
Inline Function	Function expanded at function call point.	Function call की जगह code expand होता है।
Static Data Member	Member shared by all objects of a class.	सभी objects द्वारा shared single copy.
Static Member Function	Function callable using class name.	Object के बिना class name से call होने वाला function.

19. Important Short Questions for BCA Exam

Q1. What is OOP?
OOP is a programming approach based on objects and classes. It combines data and functions into a single unit.

Q2. What is class?
Class is a user-defined data type that contains data members and member functions.

Q3. What is object?
Object is an instance of a class.

Q4. What are access specifiers?
Access specifiers control the visibility of class members. They are private, public and protected.

Q5. What is private member?
Private member can be accessed only inside the same class.

Q6. What is public member?
Public member can be accessed from inside and outside the class.

Q7. What is protected member?
Protected member can be accessed inside the same class and derived class.

Q8. What is inline function?
Inline function is a function whose code is expanded at the function call point.

Q9. What is static data member?
Static data member is a class member shared by all objects.

Q10. What is static member function?
Static member function is a class function that can access only static members directly.

20. Important Long Answer: Explain Class and Object with Example

Class is a blueprint or template used to create objects. It contains data members and member functions. Object is the real instance of a class. Through object we can access public members of the class.

```
#include <iostream>
using namespace std;

class Employee {
public:
    int id;
    string name;

    void display() {
        cout << "ID: " << id << endl;
        cout << "Name: " << name << endl;
    }
};

int main() {
    Employee e1;
    e1.id = 101;
    e1.name = "Amit";
    e1.display();
    return 0;
}
```

In this example, `Employee` is a class and `e1` is an object.

21. Important Long Answer: Explain Access Specifiers

Access specifiers are used to control the access of class members. C++ provides three access specifiers: private, public and protected. Private members are accessible only inside the class. Public members are accessible from inside and outside the class. Protected members are accessible inside the class and its derived class.

```
#include <iostream>
using namespace std;

class Demo {
private:
    int a;

protected:
    int b;

public:
    int c;

    void setData() {
        a = 10;
        b = 20;
        c = 30;
    }

    void display() {
        cout << a << " " << b << " " << c;
    }
};

int main() {
    Demo d;
    d.setData();
    d.display();
    return 0;
}
```

22. Important MCQs

1. OOP is based on which concept?
A. Function
B. Object
C. Procedure
D. Loop
Answer: B. Object

2. Which is the basic unit of OOP?
A. Function
B. Class
C. Object
D. Array
Answer: C. Object

3. Class is a _____.
A. Function
B. Variable
C. Blueprint

D. Loop
Answer: C. Blueprint

4. Object is an instance of _____.
A. Function
B. Class
C. Array
D. Pointer
Answer: B. Class

5. Default access specifier of class is _____.
A. Public
B. Private
C. Protected
D. Static
Answer: B. Private

6. Default access specifier of structure is _____.
A. Public
B. Private
C. Protected
D. None
Answer: A. Public

7. Which access specifier is accessible everywhere?
A. Private
B. Protected
C. Public
D. Static
Answer: C. Public

8. Static data member is shared by _____.
A. One object only
B. All objects
C. Only main function
D. Only private members
Answer: B. All objects

9. Inline function is used to reduce _____.
A. Data size
B. Function call overhead
C. Variable size
D. Class size
Answer: B. Function call overhead

10. Static member function can directly access _____.
A. Only non-static members
B. Only static members

C. All local variables
D. Private functions only
Answer: B. Only static members

23. Quick Revision Points

- C++ supports both procedural and object-oriented programming.
- OOP is based on class and object.
- Class is a blueprint and object is an instance of class.
- Data members store data and member functions perform operations.
- Private members are accessible only inside class.
- Public members are accessible outside class also.
- Protected members are useful in inheritance.
- Inline functions reduce function calling overhead.
- Static data member is common for all objects.
- Static member function can be called using class name.

Best One-Line Summary:
C++ Object Oriented Programming uses classes and objects to combine data and functions together, provide data security using access specifiers, improve performance using inline functions, and share common data using static data members.