Unit III: Inheritance and Polymorphism in C++

Complete UG / BCA Level Descriptive Notes - Hindi + English Mix

Syllabus Covered: Definition and concept of base and derived class; types of inheritance - single, multilevel, multiple, hierarchical and hybrid; polymorphism - compile-time and runtime; function overloading, operator overloading, constructor overloading, virtual function, pure virtual function, inline function, friend function and friend class.

1. Inheritance in C++

1.1 Definition

Inheritance C++ का important OOP concept है जिसमें एक class दूसरी class की properties और member functions को acquire करती है. It allows a new class to reuse and extend an existing class.

Simple definition: Inheritance is the process by which one class acquires the data members and member functions of another class.

1.2 Need / Importance of Inheritance

1. Code reusability: existing code को दुबारा लिखने की आवश्यकता नहीं होती.
2. Extensibility: पुरानी class में बिना change किए new features derived class में add कर सकते हैं.
3. Maintainability: common code एक जगह रहता है, इसलिए maintenance आसान होता है.
4. Real-world relationship: is-a relationship को represent करता है, जैसे Student is a Person.
5. Program organization: large software को logical classes में divide करता है.

2. Base Class and Derived Class

Base class वह class होती है जिसके members inherit किए जाते हैं. इसे parent class या super class भी कहते हैं. Derived class वह class होती है जो base class से members inherit करती है. इसे child class या subclass कहते हैं.

Term	Meaning	Example
Base Class	Class whose properties are inherited	Person
Derived Class	Class that inherits properties	Student : public Person

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

class Person {              // Base class
public:
    string name;
    int age;
    void showPerson() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};

class Student : public Person {   // Derived class
public:
    int rollNo;
    void showStudent() {
```

```
        cout << "Roll No: " << rollNo << endl;
    }
};

int main() {
    Student s1;
    s1.name = "Amit";
    s1.age = 20;
    s1.rollNo = 101;
    s1.showPerson();
    s1.showStudent();
    return 0;
}
```

Output: Name: Amit, Age: 20, Roll No: 101. यहाँ Student object base class Person के public members को access कर रहा है.

3. Syntax of Inheritance

```
class DerivedClassName : access_specifier BaseClassName {
    // members of derived class
};

class Student : public Person {
    // Student class members
};
```

Inheritance में access specifier decide करता है कि base class के public/protected members derived class में किस visibility से आएंगे.

Base Member	Public Inheritance	Protected Inheritance	Private Inheritance
public	public	protected	private
protected	protected	protected	private
private	not directly accessible	not directly accessible	not directly accessible

4. Types of Inheritance in C++

C++ में inheritance के main types हैं: Single, Multilevel, Multiple, Hierarchical और Hybrid inheritance.

4.1 Single Inheritance

जब एक derived class केवल एक base class से inherit करती है तो इसे Single Inheritance कहते हैं. Diagram: Base Class -> Derived Class.

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

class Teacher {
public:
    string subject;
    void showSubject() {
        cout << "Subject: " << subject << endl;
    }
};

class Student : public Teacher {
```

```
public:
    string name;
    void showName() {
        cout << "Student Name: " << name << endl;
    }
};

int main() {
    Student s;
    s.name = "Rahul";
    s.subject = "C++";
    s.showName();
    s.showSubject();
    return 0;
}
```

4.2 Multilevel Inheritance

जब inheritance chain form होती है, अर्थात एक class दूसरी class से और तीसरी class उस derived class से inherit करती है, उसे Multilevel Inheritance कहते हैं. Diagram: Person -> Student -> Result.

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

class Person {
public:
    string name;
    void showName() { cout << "Name: " << name << endl; }
};

class Student : public Person {
public:
    int rollNo;
    void showRollNo() { cout << "Roll No: " << rollNo << endl; }
};

class Result : public Student {
public:
    int marks;
    void showMarks() { cout << "Marks: " << marks << endl; }
};

int main() {
    Result r;
    r.name = "Neha";
    r.rollNo = 102;
    r.marks = 85;
    r.showName();
    r.showRollNo();
    r.showMarks();
    return 0;
}
```

Result class, Student और Person दोनों के accessible members को use कर सकती है.

4.3 Multiple Inheritance

जब एक derived class दो या अधिक base classes से inherit करती है तो इसे Multiple Inheritance कहते हैं. Diagram: Base1 + Base2 -> Derived.

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

class InternalMarks {
public:
    int internal;
    void getInternal() { internal = 25; }
};

class ExternalMarks {
public:
    int external;
    void getExternal() { external = 70; }
};

class Result : public InternalMarks, public ExternalMarks {
public:
    void showResult() {
        cout << "Internal Marks: " << internal << endl;
        cout << "External Marks: " << external << endl;
        cout << "Total Marks: " << internal + external << endl;
    }
};

int main() {
    Result r;
    r.getInternal();
    r.getExternal();
    r.showResult();
    return 0;
}
```

4.4 Hierarchical Inheritance

जब एक base class से कई derived classes inherit करती हैं तो इसे Hierarchical Inheritance कहते हैं. Diagram: Shape -> Rectangle and Triangle.

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

class Shape {
public:
    int x, y;
    void getData(int a, int b) {
        x = a;
        y = b;
    }
};

class Rectangle : public Shape {
public:
    void area() { cout << "Area of Rectangle: " << x * y << endl; }
};

class Triangle : public Shape {
public:
    void area() { cout << "Area of Triangle: " << 0.5 * x * y << endl; }
};

int main() {
    Rectangle r;
    Triangle t;
```

```
    r.getData(10, 5);
    t.getData(10, 5);
    r.area();
    t.area();
    return 0;
}
```

4.5 Hybrid Inheritance

Hybrid Inheritance दो या अधिक inheritance types का combination है. जैसे multilevel + multiple या hierarchical + multiple inheritance.

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

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

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

class Sports {
public:
    int sportsMarks;
    void setSportsMarks(int sm) { sportsMarks = sm; }
};

class Result : public Student, public Sports {
public:
    int academicMarks;
    void setAcademicMarks(int am) { academicMarks = am; }
    void display() {
        cout << "Name: " << name << endl;
        cout << "Roll No: " << rollNo << endl;
        cout << "Total: " << academicMarks + sportsMarks << endl;
    }
};

int main() {
    Result r;
    r.setName("Amit");
    r.setRollNo(101);
    r.setAcademicMarks(80);
    r.setSportsMarks(15);
    r.display();
    return 0;
}
```

5. Ambiguity in Multiple Inheritance

Multiple inheritance में अगर दो base classes में same name का function हो, तो derived class object से function call करते समय ambiguity हो सकती है. इसे scope resolution operator (::) से solve किया जाता है.

```
class A {
public:
    void show() { cout << "Class A" << endl; }
};

class B {
public:
    void show() { cout << "Class B" << endl; }
};

class C : public A, public B { };

int main() {
    C obj;
    obj.A::show();   // calls A version
    obj.B::show();   // calls B version
}
```

6. Polymorphism in C++

6.1 Definition

Polymorphism का अर्थ है many forms. C++ में same function name या same operator different situations में different behavior show कर सकता है.

Simple definition: Polymorphism is an OOP concept in which one name can have many forms.

6.2 Types of Polymorphism

Type	Also Called	Achieved By	Decision Time
Compile-time Polymorphism	Static/Early Binding	Function overloading, operator overloading, constructor overloading	Compile time
Runtime Polymorphism	Dynamic/Late Binding	Virtual function	Runtime

7. Compile-Time Polymorphism

Compile-time polymorphism में function call का decision compilation के समय हो जाता है. It is faster because binding is done before execution.

7.1 Function Overloading

Same scope में same name के multiple functions जिनके parameter list different हों, Function Overloading कहलाते हैं. Return type alone cannot overload a function.

- Number of parameters different हो सकते हैं.
- Type of parameters different हो सकते हैं.
- Order of parameters different हो सकता है.
- Only return type difference is not valid overloading.

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

class Calculator {
public:
```

```
    int add(int a, int b) { return a + b; }
    float add(float a, float b) { return a + b; }
    int add(int a, int b, int c) { return a + b + c; }
};

int main() {
    Calculator c;
    cout << c.add(10, 20) << endl;
    cout << c.add(5.5f, 2.5f) << endl;
    cout << c.add(1, 2, 3) << endl;
    return 0;
}
```

7.2 Operator Overloading

Operator overloading में existing operator जैसे +, -, *, == आदि को user-defined objects के लिए नया meaning दिया जाता है.

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

class Number {
private:
    int value;
public:
    Number(int v) { value = v; }
    Number operator + (Number obj) {
        Number temp(0);
        temp.value = value + obj.value;
        return temp;
    }
    void display() { cout << "Value: " << value << endl; }
};

int main() {
    Number n1(10), n2(20);
    Number n3 = n1 + n2;
    n3.display();
    return 0;
}
```

Operators that cannot be overloaded: ::, ., .*, ?:, sizeof.

7.3 Constructor Overloading

जब एक class में same name के multiple constructors होते हैं लेकिन arguments different होते हैं, तो इसे Constructor Overloading कहते हैं.

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

class Student {
private:
    string name;
    int age;
public:
    Student() { name = "Unknown"; age = 0; }
    Student(string n) { name = n; age = 18; }
    Student(string n, int a) { name = n; age = a; }
    void display() { cout << name << " " << age << endl; }
};
```

```
int main() {
    Student s1;
    Student s2("Rahul");
    Student s3("Neha", 21);
    s1.display();
    s2.display();
    s3.display();
    return 0;
}
```

8. Runtime Polymorphism

Runtime polymorphism में function call का decision program execution के दौरान होता है. यह virtual functions के द्वारा achieve होता है.

8.1 Function Overriding

जब base class और derived class दोनों में same name और same parameters वाला function हो, तो derived class function base class function को override करता है.

```
class Animal {
public:
    void sound() { cout << "Animal makes sound" << endl; }
};
class Dog : public Animal {
public:
    void sound() { cout << "Dog barks" << endl; }
};
```

8.2 Virtual Function

Virtual function base class का member function होता है जिसे derived class में override किया जा सकता है. Base class pointer जब derived class object को point करता है, तब virtual function runtime पर correct derived version call करता है.

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

class Animal {
public:
    virtual void sound() {
        cout << "Animal makes sound" << endl;
    }
};

class Dog : public Animal {
public:
    void sound() {
        cout << "Dog barks" << endl;
    }
};

int main() {
    Animal *ptr;
    Dog d;
    ptr = &d;
    ptr->sound();       // Dog's sound because function is virtual
    return 0;
```

```
}
```

Output: Dog barks. अगर virtual keyword न हो तो base class pointer base class function को call करेगा.

8.3 Pure Virtual Function and Abstract Class

Pure virtual function का base class में कोई implementation नहीं होता और इसे = 0 से declare किया जाता है. जिस class में कम से कम एक pure virtual function हो, उसे Abstract Class कहते हैं. Abstract class का object create नहीं किया जा सकता.

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

class Shape {
public:
    virtual void area() = 0;   // pure virtual function
};

class Rectangle : public Shape {
public:
    void area() { cout << "Area of Rectangle" << endl; }
};

class Circle : public Shape {
public:
    void area() { cout << "Area of Circle" << endl; }
};

int main() {
    Rectangle r;
    Circle c;
    r.area();
    c.area();
    return 0;
}
```

Basis	Virtual Function	Pure Virtual Function
Body in base class	May have body	No body in base class
Syntax	virtual void show() { }	virtual void show() = 0;
Override	Optional	Compulsory in concrete derived class
Class effect	May remain normal class	Makes class abstract
Object creation	Possible if no pure virtual	Abstract class object not possible

9. Inline Function

Inline function ऐसा function है जिसका code function call की जगह expand करने का request compiler को दिया जाता है. It reduces function call overhead for small functions.

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

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

int main() {
    cout << "Square: " << square(5);
```

```
    return 0;
}
```

Advantages	Disadvantages
Function call overhead कम होता है	Large functions inline करने से code size बढ़ सकता है
Small functions fast हो सकते हैं	Compiler inline request ignore कर सकता है
Readability अच्छी रहती है	Recursive functions generally inline नहीं होते

10. Friend Function

Friend function class का member नहीं होता, लेकिन friend declare करने पर वह class के private और protected members को access कर सकता है.

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

class Student {
private:
    int marks;
public:
    Student() { marks = 90; }
    friend void showMarks(Student s);
};

void showMarks(Student s) {
    cout << "Marks: " << s.marks << endl;
}

int main() {
    Student s1;
    showMarks(s1);
    return 0;
}
```

- Friend function member function नहीं होता.
- इसे object के through call नहीं किया जाता.
- यह private/protected members access कर सकता है.
- Friend declaration public, private या protected किसी भी section में हो सकता है.
- Friendship inherited नहीं होती.

11. Friend Class

जब एक class को दूसरी class की friend declare किया जाता है, तो friend class के member functions उस class के private/protected members को access कर सकते हैं.

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

class Student {
private:
    int marks;
public:
    Student() { marks = 95; }
    friend class Result;
```

```
};

class Result {
public:
    void show(Student s) {
        cout << "Marks: " << s.marks << endl;
    }
};

int main() {
    Student s1;
    Result r;
    r.show(s1);
    return 0;
}
```

Basis	Friend Function	Friend Class
Meaning	Specific external function को access मिलता है	पूरी class को access मिलता है
Access scope	Only declared function	All member functions of friend class
Declaration	friend void show();	friend class Result;
Security	More controlled	Broader access

12. Comparison Tables for Exam

Basis	Compile-time Polymorphism	Runtime Polymorphism
Binding	Early/static binding	Late/dynamic binding
Decision time	Compile time	Runtime
Achieved by	Function, operator, constructor overloading	Virtual functions
Speed	Generally faster	Slight overhead due to dynamic binding
Flexibility	Less flexible	More flexible

Basis	Inheritance	Polymorphism
Meaning	One class acquires properties of another	One name many forms
Purpose	Code reuse	Flexibility
Example	Student inherits Person	Overloading/virtual function
OOP role	Reuse and extension	Dynamic behavior

13. Complete Example: Inheritance + Runtime Polymorphism

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

class Employee {
protected:
    string name;
public:
    void setName(string n) { name = n; }
    virtual void calculateSalary() {
        cout << "Employee salary calculation" << endl;
```

```
    }
};

class PermanentEmployee : public Employee {
public:
    void calculateSalary() {
        cout << name << " salary includes basic pay, DA and HRA." << endl;
    }
};

class ContractEmployee : public Employee {
public:
    void calculateSalary() {
        cout << name << " salary is calculated on contract basis." << endl;
    }
};

int main() {
    Employee *ptr;
    PermanentEmployee p;
    ContractEmployee c;
    p.setName("Rahul");
    c.setName("Neha");
    ptr = &p;
    ptr->calculateSalary();
    ptr = &c;
    ptr->calculateSalary();
    return 0;
}
```

Explanation: Employee base class है. PermanentEmployee और ContractEmployee derived classes हैं. calculateSalary() virtual है, इसलिए runtime पर सही derived class function call होता है.

14. Important University Exam Definitions

Inheritance: Inheritance is an OOP concept by which one class acquires the properties and functions of another class.

Base Class: The class whose properties are inherited by another class is called base class.

Derived Class: The class that inherits properties from another class is called derived class.

Single Inheritance: Single inheritance is a type of inheritance in which one derived class inherits from one base class.

Multilevel Inheritance: Multilevel inheritance is a type of inheritance in which a class is derived from another derived class.

Multiple Inheritance: Multiple inheritance is a type of inheritance in which one derived class inherits from two or more base classes.

Hierarchical Inheritance: Hierarchical inheritance is a type of inheritance in which multiple derived classes inherit from one base class.

Hybrid Inheritance: Hybrid inheritance is a combination of two or more types of inheritance.

Polymorphism: Polymorphism is an OOP concept in which one name can have many forms.

Virtual Function: A virtual function is a base class member function that is overridden in derived class and resolved at runtime.

Pure Virtual Function: A pure virtual function has no body in base class and is declared using = 0.

Abstract Class: A class containing at least one pure virtual function is called abstract class.

Friend Function: A friend function is a non-member function that can access private and protected members of a class.

Friend Class: A friend class can access private and protected members of another class.

15. Important Long Answer Questions

Q1. Explain inheritance and its types in C++.

Inheritance is a major feature of object-oriented programming. It allows one class to acquire properties and functions of another class. The class from which properties are inherited is called base class, and the class that inherits them is called derived class. The main purpose of inheritance is code reusability and program extension. Types include single inheritance, multilevel inheritance, multiple inheritance, hierarchical inheritance and hybrid inheritance. Single inheritance has one base and one derived class. Multilevel forms a chain. Multiple inheritance has one derived class with two or more base classes. Hierarchical inheritance has one base class with many derived classes. Hybrid inheritance is a combination of two or more inheritance forms.

Q2. Explain polymorphism and its types.

Polymorphism means one name many forms. In C++, the same function name or operator can perform different tasks according to context. Polymorphism is of two types: compile-time polymorphism and runtime polymorphism. Compile-time polymorphism is resolved during compilation and is achieved by function overloading, operator overloading and constructor overloading. Runtime polymorphism is resolved during program execution and is achieved by virtual functions. Polymorphism improves flexibility and helps in writing extensible programs.

Q3. Explain virtual function with example.

A virtual function is a member function of a base class that can be overridden in a derived class. It is declared using the virtual keyword. When a base class pointer points to a derived class object and calls a virtual function, the derived class version of the function is executed. This is called runtime polymorphism or dynamic binding.

Q4. Explain friend function with example.

A friend function is not a member of a class but can access private and protected members of that class. It is declared inside the class using the friend keyword. Friend function is useful when an external function needs controlled access to private data. It is called like a normal function, not using object dot operator.

16. Short Questions with Answers

What is inheritance?

Answer: It is the process by which one class acquires properties of another class.

What is base class?

Answer: The class whose properties are inherited is called base class.

What is derived class?

Answer: The class that inherits another class is called derived class.

What is multiple inheritance?

Answer: A derived class inheriting two or more base classes is multiple inheritance.

What is polymorphism?

Answer: Polymorphism means one name many forms.

What is compile-time polymorphism?

Answer: Polymorphism resolved at compile time is compile-time polymorphism.

What is runtime polymorphism?

Answer: Polymorphism resolved during program execution is runtime polymorphism.

What is virtual function?

Answer: A virtual function is resolved dynamically at runtime.

What is pure virtual function?

Answer: A virtual function declared with = 0 is pure virtual function.

What is abstract class?

Answer: A class having at least one pure virtual function is abstract class.

What is friend class?

Answer: A class that can access private members of another class is friend class.

17. Important MCQs

1. Inheritance mainly supports:

A. Data hiding

B. Code reusability

C. Data deletion

D. File handling

Answer: B

2. The class whose properties are inherited is called:

A. Derived class

B. Base class

C. Friend class

D. Abstract object

Answer: B

3. One derived class inherits from two base classes in:

A. Single inheritance

B. Multiple inheritance

C. Hierarchical inheritance

D. None

Answer: B

4. One base class with multiple derived classes is:

A. Hierarchical inheritance

B. Multiple inheritance

C. Single inheritance

D. Hybrid inheritance

Answer: A

5. Polymorphism means:

A. One class

B. One name many forms

C. One object only

D. One function only

Answer: B

6. Function overloading is an example of:

A. Runtime polymorphism

B. Compile-time polymorphism

C. Inheritance

D. Encapsulation

Answer: B

7. Virtual function is used to achieve:

A. Compile-time polymorphism

B. Runtime polymorphism

C. Constructor overloading

D. Static data

Answer: B

8. Pure virtual function is declared using:

A. virtual void show() = 0;

B. static void show();

C. friend void show();

D. inline void show();

Answer: A

9. Friend function can access:

A. Only public members

B. Private and protected members

C. Only local variables

D. Only static members

Answer: B

10. Abstract class object creation is:

A. Allowed

B. Not allowed

C. Compulsory

D. Same as friend class

Answer: B

18. Quick Revision Points

6. Inheritance means acquiring properties of one class into another class.
7. Base class is parent class and derived class is child class.
8. Inheritance improves code reusability.
9. Single inheritance has one base and one derived class.
10. Multilevel inheritance forms a chain.
11. Multiple inheritance has more than one base class.
12. Hierarchical inheritance has one base and many derived classes.
13. Hybrid inheritance is a combination of inheritance types.
14. Polymorphism means one name many forms.
15. Compile-time polymorphism is achieved by overloading.
16. Runtime polymorphism is achieved by virtual function.
17. Pure virtual function makes a class abstract.
18. Friend function is not class member but can access private members.
19. Friend class gives access to all member functions of another class.

Best Exam Summary: Inheritance provides code reusability by allowing one class to acquire properties of another class. Polymorphism provides flexibility by allowing the same function or operator to behave differently in different situations. Compile-time polymorphism uses overloading, while runtime polymorphism uses virtual functions.