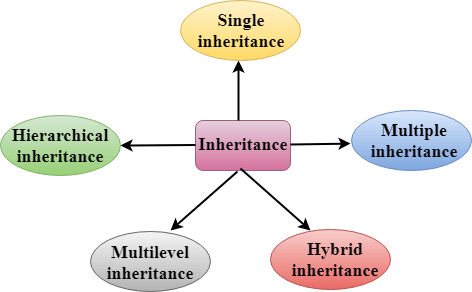
**Inheritance**

In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

**Types of Inheritance**



1. **Single Inheritance** A derived class inherits from a single base class. C++ Inheritance

**Example :**

class Base {

public:

void show() {

cout << "This is Base Class." << endl;

}

};

class Derived : public Base {

public:

void display() {

cout << "This is Derived Class." << endl;

}

};

int main() {

Derived obj;

obj.show(); // Inherited from Base

obj.display(); // From Derived

return 0;

}

**2. Multilevel Inheritance** A class derives from a class, and another class derives from it, forming a chain.

C++ Inheritance

**Example :**

class A {

public:

void methodA() {

cout << "Class A Method" << endl;

}

};

class B : public A {

public:

void methodB() {

cout << "Class B Method" << endl;

}

};

class C : public B {

public:

void methodC() {

cout << "Class C Method" << endl;

}

};

int main() {

C obj;

obj.methodA(); // From A

obj.methodB(); // From B

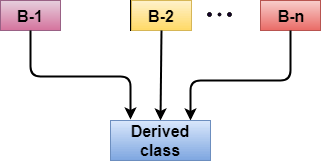
obj.methodC(); // From C

return 0;

}

1. **Multiple Inheritance**

A derived class inherits from more than one base class.



class A {

public:

void methodA() {

cout << "Class A Method" << endl;

}

};

class B {

public:

void methodB() {

cout << "Class B Method" << endl;

}

};

class C : public A, public B {

public:

void methodC() {

cout << "Class C Method" << endl;

}

};

int main() {

C obj;

obj.methodA(); // From A

obj.methodB(); // From B

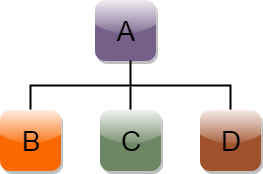
obj.methodC(); // From C

return 0;

}

1. **Hierarchical Inheritance**

Multiple derived classes inherit from a single base class.



class Base {

public:

void methodBase() {

cout << "Base Class Method" << endl;

}

};

class Derived1 : public Base {

public:

void methodDerived1() {

cout << "Derived1 Class Method" << endl;

}

};

class Derived2 : public Base {

public:

void methodDerived2() {

cout << "Derived2 Class Method" << endl;

}

};

int main() {

Derived1 obj1;

obj1.methodBase();

obj1.methodDerived1();

Derived2 obj2;

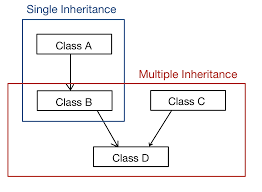
obj2.methodBase();

obj2.methodDerived2();

return 0;

}

1. **Hybrid Inheritance**

Hybrid inheritance is when a class inherits from more than one class using a combination of different inheritance types, such as single and multiple inheritance, in a single class hierarchy. 

#include <iostream>

using namespace std;

// Base class

class A {

public:

void displayA() {

cout << "Class A: Base class" << endl;

}

};

// Derived from A (Single Inheritance)

class B : public A {

public:

void displayB() {

cout << "Class B: Derived from A" << endl;

}

};

// Another base class

class C {

public:

void displayC() {

cout << "Class C: Independent class" << endl;

}

};

// Derived from both B and C (Multiple Inheritance)

class D : public B, public C {

public:

void displayD() {

cout << "Class D: Derived from B and C" << endl;

}

};

int main() {

D objD;

// Access members of all classes

objD.displayA(); // From A

objD.displayB(); // From B

objD.displayC(); // From C

objD.displayD(); // From D

return 0;

}