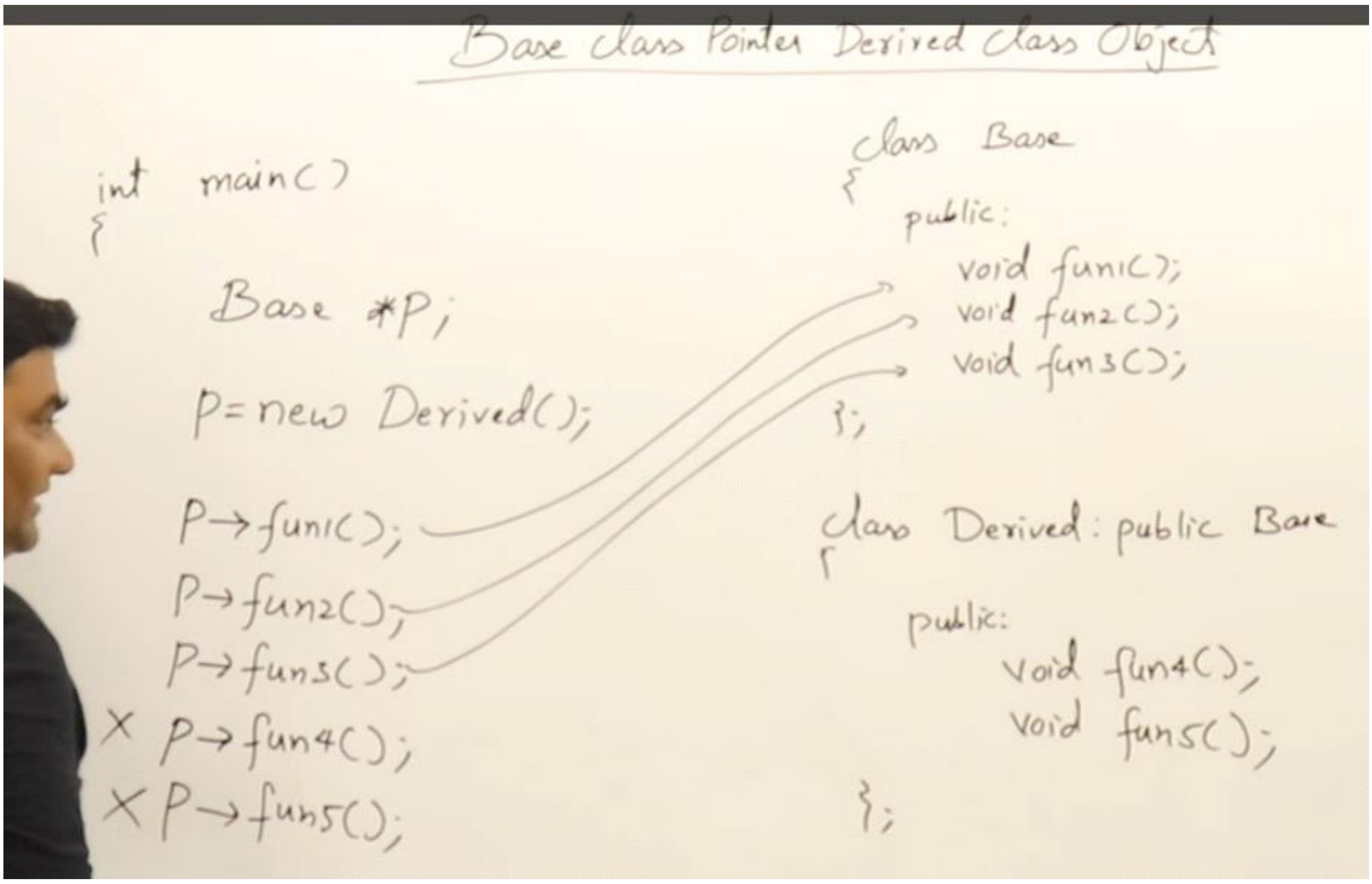
Base Class Pointer

22 September 2024 20:12

base class pointers can point to **derived class objects**. This is a fundamental concept in **polymorphism** in object-oriented programming, where a base class pointer or reference can be used to refer to any object derived from that base class.

How It Works:

- A **pointer of the base class type** can hold the address of an object of a derived class. However, by default, it will only be able to access members of the base class.
- To access derived class members through a base class pointer, **virtual functions** (runtime polymorphism) must be used.



```
int main()
{
    Base *P;
    P=new Derived();
    P->fun1();
    P->fun2();
    P->fun3();
  X P->fun4();
  X P->fun5();
```

```
class Base
{
  public:
    void fun1();
    void fun2();
    void fun3();
};

class Derived : public Base
{
  public:
    void fun4();
    void fun5();
};
```

Base class Pointer pointing to derived class object

- Base class pointer can point on derived class object
- But only those functions which are in base class, can be called
- If derived class is having overrides functions they will not be called unless base class functions are declared as virtual
- Derived class pointer cannot point on base class object

Example 1

```
class Base
{
public:
    void fun1()
    {
        cout<<"fun1 of Base "<<endl;
    }
};
```

Example 2

```
class Derived: public Base
{
public:
    void fun2()
    {
        cout<<"fun2 of Derived"<<endl;
    }
};

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

class Cuboid: public Rectangle
{
public:
    void volume()
    {
        cout<<"Volume of Cuboid"<<endl;
    }
};
```

```
#include <iostream>
using namespace std;
class Base {
public:
    void display() {
        cout << "Base class display function." << endl;
    }
    virtual void show() {
        cout << "Base class show function." << endl;
    }
};
class Derived : public Base {
public:
    void display() {
        cout << "Derived class display function." << endl;
    }
    void show() override {
        cout << "Derived class show function." << endl;
    }
};
int main() {
    Base* basePtr;         // Base class pointer
    Derived derivedObj;    // Derived class object
    basePtr = &derivedObj;  // Base class pointer pointing to 
derived class object
    basePtr->display();     // Calls Base class version (no 
polymorphism)
    basePtr->show();        // Calls Derived class version 
(polymorphism through virtual functions)
    return 0;
}
```