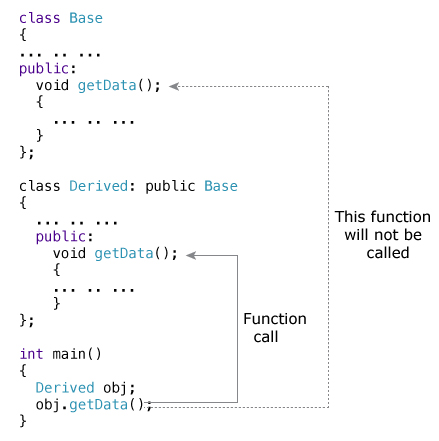
C++ Function Overriding

Inheritance allows software developers to derive a new class from the existing class. The derived class inherits features of the base class (existing class).

Suppose, both base class and derived class have a member function with same name and arguments (number and type of arguments).

If you create an object of the derived class and call the member function which exists in both classes (base and derived), the member function of the derived class is invoked and the function of the base class is ignored.

This feature in C++ is known as function overriding.

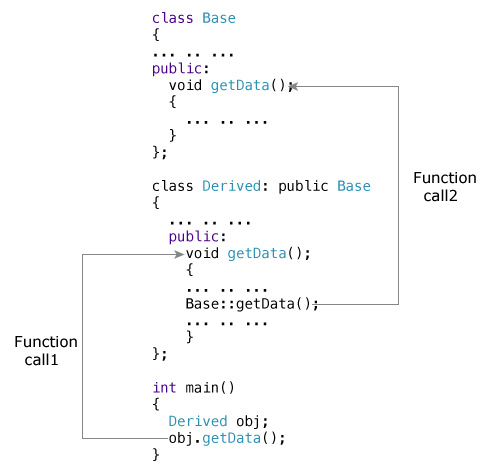


### How to access the overridden function in the base class from the derived class?

To access the overridden function of the base class from the derived class, scope resolution operator :: is used. For example,

If you want to access getData() function of the base class, you can use the following statement in the derived class.

Base::getData();



EXAMPLE:

1. #include <iostream>
2. **using** **namespace** std;
3. **class** Animal {
4. **public**:
5. **void** eat(){
6. cout<<"Eating...";
7. }
8. };
9. **class** Dog: **public** Animal {
10. **public**:
11. **void** eat()
12. {
13. cout<<"Eating bread...";
14. }
15. };
16. **int** main(**void**) {
17. Dog d = Dog();
18. d.eat();
19. **return** 0;
20. }

## Multiple Inheritance in C++ Programming

This program calculates the area and perimeter of an rectangle but, to perform this program, multiple inheritance is used.

#include <iostream>

using namespace std;

class Mammal {

public:

Mammal()

{

cout << "Mammals can give direct birth." << endl;

}

};

class WingedAnimal {

public:

WingedAnimal()

{

cout << "Winged animal can flap." << endl;

}

};

class Bat: public Mammal, public WingedAnimal {

};

int main()

{

Bat b1;

return 0;

}

### Ambiguity in Multiple Inheritance

The most obvious problem with multiple inheritance occurs during function overriding.

Suppose, two base classes have a same function which is not overridden in derived class.

If you try to call the function using the object of the derived class, compiler shows error. It's because compiler doesn't know which function to call. For example,

class base1

{

public:

void someFunction( )

{ .... ... .... }

};

class base2

{

void someFunction( )

{ .... ... .... }

};

class derived : public base1, public base2

{

};

int main()

{

derived obj;

obj.someFunction() // Error!

}

This problem can be solved using scope resolution function to specify which function to class either base1 or base2

int main()

{

obj.base1::someFunction( ); // Function of base1 class is called

obj.base2::someFunction(); // Function of base2 class is called.

}