



## Ambiguity Resolution in Inheritance in C++ | C++ Tutorials for Beginners #43



[Overview](#) [Q&A](#) [Downloads](#) [Announcements](#)

## Ambiguity Resolution in Inheritance in C++ | C++ Tutorials for Beginners #43

In this tutorial, we will discuss ambiguity resolution in inheritance in C++

### Ambiguity Resolution in Inheritance

Ambiguity in inheritance can be defined as when one class is derived for two or more base classes then there are chances that the base classes have functions with the same name. So it will confuse derived class to choose from similar name functions. To solve this ambiguity scope resolution operator is used "::". An example program is shown below to demonstrate the concept of ambiguity resolution in inheritance.

```
class Base1{  
    public:
```

```
        void greet(){
            cout<<"How are you?"<<endl;
        }
};

class Base2{
    public:
        void greet()
        {
            cout << "Kaise ho?" << endl;
        }
};

class Derived : public Base1, public Base2{
    int a;
    public:
        void greet(){
            Base2 :: greet();
        }
};
```

### Code Snippet 1: Ambiguity Resolution in Inheritance Example Program 1

As shown in a code snippet 1,

1. We have created a “Base1” class which consists of public member function “greet”. The function “greet” will

```
print "how are you?"
```

2. We have created a "Base2" class which consists of public member function "greet". The function "greet" will print "kaise ho?"
3. We have created a "Derived" class which is inheriting "Base1" and "Base2" classes. The "Derived" class consists of public member function "greet". The function "greet" will run the "greet" function of the "Base2" class because we have used a scope resolution operator to let the compiler know which function should it run otherwise it will cause ambiguity.

The code of the main function is shown below

```
int main(){  
    // Ambiguity 1  
    Base1 base1obj;  
    Base2 base2obj;  
    base1obj.greet();  
    base2obj.greet();  
    Derived d;  
    d.greet();  
  
    return 0;  
}
```

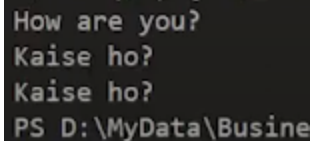
### Code Snippet 2: Main program 1

As shown in code snippet 2,

1. Object "base1obj" is created of the "Base1" data type.
2. Object "base3obj" is created of the "Base2" data type.
3. The function "greet" is called by the object "base1obj".

4. The function “greet” is called by the object “base2obj”.
5. Object “d” is created of the “Derived” data type.
6. The function “greet” is called by the object “d”.

The main thing to note here is that when the function “greet” is called by the object “d” it will run the “greet” function of the “Base2” class because we had specified it using scope resolution operator “::” to get rid of ambiguity. The output for the following program is shown in figure 1.



```
How are you?  
Kaise ho?  
Kaise ho?  
PS D:\MyData\Busine
```

**Figure 1: Output**

Another example of ambiguity resolution in inheritance is shown below.

```
class B{  
    public:  
        void say(){  
            cout<<"Hello world"<<endl;  
        }  
};  
  
class D: public B{  
    int a;  
    // D's new say() method will override base class's say() method  
    public:  
        void say()  
        {
```

```
        cout << "Hello my beautiful people" << endl;
    }
};
```

### Code Snippet 3: Ambiguity Resolution in Inheritance Example Program 2

As shown in a code snippet 3,

1. We have created a “B” class which consists of public member function “say”. The function “say” will print “hello world”
2. We have created a “D” class that is inheriting the “B” class. The “D” class consists of the public member function “say”. The function “say” will print “Hello my beautiful people”

The main thing to note here is that both “B” and “D” classes have the same function “say”, So if the class “D” will call the function “say” it will override the base class “say” method because compiler by default run the method which is already written in its own body. But if the function “say” was not present in the class “D” then the compiler will run the method of the class “B”.

The code of the main function is shown below,

```
int main(){
    // Ambiguity 2
    B b;
    b.say();

    D d;
    d.say();

    return 0;
```

```
}
```

### Code Snippet 4: Main Program 2

As shown in code snippet 4,

1. Object “b” is created of the “B” data type.
2. The function “say” is called by the object “b”.
3. Object “d” is created of the “D” data type.
4. The function “say” is called by the object “d”.

The output for the following program is shown in figure 2.

```
Hello world  
Hello my beautiful people
```

**Figure 2: Output**

[Previous](#)[Next](#)

CodeWithHarry

Copyright © 2022 CodeWithHarry.com

