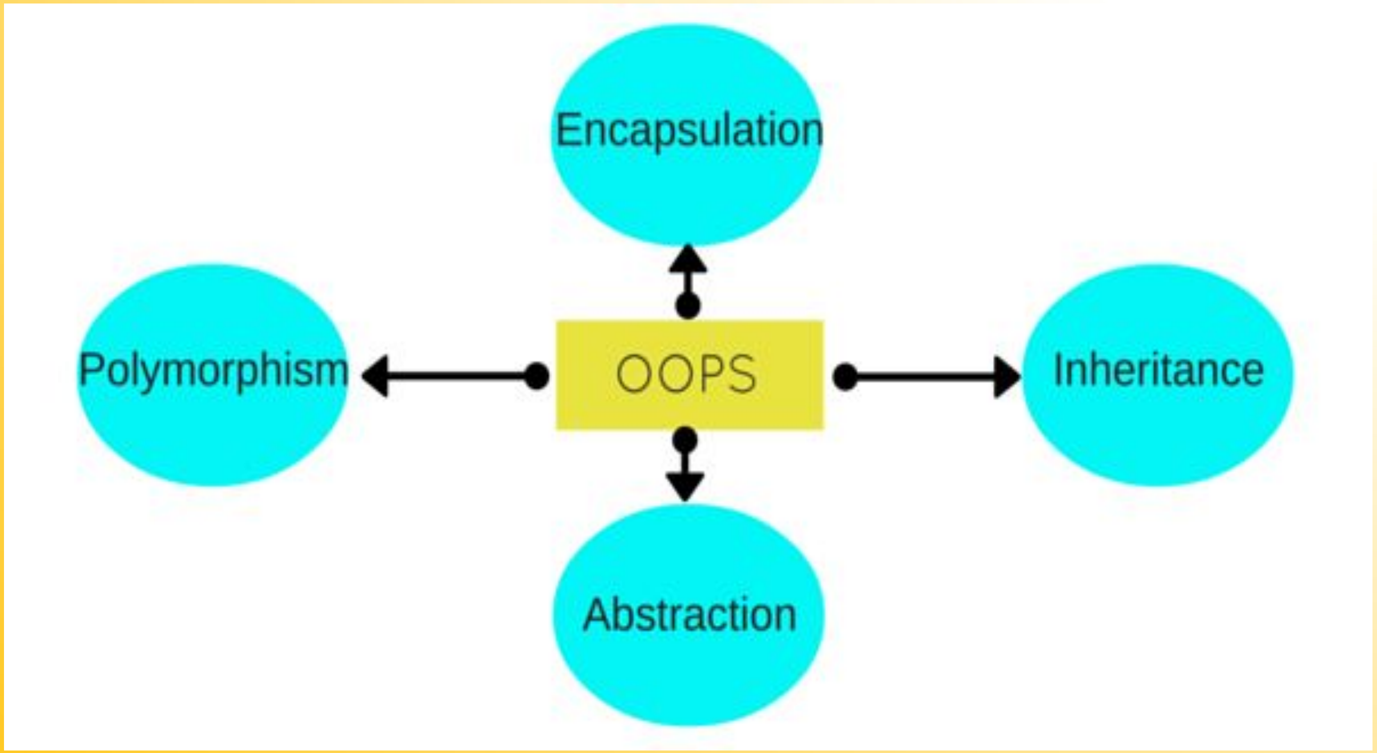
Polymorphism in C++

CONTENTS

C++ and Object Oriented Programming

- Object Oriented programming is a programming style that is associated with the concept of Class, Objects and various other concepts revolving around these two, like Inheritance, Polymorphism, Abstraction, Encapsulation etc.



Oops Concept in C++

- **OBJECTS** -Objects are the basic unit of OOP. They are instances of class, which have data members and uses various member functions to perform tasks.
- **CLASS** -It is similar to structures in C language. Class can also be defined as user defined data type but it also contains functions in it.
- **ABSTRACTION** -Abstraction refers to showing only the essential features of the application and hiding the details.

Encapsulation -It can also be said data binding. Encapsulation is all about binding the data variables and functions together in class.

Inheritance

Inheritance is a way to reuse once written code again and again. The class which is inherited is called base calls & the class which inherits is called derived class. So when, a derived class inherits a base class, the derived class can use all the functions which are defined in base class, hence making code reusable.

Polymorphism

It is a feature, which lets us create functions with same name but different arguments, which will perform differently. That is function with same name, functioning in different way. Or, it also allows us to redefine a function to provide its new definition. You will learn how to do this in details soon in coming lessons.

Polymorphism in C++

- The process of representing one Form in multiple forms is known as Polymorphism.
- Here one form represent original form or original method always resides in base class and multiple forms represents overridden method which resides in derived classes.
- Polymorphism is derived from 2 Greek words: **poly** and morphs. The word "poly" means many and **morphs** means forms. So polymorphism means many forms.

Real life example of Polymorphism in C++

- Suppose if you are in class room that time you behave like a student, when you are in market at that time you behave like a customer, when you at your home at that time you behave like a son or daughter, Here one person have different-different behaviors.

T4us
In Shopping malls behave like Customer
In Bus behave like Passenger
In School behave like Student
At Home behave like Son
Tutorial4us.com

Type of Polymorphism

- Compile time polymorphism
- Run time polymorphism

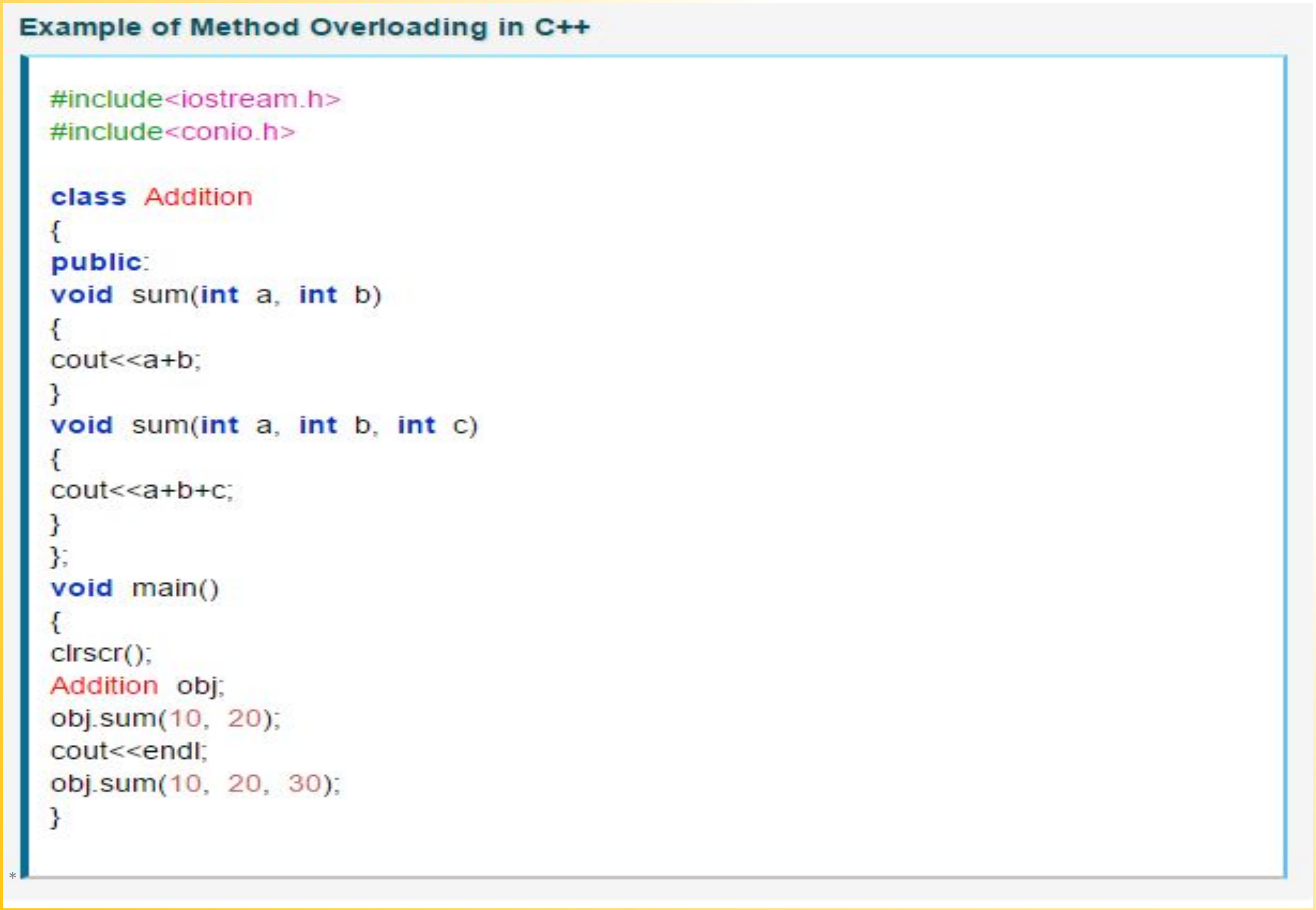
Compile time polymorphism

- In C++ programming you can achieve compile time polymorphism in two way, which is given below;
- Method overloading
- Method overriding

Method Overloading in C++

- Whenever same method name is exiting multiple times in the same class with different number of parameter or different order of parameters or different types of parameters is known as method overloading.

FIGURE 1:Example of Method Overloading in C++

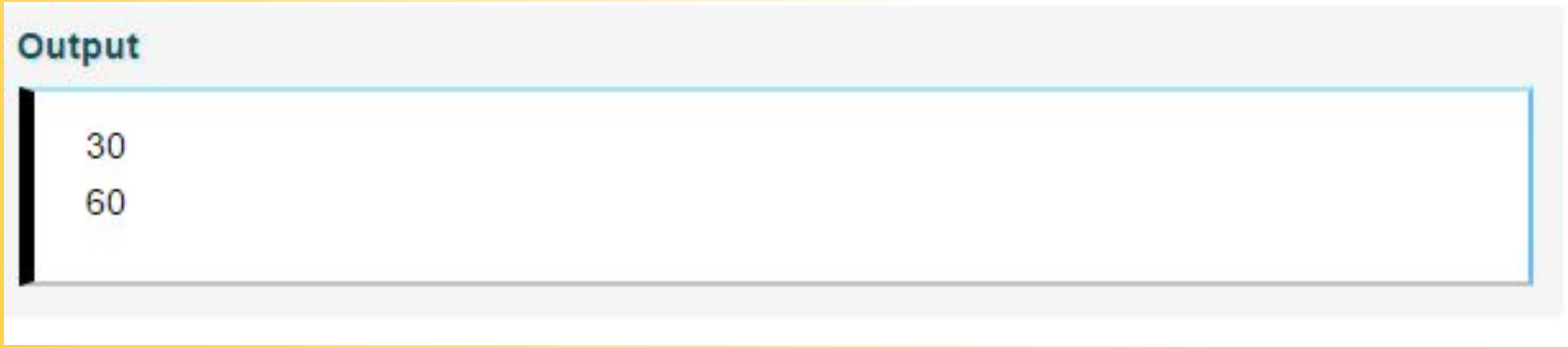


Example of Method Overloading in C++

```
#include<iostream.h>
#include<conio.h>

class  Addition
{
public:
void  sum(int  a,  int  b)
{
cout<<a+b;
}
void  sum(int  a,  int  b,  int  c)
{
cout<<a+b+c;
}
};
void  main()
{
clrscr();
Addition  obj;
obj.sum(10,  20);
cout<<endl;
obj.sum(10,  20,  30);
}
```

*

FIGURE 2:OUTPUT



```
30
60
```

Questions:

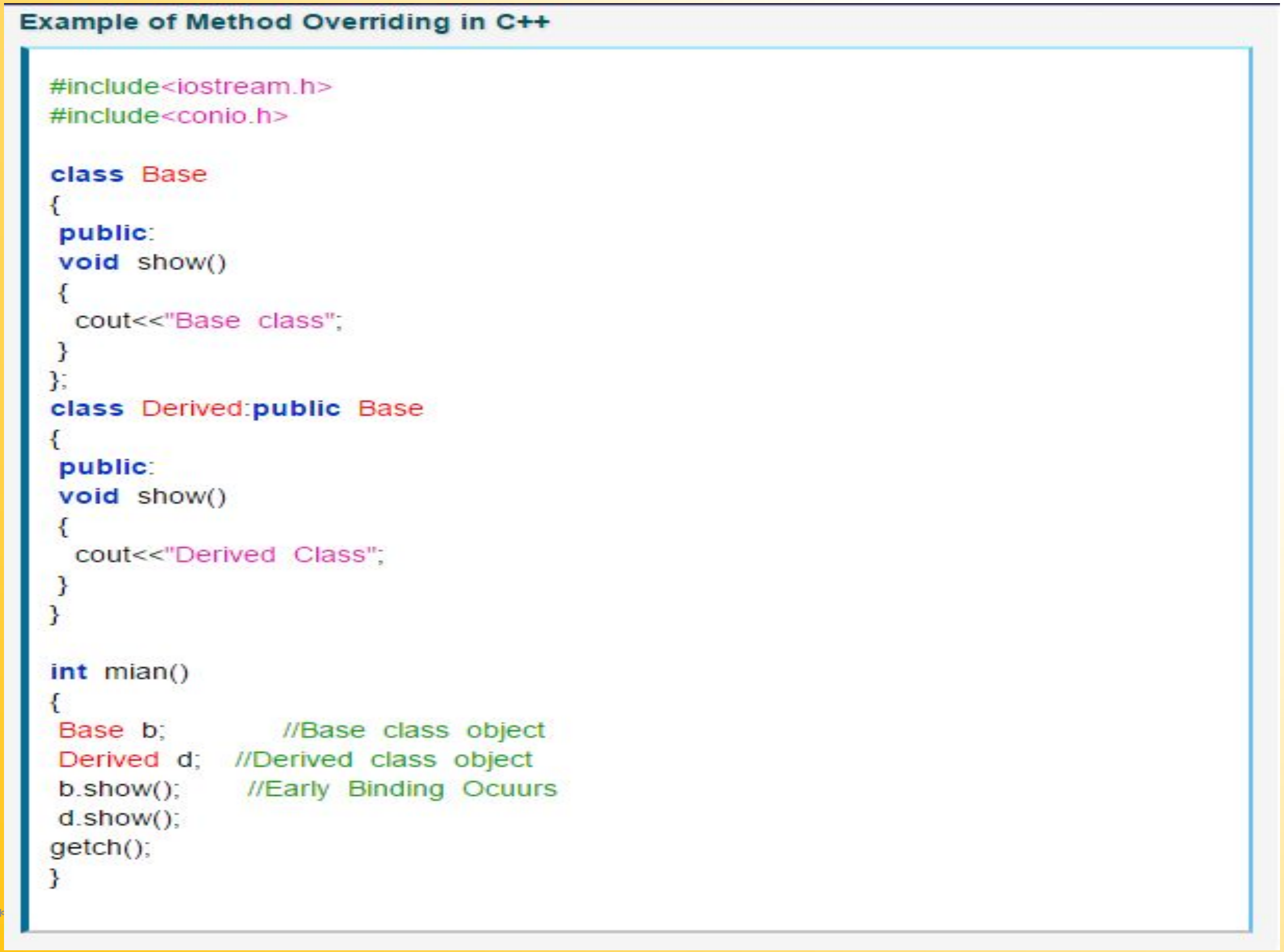
Write a c++ program to calculate area of triangle and square

Write a c++ program to calculate perimeter of triangle and square

Method Overriding in C++

- Define any method in both base class and derived class with same name, same parameters or signature, this concept is known as **method overriding**. In below example same method "show()" is present in both base and derived class with same name and signature.

FIGURE 3:Method Overriding in C++

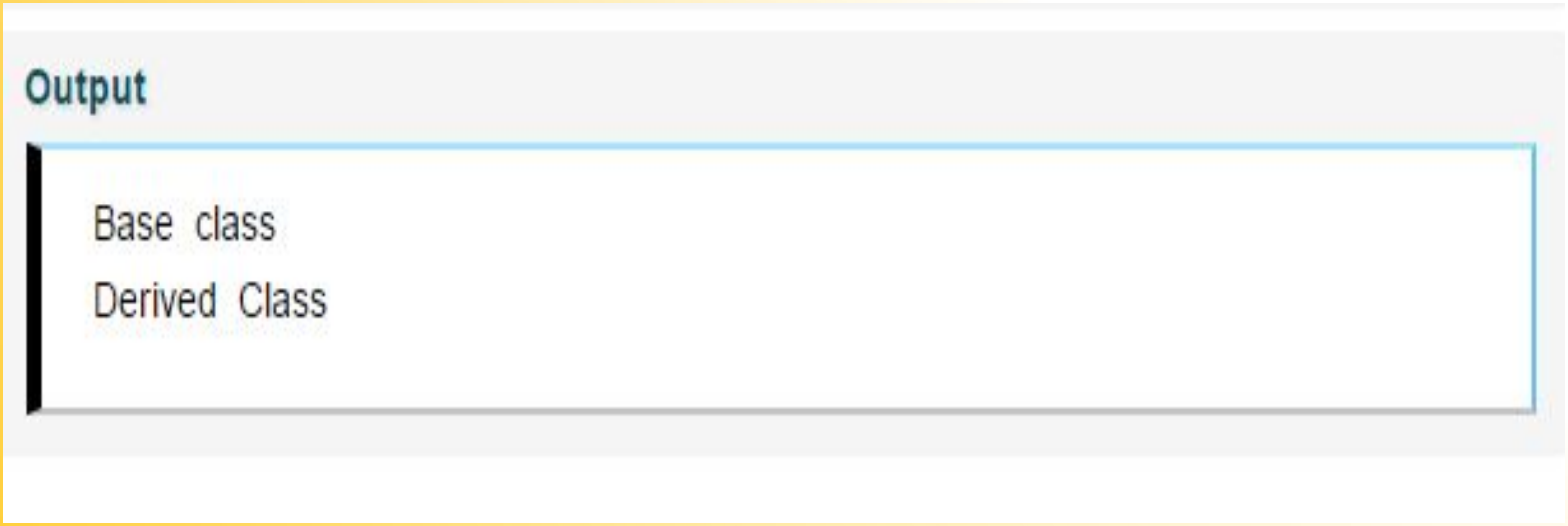


Example of Method Overriding in C++

```
#include<iostream.h>
#include<conio.h>

class Base
{
 public:
 void show()
 {
  cout<<"Base class";
 }
};
class Derived:public Base
{
 public:
 void show()
 {
  cout<<"Derived Class";
 }
}

int mian()
{
 Base b;         //Base class object
 Derived d;   //Derived class object
 b.show();      //Early Binding Ocuurs
 d.show();
getch();
}
```

*

FIGURE 4:Output


Output

```
Base class
Derived Class
```

*

Run time polymorphism

- In C++ Run time polymorphism can be achieve by using virtual function.

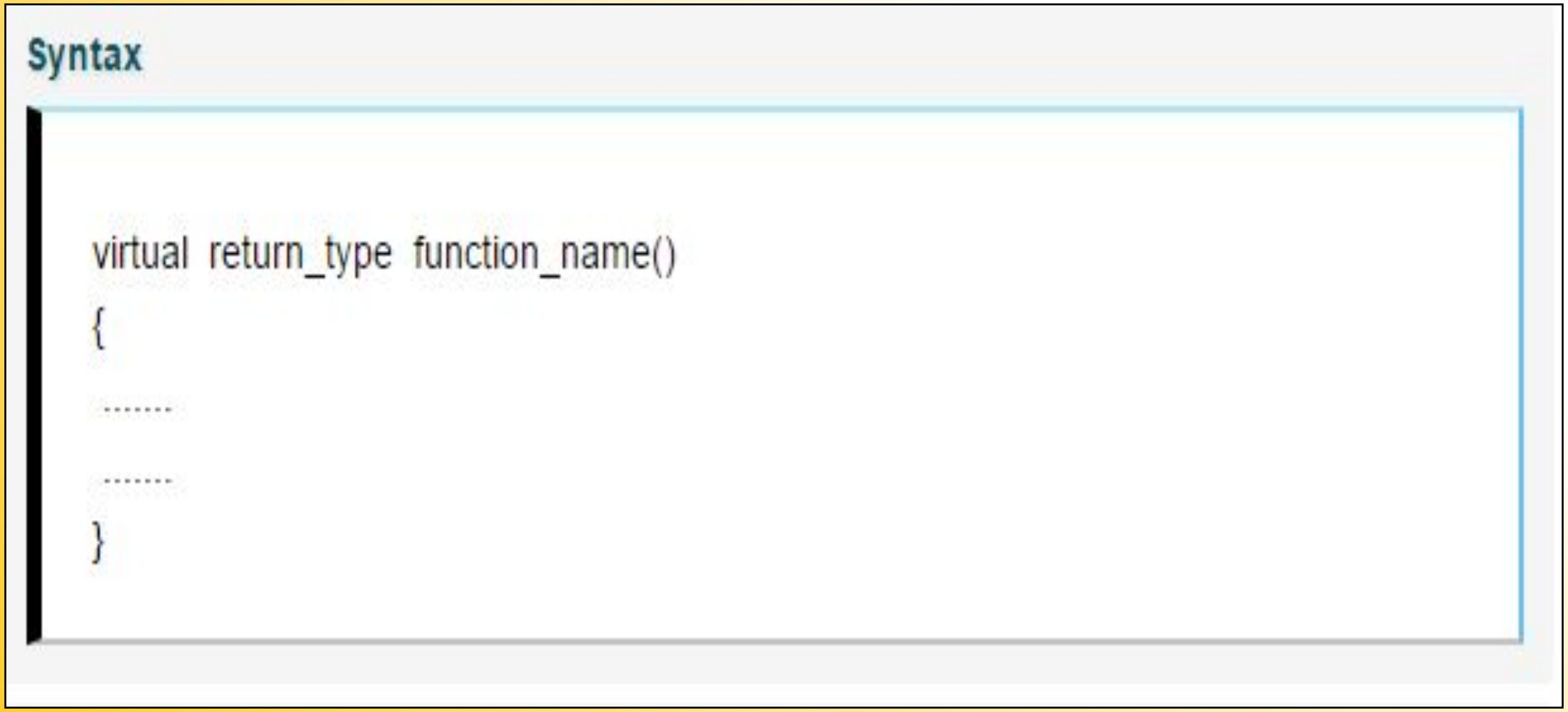
- It works by means of an indirect call, calling a virtual member function of a base object by means of reference or a pointer to what is actually a derived object re-implementing the virtual functions.

Virtual Function in C++

- A virtual function is a member function of class that is declared within a base class and re-defined in derived class.

- When you want to use same function name in both the base and derived class, then the function in base class is declared as virtual by using the virtual keyword and again re-defined this function in derived class without using virtual keyword.

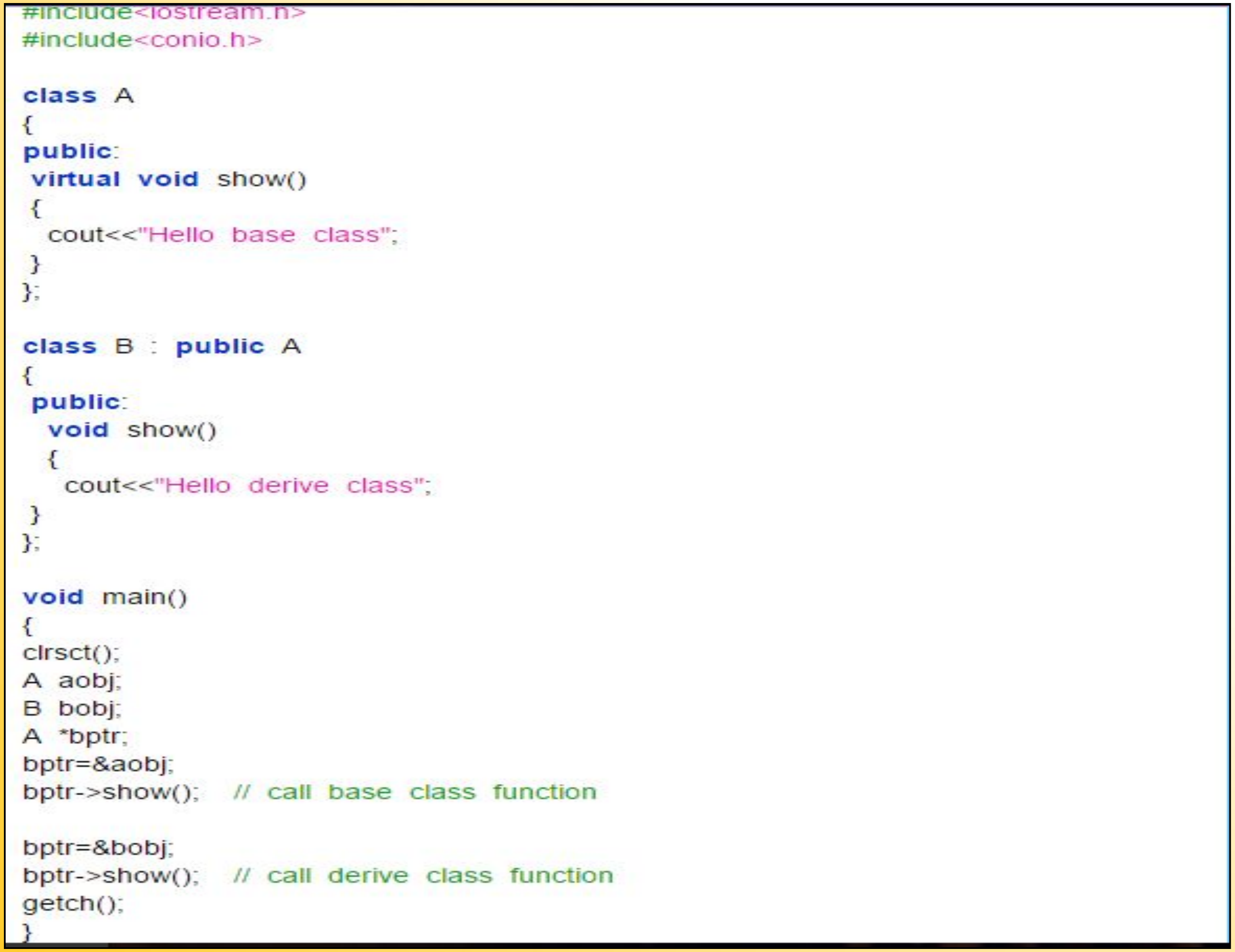
FIGURE 5: SYNTAX FOR Virtual Function in C++


Syntax

```
virtual return_type function_name()
{
........
........
}
```

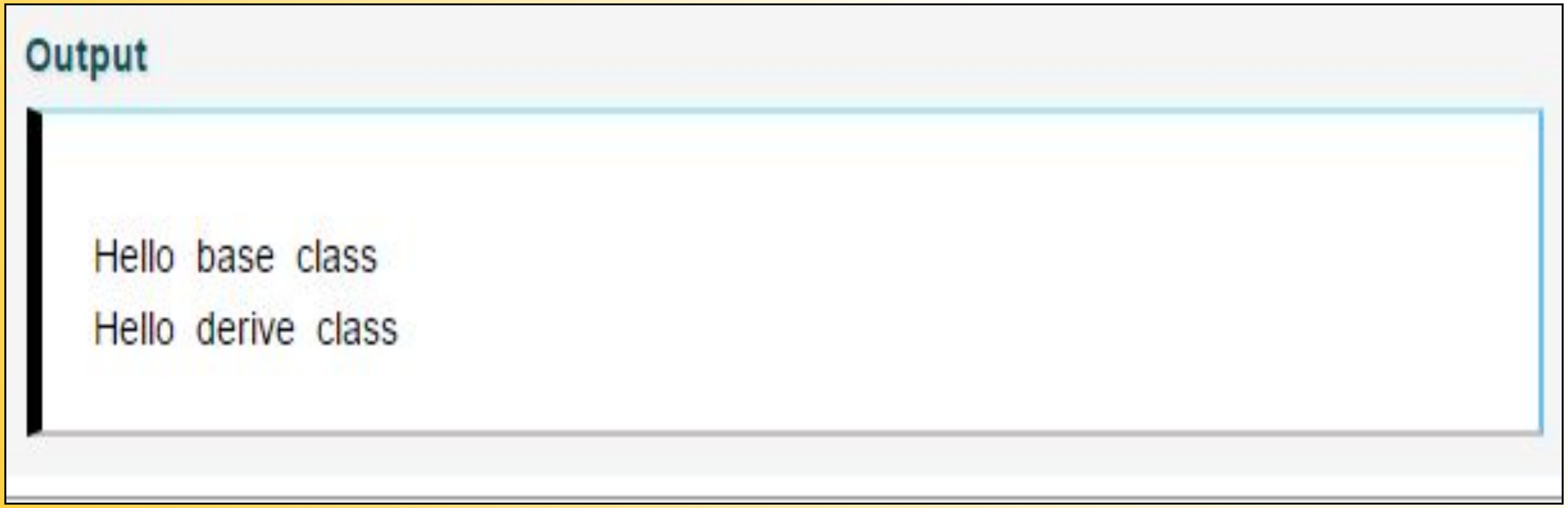
*

FIGURE 6:Example of Method Overriding in C++



```
#include<iostream.h>
#include<conio.h>

class A
{
public:
 virtual void show()
 {
   cout<<"Hello base class";
 }
};

class B : public A
{
 public:
   void show()
   {
    cout<<"Hello derive class";
 }
};

void main()
{
clrsct();
A aobj;
B bobj;
A *bptr;
bptr=&aobj;
bptr->show();   // call base class function

bptr=&bobj;
bptr->show();   // call derive class function
getch();
}
```

FIGURE 7: Output



```
Hello  base  class
Hello  derive  class
```

Reference & Research

- http://www.studytonight.com/cpp/virtual-destructors.php
- http://www.sitesbay.com/cpp/cpp-polymorphism
- https://www.tutorialspoint.com/cplusplus/pdf/cpp_polymorphism.pdf

Thank you