```
// C++ program for function overriding

#include <iostream>
using namespace std;

class base
{
public:
   virtual void print ()
   { cout<< "print base class" <<endl; }

   void show ()
   { cout<< "show base class" <<endl; }
};

class derived:public base
{
public:
   void print () //print () is already virtual function in derived class, we could also declared as virtual
void print () explicitly
   { cout<< "print derived class" <<endl; }

   void show ()
   { cout<< "show derived class" <<endl; }
};

//main function
int main()
{
   base *bptr;
```

```
    derived d;
    bptr = &d;

    //virtual function, binded at runtime (Runtime polymorphism)
    bptr->print();

    // Non-virtual function, binded at compile time
    bptr->show();

    return 0;
}
```

```
#include <iostream>
#include <string>
using namespace std;

// Base class
class Animal {
  public:
    void animalSound() {
      cout << "The animal makes a sound \n" ;
    }
};

// Derived class
class Pig : public Animal {
  public:
    void animalSound() {
      cout << "The pig says: wee wee \n" ;
    }
```

```
};

// Derived class
class Dog : public Animal {
  public:
   void animalSound() {
      cout << "The dog says: bow wow \n" ;
    }
};

int main() {
  Animal myAnimal;
  Pig myPig;
  Dog myDog;

  myAnimal.animalSound();
  myPig.animalSound();
  myDog.animalSound();
  return 0;
}



#include <iostream>
using namespace std;

class Shape {
   protected:
      int width, height;

   public:
      Shape( int a = 0, int b = 0){
         width = a;
         height = b;
      }
      virtual int area() {
         cout << "Parent class area :" <<endl;
         return 0;
      }
};
class Rectangle: public Shape {
   public:
      Rectangle( int a = 0, int b = 0):Shape(a, b) { }

      int area () {
         cout << "Rectangle class area :" <<endl;
         return (width * height);
      }
};

class Triangle: public Shape {
   public:
      Triangle( int a = 0, int b = 0):Shape(a, b) { }

      int area () {
         cout << "Triangle class area :" <<endl;
         return (width * height / 2);
      }
};

// Main function for the program
int main() {
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);

   // store the address of Rectangle
   shape = &rec;

   // call rectangle area.
   shape->area();

   // store the address of Triangle
   shape = &tri;

   // call triangle area.
   shape->area();

   return 0;
}
```