```
/* virtual.cpp
   Simple program to demonstrate virtual member functions -
   1. funtions with pass by reference/pointer, the class type of the argument determines ↙
    which instance to call (dynamic binding)
   2. funtions with pass by value, the class type of the parameter determines which ↙
    instance to call (static binding)

   Sophie wang
   Created: October 10, 2006
   Current: October 10, 2006
*/

#include <iostream>

class One{
    public:
        virtual void Print(){std::cout <<"Print from One\n";}
};

class Two:public One{
    public:
        void Print(){std::cout <<"Print from Two\n";}
};

void Print0(One one)
{
    one.Print();
}

void Print1(One& one)
{
    one.Print();
}

void Print2(One* onePtr)
{
    onePtr->Print();
}

void Print3(One one, One &oneRef, One *onePtr)
{
    one.Print();
    oneRef.Print();
    onePtr->Print();
}

int main()
{

    One one;
    Two two;

    Print0(one); // One::Print is called
    Print0(two); // One::Print is called

    Print1(one); // One::Print is called
    Print1(two); // Two::Print is called
    one = two;
    Print1(one); // One::Print is called

    Print3(one, one, &one); // One::Print, One::Print, One::Print
    Print3(two, two, &two); // One::Print, Two::Print, Two::Print

    One *onePtr;
    onePtr = new One;
    Print2(onePtr); // One::Print is called
    onePtr = new Two;
    Print2(onePtr); // Two::Print is called
    Two *twoPtr;
    twoPtr = new Two;
    Print2(twoPtr); // Two::Print is called

    return 0;
}
```