```h
#pragma once
#include <iostream>

enum Coordinate
{
    Cartisian = 1,
    Polar
};
class CPlane
{
public:
    CPlane();
    virtual ~CPlane();
    virtual void GetArguments();
    int input1;
    int input2;
};
```

```cpp
#include "CPlane.h"

CPlane::CPlane()
    :input1 (0)
    ,input2 (0)
{
}
CPlane::~CPlane()
{

}
void CPlane::GetArguments()
{

}
```

```h
#pragma once
#include "CPlane.h"

class CCartesian :
    public CPlane
{
public:
    CCartesian();
    ~CCartesian();
    void GetArguments();
};
```

```cpp
#include "CCartesian.h"

CCartesian::CCartesian()
{

}
CCartesian::~CCartesian()
{

}
void CCartesian::GetArguments()
{
    CPlane::GetArguments();
    std::cout << "輸入X值" << std::endl;
    std::cin >> input1;
    std::cout << "輸入Y值" << std::endl;
    std::cin >> input2;
    std::cout << "輸入座標為(" << input1 << "," << input2 << ")" << std::endl;
    std::cout << "對X軸的鏡像座標為 : (" << input1 << "," << -input2 << ")" << std::endl;
    std::cout << "對Y軸的鏡像座標為 : (" << -input1 << "," << input2 << ")" << std::endl;
}
```

```h
class CPolar :
    public CPlane
{
public:
    CPolar();
    ~CPolar();
    void GetArguments();
};
```

```cpp
#include "CPolar.h"

CPolar::CPolar()
{

}
CPolar::~CPolar()
{

}
void CPolar::GetArguments()
{
    CPlane::GetArguments();

        std::cout << "輸入r值" << std::endl;
        std::cin >> input1;
        std::cout << "輸入Theta值" << std::endl;
        std::cin >> input2;

        if (input2 > 360)
        {
            std::cout << "角度為不合法的輸入" << std::endl;
            return;
        }
        else if (input2 >= 180)
        {
            std::cout << "輸入座標為(" << input1 << "," << input2 << ")" << std::endl;
            std::cout << "對L軸的鏡像座標為 : (" << input1 << "," << (360 - input2) << ")" << std::endl;
            std::cout << "對垂直於L軸且通原點的直線鏡像座標為 : (" << input1 << "," << ((360 -input2) + 180) << ")" << std::endl;
        }
        else if (input2 >= 0)
        {
            std::cout << "輸入座標為(" << input1 << "," << input2 << ")" << std::endl;
            std::cout << "對L軸的鏡像座標為 : (" << input1 << "," << (360 - input2) << ")" << std::endl;
            std::cout << "對垂直於L軸且通原點的直線鏡像座標為 : (" << input1 << "," << ((360 - input2) - 180) << ")" << std::endl;
        }
        else if(input2 < 0)
        {
            std::cout << "角度為不合法的輸入" << std::endl;
            return;
        }


    //std::cout << "對垂直於L軸且通原點的直線鏡像座標為 : (" << input1 << "," << (input2 + 180) << ")" << std::endl;
}
```

```cpp
#include <iostream>
#include "CCartesian.h"
#include "CPolar.h"

int main()
{
    CPlane* CurrentCoordinateSystem = NULL;
    int selection;
    int coordinate;
    std::cout << "請輸入號碼選擇座標系 : \n1.直角座標系\t2.極座標系\n:";
    std::cin >> selection;
    switch (selection)
    {
    case Cartisian:
        CurrentCoordinateSystem = new CCartesian;
        std::cout << "選擇了直角座標系" << std::endl;
        break;
    case Polar:
        CurrentCoordinateSystem = new CPolar;
        std::cout << "選擇了極座標系" << std::endl;
        break;
    }
    std::cout << "請輸入座標值 : " << std::endl;
    CurrentCoordinateSystem->GetArguments();
    //CurrentCoordinateSystem->CPlane();

    delete CurrentCoordinateSystem;
    return 0;

}
```

```
Microsoft Visual Studio 偵錯主控台
請輸入號碼選擇座標系 :
1.直角座標系      2.極座標系
:1
選擇了直角座標系
請輸入座標值 :
輸入X值
2
輸入Y值
3
輸入座標為(2,3)
對X軸的鏡像座標為 : (2,-3)
對Y軸的鏡像座標為 : (-2,3)
```

```
Microsoft Visual Studio 偵錯主控台
請輸入號碼選擇座標系 :
1.直角座標系      2.極座標系
:2
選擇了極座標系
請輸入座標值 :
輸入r值
8
輸入Theta值
170
輸入座標為(8,170)
對L軸的鏡像座標為 : (8,190)
對垂直於L軸且通原點的直線鏡像座標為 : (8,10)

C:\Users\88698\OneDrive\Desktop\practice\C++\
代碼 0。
按任意鍵關閉此視窗…
```