

HW1: 平面座標系

HW1CPP.cpp

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

int main()
{
    CPlane *plane = NULL;
    int type = 0, lhs = 0, rhs = 0;

    std::cout << "請問你要使用哪種座標系統?" << "\n";
    std::cout << "1.直角座標系(輸入1)" << "\t" << "2.極座標系(輸入2)" << "\n";
    std::cin >> type; // type of coordinate system = cartesian or polar

    switch (type)
    {
        case cartesian :
            plane = new CCartesian;
            break;
        case polar:
            plane = new CPolar;
            break;
    }

    std::cout << "請輸入欲鏡像的點座標(皆為整數): " << "\n";
    std::cin >> lhs >> rhs; // left hand side & right hand side
    plane -> getPoint(lhs, rhs); // get the point user gave
    std::cout << "原座標(" << lhs << ", " << rhs << ") 在對兩軸分別鏡像後的座標分別為: \n";
    plane -> reflect(); // show the points after reflected with respect to two axis respectively

    delete plane;
    return 0;
}
```

CPlane.h

```
#pragma once
#include <iostream>
enum PlaneType {cartesian = 1 , polar};
class CPlane
{
public:
    CPlane();
    virtual ~CPlane();
    void getPoint(int , int);
    virtual void reflect();
protected:
    int lhs, rhs;
};
```

CPlane.cpp

```
#include "CPlane.h"
CPlane::CPlane() : lhs(0), rhs(0)
{
}
CPlane::~~CPlane()
{
}

void CPlane::getPoint(int lElement, int rElement)
{
    lhs = lElement;
    rhs = rElement;
}

void CPlane::reflect()
{
}
```

CCartesian.h

```
#pragma once
#include "CPlane.h"
class CCartesian : public CPlane
{
public:
    CCartesian();
    ~CCartesian();
    void reflect();
};
```

CCartesian.cpp

```
#include "CCartesian.h"
CCartesian::CCartesian()
{
}

CCartesian::~~CCartesian()
{
}

void CCartesian::reflect()
{
    std::cout << "對x軸鏡像後的點座標: " << "(" << lhs << ", " << -rhs << ")\n";
    std::cout << "對y軸鏡像後的點座標: " << "(" << -lhs << ", " << rhs << ")\n";
}
```

CPolar.h

```
#pragma once
#include "CPlane.h"
class CPolar : public CPlane
{
public:
    CPolar();
    ~CPolar();
    void reflect();
};
```

CPolar.cpp

```
#include "CPolar.h"

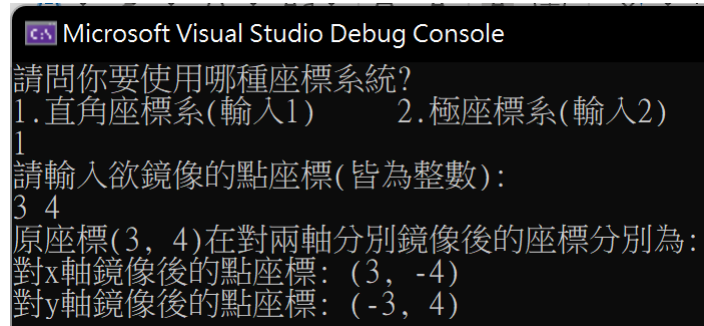
CPolar::CPolar()
{
}

CPolar::~CPolar()
{
}

void CPolar::reflect()
{
    std::cout << "對L軸鏡像後的點座標: " << "(" << lhs << ", " << 360 - rhs << "°)\n";
    std::cout << "對虛軸鏡像後的點座標: " << "(" << lhs << ", ";
    if (rhs <= 180)
        std::cout << 180 - rhs;
    else
        std::cout << 540 - rhs;
    std::cout << "°)\n";
}
```

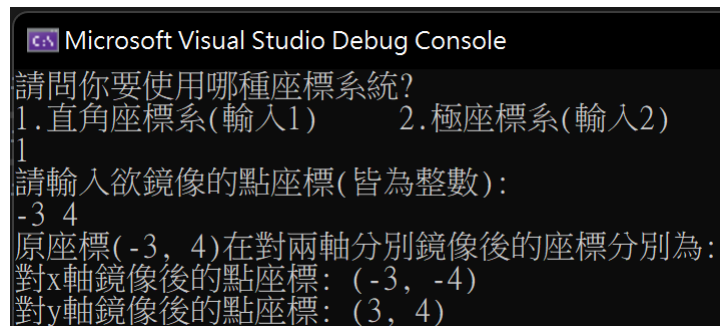
1. 直角坐標

(1) 輸入的點第一象限



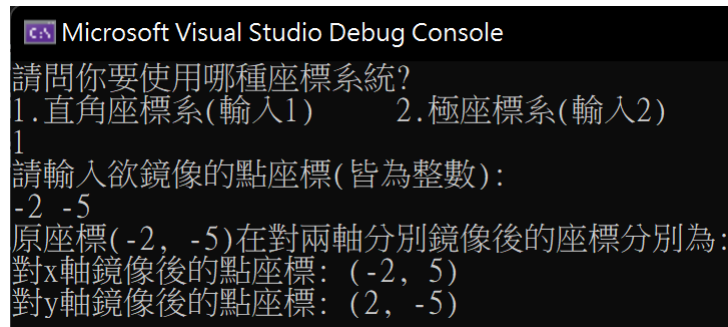
```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
1
請輸入欲鏡像的點座標(皆為整數):
3 4
原座標(3, 4)在對兩軸分別鏡像後的座標分別為:
對x軸鏡像後的點座標: (3, -4)
對y軸鏡像後的點座標: (-3, 4)
```

(2) 輸入的點在第二象限



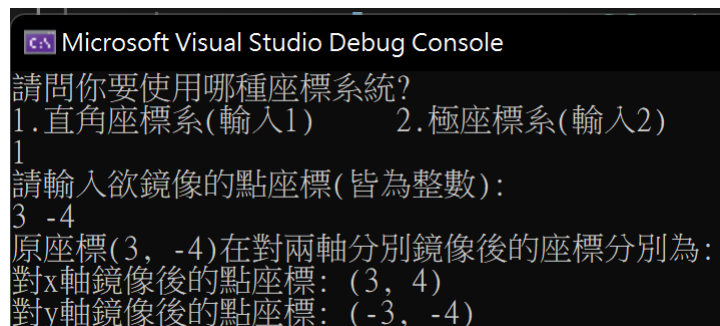
```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
1
請輸入欲鏡像的點座標(皆為整數):
-3 4
原座標(-3, 4)在對兩軸分別鏡像後的座標分別為:
對x軸鏡像後的點座標: (-3, -4)
對y軸鏡像後的點座標: (3, 4)
```

(3) 輸入的點在第三象限



```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
1
請輸入欲鏡像的點座標(皆為整數):
-2 -5
原座標(-2, -5)在對兩軸分別鏡像後的座標分別為:
對x軸鏡像後的點座標: (-2, 5)
對y軸鏡像後的點座標: (2, -5)
```

(4) 輸入的點在第四象限



```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
1
請輸入欲鏡像的點座標(皆為整數):
3 -4
原座標(3, -4)在對兩軸分別鏡像後的座標分別為:
對x軸鏡像後的點座標: (3, 4)
對y軸鏡像後的點座標: (-3, -4)
```

2. 極座標

(1) $0^\circ \leq \theta < 90^\circ$

```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
2
請輸入欲鏡像的點座標(皆為整數):
2 30
原座標(2, 30)在對兩軸分別鏡像後的座標分別為:
對L軸鏡像後的點座標: (2, 330°)
對虛軸鏡像後的點座標: (2, 150°)
```

(2) $90^\circ \leq \theta < 180^\circ$

```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
2
請輸入欲鏡像的點座標(皆為整數):
3 120
原座標(3, 120)在對兩軸分別鏡像後的座標分別為:
對L軸鏡像後的點座標: (3, 240°)
對虛軸鏡像後的點座標: (3, 60°)
```

(3) $180^\circ \leq \theta < 270^\circ$

```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
2
請輸入欲鏡像的點座標(皆為整數):
4 210
原座標(4, 210)在對兩軸分別鏡像後的座標分別為:
對L軸鏡像後的點座標: (4, 150°)
對虛軸鏡像後的點座標: (4, 330°)
```

(4) $270^\circ \leq \theta < 370^\circ$

```
Microsoft Visual Studio Debug Console
請問你要使用哪種座標系統?
1. 直角座標系(輸入1)      2. 極座標系(輸入2)
2
請輸入欲鏡像的點座標(皆為整數):
5 320
原座標(5, 320)在對兩軸分別鏡像後的座標分別為:
對L軸鏡像後的點座標: (5, 40°)
對虛軸鏡像後的點座標: (5, 220°)
```