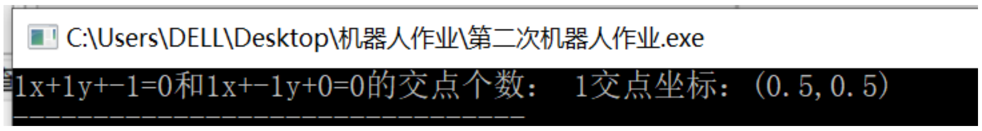
机器人第二次作业
姓名：曾宇杰
学号 2019218164
运行示例：

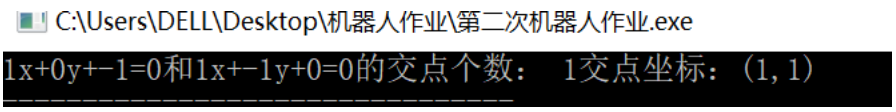
直线和直线交点：

有一个交点：

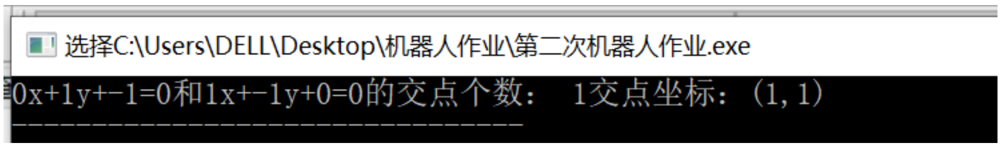
输入：直线 x+y=1 和直线 x=y


C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
```
1x+1y+-1=0和1x+-1y+0=0的交点个数： 1交点坐标：(0.5,0.5)
---------------------------------
```

输入：直线 x=1 和直线 x=y

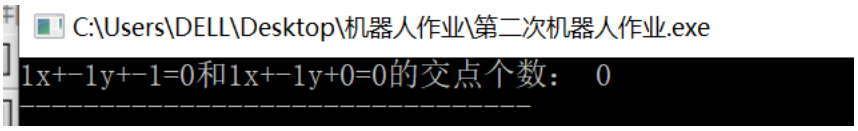

C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
```
1x+0y+-1=0和1x+-1y+0=0的交点个数： 1交点坐标：(1,1)
---------------------------------
```

输入：直线 y=1 和直线 x=y


选择C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
```
0x+1y+-1=0和1x+-1y+0=0的交点个数： 1交点坐标：(1,1)
---------------------------------
```

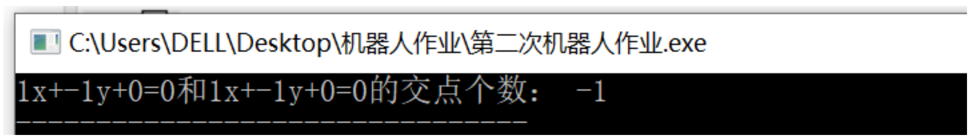
没有交点：

输入：直线 x=1+y 和直线 x=y


C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
```
1x+-1y+-1=0和1x+-1y+0=0的交点个数： 0
---------------------------------
```

直线重合：

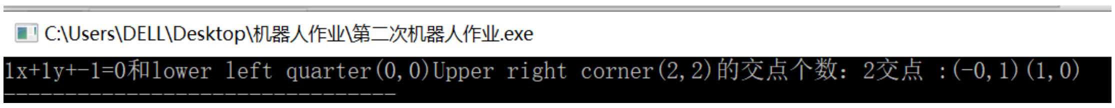
输入直线 x=y 和直线 x=y


C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
```
1x+-1y+0=0和1x+-1y+0=0的交点个数： -1
---------------------------------
```

直线和矩形交点：

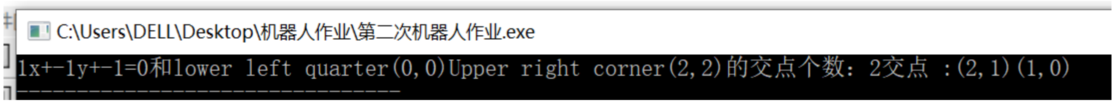
有两个交点：

输入：矩形左下角点（0,0）右上角点（2,2），直线 x+y=1



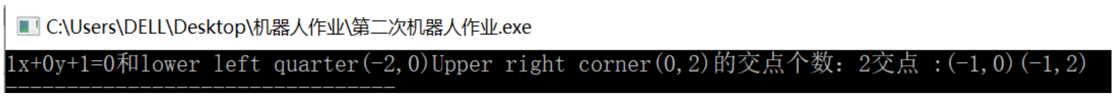
```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+1y+-1=0和lower left quarter(0,0)Upper right corner(2,2)的交点个数：2交点 :(-0,1)(1,0)
-----------------------------------------
```

输入：矩形左下角点（0,0）右上角点（2,2），直线 x-y=1



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+-1y+-1=0和lower left quarter(0,0)Upper right corner(2,2)的交点个数：2交点 :(2,1)(1,0)
-----------------------------------------
```

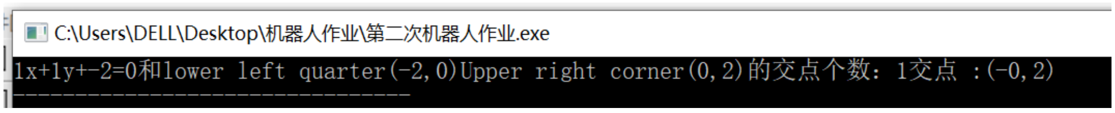
输入：矩形左下角（-2,0）右上角（0,2），直线 x=-1



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+0y+1=0和lower left quarter(-2,0)Upper right corner(0,2)的交点个数：2交点 :(-1,0)(-1,2)
-----------------------------------------
```

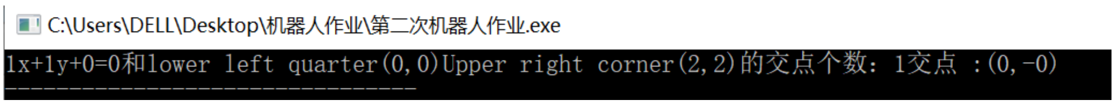
有一个交点：

输入：矩形左下角（-2,0）右上角（0,2），直线 x+y=2



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+1y+-2=0和lower left quarter(-2,0)Upper right corner(0,2)的交点个数：1交点 :(-0,2)
-----------------------------------------
```

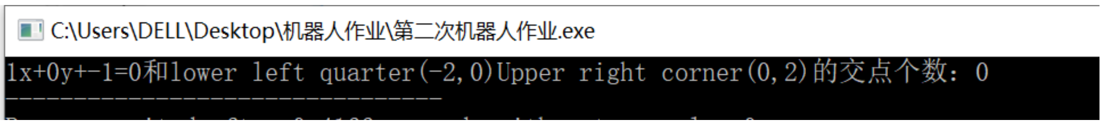
输入：矩形左下角点（0,0）右上角点（2,2），直线 x+y=0



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+1y+0=0和lower left quarter(0,0)Upper right corner(2,2)的交点个数：1交点 :(0,-0)
-----------------------------------------
```

没有交点：

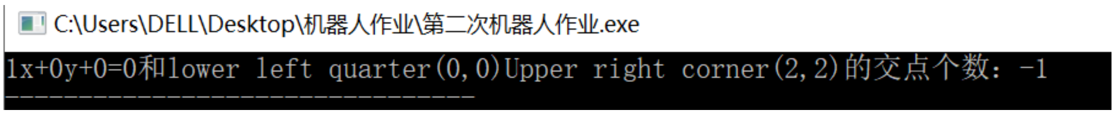
输入：矩形左下角（-2,0）右上角（0,2），直线 x=1



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+0y+-1=0和lower left quarter(-2,0)Upper right corner(0,2)的交点个数：0
-----------------------------------------
```

有无数个交点：

输入：矩形左下角点（0,0）右上角点（2,2），直线 x=0

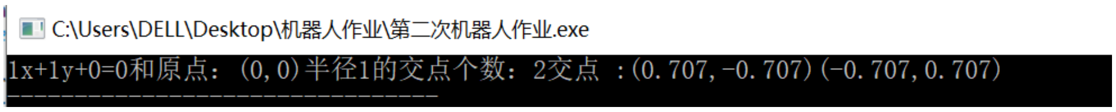


```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+0y+0=0和lower left quarter(0,0)Upper right corner(2,2)的交点个数：-1
-----------------------------------------
```

直线和圆

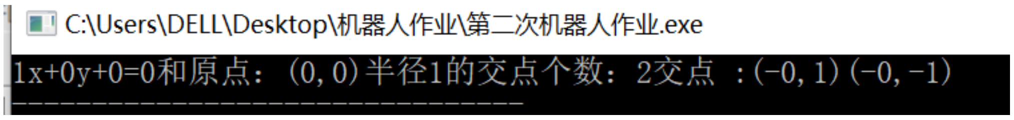
有两个交点：

输入：直线 x+y=0，圆（0,0）半径 1



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+1y+0=0和原点：(0,0)半径1的交点个数：2交点 :(0.707,-0.707)(-0.707,0.707)
--------------------------------
```

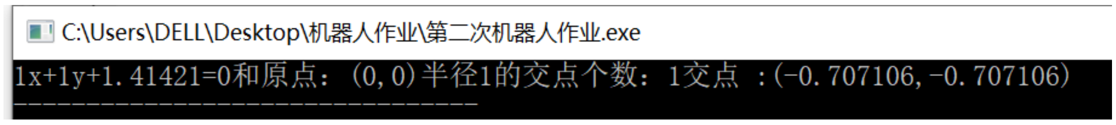
输入：直线 x=0，圆（0,0）半径 1



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+0y+0=0和原点：(0,0)半径1的交点个数：2交点 :(-0,1)(-0,-1)
--------------------------------
```

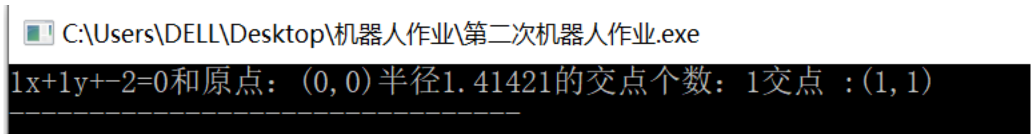
有一个交点：

输入：直线 x+y+1.414213=0，圆（0,0）半径 1



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+1y+1.41421=0和原点：(0,0)半径1的交点个数：1交点 :(-0.707106,-0.707106)
--------------------------------
```

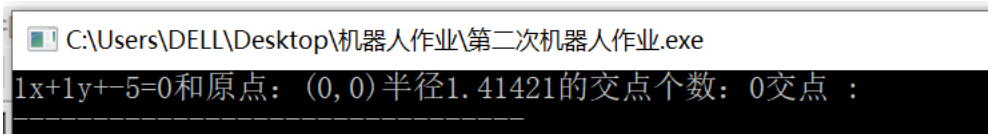
输入：直线 x+y-2=0，圆（0,0）半径 1.414213



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+1y+-2=0和原点：(0,0)半径1.41421的交点个数：1交点 :(1,1)
--------------------------------
```

没有交点：

输入：直线 x+y-5=0，圆（0,0）半径 1.414213



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
1x+1y+-5=0和原点：(0,0)半径1.41421的交点个数：0交点 :
--------------------------------
```

```cpp
#include <iostream>
#include <math.h>
#ifndef ROBOCUB_SECOND_HOMEWORK
using namespace std;
class point
{
private:
    double x;
    double y;
public:
    point(double xValue, double yValue);
    point();
    ~point();
    point & operator = (const point &p);
    inline void print(){
        cout << "(" << x << "," << y << ")";
    }

    friend class line;
    friend class rectangle;
    friend class circle;
};
point::point()
{
};
point::point(double xValue, double yValue):x(xValue),y(yValue)
{
};
point::~point()
{
};

point & point::operator = (const point &p)
{
    x = p.x;
    y = p.y;
    return *this;
}

class line
{
private:
    double A;
    double B;
    double C;

public:
    line(double AValue, double BValue, double CValue);
    line();
    ~line();
    friend int intersection_line(const line & FirstLine, const line & S
econdLine, point &t);
    friend class circle;
    line & operator = (const line & l);
    inline void print()
    {
        cout << A << "x+" << B << "y+" << C << "=0";
    }
    /**
     * 求点到直线的距离
     */
    inline double disOfPointToLine(const point &p)
    {
        return fabs((A*p.x + B*p.y + C)/sqrt(A*A+B*B));
    }
};

/**
 * return : -1 代表无数个交点，0 代表没有交点，1 代表一个交点
 */
int intersection_line(const line & FirstLine, const line & SecondLine,
point &t)
{
    double m = FirstLine.A * SecondLine.B - FirstLine.B * SecondLine.A;
    if (m == 0)
    {
        if (FirstLine.C == SecondLine.C && FirstLine.A == SecondLine.A
&& FirstLine.B == SecondLine.B)
        {
            return -1;
        }
        else
        {
            return 0;
        }
    }
    else
    {
        t = point(((-FirstLine.C) * SecondLine.B - FirstLine.B * (-
SecondLine.C))/m, (FirstLine.A * (-SecondLine.C) - (-
FirstLine.C) * SecondLine.A)/m);
        return 1;
    }
}


line & line::operator = (const line & l)
{
        A = l.A;
        B = l.B;
        C = l.C;
}

line::line()
{
};

line::line(double AValue, double BValue, double CValue)
{   if (A == 0&&B == 0)
    {
        cout << "This line don't exit!";
    }
    else
    {
        A = AValue;
        B = BValue;
        C = CValue;
    }
}

line::~line()
{
}


////////////////////////////////////////////////////////////////////////
/////////////

class rectangle
{
private:
    point lLim;
    point rLim;
```

```
public:
    line leftX;
    line rightX;
    line topY;
    line floorY;
    rectangle(point l,point r);
    rectangle(double xL, double yL, double xR, double yR);
    ~rectangle();

    inline void print()
    {cout<<"lower left quarter";lLim.print();cout<<"Upper right corner"
;rLim.print();}

    int intersection_rectangle(const line & line, point &p1, point &p2)
;
};

//左下右上两个点
rectangle::rectangle(point l, point r)
{
    lLim = l;
    rLim = r;

    line tLeftLim_x (1, 0, -lLim.x);
    line tRightLim_x (1, 0, -rLim.x);
    line tLeftLim_y (0, 1,-lLim.y);
    line tRightLim_y (0, 1,-rLim.y);

    leftX = tLeftLim_x;
    rightX = tRightLim_x;
    topY = tRightLim_y;
    floorY = tLeftLim_y;

    /*
    leftX.print();
    rightX.print();
    topY.print();
    floorY.print(); */
};

rectangle::rectangle(double xL, double yL, double xR, double yR)
{
    point l (xL, yL);
    point r (xR, yR);

    lLim = l;
    rLim = r;

    line tLeftLim_x (1, 0, -lLim.x);
    line tRightLim_x (1, 0, -rLim.x);
    line tLeftLim_y (0, 1,-lLim.y);
    line tRightLim_y (0, 1,-rLim.y);

    leftX = tLeftLim_x;
    rightX = tRightLim_x;
    topY = tLeftLim_y;
    floorY = tRightLim_y;

    /*
    leftX.print();
    rightX.print();
    topY.print();
    floorY.print(); */
}

rectangle::~rectangle()
{
};

int rectangle::intersection_rectangle(const line & line, point &p1, poi
nt &p2)
{
    point t[4] = {point(65535,65535),point(65535,65535),point(65535,655
35),point(65535,65535)};

    if (intersection_line(line,leftX,t[0]) == -1)
    {
        return -1;
    }
    if (intersection_line(line,rightX,t[1]) == -1)
    {
        return -1;
    }
    if (intersection_line(line,topY,t[2]) == -1)
    {
        return -1;
    }
    if (intersection_line(line,floorY,t[3]) == -1)
    {
        return -1;
    }

    int j = 0;
    int m = 0;
    if (t[0].y >= lLim.y && t[0].y <= rLim.y)
    {

        if (j == 0)
        {
            p1 = t[0];
        }
        else
        {
            p2 = t[0];
        }
        j++;
        m++;
    }

    if (t[1].y >= lLim.y && t[1].y <= rLim.y)
    {

        if (j == 0)
        {
            p1 = t[1];
        }
        else
        {
            p2 = t[1];
        }
        j++;
        m++;
    }

    if (t[2].x > lLim.x && t[2].x < rLim.x)
    {

        if (j == 0)
        {
            p1 = t[2];
        }
        else
        {
            p2 = t[2];
        }
        j++;
        m++;
    }

    if (t[3].x > lLim.x && t[3].x < rLim.x)
    {

        if (j == 0)
        {
            p1 = t[3];
        }
        else
        {
            p2 = t[3];
        }
        j++;
        m++;
    }

    return m;
}

inline double myRound(double x)
{
    return floor(x*1000+0.5)/1000.0;
}

class circle
{
private:
    point centre;
    double R;
public:
    circle(point c, double r);
    ~circle();

    inline void print()
    {cout<<"圆心坐标：";centre.print();cout<<"半径："<<R;}

    int intersection(line & l,point &p1, point &p2);
```

```
};

int circle::intersection(line & l,point &p1, point &p2)
{
    double dis = l.disOfPointToLine(centre);
    //cout << "*" << dis;
    line m (l.B,-l.A,-l.B*centre.x + l.A*centre.y);
    //m.print();
    if (myRound(dis) > myRound(R))
    {
        return 0;
    }
    else if (myRound(dis) < myRound(R))
    {
        point s (0,0);
        point s1 (0,0);
        point s2 (0,0);
        intersection_line(l,m,s);

        if ( myRound(l.B) == 0)
        {
            s1.x = s.x;
            s2.x = s.y;
            s1.y = sqrt(R*R - dis*dis) + s.y;
            s2.y = -sqrt(R*R - dis*dis) + s.y;
        }
        else
        {
            double sinValue = sin(atan(-l.A/l.B));
            double cosValue = cos(atan(-l.A/l.B));
            s1.x = myRound(sqrt(R*R - dis*dis)*cosValue + s.x);
            s2.x = myRound(-sqrt(R*R - dis*dis)*cosValue + s.x);
            s1.y = myRound(sqrt(R*R - dis*dis)*sinValue + s.y);
            s2.y = myRound(-sqrt(R*R - dis*dis)*sinValue + s.y);
            /*
            s2.x = -sqrt(R*R - dis*dis)*cosValue + s.x;
            s1.y = sqrt(R*R - dis*dis)*sinValue + s.y;
            s2.y = -sqrt(R*R - dis*dis)*sinValue + s.y;
            */
        }

        p1 = s1;
        p2 = s2;
        return 2;
    }
    else
    {
        intersection_line(l,m,p1);
        return 1;
    }
}

circle::circle(point c, double r)
{
    centre = c;
    R = r;
}

circle::~circle()
{
}


#endif // !ROBOCUB_SECOND_HOMEWORK
```