源码

```
#include <iostream>
#include<cmath>
using namespace std;
class Point
{
   public:
   double x;
   double y;
   bool sign;  //设置为这个点是否是交点
   Point(double x,double y):x(x),y(y)
   {sign = false;}
   Point(){sign = false;}
   void print()
   {
      cout<<"交点为"<<endl;
      cout<<"("<<x<<','<<y<<")"<<endl;
   }
};
class line
{
public:
double A;
double B;
double C;

line(double A,double B,double C)
{
   if(A==0&&B==0)
   {
      cout<<"false"<<endl;
   }
   else
   {
      this->A = A;
      this->B = B;
      this->C = C;
   }

}

//得到交点
void  getPoint(line &line1,Point *p);
};
void  line::getPoint(line &line1,Point *p)
{
```

```
    //判断平行或者重合
    if(A == 0&&line1.A == 0&&line1.B!=0&&B!=0)
    {
        p ->sign = false;
        return;
    }
    if(B==0&&line1.B==0&&line1.A!=0&&A!=0)
    {
        p ->sign = false;
        return;
    }
    if(B!=0)
    {
            double A1 = line1.A;
            double B1 = line1.B;
            double C1 = line1.C;
        if(line1.B!=0)
        {
            //判断平行或者重合
            if(fabs(A/B-line1.A/line1.B)<1e-6)
            {
                p->sign = false;
                return;
            }
            p->sign = true;
            p->x = (B*C1/B1-C)/(A-A1*B/B1);
            p->y = -A1/B1*p->x-C1/B1;
            return ;
        }
        else
        {
            p->sign = true;
            p->x = -C1/A1;
            p->y = -A/B*p->x-C/B;
            return;
        }
    }
    if (line1.B!=0)
    {
        p->sign = true;
        p->x = -C/A;
        p->y = line1.A*C/(line1.B*A) - line1.C/line1.B;
    }
}


class circle
{
    public:
```

```
    double R;  //半径
    Point P0;    //圆点
    circle(Point &p,double r):P0(p),R(r) {}
    void getPoint_Line_circle(line &l,Point *p1,Point *p2);
};

void circle::getPoint_Line_circle(line &l,Point *p1,Point *p2)
{
    double X0 = P0.x;
    double Y0 = P0.y;
    double A = l.A;
    double B = l.B;
    double C = l.C;
    double  d = fabs(A*X0+B*Y0+C)/sqrt(A*A+B*B); //表示圆点到直线的距离
    if(d > R) //圆点到直线的距离大于R，没有交点
    {
        p1->sign = p2->sign = false;
        return ;
    }
    //当斜率不存在时，单独求点
    p1->sign = true;
    p2->sign = true;
    if(l.B==0)
    {
        p1->x = p2->x =  -l.C/l.A;
    p1->y =  Y0+sqrt(R*R-(l.C/A+X0)*(l.C/A));
    p2->y =  Y0-sqrt(R*R-(l.C/A+X0)*(l.C/A));
    return ;
    }

    //获得二次方程的a,b,c计算坐标
    double a = 1+(A/B)*(A/B);
    double b = 2*A*C/(B*B) - 2*X0 + 2*Y0*A/B;
    double c = X0*X0 + (C/B)*(C/B) - R*R+2*Y0*C/B;
    d = b*b-4*a*c;

    if(fabs(d) < 1e-6)
    {
        p2->x = p1->x = -b/(2*a);
        p2->y = p1->y = -A/B*p1->x - C/B;
    }
    else
    {
        p1->x = (-b+sqrt(d))/(2*a);
        p2->x =  (-b-sqrt(d))/(2*a);
        p1->y = -A/B*p1->x - C/B;
        p2->y = -A/B*p2->x - C/B;p1->sign = p2->sign = true;
    }
}
```

```
class rectangle
{
  //默认左下和右上两点确定矩形
  Point p1;
  Point p2;
  public:
rectangle(double x1,double y1,double x2,double y2)
  {
    p1.x = x1;
    p1.y = y1;
    p2.x = x2;
    p2.y = y2;
  }
  rectangle(Point &p1,Point&p2)
  {
    this->p1.x = p1.x;
    this->p1.y = p1.y;
    this->p2.x = p2.x;
    this->p2.y = p2.y;
  }
  void getPoint_line_rectangle(line &l ,Point *p1,Point *p2);
  bool judgePointInRectangle(Point *p);
};

void rectangle::getPoint_line_rectangle( line &l,Point *p3,Point *p4)
{
  double A = l.A;
  double B = l.B;
  double C = l.C;
  if(l.B == 0)
  {
    //如果直线在矩形范围内
    double x = -C/A;
    if( x >p1.x&&x < p2.x)
    {
      p3->sign = p4->sign = true;
      p3->x = p4->x = x;
      p3->y = p1.y;
      p4->y = p2.y;
      return;
    }
    else if(fabs(x-p1.x)<1e-6||fabs(x-p2.x)<1e-6)
    {
      cout<<"直线和矩形的边界重合"<<endl;
      p3 ->sign = p4->sign = false;

    }
    {
      p3 ->sign = p4->sign = false;
```

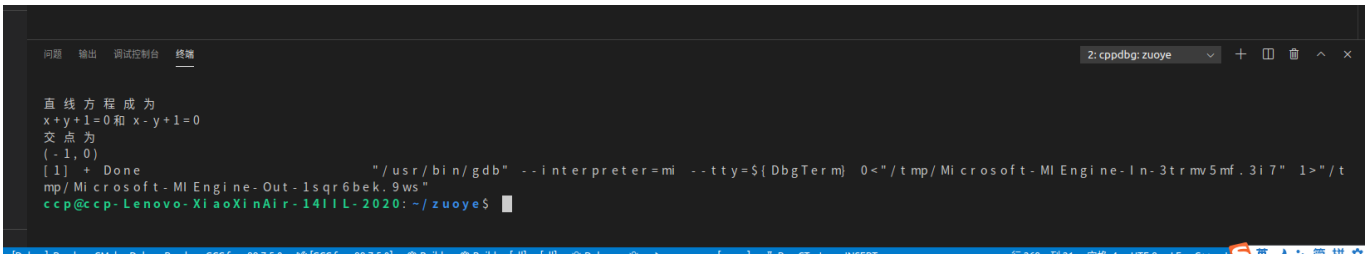
```
            return;
        }
    }
else if(l.A == 0)
    {
        double y = -C/B;
        if(y>p1.y&&y<p2.y)
        {
            p3->sign = p4->sign = true;
            p3->y = p4->y = y;
            p3->x = p1.x;
            p4->y = p2.y;
            return;
        }
        else if(fabs(y-p1.y) < 1e-6||fabs(y-p2.y) < 1e-6)
        {
            cout<<"直线和矩形边界重合"<<endl;
            p3 ->sign = p4->sign = false;
        }
        else
        {
            p3->sign = p4->sign = false;
            return;
        }
    }
    else
    {
        //求出直线和组成矩形的四条直线的交点，判断交点是否在矩形范围内就可以得出矩形和直线的交
点
        line l_x1(1,0,-p1.x);
        line l_x2(1,0,-p2.x);
        line l_y1(0,1,-p1.y);
        line l_y2(0,1,-p2.y);
        Point *P[4];
        int sign = 0; //判断是否已经有交点
        for (int i = 0;i < 4;++i)
        {
            P[i] = new Point();
        }
            l.getPoint(l_x1,P[0]);
            l.getPoint(l_x2,P[1]);
            l.getPoint(l_y1,P[2]);
            l.getPoint(l_y2,P[3]);
        for(int i = 0;i < 4;++i)
        {
            if(P[i]->sign&&judgePointInRectangle(P[i]))
            {
                if(sign == 0)
                {
```

```
                p3 ->x = P[i]->x;
                p3->y = P[i]->y;
                p4->x = P[i]->x;
                p4->y = P[i]->y;
                sign = 1;
            }
            else if(sign==1)
            {
                p4->x = P[i]->x;
                p4->y = P[i]->y;
                sign++;
            }
        }
        delete P[i];
    }
    if(sign == 0)
    {
        p3->sign = p4->sign = false;
    }
    else
    {
        p3->sign = p4->sign = true;
    }
  }
}
bool rectangle::judgePointInRectangle(Point *p)
{
  bool aboutX1 = p->x>p1.x||fabs(p->x-p1.x) < 1e-6;
  bool aboutX2 = p->x<p2.x||fabs(p->x-p2.x) < 1e-6;
  bool aboutY1 = p->y>p1.y||fabs(p->y-p1.y) < 1e-6;
  bool aboutY2 = p->y<p2.y||fabs(p->y-p2.y) < 1e-6;
  if(aboutX1&&aboutX2&&aboutY1&&aboutY2)
  {
    return true;
  }
  else
  {
    return false;
  }

}

int main(int, char**) {

//测试
line l1(1,1,-1);
//line l2(1,-1,2);
Point *p = new Point();
cout<<"直线方程成为"<<endl;
```

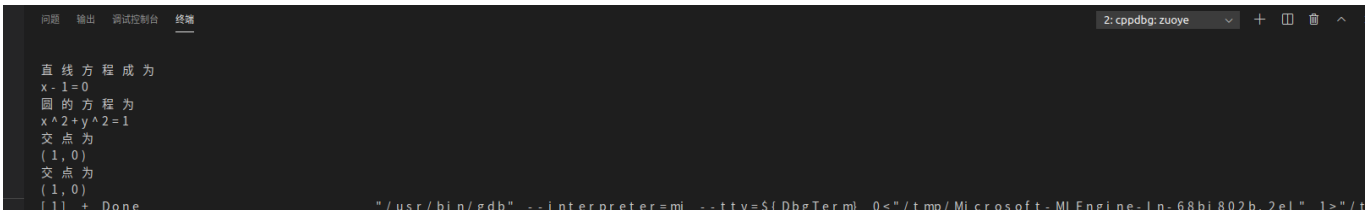
```
cout<<"x+y-1=0"<<endl;
/*cout<<"圆的方程为"<<endl;
cout<<"(x-1)^2+(y-1)^2=1"<<endl;*/
Point p0(2,1);
Point p3(1,0);
Point *p1 = new Point();
Point *p2  = new Point();
//circle c(p0,1);
//c.getPoint_Line_circle(l1,p1,p2);
rectangle r(p3,p0);
r.getPoint_line_rectangle(l1,p1,p2);
cout << "矩形的左下坐标为(1,),右上坐标为(2,1)"<<endl;
if(p1->sign&&p2->sign)
{
   p1->print();
   p2->print();
}
else
{
   cout<<"没有交点或直线与边界重合有无数个交点"<<endl;
}

delete p;
delete p1,p2;
}
```

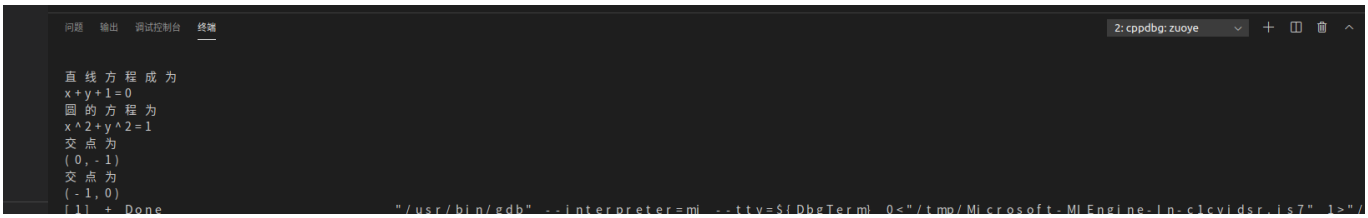
结果示例

直线



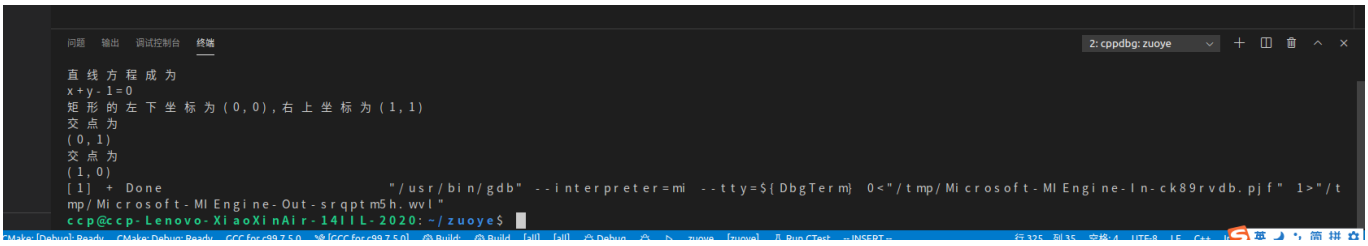
圆






```
[Running] cd "/home/ccp/zuoye/" && g++ main.cpp -o main && "/home/ccp/zuoye/"main
直线方程成为
x+y+1=0
圆的方程为
(x-1)^2+(y-1)^2=1
没有交点或者与边界重合有无数个交点

[Done] exited with code=0 in 0.231 seconds
```

矩形




```
[Running] cd "/home/ccp/zuoye/" && g++ main.cpp -o main && "/home/ccp/zuoye/"main
直线方程成为
x+y-1=0
矩形的左下坐标为(1,0),右上坐标为(2,1)
交点为
(1,0)
交点为
(1,0)
```

```
[Done] exited with code=0 in 0.241 seconds

[Running] cd "/home/ccp/zuoye/" && g++ main.cpp -o main && "/home/ccp/zuoye/"main
直线方程成为
x+y-1=0
矩形的左下坐标为(1,1),右上坐标为(2,2)
没有交点或者与边界重合有无数个交点

[Done] exited with code=0 in 0.245 seconds
```