class MyCircle2D {

public:

MyCircle2D(double x, double y, double r);

double getCirX();

double getCirY();

double getCirR();

private:

double x;

double y; //圆心坐标

double r; //半径

};

class MyRectangle2D {

public:

MyRectangle2D(double xLeftB,

double yLeftB,

double xRightT,

double yRightT);

double getRecXL();

double getRecYL();

double getRecXR();

double getRecYR();

private:

double xLeftB;

double yLeftB;

double xRightT;

double yRightT; //分别代表矩形左下角和右上角的坐标

};

class MyLine2D {

public:

MyLine2D(double a, double b, double c);

void PointOfIntersection(MyLine2D &line);

void PointOfIntersection(MyCircle2D &circle);

void PointOfIntersection(MyRectangle2D &rectangle);//求交点

double getLineA();

double getLineB();

double getLineC();

private:

double a;

double b;

double c; //ax + by + c = 0;

};

/\*file:robocup2d2cd.cpp

compiler: VS Code

function: 求直线与直线、直线与圆、直线与矩形的交点

\*/

#include<iostream>

#include<math.h>

#include"robocup2d2cd.h"

using namespace std;

MyLine2D::MyLine2D(double a, double b, double c){

this->a = a;

this->b = b;

this->c = c;

}

double MyLine2D::getLineA(){

return this->a;

}

double MyLine2D::getLineB(){

return this->b;

}

double MyLine2D::getLineC(){

return this->c;

}

void MyLine2D::PointOfIntersection(MyLine2D &line){ //直线与另一条直线的交点

if((this->a==0 && line.a==0) || (this->b==0 && line.b==0)){

cout<<"wrong line!"<<endl;

}

if((this->a)\*(line.b) != (this->b)\*(line.a)){ //两直线相交

cout<<"两直线相交,交点：("

<<((this->b)\*(line.c) - (this->c)\*(line.b))/((this->a)\*(line.b) - (this->b)\*(line.a))

<<","

<<((this->c)\*(line.a) - (this->a)\*(line.c))/((this->a)\*(line.b) - (this->b)\*(line.a))

<<")"<<endl;

}else {

if(this->a!=0 && line.a!=0){

if((this->a)/(line.a) == (this->c)/(line.c)) {

cout<<"两直线重合,有无数交点"<<endl;

}else {

cout<<"两直线平行--无交点"<<endl;

}

}else if(this->b!=0 && line.b!=0){

if((this->b)/(line.b) == (this->c)/(line.c)) {

cout<<"两直线重合,有无数交点"<<endl;

}else {

cout<<"两直线平行,无交点"<<endl;

}

}

}

}

void MyLine2D::PointOfIntersection(MyCircle2D &circle){

if(this->a==0 && this->b==0){

cout<<"wrong line!"<<endl;

}

double dist; //圆心到直线的距离

double k;

double d;

double f;

dist = fabs((this->a)\*(circle.getCirX()) + (this->b)\*(circle.getCirY()) + this->c)

/sqrt((this->a)\*(this->a) + (this->b)\*(this->b));

if(this->b != 0){//斜率存在时,转化

k = (-1)\*this->a/this->b;

d = (-1)\*this->c/this->b; //截距

f = sqrt((k\*k+1)\*circle.getCirR()\*circle.getCirR()

- circle.getCirX()\*circle.getCirX()\*k\*k

+ 2\*k\*(circle.getCirX()\*circle.getCirY()+d\*circle.getCirX())

- circle.getCirY()\*circle.getCirY()-2\*d\*circle.getCirY()-d\*d);

}

if(dist == circle.getCirR()){

cout<<"直线与圆相切，一个交点，交点：";

if(this->a == 0){

cout<<"("<<circle.getCirX()<<","<<(-1)\*(this->c/this->b)<<")"<<endl;

}else if(this->b == 0){

cout<<"("<<(-1)\*(this->c/this->a)<<","<<circle.getCirY()<<")"<<endl;

}else {

cout<<"("<<((-1)\*f+(circle.getCirY()+d)\*k+circle.getCirX())/(k\*k+1)<<","

<<((-1)\*(k\*(f+circle.getCirX())+circle.getCirY()\*k\*k-d)/(k\*k+1))<<")"<<endl;

}

}else if(dist > circle.getCirR()){

cout<<"直线与圆相离，无交点"<<endl;

}else {

cout<<"直线与圆相交，两个交点，交点：";

double f1 = sqrt(circle.getCirR()\*circle.getCirR()-(dist/2)\*(dist/2));

if(this->a == 0){

cout<<"("<<circle.getCirX()-f1<<","<<d<<"),"

<<"("<<circle.getCirX()+f1<<","<<d<<")"<<endl;

}else if(this->b == 0){

cout<<"("<<(-1)\*this->c/this->a<<","<<circle.getCirY()-f1<<"),"

<<"("<<(-1)\*this->c/this->a<<","<<circle.getCirY()+f1<<")"<<endl;

}else{

cout<<"("<<((-1)\*f+(circle.getCirY()+d)\*k+circle.getCirX())/(k\*k+1)<<","

<<((-1)\*(k\*(f+circle.getCirX())+circle.getCirY()\*k\*k-d)/(k\*k+1))<<")"

<<","

<<"("<<(f+(-1)\*(circle.getCirY()+d)\*k-circle.getCirX())/(k\*k+1)<<","

<<k\*((f+(-1)\*(circle.getCirY()+d)\*k-circle.getCirX())/(k\*k+1))+d<<")"<<endl;

}

}

}

void MyLine2D::PointOfIntersection(MyRectangle2D &rec){

if(this->a==0 && this->b==0){

cout<<"wrong line!"<<endl;

}

//假设矩形的边平行于坐标轴

double kRec;

double bRec; //矩形对角线所在的直线的斜率，截距

double kLine; //直线斜率存在时的斜率，截距

double bLine;

kRec = tan((rec.getRecYR()-rec.getRecYL())/(rec.getRecXR()-rec.getRecXL()));

bRec = rec.getRecYL() - kRec\*rec.getRecXL();

if(this->b != 0){//直线斜率存在

kLine = (-1)\*this->a/this->b;

bLine = (-1)\*this->c/this->b;

double y1 = kLine\*rec.getRecXL()+bLine;

double y2 = kLine\*rec.getRecXR()+bLine;

//矩形顶点到对角线的距离

double dist = (rec.getRecXR()-rec.getRecXL())\*(rec.getRecYR()-rec.getRecYL())

/sqrt((rec.getRecXR()-rec.getRecXL())\*(rec.getRecXR()-rec.getRecXL())+

(rec.getRecYR()-rec.getRecYL())\*(rec.getRecYR()-rec.getRecYL()));

//矩形的左下角点到直线距离

double dist1 = fabs(this->a\*rec.getRecXL()+this->b\*rec.getRecYL()+this->c)

/sqrt(this->a\*this->a + this->b\*this->b);

//矩形的右上角点到直线距离

double dist2 = fabs(this->a\*rec.getRecXR()+this->b\*rec.getRecYR()+this->c)

/sqrt(this->a\*this->a + this->b\*this->b);

//对角线段（除去端点）与直线相交

if((rec.getRecYL()<y1 && rec.getRecYR()>y2) || (rec.getRecYL()>y1 && rec.getRecYR()<y2)){

cout<<"直线与矩形有两个交点:";

if(kLine > 0){

if(kLine > kRec){

cout<<"("<<(rec.getRecYL()-bLine)/kLine<<","<<rec.getRecYL()<<")"

<<"("<<(rec.getRecYR()-bLine)/kLine<<","<<rec.getRecYR()<<")"<<endl;

}else {

cout<<"("<<rec.getRecXL()<<","<<kLine\*rec.getRecXL()<<"),"

<<"("<<rec.getRecXR()<<","<<kLine\*rec.getRecXR()<<")"<<endl;

}

}else if(kLine < 0){

if(kLine\*rec.getRecXR()+bLine > rec.getRecYL()){

if(kLine\*rec.getRecXL()+bLine < rec.getRecYL()){

cout<<"("<<rec.getRecXL()<<","<<kLine\*rec.getRecXL()<<"),"

<<"("<<rec.getRecXR()<<","<<kLine\*rec.getRecXR()<<")"<<endl;

}else{

cout<<"("<<(rec.getRecYR()-bLine)/kLine<<","<<rec.getRecYR()<<")"

<<"("<<rec.getRecXR()<<","<<rec.getRecXR()\*kLine+bLine<<")"<<endl;

}

}else {

cout<<"("<<(rec.getRecYL()-bLine)/kLine<<","<<rec.getRecYL()<<")"

<<"("<<rec.getRecXL()<<","<<rec.getRecXL()\*kLine+bLine<<")"<<endl;

}

}

}else if(rec.getRecYL()<y1 && rec.getRecYR()<y2){//两点在直线的下方

if(kLine\*rec.getRecXL()+bLine > rec.getRecYR()){

cout<<"直线与矩形没有交点"<<endl;

}else if(kLine\*rec.getRecXL()+bLine == rec.getRecYR()){

cout<<"直线与矩形有一个交点:";

cout<<"("<<rec.getRecXL()<<","<<rec.getRecYR()<<")"<<endl;

}else {

cout<<"直线与矩形有两个交点:";

cout<<"("<<rec.getRecXL()<<","<<kLine\*rec.getRecXL()+bLine<<"),"

<<"("<<(rec.getRecYR()-bLine)/kLine<<","<<rec.getRecYR()<<")"<<endl;

}

}else if(rec.getRecYL()>y1 && rec.getRecYR()>y2){//两点在直线的上方

if(kLine\*rec.getRecXR()+bLine < rec.getRecYL()){

cout<<"直线与矩形没有交点"<<endl;

}else if(kLine\*rec.getRecXR()+bLine == rec.getRecYL()){

cout<<"直线与矩形有一个交点:";

cout<<"("<<rec.getRecXR()<<","<<rec.getRecYL()<<endl;

}else {

cout<<"直线与矩形有两个交点:";

cout<<"("<<(rec.getRecYL()-bLine)/kLine<<","<<rec.getRecYL()<<"),"

<<"("<<rec.getRecXR()<<","<<kLine\*rec.getRecXR()+bLine<<")"<<endl;

}

}

}else{

bLine = (-1)\*this->a/this->c;

if(bLine>rec.getRecXR() || bLine<rec.getRecXL()){

cout<<"直线与矩形没有交点"<<endl;

}else if(bLine<rec.getRecXR() || bLine>rec.getRecXL()){

cout<<"直线与矩形有两个交点:";

cout<<"("<<bLine<<","<<rec.getRecYL()<<"),"

<<"("<<bLine<<","<<rec.getRecYR()<<")"<<endl;

}else{

cout<<"直线与矩形一条边重合，有无数交点"<<endl;

}

}

}

MyCircle2D::MyCircle2D(double x, double y, double r){

this->x = x;

this->y = y;

this->r = r;

}

double MyCircle2D::getCirX(){

return this->x;

}

double MyCircle2D::getCirY(){

return this->y;

}

double MyCircle2D::getCirR(){

return this->r;

}

MyRectangle2D::MyRectangle2D(double x1, double y1,

double x2, double y2){

this->xLeftB = x1;

this->yLeftB = y1;

this->xRightT = x2;

this->yRightT = y2;

}

double MyRectangle2D::getRecXL(){

return this->xLeftB;

}

double MyRectangle2D::getRecYL(){

return this->yLeftB;

}

double MyRectangle2D::getRecXR(){

return this->xRightT;

}

double MyRectangle2D::getRecYR(){

return this->yRightT;

}

int main(){

MyLine2D line1(2.0, 3.0, 1.0);

MyLine2D line2(1.0, 4.0, 8.0);

MyLine2D line3(4.0, -2.0, 3.0);

MyLine2D line4(0.0, 3.0, 8.0);

MyCircle2D circle(2.0, 2.0, 2.0);

MyRectangle2D rec(-4.0, 0.0, 0.0, 3.0);

line1.PointOfIntersection(line2);

line1.PointOfIntersection(circle);

line1.PointOfIntersection(rec);

line2.PointOfIntersection(line3);

line2.PointOfIntersection(circle);

line2.PointOfIntersection(rec);

line3.PointOfIntersection(line4);

line3.PointOfIntersection(circle);

line3.PointOfIntersection(rec);

return 0;

}