**第三次作业**

**说明：程序中输入信息处提供有部分样例，复制到msg[50]处即可进行测试。**

**思路：通过圆方程求交点得到两个疑似位置，用向量叉积来甄别正确点。**

#include <iostream>

#include <cstdio>

#include <math.h>

#include <string>

#include <cstring>

#include <stdio.h>

#include <iostream>

#include <bits/stdc++.h>

#define PI 3.1415926

#define EPS 1e-6

**using** **namespace** std**;**

struct Object**{**//场上标志结构

string name**;**

float x**;**

float y**;**

**};**

struct Point**{**//点结构

float x**;**

float y**;**

**};**

string seeobj**[**200**];**//存储输入信息

Object obj**[**14**];**//存储标志点

Point pt**[**3**];**//存储可能的位置

void split**(**char str**[],**char delims**[]);**//分割字符串

void initobj**(**Object a**[]);**//初始化点集数据

void printobj**(**Object a**[]);**//打印点集

Object getobj**(**string str**);**//通过输入返回固定点坐标

void InterCircles**(**float fr**,** float gr**,** float fx**,** float fy**,** float gx**,** float gy**);**//求两个可能交点

void printPoint**();**//打印可能的点

float LR**(**Point P1**,** Point P2**,** Point P3**);**//面积量判定点线位置关系

Point solve2**(**float t**,** Point A**,** Point B**,** Point C **,** Point D**);**//甄别函数2

float normalize**(**float t1**,** float t2**);**//夹角规范化

int main**(){**

int i **=** 1**;**

float fr**,** ft**,** gr**,** gt**;**//分别为两个点的相对距离和极角

float fx**,** fy**,** gx**,** gy**;**//分别为两个点的绝对坐标

float ang**;**//角度

Point A**,** C**;**//规范化需要的中间量

Point B**,** D**;**

Point result**;**//最终返回的结果点

char msg**[**50**];**

cout **<<** "请输入测试点！\n"**;**

cin**.**get**(**msg**,** 50**);**

//char msg[50] = "(P10 7 135) (G2 22.5 -135)";//也可以直接把信息赋给数组

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 部分测试点集 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//以下左侧是输入信息 右侧是所得点

/\*

(P8 22 0) (P7 27.7128 30) (-8.22565 10.1429)

(C 7 45) (P8 30 -45) (0 7)

(P8 7 135) (C 30 45) (-30 0)

(P10 7 135) (G2 22.5 -135) (30 0)

(P6 10 0) (P3 62.5 0) (-10 32)

(P9 22.5 45) (G2 7 135) (52.5 7)

(P8 39.051248 -56.22368) (P9 39.051248 56.22368) (0 , -18)

\*/

char delims**[]** **=** ") ("**;**

Object F**,** G**;**//两个观测点

split**(**msg**,** delims**);**

initobj**(**obj**);**

//printobj(obj);

//以下内容是从输入获得数据

F **=** getobj**(**seeobj**[**1**])** **;**

G **=** getobj**(**seeobj**[**4**])** **;**

fx **=** F**.**x**;**

fy **=** F**.**y**;**

gx **=** G**.**x**;**

gy **=** G**.**y**;**

fr **=** atof**(**seeobj**[**2**].**c\_str**());** //参数赋值

ft **=** atof**(**seeobj**[**3**].**c\_str**());**

gr **=** atof**(**seeobj**[**5**].**c\_str**());**

gt **=** atof**(**seeobj**[**6**].**c\_str**());**

//cout << fr << " " << ft << " "<< gr << " "<< gt;

**if(**gy **<** fy**)**//得到中间量A,C 视情况规范角度

**{**

A**.**x **=** G**.**x**;**

A**.**y **=** G**.**y**;**

C**.**x **=** F**.**x**;**

C**.**y **=** F**.**y**;**

ang **=** normalize**(**ft**,** gt**);**

**}**

**else**

**{**

C**.**x **=** G**.**x**;**

C**.**y **=** G**.**y**;**

A**.**x **=** F**.**x**;**

A**.**y **=** F**.**y**;**

ang **=** normalize**(**gt**,** ft**);**

**}**

//cout << " A点 "<< A.x << " " << A.y << "\n C点 "<< C.x << " " << C.y << "\n";

//数据获取完成，以下开始处理

InterCircles**(**fr**,** gr**,** fx**,** fy**,** gx**,** gy**);**//获取两个可能位置存入Point[]

**if(**pt**[**1**].**x **<** pt**[**2**].**x**)** //视横坐标选取中间量B,D

**{**

B **=** pt**[**1**];**

D **=** pt**[**2**];**

**}**

**else**

**{**

B **=** pt**[**2**];**

D **=** pt**[**1**];**

**}**

cout **<<** "\nB点是： (" **<<** B**.**x **<<** " , " **<<** B**.**y **<<** " )\n"**;**

//角度规范化

cout **<<** "规范化夹角是：" **<<** ang **<<** endl**;**

float t **=** fr **\*** gr **\*** sin**(**ang **/** 180 **\*** PI**);** //叉积的一种算法

cout **<<** "向量叉积1是：" **<<** t **<<** endl**;**

result **=** solve2**(**t**,** A**,** B**,** C**,** D**);**//判定函数 传入一种叉积和若干中间量供甄别 返回求得的点

cout **<<** "\n机器人的坐标是： ( " **<<** result**.**x **<<** " , " **<<** result**.**y **<<** " )"**;**

**}**

void initobj**(**Object a**[])**//初始化点集数据

**{**

a**[**1**].**name **=** "C"**;**

a**[**1**].**x **=** 0.0f**;**

a**[**1**].**y **=** 0.0f**;**

a**[**2**].**name **=** "P1"**;**

a**[**2**].**x **=** **-**52.5f**;**

a**[**2**].**y **=** **-**32**;**

a**[**3**].**name **=** "P2"**;**

a**[**3**].**x **=** **-**52.5f**;**

a**[**3**].**y **=** 32**;**

a**[**4**].**name **=** "P3"**;**

a**[**4**].**x **=** 52.5f**;**

a**[**4**].**y **=** 32**;**

a**[**5**].**name **=** "P4"**;**

a**[**5**].**x **=** 52.5f**;**

a**[**5**].**y **=** **-**32**;**

a**[**6**].**name **=** "P5"**;**

a**[**6**].**x **=** 0.0f**;**

a**[**6**].**y **=** **-**32**;**

a**[**7**].**name **=** "P6"**;**

a**[**7**].**x **=** 0.0f**;**

a**[**7**].**y **=** 32**;**

a**[**8**].**name **=** "P7"**;**

a**[**8**].**x **=** **-**30**;**

a**[**8**].**y **=** **-**7**;**

a**[**9**].**name **=** "P8"**;**

a**[**9**].**x **=** **-**30**;**

a**[**9**].**y **=** 7**;**

a**[**10**].**name **=** "P9"**;**

a**[**10**].**x **=** 30**;**

a**[**10**].**y **=** 7**;**

a**[**11**].**name **=** "P10"**;**

a**[**11**].**x **=** 30**;**

a**[**11**].**y **=** **-**7**;**

a**[**12**].**name **=** "G1"**;**

a**[**12**].**x **=** **-**52.5f**;**

a**[**12**].**y **=** 0**;**

a**[**13**].**name **=** "G2"**;**

a**[**13**].**x **=** 52.5f**;**

a**[**13**].**y **=** 0**;**

**}**

void printobj**(**Object a**[])**//打印参考点集

**{**

int i **=** 1**;**

**for(**i **=** 1**;** i **<** 14 **;** i**++)**

**{**

cout **<<** a**[**i**].**name **<<** " " **<<** a**[**i**].**x **<<** " " **<<** a**[**i**].**y **<<** "\n"**;**

**}**

**}**

void split**(**char str**[],**char delims**[])**//分割字符串

**{**

char **\***result **=** **NULL;**

int i **=** 1**;**

result **=** strtok**(**str**,** delims**);**

**while(**result **!=** **NULL** **)**

**{**

//printf("result is \"%s\"\n", result);

//cout << result << "\n";

seeobj**[**i**]** **=** result**;**

result **=** strtok**(** **NULL,** delims**);**

i**++;**

**}**

**}**

Object getobj**(**string str**)**//给定名称返回obj结构体

**{**

**for(**int i **=** 1**;** i **<** 14**;** i**++)**

**{**

**if(**obj**[**i**].**name **==** str**)**

**return** obj**[**i**];**

**}**

**}**

void printPoint**()**//打印point数组

**{**

cout **<<** pt**[**1**].**x **<<** " " **<<** pt**[**1**].**y **<<** endl**;**

cout **<<** pt**[**2**].**x **<<** " " **<<** pt**[**2**].**y **<<** endl**;**

**}**

void InterCircles**(**float fr**,** float gr**,** float fx**,** float fy**,** float gx**,** float gy**)**//产生可能的位置以供甄别

**{**

float a1**,** b1**,** R1**,** a2**,** b2**,** R2**;**

float p1x**,** p1y**,** p2x**,** p2y**;**

a1 **=** fx**;**

b1 **=** fy**;**

R1 **=** fr**;**

a2 **=** gx**;**

b2 **=** gy**;**

R2 **=** gr**;**

//

float R1R1 **=** R1**\***R1**;**

float a1a1 **=** a1**\***a1**;**

float b1b1 **=** b1**\***b1**;**

float a2a2 **=** a2**\***a2**;**

float b2b2 **=** b2**\***b2**;**

float R2R2 **=** R2**\***R2**;**

float subs1 **=** a1a1 **-** 2 **\*** a1**\***a2 **+** a2a2 **+** b1b1 **-** 2 **\*** b1 **\*** b2 **+** b2b2**;**

float subs2 **=** **-**R1R1 **\*** a1 **+** R1R1 **\*** a2 **+** R2R2 **\*** a1 **-** R2R2 **\*** a2 **+** a1a1 **\*** a1 **-** a1a1 **\*** a2 **-** a1**\***a2a2 **+** a1**\***b1b1 **-** 2 **\*** a1**\***b1**\***b2 **+** a1**\***b2b2 **+** a2a2**\***a2 **+** a2**\***b1b1 **-** 2 **\*** a2**\***b1**\***b2 **+** a2**\***b2b2**;**

float subs3 **=** **-**R1R1 **\*** b1 **+** R1R1 **\*** b2 **+** R2R2 **\*** b1 **-** R2R2 **\*** b2 **+** a1a1 **\*** b1 **+** a1a1 **\*** b2 **-** 2 **\*** a1**\***a2**\***b1 **-** 2 **\*** a1**\***a2**\***b2 **+** a2a2 **\*** b1 **+** a2a2 **\*** b2 **+** b1b1**\***b1 **-** b1b1 **\*** b2 **-** b1**\***b2b2 **+** b2b2**\***b2**;**

float sigma **=** sqrt**((**R1R1 **+** 2 **\*** R1 **\*** R2 **+** R2R2 **-** a1a1 **+** 2 **\*** a1**\***a2 **-** a2a2 **-** b1b1 **+** 2 **\*** b1**\***b2 **-** b2b2**)\*(-**R1R1 **+** 2 **\*** R1**\***R2 **-** R2R2 **+** subs1**));**

**if(**abs**(**subs1**)>**0.0000001**)**//分母不为0

**{**

p1x **=** **(**subs2 **-** sigma **\*** b1 **+** sigma **\*** b2**)** **/** **(**2 **\*** subs1**);**

p2x **=** **(**subs2 **+** sigma **\*** b1 **-** sigma **\*** b2**)** **/** **(**2 **\*** subs1**);**

p1y **=** **(**subs3 **+** sigma **\*** a1 **-** sigma **\*** a2**)** **/** **(**2 **\*** subs1**);**

p2y **=** **(**subs3 **-** sigma **\*** a1 **+** sigma **\*** a2**)** **/** **(**2 **\*** subs1**);**

**}**

**if((**fabs**(**p1x**)** **<=** 52.5 **&&** fabs**(**p1y**)** **<=** 32**)** **&&** **(**fabs**(**p2x**)** **<=** 52.5 **&&** fabs**(**p2y**)** **<=** 32**))** //两个可能点

**{**

**if(**fabs**(**p1x **-** p2x**)** **<** EPS **&&** fabs**(**p1y **-** p2y**)** **<** EPS**)**//两圆相切 输出

**{**

cout **<<** "\n机器人的坐标是： (" **<<** p1x **<<** " , " **<<** p1y **<<** " )"**;**

exit**(**100**);**

**}**

**else**

**{**

cout **<<** "有两个可能点！" **<<** endl**;**

cout **<<** "\n机器人的坐标可能1是： (" **<<** p1x **<<** " , " **<<** p1y **<<** " )"**;**

cout **<<** "\n机器人的坐标可能2是： (" **<<** p2x **<<** " , " **<<** p2y **<<** " )\n"**;**

**}**

**}**

**else**//只有一个点合法 直接输出

**{**

**if(**fabs**(**p1x**)** **<=** 52.5 **&&** fabs**(**p1y**)** **<=** 32**)**

cout **<<** "\n机器人的坐标是： (" **<<** p1x **<<** " , " **<<** p1y **<<** " )"**;**

**else**

cout **<<** "\n机器人的坐标是： (" **<<** p2x **<<** " , " **<<** p2y **<<** " )"**;**

exit**(**100**);**

**}**

pt**[**1**].**x **=** p1x**;**

pt**[**1**].**y **=** p1y**;**

pt**[**2**].**x **=** p2x**;**

pt**[**2**].**y **=** p2y**;**

**}**

float LR**(**Point P1**,** Point P2**,** Point P3**)**//用面积量判定点线关系

**{**

float x1**,** x2**,** x3**;**

float y1**,** y2**,** y3**;**

x1 **=** P1**.**x**;**

y1 **=** P1**.**y**;**

x2 **=** P2**.**x**;**

y2 **=** P2**.**y**;**

x3 **=** P3**.**x**;**

y3 **=** P3**.**y**;**

float S **=** **((**x1 **-** x3**)** **\*** **(**y2 **-** y3**)** **-** **(**y1 **-** y3**)** **\*** **(**x2 **-** x3**))** **/** 2**;**

**return** 1 **\*** S**;**

**}**

float normalize**(**float t1**,** float t2**)**//夹角规范化 其中t1是和C的夹角，t2是和A的夹角

**{**

float ang**;**

**if(**t1 **\*** t2 **>** 0**)**

**{**

ang **=** t1 **-** t2**;**

**}**

**else** **if(**t1 **>** 0 **&&** t2 **<** 0**)**

**{**

ang **=** t1 **-** t2**;**

**}**

**else** **if(**t1 **<** 0 **&&** t2 **>** 0**)**

**{**

ang **=** 360 **-** t2 **+** t1**;**

**}**

**return** ang**;**

**}**

Point solve2**(**float t**,** Point A**,** Point B**,** Point C**,** Point D**)**//甄别函数

**{**

float a**,** b**,** c**,** d**,** x**,** y**;**

a **=** A**.**x**;**

b **=** A**.**y**;**

c **=** C**.**x**;**

d **=** C**.**y**;**

x **=** B**.**x**;**//代入一个疑似点

y **=** B**.**y**;**

float flag**;** //使用叉积的一种算法来判定方向

flag **=** **(**b **-** d**)** **\*** x **+** **(**c **-** a**)** **\*** y **+** a **\*** d **-** c **\*** b**;**

//flag = -flag;

cout **<<** "向量叉积2是：" **<<** flag **<<** endl**;**//校验式

//flag = flag - t;

cout **<<** flag **<<** endl**;**

**if(**flag **\*** t **>** 0 **&&** **(**fabs**(**flag**)** **-** fabs**(**t**))** **<** EPS**)**//如果两种叉积相等 说明疑似点是正确的

**{**

**return** B**;**

cout **<<** "\n机器人的坐标11是： (" **<<** x **<<** " , " **<<** y **<<** " )"**;**

**}**

**else**//说明疑似点错误，是另外一个点

**{**

**return** D**;**

cout **<<** "\n机器人的坐标22是： (" **<<** D**.**x **<<** " , " **<<** D**.**y **<<** " )"**;**

**}**

**}**

**运行结果示例：**



