判断点是否在多边形内的算法和C语言程序

判断点是否在凸多边形内，有多种方法，方法简单，计算速度也快。

但实际问题中遇到的多边形不一定是凸多边形,它可能是凹多变形,或几何形状复杂如同迷宫般的多边形。

判断一个点在多边形内或多边形外，比较可靠，也容易理解的方法是射线法。

射线法，把多边形理解为一个有围墙的大院，一个人从院外越过一道墙，他就进了大院，如果他再越过一道墙，就 出了大院。无论大院的形状如何奇特，只要从院外越过奇数道围墙，他就在院内，越过偶数道围墙就在院外。所以，判断一点是否在多边形内或多边形外，只要从这 点起，作一条射线，例如，沿x向直到负无穷，如果越过的边数是单数，这点就在多边形内，越过的边数是偶数，这点就在多边形外。

注意到如果从P作水平向左的射线的话，如果P在多边形内部，那么这条射线与多边形的交点必为奇数，如果P在多边形外部，则交点个数必为偶数（0也在内）。所以，我们可以顺序考虑多边形的每条边，求出交点的总个数。还有一些特殊情况要考虑。假如考虑边(P1,P2)，  
1)如果射线正好穿过P1或者P2,那么这个交点会被算作2次，处理办法是如果P的从坐标与P1,P2中较小的纵坐标相同，则直接忽略这种情况  
2)如果射线水平，则射线要么与其无交点，要么有无数个，这种情况也直接忽略。  
3)如果射线竖直，而P0的横坐标小于P1,P2的横坐标，则必然相交。  
4)再判断相交之前，先判断P是否在边(P1,P2)的上面，如果在，则直接得出结论：P再多边形内部

// 功能：判断点是否在多边形内

// 方法：求解通过该点的水平线与多边形各边的交点

// 结论：单边交点为奇数，成立!

//参数：

// POINT p 指定的某个点

// LPPOINT ptPolygon 多边形的各个顶点坐标（首末点可以不一致）

// int nCount 多边形定点的个数

BOOL PtInPolygon (POINT p, LPPOINT ptPolygon, int nCount)

{

int nCross = 0;

for (int i = 0; i < nCount; i++)

{

POINT p1 = ptPolygon[i];

POINT p2 = ptPolygon[(i + 1) % nCount];

// 求解 y=p.y 与 p1p2 的交点

if ( p1.y == p2.y ) // p1p2 与 y=p0.y平行

continue;

if ( p.y < min(p1.y, p2.y) ) // 交点在p1p2延长线上

continue;

if ( p.y >= max(p1.y, p2.y) ) // 交点在p1p2延长线上

continue;

// 求交点的 X 坐标 --------------------------------------------------------------

double x = (double)(p.y - p1.y) \* (double)(p2.x - p1.x) / (double)(p2.y - p1.y) + p1.x;

if ( x > p.x )

nCross++; // 只统计单边交点

}

// 单边交点为偶数，点在多边形之外 ---

return (nCross % 2 == 1);

}

这里只考虑“简单多边形”，所谓简单多边形并非说形状简单，而是按数学定义，这类多边形不相邻的边没有交点。形象来说，多边形像一张纸，是平的，没有折叠。

多边形，可用顶点坐标表述，逆时针方向为正。

多边形边数可能很多，要判断的点也可能很多，所以既要判断正确，速度也尽可能快。

为此，算法的第一步，是计算出多边形的Bounding Box, 也就是顶点最大最小坐标值的范围，如果一个点在Bounding Box以外，它必在多边形以外，不需要进一步判断，这样，可以达到加速的目的。

第二步，判断点是否与顶点重合，或是否落在边线上。如果是，则结束。

第三步，作射线。射线１，指向-x方向，射线２，指向-y方向，射线3，x向，射线4，y向，射线5，指向Bounding Box 左下角，射线６，右下角，射线７，右上角，射线８，左上角。

第四步，选择射线。  
射线与墙向交，有可能没有交点，有可能有一个交点，有可能两线重合有无数交点。有一个交点时，交点有可能正好是多边形顶点。两线重合和交点正好是顶点的情况，处理起来比较麻烦。所以，要选第一条既不与边线线重合，也不通过任何顶点的射线。

第五步，计算选中的射线跨越墙数。墙数 NN\_cross % 2　为１，点在多边形内，为０，点在多边形外。

下面是示范程序。主函数中，输入一个多边形数据，然后while循环中输入要判断的点的坐标，调用is\_inside（）作判断，直到输入-9999，循环结束。

多边形数据存放在数据文件中。文件的第一行是顶点个数，第二行起是顶点坐标 x1 y1 x2 y2 ...xn yn, 按逆时针顺序排列。

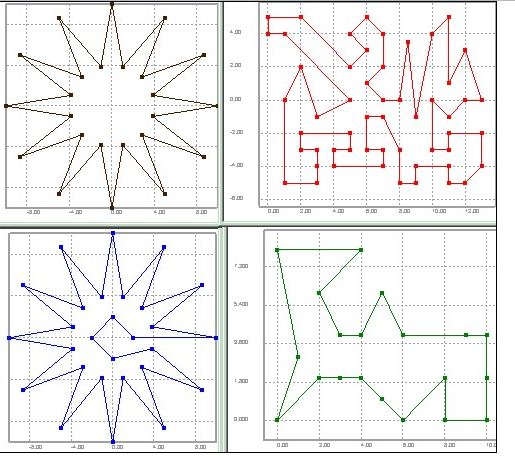
例二（green)  
20  
0 0 2 2 3 2 4 2 5 1 6 0 8 2 8 0 10 0 10 2  
10 4 9 4 6 4 5 6 4 4 3 4 2 6 4 8 0 8 1 3

例三(black)  
24  
10.00 0.00 3.86 1.04 8.66 5.00 2.83 2.83 5.00 8.66  
1.04 3.86 0.00 10.00 -1.04 3.86 -5.00 8.66 -2.83 2.83  
-8.66 5.00 -3.86 1.04 -10.00 0.00 -3.86 -1.04 -8.66 -5.00  
-2.83 -2.83 -5.00 -8.66 -1.04 -3.86 -0.00 -10.00 1.04 -3.86  
5.00 -8.66 2.83 -2.83 8.66 -5.00 3.86 -1.04

例四(blue)  
28  
10.00 0.00 3.86 1.04 8.66 5.00 2.83 2.83 5.00 8.66  
1.04 3.86 0.00 10.00 -1.04 3.86 -5.00 8.66 -2.83 2.83  
-8.66 5.00 -3.86 1.04 -10.00 0.00 -3.86 -1.04 -8.66 -5.00  
-2.83 -2.83 -5.00 -8.66 -1.04 -3.86 -0.00 -10.00 1.04 -3.86  
5.00 -8.66 2.83 -2.83 8.66 -5.00 3.86 -1.04 0.0 -2.0   
-2.0 0.0 0.0 2.0 2.0 0.0

例五（red)  
54  
1 -5 3 -5 3 -4 3 -3 2 -3 2 -2 5 -2 5 -3 4 -3 4 -4  
7 -4 7 -3 6 -3 6 -1 7 -1 8 -3 8 -5 9 -5 9 -4 11 -4  
11 -5 12 -5 13 -4 13 -2 11 -2 11 -3 10 -3 10 0 11 -1 12 0  
13 0 12 3 11 1 11 5 10 4 9 -1 8.5 3.5 8 0 7 0 6 1  
7 2 7 4 6 5 5 4 6 3 5 2 2 5 0 5 0 4 1 4 5 0 3 -1 2 2 1 0

试验点数据，你可以随便给。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=75e063c101019f1c&url=http://s16.sinaimg.cn/orignal/75e063c1gc691bccd171f)

程序中有下列函数：  
is\_inside（）　主要函数，计算一点在多边形内外或边线上，它调用  
get\_bounding\_box（）　计算　Bounding Box。  
p\_outside\_bounding\_box（）　判断点在Bounding Box以内还是以外。  
cross\_pro（）　计算３点构成的向量的叉乘积，叉乘积为０则３点一直线。  
p\_online(）　判断点是否在边线上或顶点上。  
intersect()　计算线段AB和CD相交状态和交点。

完整程序  
#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
#define DEBUG 0  
#define EPS 1E-07  
typedef struct {double x, y; } POINT;  
typedef struct {double min\_x, min\_y, max\_x, max\_y; } BOX;  
void get\_bounding\_box(BOX \*box, POINT \*poly, int N);   
int p\_outside\_bounding\_box(BOX box, POINT p);   
double cross\_pro(POINT p1, POINT p2, POINT p);   
int p\_online(POINT p1,POINT p2, POINT p);   
int intersect(POINT A,POINT B, POINT C, POINT D);  
int is\_inside(POINT poly[],int NN, POINT q);  // is q inside the polygon

FILE \*fin;  
void main(int argc, char \*argv[])  
{  
char namein[80];  
POINT \*poly,q;  
int NN,i;  
int flag;  
if (argc < 2){  
printf("\007Usage: %s namein\n",argv[0]);  
printf("record-1: Number of vertex of the polygon\n");  
printf("record-2: x y x y ....\n");  
printf("Example: %s test\_poly01.dat\n",argv[0]);  
exit(0);  
}  
strcpy(namein,argv[1]);  
if (  (fin = fopen(namein,"r"))==NULL ){  
printf("Can not open %s\n",argv[1]);  
}  
fscanf(fin,"%d",&NN);  
if (NN < 3) {printf("\007NN must >= 3\n");exit(0);};  
poly = (POINT \*) malloc( (NN+1) \* sizeof(POINT));   // array size NN+1  
if (!poly) {printf("\007No enough memory for the polygon\n"); exit(0);}  
for (i=0;i<NN;i++) fscanf(fin,"%lf %lf",&poly[i].x,&poly[i].y);  
fclose(fin);  
poly[NN]=poly[0];  
while(1)  
{  
printf("\007input a point x,y, -9999 0 exit\n");  
scanf("%lf %lf",&q.x,&q.y);  
if (q.x < -9998.0)break;  
flag = is\_inside(poly,NN,q);   
if (flag == 2) { printf("the point q is on line of the polygon\n");}  
else if (flag == 1)  { printf("the point q is inside of the polygon\n");}  
else printf("the point q is outside of the polygon\n");  
printf("\n");  
}  
exit(0); // exit main()  
}

void get\_bounding\_box(BOX \*box, POINT \*poly, int N){  
int i;  
box->min\_x = poly[0].x;  
box->max\_x = poly[0].x;  
box->min\_y = poly[0].y;  
box->max\_y = poly[0].y;  
for (i=1;i<N;i++){  
if  (poly[i].x > box->max\_x) box->max\_x=poly[i].x;  
if  (poly[i].x < box->min\_x) box->min\_x=poly[i].x;  
if  (poly[i].y > box->max\_y) box->max\_y=poly[i].y;    
if  (poly[i].y < box->min\_y) box->min\_y=poly[i].y;   
}  
}

// return 1 if outside bonding box, otherwise 0  
int p\_outside\_bounding\_box(BOX box, POINT p){  
int flag = 0;  
if (p.x < box.min\_x || p.x > box.max\_x) flag = 1;  
if (p.y < box.min\_y || p.y > box.max\_y) flag = 1;  
return flag;  
}

double cross\_pro(POINT p1, POINT p2, POINT p)   
{   
return ((p1.x - p.x) \* (p2.y - p.y) - (p2.x - p.x) \* (p1.y - p.y) );   
}

// return 1 if on-line  
int p\_online(POINT p1,POINT p2, POINT p){  
double area2;  
if  ((p.x == p1.x) && (p.y == p1.y)) return 1;  
if  ((p.x == p2.x) && (p.y == p2.y)) return 1;  
if  ((p.x >= p1.x) && (p.x >= p2.x)) return 0;  
if  ((p.x <= p1.x) && (p.x <= p2.x)) return 0;  
if  ((p.y >= p1.y) && (p.y >= p2.y)) return 0;  
if  ((p.y <= p1.y) && (p.y <= p2.y)) return 0;  
area2=cross\_pro(p1,p2,p);  
if (fabs(area2)<EPS)  return 1;  
return 0;  
}

// if (t2==0) AB & CD are parallel  
// if (t2==0 && t1==0) AB & CD are collinear.  
// If 0<=r<=1 && 0<=s<=1, intersection exists  
// r<0 or r>1 or s<0 or s>1 line segments do not intersect  
// return 1 -- intersect  
// return 0 -- not intersect  
// return 0 --  (t2==0  && t1 != 0) parallel  
// return -1 -- ( t2==0 && t1==0)  collinear.  
int intersect(POINT A,POINT B, POINT C, POINT D){  
double r,s;  
double t1,t2,t3;  
t2 = (B.x - A.x) \* (D.y - C.y) - (B.y-A.y) \* (D.x - C.x);  
t1 = (A.y - C.y) \* (D.x - C.x) - (A.x-C.x) \* (D.y-C.y);  
if  ( (fabs(t2) < EPS) && (fabs(t1) < EPS) ) return -1;  
if  ( (fabs(t2) < EPS) && (fabs(t1) >= EPS) ) return 0;  
r = t1 / t2;  
if  ( (r < 0.0) || (r > 1.0)) return 0;  
t3 = (A.y-C.y)\*(B.x-A.x)-(A.x-C.x)\*(B.y-A.y);  
s = t3 / t2;  
if  ( (s < 0.0) || (s > 1.0)) return 0;  
return 1;  
}

// return 0 -- outside the polygon  
// return 1 -- inside the polygon  
// return 2 -- online the polygon  
int is\_inside(POINT poly[],int NN, POINT q)  
{  
BOX box;  
int flag\_box,flag\_online,flag;  
int NN\_cross=0;  
POINT qi[9],qk;   // not use 0  
int i,j,k;  
double slope,dy,dx,dy2,dx2;  
double shift\_rd=0.0;  
get\_bounding\_box(&box, poly,NN);   
if (DEBUG == 1)  
printf("Bounding Box: %lf %lf %lf %lf\n",box.min\_x,box.min\_y,box.max\_x,box.max\_y);   
//check 1 -- BoundingBox check  
flag\_box = p\_outside\_bounding\_box(box, q);  
if (DEBUG == 1)  
{  
if (flag\_box == 1) {printf("the point is outside of the bounding box\n");}  
else printf("the point is inside of the bounding box\n");  
};  
if (flag\_box == 1)  return 0;

//check 2 -- Vertex and Online check  
flag\_online = 0;  
for (i=0;i<NN;i++){  
if (p\_online(poly[i],poly[i+1], q) == 1){flag\_online = 1; break;};   
};  
if (DEBUG == 1)  
{  
if (flag\_online == 1) printf("Point q is on a line of the polygon\n");  
else printf("Point q is not on a line of the polygon\n");  
};  
if (flag\_online == 1) return 2;

// make qi[1]  
srand((unsigned)time(NULL));  
qi[1].x = box.min\_x - 1.0;  qi[1].y = q.y;  
qi[2].x = box.max\_x + 1.0;  qi[2].y = q.y;  
qi[3].y = box.min\_y - 1.0;  qi[3].x = q.x;  
qi[4].y = box.max\_y + 1.0;  qi[4].x = q.x;  
shift\_rd =  (double)rand() / ((double)(RAND\_MAX)+(double)(1)) \* 2.0;  
qi[5].x = box.min\_x - 1.0; qi[5].y=box.min\_y - shift\_rd;  
qi[6].x = box.max\_x + shift\_rd; qi[6].y=box.min\_y - 1.0;  
shift\_rd =  (double)rand() / ((double)(RAND\_MAX)+(double)(1)) \* 2.0;  
qi[7].x = box.max\_x + 1.0; qi[7].y=box.max\_y + shift\_rd;  
qi[8].x = box.min\_x - shift\_rd; qi[8].y=box.max\_y + 1.0;  
// which Ray  
k = 1;  
flag = 0;  
for (i=0;i<NN;i++){  
if ( (poly[i].y == q.y) && (poly[i].x < q.x)) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
k=2;  
flag=0;  
for (i=0;i<NN;i++){  
if ( (poly[i].y == q.y) && (poly[i].x > q.x)) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
k=3;  
flag=0;  
for (i=0;i<NN;i++){  
if ( (poly[i].x == q.x) && (poly[i].y < q.y)) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
k=4;  
flag=0;  
for (i=0;i<NN;i++){  
if ( (poly[i].x == q.x) && (poly[i].y > q.y)) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
k=5;  
flag=0;  
dx = q.x - qi[5].x;  
if  ( fabs(dx) < EPS) goto Lab\_6;  
dy = q.y - qi[5].y;  
for (i=0;i<NN;i++){  
dx2= poly[i].x - qi[5].x;  
dy2= poly[i].y - qi[5].y;  
if ( fabs( dy \* dx2 - dx \* dy2) < EPS) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
Lab\_6:;  
k=6;  
flag = 0;  
dx = q.x - qi[6].x;  
if  ( fabs(dx) < EPS) goto Lab\_7;  
dy = q.y - qi[6].y;  
for (i=0;i<NN;i++){  
dx2= poly[i].x - qi[6].x;  
dy2= poly[i].y - qi[6].y;  
if ( fabs( dy \* dx2 - dx \* dy2) < EPS) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
Lab\_7:;  
k=7;  
flag = 0;  
dx = q.x - qi[7].x;  
if  ( fabs(dx) < EPS) goto Lab\_8;  
dy = q.y - qi[7].y;  
for (i=0;i<NN;i++){  
dx2= poly[i].x - qi[7].x;  
dy2= poly[i].y - qi[7].y;  
if ( fabs( dy \* dx2 - dx \* dy2) < EPS) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
Lab\_8:;  
k=8;  
flag = 0;  
dx = q.x - qi[8].x;  
if  ( fabs(dx) < EPS) goto Lab\_9;  
dy = q.y - qi[8].y;  
for (i=0;i<NN;i++){  
dx2= poly[i].x - qi[8].x;  
dy2= poly[i].y - qi[8].y;  
if ( fabs( dy \* dx2 - dx \* dy2) < EPS) {flag=1;break;};  
};  
if (flag == 0) goto Lab\_K;  
Lab\_9: printf("\007All Rays do not work -- use k=8 !\n");  
flag = 0;  
Lab\_K: qk=qi[k];  
printf("Ray-%d\n",k);  
// check 3  
for (i=0;i<NN;i++){  
flag =  intersect(poly[i],poly[i+1], qk, q);  
if (flag == 1) NN\_cross++;  
};  
if  ( NN\_cross % 2 == 1) {  
// printf("point Q is inside of the polygon\n");  
return 1;  
}  
else  {  
// printf("point Q is outside of the polygon\n");  
return 0;  
}  
}