**HỘI THẢO KHOA HỌC**

**CÁC TRƯỜNG THPT CHUYÊN**

**KHU VỰC DUYÊN HẢI VÀ ĐỒNG BẰNG BẮC BỘ**

**NĂM 2021**

**Môn: Tin học**

**CHUYÊN ĐỀ: HÌNH HỌC**

**Tháng 9/2021**

**MỤC LỤC**

[Mở đầu: 3](#_Toc82633059)

[1. Kiến thức hình học cần thiết 3](#_Toc82633060)

[2. Phương pháp giải quyết các bài toán hình học 3](#_Toc82633061)

[3. Cách biểu diễn các đối tượng hình học trong máy tính 2](#_Toc82633062)

[4. Giải thuật cho các dạng bài tập cơ bản 4](#_Toc82633063)

[5. Bài tập áp dụng 10](#_Toc82633064)

[Bài 1. Lập trình giải quyết bài tập sau: 10](#_Toc82633065)

[Bài 2. Lập trình giải quyết bài tập sau: 12](#_Toc82633066)

[Bài 3. LOA 13](#_Toc82633067)

[Bài 4. Vườn táo — APPLE 15](#_Toc82633068)

[Bài 5. Diện tích đa giác – POLAREA 18](#_Toc82633069)

[Bài 6. Robot hút bụi 21](#_Toc82633070)

[Bài 7. Bao lồi – CONVEX 25](#_Toc82633071)

[Bài 8. Câu chuyện người lính (https://oj.vnoi.info/problem/military) 27](#_Toc82633072)

[6. Một số bài tập tự luyện: 30](#_Toc82633073)

# **Mở đầu:**

Trong quá trình bồi dưỡng học sinh giỏi tôi nhận thấy học sinh đội tuyển của tôi rất yếu trong việc lập trình giải quyết các bài toán hình, các em sợ một cách tiêu cực. Lý do có thể là các em khó cài đặt, hoặc không tìm được thuật giải thích hợp, hoặc có thuật giải rồi nhưng kết quả vẫn sai do sai số trong quá trình tính toán.

Chuyên đề này tôi đưa ra phương pháp giải quyết các bài toán tin liên quan đến hình học, những lưu ý khi giải bài toán hình và bài tập minh họa (có code và test kèm theo). Chuyên đề phù hợp với những học sinh bắt đầu học hình học, giúp các em phát triển tư duy lôgic, rèn luyện kĩ năng từ đó có thể giải quyết những bài toán hình nâng cao hơn.

# **1. Kiến thức hình học cần thiết**

❶ Vectơ: Tổng/hiệu của hai vectơ; tích của vectơ với một số; hệ trục tọa độ.

❷ Tích vô hướng của hai vectơ; các hệ thức lượng trong tam giác.

❸ Phương pháp tọa độ trong mặt phẳng: phương trình đường thẳng, phương trình đường tròn...; đổi hệ tọa độ.

❹ Phép dời hình và phép đồng dạng trong mặt phẳng: phép quay, phép tịnh tiến, phép quay quanh một điểm cho trước, phép vị tự...

Giáo viên và học sinh có thể xem lại các kiến thức này trong sách hình học 10 và 11, tài liệu chuyên tin học – quyển 3. Bên dưới, chuyên đề có nhắc lại một số kiến thức hình học cơ bản, hay dùng nhất đi kèm với giải thuật.

# **2. Phương pháp giải quyết các bài toán hình học**

***Thứ nhất***, như việc giải quyết các bài toán tin khác, học sinh phải xác định đúng Input và Output của bài toán.

***Thứ hai***:

* Với bài tập đơn giản, chỉ cần vẽ hình học ra và áp dụng những công thức hình học cơ bản là được. Loại bài tập này, giải thuật cũng đơn giản.
* Với bài tập khó, phải xác định, tìm cách biểu diễn và quản lý tốt các **đối tượng hình học**. Loại bài tập này, giải thuật phức tạp hơn.

***Thứ ba***, viết chương trình máy tính để biến Input thành Output cần tìm.

**Các bài toán hình học khó vì:**

* + Dễ bỏ sót các trường hợp đặc biệt:
  + Trường hợp một đường thẳng là thẳng đứng (có hệ số góc là vô hạn)
  + Trường hợp các điểm là thẳng hàng với nhau
  + Trường hợp đa giác là lõm.
  + Có thể gây ra lỗi về độ chính xác sau dấu phẩy động nên dù thuật toán đúng nhưng quá trình tính toán vẫn dẫn đến kết quả sai.
* Cần nắm vững nguyên lý lưu trữ biến và tính toán trong máy tính để có thể ước lượng được sai số tính toán.
* Khi so sánh hai giá trị với nhau, không được dùng dấu “==”, mà phải xét trị tuyệt đối hiệu hai giá trị với một giá trị Epsilon nào đó. Ở đây, Epsilon là một số tương đối bé, tuỳ vào yêu cầu của bài toán mà ta lựa chọn cho phù hợp.

Ví dụ:

Không được dùng:

if (x1 == x2) …

mà phải dùng:

if (abs(x1 – x2) < Eps) …

**Lưu ý:**

Trong C++, sử dụng đơn vị radian với góc.

Quy đổi góc α độ thành radian: .

Thủ thuật: PI = acos(-1).

# **3. Cách biểu diễn các đối tượng hình học trong máy tính**

Các đối tượng chủ yếu trong hình học đó là: điểm, vectơ, đường thẳng và đa giác. Khi dạy học, giáo viên nên thống nhất cách biểu diễn các đối tượng này, như thế sẽ dễ dàng hơn trong việc thể hiện thuật toán.

**❶ Điểm hoặc vectơ**

Điểm hoặc vectơ được biểu diễn bởi 1 cặp số thực.

struct point {

double x, y;

};

Hoặc có thể khai báo như sau:

typedef pair<double, double> point;

**Lưu ý:**

* Coi điểm như vectơ trong nhiều trường hợp giúp việc xử lý dễ dàng hơn.
* Chuyên đề này chủ yếu sử dụng cách khai báo thứ hai.

**❷ Đường thẳng**

Có nhiều cách để biểu diễn một đường thẳng và do đó có nhiều cách biểu diễn đường thẳng trong máy tính, sau đây là hai cách thường dùng:

* Dạng tổng quát của đường thẳng trong mặt phẳng tọa độ: ax + by + c = 0

struct line {

double a , b , c ;

};

**Lưu ý**: b = 0, khi đó đường thẳng có dạng thẳng đứng và có hệ số góc là vô cùng.

* Đường thẳng đi qua hai điểm:

struct line {

point p1, p2;

};

**❸ Đa giác**

point polygon[maxN];

Đa giác đơn giản được biểu diễn bằng cách liệt kê các đỉnh (theo chiều kim đồng hồ hoặc theo chiều ngược kim đồng hồ). Ta nối đỉnh 1 và đỉnh n tạo một liên kết vòng, để thuận lợi cho việc xử lý tính toán, ta để đỉnh 0 bằng đỉnh n và đỉnh n + 1 bằng đỉnh 1.

polygon[n+1] = polygon[1];

polygon[0] = polygon[n];

Trong C++, có thể lưu các điểm của đa giác bằng 1 vectơ cho đơn giản. Ví dụ :

// da giac voi 6 diem

vector <point> p;

p.push\_back(point(1,1)); // P0

p.push\_back(point(3,3)); // P1

p.push\_back(point(9,1)); // P2

p.push\_back(point(12,4)); // P3

p.push\_back(point(9,7)); // P4

p.push\_back(point(1,7)); // P5

# **4. Giải thuật cho các dạng bài tập cơ bản**

***❶ Các phép toán cơ bản với điểm và vectơ***

Đó là các phép toán cộng, trừ vectơ, tích một vectơ với một số, tích có hướng và tích vô hướng, được thể hiện dưới các hàm như sau :

* **Phép cộng**

point operator +(point A, point B)

{

return point(A.first + B.first, A.second + B.second);

}

* **Phép trừ**

point operator -(point A, point B)

{

return point(A.first - B.first, A.second - B.second);

}

* **Tích của một vectơ với một số**

point operator \*(int k, point A)

{

return point(k\*A.first, k\*A.second);

}

* **Tích vô hướng (còn gọi là tích chấm) của và**

Ta có :

Biểu thức tọa độ của tích vô hướng :

Góc tạo bởi hai vectơ sẽ là :

Tích vô hướng có thể coi như một độ đo về mức độ cùng chiều giữa hai vectơ. Nó là cơ sở để xây dựng các khái niệm về khoảng cách và góc. Ví dụ, khoảng cách giữa hai điểm.

* **Tích có hướng (còn gọi là tích chéo)**

Tích có hướng của và là một “giả” vec-tơ vuông góc với mặt phẳng chứa , có tọa độ trên trục Z là :

còn hướng thì tuân theo quy tắc bàn tay phải.

***Ý nghĩa của tích chéo:***

Ta có,

Suy ra,

sinα mang giá trị dương nếu chiều quay và là chiều thuận (ngược chiều kim đồng hồ) và mang giá trị âm nếu chiều quay và là chiều nghịch (theo chiều kim đồng hồ)

double operator \*(point A, point B)

{

return (A.first \* B.second – A.second \* B.first);

}

* **Kiểm tra 3 điểm bất kì**

Tích chéo được dùng để khảo sát chiều: Giả sử ta đi từ A đến B theo đường thẳng và đi tiếp sang C theo đường thẳng, khi đó:

+ Tích chéo > 0 nếu chỗ rẽ tại B là “rẽ trái”

+ Tích chéo < 0 nếu chỗ rẽ tại B là “rẽ phải”

+ Tích chéo = 0 có nghĩa là ba điểm A, B, C thẳng hàng

CCW

CCW

Hàm ccw sau đây sẽ giúp chúng ta xác định được chiều đi từ A đến C tại ngã rẽ B:

int ccw(point A, point B, point C)

{

double t = (B-A)\*(C-B);

if (t>0) return 1; //rẽ trái

if (t<0) return -1; //rẽ phải

return 0; // 3 điểm thẳng hàng

}

***❷ Khoảng cách đại số giữa hai điểm***

Dựa vào tích chấm (tích vô hướng), ta tính khoảng cách giữa hai điểm như sau:

double dist(point A, point B)

{

return sqrt( (B.first - A.first)\*(B.first - A.first) + (B.second- A.second)\*(B.second-A.second));

}

***❸ Tìm phương trình tổng quát của đường thẳng đi qua hai điểm đã cho***

Phương trình đường thẳng đi qua hai điểm A(xA,yA) và B(xB,yB) là :

=

Từ đó ta có:

(yB - yA)x + (xA – xB)y – xA.(yB – yA) + yA.(xB – xA) = 0

Như vậy, phương trình tổng quát của đường thẳng là ax+by+c = 0 thì hàm trích xuất ra các hệ số a, b, c sẽ như sau:

void extract(point A, point B, double &a, double &b, double &c){

a = B.second – A.second;

b = A.first – B.first;

c = - A.first \* (B.second - A.second) + A.second \*(B.first - A.first);

}

***❹ Tìm tọa độ giao điểm của hai đường thẳng***

Cho hai phương trình đường thẳng:

a1x+b1y+c1=0 (d1) và

a2x+b2y+c2=0 (d2)

Tính các định thức:

D = a1b2 - a2b1, Dx = b1c2 - b2c1, Dy = c1a2 - c2a1

- Nếu D = Dx = Dy = 0 thì kết luận 2 đường thẳng trùng nhau.

- Nếu D = 0 và ((Dx≠0) hoặc (Dy≠0)) thì kết luận hai đường thẳng song song.

- Nếu D ≠ 0 thì kết luận 2 đường thẳng cắt nhau tại điểm có tọa độ (Dx/D, Dy/D)

Hàm **IntersectP** trả về giá trị logic **true** và tọa độ của giao điểm **p** nếu hai đường thẳng cắt nhau và ngược lại trả về giá trị logic **false**.

bool intersectP(line l1, line l2, point &p)

{

double a1, b1, c1, a2, b2, c2, D, Dx, Dy;

extract(l1.p1, l1.p2, a1, b1, c1);

extract(l2.p1, l2.p2, a2, b2, c2);

D = a1\*b2 - a2\*b1;

Dx = b1\*c2 - b2\*c1;

Dy = c1\*a2 - c2\*a1;

if (D!=0)

{

p.first = Dx/D;

p.second = Dy/D;

return true;

}

return false;

}

***❺ Khoảng cách giữa điểm và đường thẳng***

(d) là đường thẳng có phương trình: ax + by + c = 0

Khoảng cách đại số từ điểm p đến đường thẳng (d) được tính theo công thức:

double distPL(line L, point P)

{

return (abs(L.a\*p.first + L.b\*p.second + L.c)/sqrt(L.a\*L.a + L.b\*L.b));

}

***➏ Diện tích đa giác***

Một đa giác được lưu bởi một dãy các đỉnh liên tiếp nhau A1, A2, … , AN.

Diện tích đại số của một đa giác không tự cắt có thể xác định bởi công thức sau:

|S| chính là diện tích của đa giác.

Để thuận lợi cho việc xử lý tính toán, ta để đỉnh đỉnh n + 1 bằng đỉnh 1.

double Area(polygon p[])

{

int i;

double s = 0;

p[n + 1] = p[1];

for (i = 1; i<=n; i++)

s += (p[i].first – p[i+1].first)\*(p[i].second + p[i+1].second);

return abs(s)/2;

}

***➐ Sắp xếp các điểm theo góc tăng dần***

Để sắp xếp các điểm A1, A2, …, An theo góc của vectơ OAi tăng dần ta viết hàm so sánh và sort như sau:

bool cmp(point A, point B)

{

return A\*B > 0;

}

sort(A+1, A+1+n, cmp);

***Chú ý:*** Tất cả các điểm phải có cùng hoặc tung độ dương, hoặc tung độ âm.

***➑ Tìm bao lồi của một tập điểm***

Bao lồi của một tập hợp điểm được định nghĩa là một đa giác lồi nhỏ nhất chứa toàn bộ tập điểm này.

Cho tập điểm A1, A2, …, An. Hãy tìm bao lồi của tập điểm này.

**Thuật toán quét Graham:**

* Chọn điểm p0 với tọa độ y nhỏ nhất. O(n)
* Sắp xếp tăng dần tất cả các điểm theo góc tạo với p0 và trục hoành. O(nlogn)
* Duyệt lần lượt qua tất cả các điểm (O(n))
  + Nếu điểm hiện tại tạo thành một góc theo chiều kim đồng hồ với hai điểm ngay trước đó, loại bỏ điểm cuối cùng ra khỏi bao lồi.
  + Nếu không, thêm điểm hiện tại vào bao lồi.

Độ phức tạp của thuật toán là: O(nlogn)

**Lưu ý:** Trường hợp các điểm đã cho là đường thẳng.

**Code tham khảo:**

Sắp xếp danh sách điểm theo hệ số góc, với p[0] là điểm có tung độ nhỏ nhất.

Điểm A nằm trước điểm B nếu như ccw(P, A, B) = true.

int convex\_hull ( vector <point > P)

{

int n = size (P),

sort (P. begin (), P. end (), cmp );

vector <point > S;

S. push\_back (P[n -1]);

S. push\_back (P[0]);

S. push\_back (P[1]);

i = 2;

while (i < n)

{

j = ( int )S. size () -1;

if ( ccw (S[j -1] , S[j], P[i]) || j==2 )

S. push\_back (P[i++]);

// (j==2) de kiem tra cac diem thang hang

else S. pop\_back ();

}

return S;

}

**Thuật toán chuỗi đơn điệu:**

* **Sắp xếp các điểm** được cho theo thứ tự tăng dần theo hoành độ. Có cùng hoành độ, điểm có tung độ nhỏ hơn sẽ đứng trước. Khi đó:
  + Điểm đầu A1 – điểm trái nhất
  + Điểm cuối An – điểm phải nhất
* **Tìm chuỗi trên:** Gọi H là chuỗi trên hiện tại và độ lớn của bao là h. Với mỗi điểm Ai được xét:
  1. Nếu h<2, thêm Ai vào chuỗi H rồi xét điểm tiếp theo.
  2. Gọi  = Ah−1Ah và  = AhAi. Lần này, do ta đang di chuyển theo chiều kim đồng hồ để tìm chuỗi trên, ta kiểm tra xem  có lớn hơn 0 hay không. Nếu có, ta loại bỏ điểm cuối của chuỗi H và quay lại bước ii.

Nếu không, ta thêm Ai vào chuỗi H rồi xét điểm tiếp theo.

Sau khi xét hết các điểm từ 1 đến n, H sẽ chứa toàn bộ các điểm ở chuỗi trên.

* **Tìm chuỗi dưới:** tương tự như tìm chuỗi trên, chỉ khác là ta xét các điểm theo thứ tự ngược lại.

⇒ **Bao lồi** gồm cả chuỗi trên và chuỗi dưới,

**Lưu ý:** Không thêm điểm phải nhất hai lần. Thuật toán này cũng có độ phức tạp O(nlogn). Thuật toán chuỗi đơn điệu được khuyên dùng ở mọi bài toán tìm bao lồi, do nó đơn giản hơn thuật toán Graham và nhanh hơn một chút (do ta không phải tính góc).

void Hull()

{

sort(a+1, a+n+1);

int h = 0;

for(int i = 1; i<=n; i++) //Tim chuoi tren

{

if (h<2)

b[++h] = a[i];

else

{

while (h>1 && ccw(b[h-1],b[h],a[i])>=0)

h--;

b[++h] = a[i];

}

}

for(int i = n-1; i>=1; i--) //Tim chuoi duoi

if(h<2)

b[++h] = a[i];

else

{

while (h>1 && ccw(b[h-1],b[h],a[i])>=0)

h--;

b[++h] = a[i];

}

h--;

cout << h << endl;

for(int i = 1; i<=h; i++)

cout << b[i].first << ' '<< b[i].second << endl;

}

# **5. Bài tập áp dụng**

Các bài sau đây, từ dễ đến khó sẽ giúp các em rèn luyện kĩ năng biểu diễn, quản lý các đối tượng hình học để giải quyết yêu cầu bài toán.

## **Bài 1. Lập trình giải quyết bài tập sau:**

a) Tìm tọa độ của vectơ

b) Tìm tọa độ vectơ sao cho

 Bài tập này đơn giản chỉ sử dụng kiến thức vectơ: tổng/hiệu vectơ, tích của vectơ với một số, nhằm giúp cho học sinh rèn luyện kỹ năng lập trình: biểu diễn vectơ và tính toán đơn giản với vectơ. Dữ liệu nhập trực tiếp từ bàn phím.

*Bước 1*: Xác định Input và Output

Input: tọa độ 3 vectơ , ,

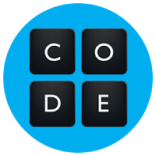
Output: tọa độ vectơ và

*Bước 2*: Giải bài toán bằng kiến thức toán hình đã biết.

* Phép cộng, trừ vectơ và phép nhân một vectơ với một số.

*Bước 3*: Xác định đối tượng hình học là: Vectơ. Khai báo vectơ và viết các hàm cộng, trừ hai vectơ; nhân một vectơ với một số.

*Bước 4*: Hoàn thiện chương trình

 Tham khảo:

#include <bits/stdc++.h>

using namespace std;

typedef pair<int,int> Vecto;

Vecto a, b, c, u, vx;

Vecto operator + (Vecto a, Vecto b)

{ return Vecto(a.first + b.first, a.second + b.second);}

Vecto operator - (Vecto a, Vecto b)

{ return Vecto(a.first - b.first, a.second - b.second);}

Vecto operator \* (double k, Vecto a)

{ return Vecto(k\*a.first, k\*a.second);}

void nhap()

{

cin >> a.first >> a.second >> b.first >> b.second >>c.first >> c.second;

return;

}

void xuly()

{

u = 3\*a + 2\*b - 4\*c;

vx = b - c - a; //x + a = b - c => x = b - c - a

cout << "vecto u(" << u.first << "," << u.second << ")" << endl;

cout << "vecto x(" << vx.first << "," << vx.second << ")" << endl;

return;

}

int main()

{

nhap();

xuly();

return 0;

}

## **Bài 2. Lập trình giải quyết bài tập sau:**

Tìm khoảng cách từ một điểm đến đường thẳng trong các trường hợp sau:

a)

b)

c)

 Bài tập này giúp học sinh rèn luyện kĩ năng lập trình: biểu diễn điểm, biểu diễn đường thẳng tổng quát, xây dựng hàm tính khoảng cách từ một điểm đến một đường thẳng. Dữ liệu nhập trực tiếp từ bàn phím.

*Bước 1*: Xác định Input và Output

Input: tọa độ của 1 điểm và các hệ số của đường thẳng.

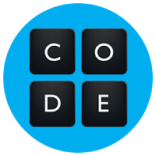
Output: khoảng cách từ điểm đến đường thẳng đã cho.

*Bước 2*: Giải bài toán bằng kiến thức toán hình đã biết.

Áp dụng công thức tính khoảng cách từ 1 điểm tới 1 đường thẳng.

*Bước 3*: Xác định đối tượng hình học là: điểm, đường thẳng. Khai báo điểm, đường thẳng, xây dựng hàm tính khoảng cách từ 1 điểm đến 1 đường thẳng.

*Bước 4:* Hoàn thiện chương trình

Tham khảo:

#include <bits/stdc++.h>

using namespace std;

typedef pair<double, double> Point;

Point p;

struct Line{

double a, b, c;

} L;

void nhap()

{

cin >> p.first >> p.second;

cin >> L.a >> L.b >> L.c;

return;

}

double distPL(Line L, Point P)

{

return (abs(L.a\*p.first + L.b\*p.second + L.c)/sqrt(L.a\*L.a + L.b\*L.b));

}

int main()

{

nhap();

cout << distPL(L,p);

return 0;

}

## **Bài 3. LOA**

Trong kỳ thi học sinh giỏi môn Tin có N học sinh tại sân trường XYZ. Có thể coi sân trường là một hệ trục tọa độ, mỗi học sinh có một tọa độ xi, yi. Ban tổ chức cần đứng tại một vị trí thật thuận tiện để có thể phát loa thông báo cho N em học sinh. Cho rằng loa có thể thông báo cho các em học sinh trong phạm vi bán kính R nghe thấy. Hãy xác định R nhỏ nhất, đảm bảo cho N học sinh có thể nghe được để đi thi đúng giờ.

**Input:**File **LOA.INP** gồm :

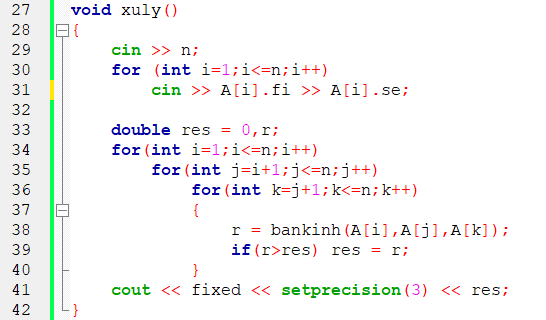
* Dòng 1 ghi *n* (*n* ≤100)
* *n* dòng tiếp theo, dòng thứ i ghi hai số nguyên xi, yi thể hiện tọa độ của một điểm

**Output:**File **LOA.OUT** gồm một số thực với 3 chữ số thập phân là kết quả cần tìm.

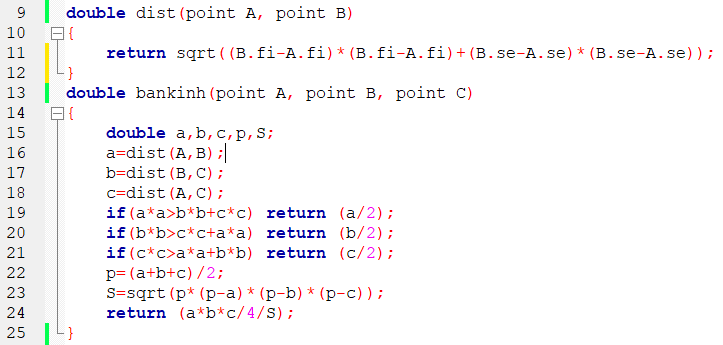
|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  0 0  0 1  1 1 | 0.707 |

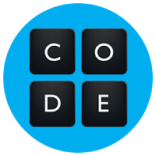
* Link tải test: <https://bitly.com.vn/9qdpk1>

 Bài tập này cũng không yêu cầu quá khó. Vì n nhỏ, nên ta chỉ cần duyệt 3 điểm khác nhau và tìm bán kính nhỏ nhất chứa 3 điểm đó. Khi đó bán kính R cần tìm chính là bán kính R lớn nhất.



Bài này học sinh được rèn luyện cách khai báo điểm, xây dựng hàm tính khoảng cách giữa hai điểm bất kì, xây dựng hàm tính bán kính đường tròn nhỏ nhất bao 3 điểm bất kì.



Tham khảo code đầy đủ:

#include <bits/stdc++.h>

#define fi first

#define se second

using namespace std;

typedef pair<double,double> point;

point A[105];

int n;

double dist(point A, point B)

{

return sqrt((B.fi-A.fi)\*(B.fi-A.fi)+(B.se-A.se)\*(B.se-A.se));

}

double bankinh(point A, point B, point C)

{

double a,b,c,p,S;

a=dist(A,B);

b=dist(B,C);

c=dist(A,C);

if(a\*a>b\*b+c\*c) return (a/2);

if(b\*b>c\*c+a\*a) return (b/2);

if(c\*c>a\*a+b\*b) return (c/2);

p=(a+b+c)/2;

S=sqrt(p\*(p-a)\*(p-b)\*(p-c));

return (a\*b\*c/4/S);

}

void xuly()

{

cin >> n;

for (int i=1;i<=n;i++)

cin >> A[i].fi >> A[i].se;

double res = 0,r;

for(int i=1;i<=n;i++)

for(int j=i+1;j<=n;j++)

for(int k=j+1;k<=n;k++)

{

r = bankinh(A[i],A[j],A[k]);

if(r>res) res = r;

}

cout << fixed << setprecision(3) << res;

}

int main()

{

freopen("LOA.INP","r", stdin);

freopen("LOA.OUT", "w", stdout);

xuly();

return 0;

}

## **Bài 4. Vườn táo — APPLE**

An mua một mảnh đất. Vùng đất này chứa N cây táo, nhưng mảnh của An mua có hình tam giác và không dễ để An xác định cây táo nào thuộc về mình.

**Yêu cầu**: Cho biết tọa độ của các đỉnh tam giác tạo nên mảnh đất của An và tọa độ của tất cả các cây táo. Xác định diện tích đất thuộc về An và số lượng cây táo thuộc về anh ta. Biết rằng cây táo ở biên giới (cạnh tam giác) vùng đất của An là thuộc về anh ta.

**Input**: File **APPLE.INP** gồm:

* Ba dòng đầu tiên chứa tọa độ của các đỉnh của tam giác.
* Dòng tiếp theo chứa số nguyên N (1 ≤ N ≤ 100) là số lượng cây táo. Mỗi dòng trong số N dòng tiếp theo chứa tọa độ của một cây táo.
* Tất cả tọa độ là các cặp số nguyên dương nhỏ hơn 1000, cách nhau bởi một dấu cách.

**Output:** File **APPLE.OUT** gồm:

* Dòng đầu tiên ghi ra diện tích đất thuộc về An, và có đúng một chữ số sau dấu phẩy.
* Dòng thứ hai ghi ra số cây thuộc về An.

**Ví dụ:**

|  |  |
| --- | --- |
| **APPLE.INP** | **APPLE.OUT** |
| 1 1  5 1  3 3  4  3 1  3 2  3 3  3 4 | 4.0  3 |

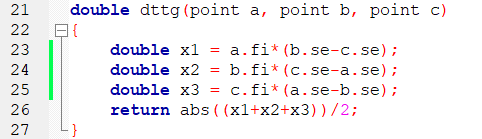
* Link tải test: <https://bitly.com.vn/9qdpk1>

Bài tập này học sinh được rèn luyện kĩ năng làm việc với điểm và tam giác.



Cần phải xây dựng hàm tính diện tích tam giác khi biết 3 đỉnh của tam giác đó.

Ta có thể áp dụng công thức tính diện tích đa giác, mà cụ thể trong bài này là tam giác, ta có công thức như sau:

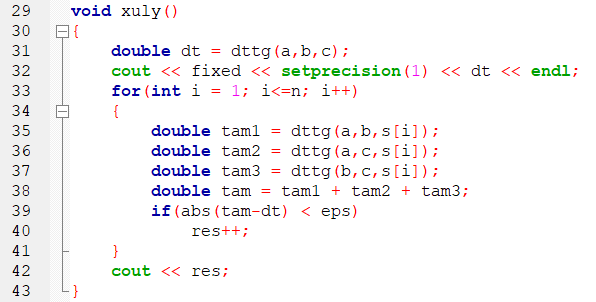


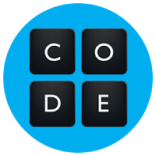
Mỗi cây táo sẽ chia mảnh đất tam giác của An thành 3 tam giác con, nếu tổng diện tích của 3 tam giác con này bằng diện tích mảnh đất mà an sở hữu thì cây táo đó là của An và ngược lại.

Cây nằm ngoài

Cây nằm trong

**Lưu ý:** Trong tính toán có sai số, nên khi thực hiện phép so sánh “==” phải sử dụng phép tính gần đúng như đã nêu ở phần lý thuyết.



 **Tham khảo:**

#include <bits/stdc++.h>

#define fi first

#define se second

using namespace std;

const double eps = 1e-9;

typedef pair <double,double> point;

point a, b, c, s[105];

int n,res;

void nhap()

{

cin >> a.fi >> a.se;

cin >> b.fi >> b.se;

cin >> c.fi >> c.se;

cin >> n;

for(int i = 1; i<=n; i++)

cin >> s[i].fi >> s[i].se;

return;

}

double dttg(point a, point b, point c)

{

double x1 = a.fi\*(b.se-c.se);

double x2 = b.fi\*(c.se-a.se);

double x3 = c.fi\*(a.se-b.se);

return abs((x1+x2+x3))/2;

}

void xuly()

{

double dt = dttg(a,b,c);

cout << fixed << setprecision(1) << dt << endl;

for(int i = 1; i<=n; i++)

{

double tam1 = dttg(a,b,s[i]);

double tam2 = dttg(a,c,s[i]);

double tam3 = dttg(b,c,s[i]);

double tam = tam1 + tam2 + tam3;

if(abs(tam-dt) < eps)

res++;

}

cout << res;

}

int main()

{

freopen("apple.inp","r",stdin);

freopen("apple.out","w",stdout);

nhap();

xuly();

return 0;

}

Hình ảnh Biểu Tượng Toàn Màn Hình Trong Phong Cách Hợp Thời Trang Nền, Lối  Ra, Mở Rộng, Toàn Màn Hình Vector và PNG với nền trong suốt để tải xuống  miễnCó thể áp dụng phương pháp giải bài này để giải quyết bài toán: “Kiểm tra một điểm nằm trong đa giác lồi”

## **Bài 5. Diện tích đa giác – POLAREA**

**Input**: File **POLAREA.INP** gồm:

* Dòng đầu tiên ghi một số nguyên n (3 ≤ n ≤ 1000).
* Mỗi dòng trong số n dòng tiếp theo chứa x, y là tọa độ một đỉnh của đa giác đơn.
* Các đỉnh được cho lần lượt theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ. Các tọa độ đều là số nguyên với giá trị tuyệt đối nhỏ hơn hoặc bằng 10000.

**Output**: File **POLAREA.OUT** gồm:

Đối với mỗi bộ test, ghi ra trên một dòng có dạng D A, trong đó D hoặc là CW hoặc là CCW, cho biết các đỉnh của đa giác được cho theo chiều kim đồng hồ hay ngược chiều kim đồng hồ. A là diện tích của đa giác, và có đúng một chữ số sau dấu phẩy.

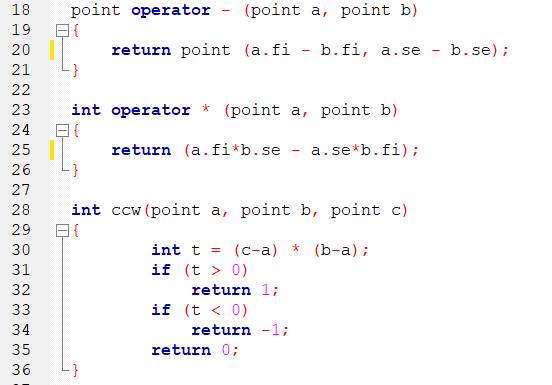
**Ví dụ:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  0 0  10 0  0 10 | CCW 50.0 |

* Link tải test: <https://bitly.com.vn/9qdpk1>

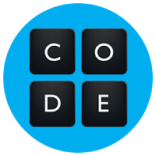
Để biết các đỉnh được cho lần lượt theo chiều kim đồng hồ hay ngược chiều kim đồng hồ: Ta sử dụng hàm ccw, áp dụng hàm này cho 3 điểm liên tiếp nào đó, nếu ccw của 3 điểm đó > 0 thì n điểm đã cho ngược chiều kim đồng hồ, ngược lại ccw của 3 điểm đó <0 thì n điểm đã cho theo chiều kim đồng hồ.





Để tính diện tích của đa giác, ngoài cách tính như trong phần lý thuyết, ta có thể tính bằng cách: Diện tích đa giác bằng tổng diện tích của các hình tam giác con.

Đỉnh cố định

 **Tham khảo đầy đủ:**

#include <bits/stdc++.h>

#define fi first

#define se second

using namespace std;

const double eps = 1e-9;

typedef pair<int,int> point;

int n;

point s[1006];

void nhap()

{

cin >> n;

for(int i = 1; i<=n; i++)

cin >> s[i].fi >> s[i].se;

return;

}

point operator - (point a, point b)

{

return point (a.fi - b.fi, a.se - b.se);

}

int operator \* (point a, point b)

{

return (a.fi\*b.se - a.se\*b.fi);

}

int ccw(point a, point b, point c)

{

int t = (c-a) \* (b-a);

if (t > 0)

return 1;

if (t < 0)

return -1;

return 0;

}

double dttg(point a, point b, point c)

{

double tam1 = a.fi\*(b.se - c.se);

double tam2 = b.fi\*(c.se - a.se);

double tam3 = c.fi\*(a.se - b.se);

return abs(tam1+tam2+tam3)/2;

}

void xuly()

{

for(int i = 3; i<=n; i++)

{

if (ccw(s[i-2],s[i-1],s[i])>0)

{

cout << "CW " << endl;

break;

}

if (ccw(s[i-2],s[i-1],s[i])<0)

{

cout << "CCW " << endl;

break;

}

}

double res = 0;

for(int i = 2; i<n; i++)

{

double tam = dttg(s[1],s[i],s[i+1]);

res = res + tam;

}

cout << fixed << setprecision(1) << res;

return;

}

int main()

{

freopen("polarea.inp","r",stdin);

freopen("polarea.out","w",stdout);

nhap();

xuly();

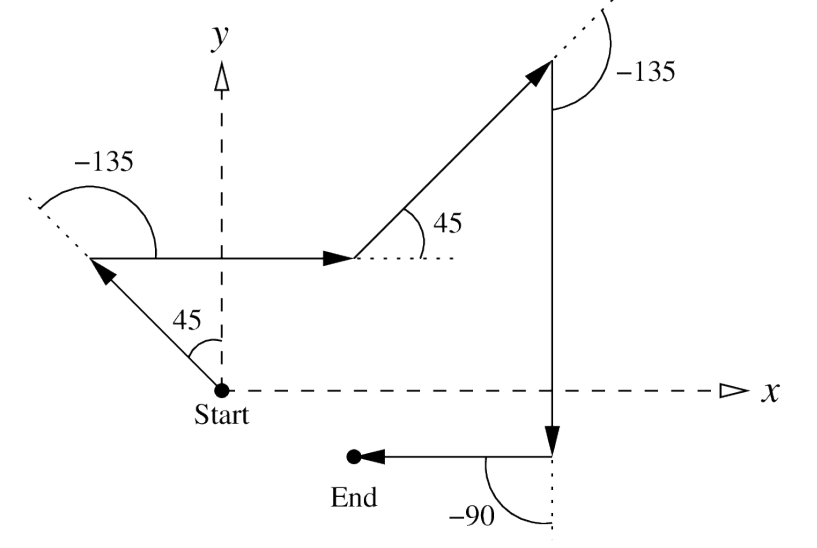
return 0;

}

## **Bài 6. Robot hút bụi**

Công ty ABC đang phát triển phần mềm cho một thế hệ mới của robot hút bụi sàn nhà. Là một robot thị trường đại chúng rẻ tiền, thiết bị này có một cách điều hướng quanh căn phòng khá đơn giản. Nó có thể quay sang trái hoặc sang phải một góc nào đó và sau đó di chuyển về phía trước theo một đường thẳng. Một kế hoạch cho robot bao gồm một chuỗi các đoạn thẳng này. Robot bắt đầu tại điểm gốc toạ độ, quay mặt về hướng Y dương.

**Yêu cầu:** Hãy dự đoán robot sẽ ở đâu sau khi thực hiện theo một kế hoạch cho trước.



**Input:** File **ROBOT.INP** gồm:

* Dòng đầu tiên chứa một số nguyên 1 ≤ n ≤ 25 là số lượng bộ test.
* Mỗi bộ test bắt đầu bằng một số nguyên, 1 ≤ m ≤ 10, cho biết số lượng đoạn thẳng trong kế hoạch của rô bốt.
* Tiếp theo là m dòng, mỗi dòng mô tả một đoạn thẳng trong kế hoạch. Mỗi đoạn được mô tả bằng một cặp số thực, một góc quay theo đơn vị độ, tiếp theo là giá trị khoảng cách. Góc nằm trong phạm vi [-360, 360] và giá trị khoảng cách nằm trong phạm vi [-100,100]. Góc quay cho biết robot nên quay bao nhiêu độ trước khi di chuyển về phía trước khoảng cách đã cho. Góc xoay dương chỉ quay ngược chiều kim đồng hồ. Góc âm chỉ quay theo chiều kim đồng hồ. Tất cả các số thực có tối đa 6 chữ số thập phân sau dấu phẩy.

**Output:** File **ROBOT.OUT**:

* Đối với mỗi bộ test, ghi ra vị trí X và Y dự kiến của robot sau khi đi theo kế hoạch.
* Kết quả phải chính xác trong vòng 10-6 đối với mỗi tọa độ.

**Ví dụ:**

|  |  |
| --- | --- |
| **ROBOT.INP** | **ROBOT.OUT** |
| 2 | -5.000000 10.500000 |
| 2 | 7.071068 7.071068 |
| 0 10.5 |  |
| 90 5 |  |
| 1 |  |
| -45 10 |  |

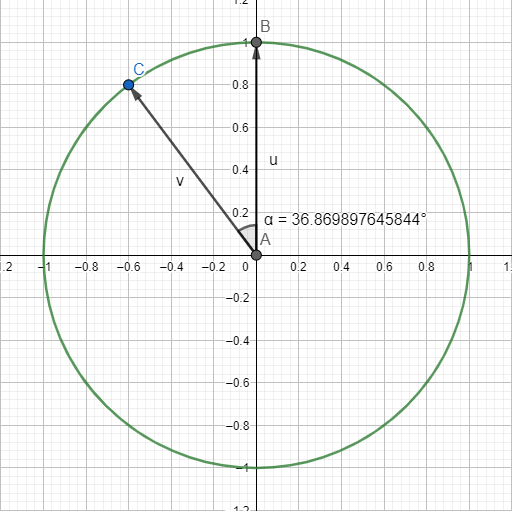
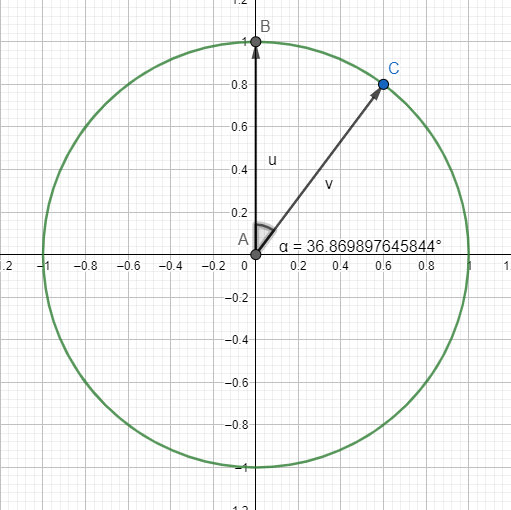
* Link tải test: <https://bitly.com.vn/9qdpk1>

**Nhắc lại bài toán**: Cho điểm trên hệ trục toạ độ đang hướng theo góc . Xoay điểm đã cho một góc và di chuyển điểm theo hướng đã cho một đoạn bằng , xác định toạ độ mới .



- Trước hết, ta thực hiện xác định vector chỉ hướng di chuyển của điểm. Lấy ví dụ lúc ban đầu, khi chưa di chuyển, vector chỉ hướng tạo với trục Ox một góc (trong hình lấy độ dài đơn vị, gốc là A).

- Sau một lần quay với góc , ta xác định được hướng mới của vector (tạo với trục hoành góc ). Hai ảnh dưới thể hiện lần lượt (, vector quay sang trái) và (, vector quay sang phải):

- Sau khi xác định được hướng mới, ta có thể tìm được chênh lệch hai toạ độ dựa theo khoảng cách và hướng đã cho như sau:

+ Tung độ của điểm C, tức , là chênh lệch của với mỗi 1 đơn vị độ dài mà điểm di chuyển

+ Hoành độ điểm C, tức , thể hiện chênh lệch với mỗi 1 đơn vị độ dài mà điểm di chuyển.

Tóm lại, với 1 đơn vị độ dài mà điểm di chuyển: .

Khi điểm di chuyển đơn vị độ dài thì: .

Kết luận: Có tất cả hai bước chính với là các chỉ số của hiện tại:

.

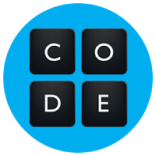
Khởi tạo: .

* **Lưu ý:** Bài này chủ yếu giúp học sinh rèn luyện kỹ năng tính toán hình học với góc trong C++:

+ Sử dụng đơn vị radian với góc. Quy đổi góc α độ thành radian: .

+ Số trong C++ có thể thu được bằng nhiều cách, ví dụ số có sẵn M\_PI hay các biểu thức lượng giác như acos(-1), ).

+ Độ chính xác khi làm việc với số thực.

 **Tham khảo đầy đủ:**

#include <bits/stdc++.h>

using namespace std;

const double pi=acos(-1);

double x,y;

int t;

void dichuyen(double m, double n)

{

m=(pi\*m)/180;

y+=n\*sin(m);

x+=n\*cos(m);

}

void xuly()

{

cin >> t;

while (t--)

{

int n;

cin >> n;

double goc=90;

x=y=0;

for (int i=1; i<=n; i++)

{

double a,b;

cin >> a >> b;

goc=(goc+a);

if (goc>360) goc-=360;

if (goc<0) goc+=360;

dichuyen(goc,b);

}

cout << fixed << setprecision(6) << x << " " << y << '**\n**';

}

}

int main()

{

freopen("robot.inp","r",stdin);

freopen("robot.out","w",stdout);

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

cout.tie(NULL);

xuly();

return 0;

}

## **Bài 7. Bao lồi – CONVEX**

Cho n điểm trên mặt phẳng. Hãy tìm chu vi nhỏ nhất của đa giác lồi chứa n điểm trên (các điểm có thể nằm trên biên của đa giác này).

**Input:** File **CONVEX.INP** gồm:

* Dòng đầu tiên ghi số nguyên dương n (n ≤ 10000)
* n dòng tiếp theo, mỗi dòng thứ i ghi hai số thực xi, yi là tọa độ của một điểm. Các số thựcnày được mô tả dưới dạng tĩnh trong hệ thập phân.

**Output**: File **CONVEX.OUT** gồm một số thực duy nhất là kết quả tìm được. Số thực được ghi với 2 chữ số phần thập phân.

**Ví dụ**:

|  |  |
| --- | --- |
| **CONVEX.INP** | **CONVEX.OUT** |
| 4  4 8  4 12  5 9.3  7 8 | 12.00 |

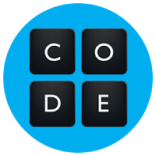
* Link tải test: <https://bitly.com.vn/9qdpk1>

Tìm đa giác lồi chứa n điểm, tức là tìm bao lồi của tập điểm. Cách làm đã nói trong phần lý thuyết ở trên.



Chu vi của đa giác lồi được tính bằng tổng độ dài của các cạnh trong đa giác lồi đó.

Bài tập này học sinh được rèn luyện thuật toán tính độ dài hai điểm bất kì, thuật toán tìm bao lồi của một tập điểm.

 **Tham khảo đầy đủ:**

#include <bits/stdc++.h>

#define fi first

#define se second

using namespace std;

const int nmax = 1e4+1;

typedef pair<double,double> point;

int n;

point a[nmax], b[nmax];

double res = 0;

void nhap()

{

cin >> n;

for(int i = 1; i<=n; i++)

cin >> a[i].fi >> a[i].se;

return;

}

point operator - (point a, point b)

{

return point (a.fi-b.fi, a.se - b.se);

}

double operator \* (point a, point b)

{

return (a.fi\*b.se - a.se\*b.fi);

}

double ccw(point a, point b, point c)

{

double t = (b-a) \* (c-a);

if (t > 0)

return 1;

if (t < 0)

return -1;

return 0;

}

double dist(point A, point B)

{

return sqrt( (B.fi - A.fi)\*(B.fi - A.fi) + (B.se- A.se)\*(B.se-A.se));

}

void Hull()

{

sort(a+1, a+n+1);

int h = 0;

for(int i = 1; i<=n; i++)

{

if (h<2)

b[++h] = a[i];

else

{

while (h>1 && ccw(b[h-1],b[h],a[i])>=0)

h--;

b[++h] = a[i];

}

}

for(int i = n-1; i>=1; i--)

if(h<2)

b[++h] = a[i];

else

{

while (h>1 && ccw(b[h-1],b[h],a[i])>=0)

h--;

b[++h] = a[i];

}

h--;

for(int i =1; i<h ; i++)

res = res + dist(b[i],b[i+1]);

res = res + dist(b[h],b[1]);

cout << fixed << setprecision(2) << res;

return;

}

int main()

{

freopen("convex.inp","r",stdin);

freopen("convex.out","w",stdout);

nhap();

Hull();

return 0;

}

## **Bài 8. Câu chuyện người lính (**<https://oj.vnoi.info/problem/military>**)**

*Tôi vẫn nhớ chiến trường Điện Biên năm đó rất ác liệt, rất nhiều người lính đã ngã xuống. Tại vùng căn cứ này, địch cho xây dựng lô cốt, hàng rào dây thép gai rất nhiều, vòng trong nối vòng ngoài, tạo thành nhiều vòng bảo vệ...*

Đó là dòng hồi tưởng của một người lính già đã từng tham gia chiến dịch Tây Bắc lịch sử. Lần theo những trang sử được ghi chép lại, người ta biết rằng tướng Đờ Cát lúc đầu chưa chọn vị trí để đặt sở chỉ huy mà tìm cách thiết lập các vòng bảo vệ bằng dây thép gai nối các cứ điểm lại với nhau, sau đó sẽ chọn đặt sở chỉ huy tại vị trí an toàn nhất là ở vị trí mà có nhiều vòng bảo vệ bao quanh nhất. Mỗi một vòng bảo vệ là một đa giác không tự cắt tạo thành bằng cách nối một số cứ điểm lại với nhau bằng dây thép gai, một cứ điểm thuộc về không quá một vòng bảo vệ, các vòng bảo vệ phải được thiết lập sao cho giữa hai vòng bảo vệ bất kỳ X và Y thì phần diện tích chung của X và Y bằng min(diện tích X, diện tích Y) hoặc bằng 0. Trên mặt phẳng toạ độ, các cứ điểm được coi như các điểm có toạ độ nguyên. Bạn hãy xác định xem, sở chỉ huy của tướng Đờ Cát sẽ được bảo vệ tối đa bởi mấy vòng bảo vệ.

**Input:**

Dòng 1: số nguyên N là số cứ điểm. (1≤N≤4000).

N dòng tiếp theo, dòng thứ i gồm 2 số nguyên xi,yi tương ứng là toạ độ của cứ điểm i. Các toạ độ đều là số nguyên dương nhỏ hơn hoặc bằng 10000.

**Output:**

Gồm một dòng duy nhất ghi ra số lượng vòng bảo vệ tối đa mà sở chỉ huy của tướng Đờ Cát có thể được bao bọc.

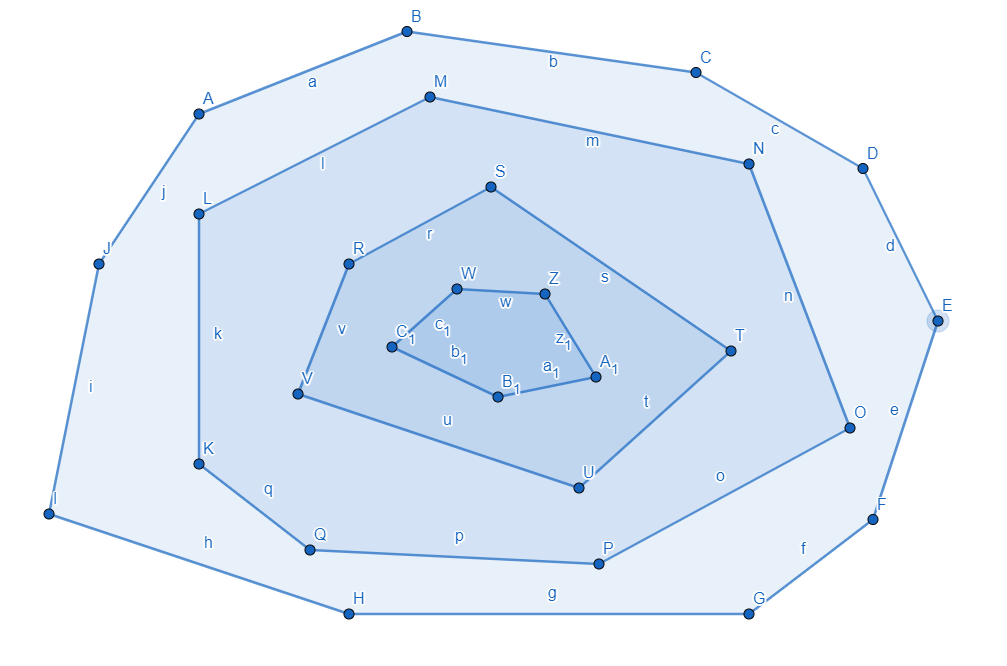
**Sample:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  100 100  200 100  100 200  300 300 | 1 |



- Nhắc lại bài toán: Cho điểm trên hệ trục toạ độ Oxy. Ta có thể thực hiện bao vòng một tập các điểm sao cho khi xét tất cả các vòng. không có điểm nào thuộc vào 2 vòng và không có 2 vòng nào tồn tại điểm chung. Xác định số vòng tối đa.

- Ví dụ với một tập các điểm dưới đây, đáp án có thể thấy là 4:



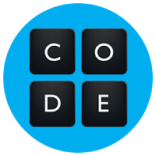
- Có thể thấy để tối ưu hoá kết quả, ta cần dùng ít điểm nhất có thể với mỗi vòng bao và cần bao được nhiều điểm nhất có thể nên thuật toán nhìn chung là chừng nào còn tìm được bao lồi thì ta tăng kết quả

Áp dụng thuật toán **tìm bao lồi và kĩ thuật đánh dấu** để giải quyết bài toán.

Một cách khác để xác định thứ tự đi là xuôi hay ngược chiều kim đồng hồ. Đó là ta tính diện tích tam giác theo toạ độ 3 điểm bằng giá trị tuyệt đối của biểu thức sau:

Tuy nhiên, biểu thức trên chỉ trả về giá trị dương khi và chỉ khi 3 điểm đi ngược chiều kim đồng hồ

Và việc xét dấu của định thức trên quay về xét:

 **Tham khảo có thể chấm bài trên** [**https://oj.vnoi.info/problem/military**](https://oj.vnoi.info/problem/military)**:**

#include <bits/stdc++.h>

#define nmax 4007

#define X first

#define Y second

#define ll long long

using namespace std;

typedef pair<ll,ll> pii;

pii a[nmax];

int s[nmax];

bool ck[nmax];

int n,m,kq;

bool ccw(pii a, pii b, pii c) //xét a -> b -> c là ngược chiều

{

return a.X\*(b.Y-c.Y)+b.X\*(c.Y-a.Y)+c.X\*(a.Y-b.Y) > 0;

}

void Hull() //tìm bao lồi để loại điểm

{

int i,k,t;

k=1;

for (i=1; i<=n; i++) //tìm chuỗi trên

{

while (k>2 && ccw(a[s[k-2]],a[s[k-1]],a[i])) k--;

s[k++]=i;

}

for (i=n,t=k+1; i>=1; i--) //tìm chuỗi dưới

{

while (k>t && ccw(a[s[k-2]],a[s[k-1]],a[i])) k--;

s[k++]=i;

}

memset(ck,0,sizeof(ck));

for (int i=1; i<k; i++) ck[s[i]]=1;

int m=0;

for (i=1; i<=n; i++)

if (!ck[i]) a[++m]=a[i];

n=m;

}

int main()

{

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

cout.tie(NULL);

cin >> n;

for (int i=1; i<=n; i++)

cin >> a[i].X >> a[i].Y;

sort(a+1,a+n+1);

while (n>2) //khi còn 2 điểm thì không còn bao lồi

{

kq++;

Hull();

}

cout << kq;

return 0;

}

# **6. Một số bài tập tự luyện:**

1. <https://oj.vnoi.info/problem/pvoi14_1>
2. <https://oj.vnoi.info/problem/ballgmvn>
3. <https://oj.vnoi.info/problem/pravo>
4. <https://oj.vnoi.info/problem/nkpoli>

Chuyên đề này giúp học sinh tổng hợp được các kiến thức hình học cần thiết và biết phương pháp giải các bài tập lập trình hình học đơn giản và dần dần nâng cao hơn. Mỗi bài tập lập trình hình học, tôi đã chuẩn bị code tham khảo và test để chấm bằng phần mềm Themis.

Hãy đăng nhập vào link sau để tải code và test: <https://bitly.com.vn/9qdpk1>

Kiến thức bản thân còn hạn chế, tôi rất mong nhận được sự góp ý từ các thầy cô. Tôi xin chân thành cảm ơn!