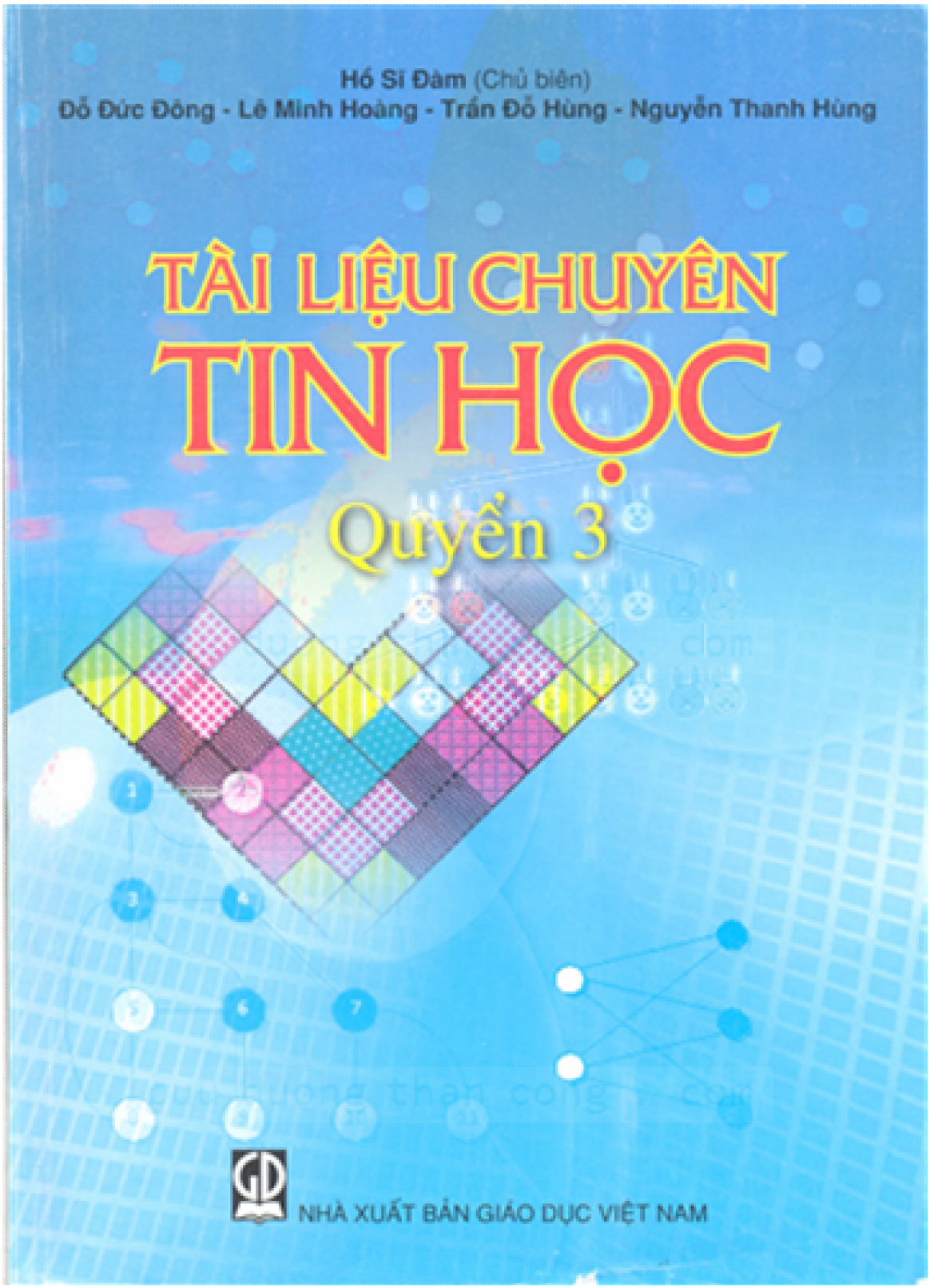

Hồ Sĩ Đàm (Chủ biên)
Đỗ Đức Đông - Lê Minh Hoàng - Trần Đỗ Hùng - Nguyễn Thanh Hùng
TÀI LIỆU CHUYÊN
TIN HỌC
Quyển 3
NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

MỤC LỤC

Chuyên đề 8

HÌNH HỌC TÍNH TOÁN

Hình học tính toán (computational geometry) là một nhánh của ngành khoa học máy tính, chuyên nghiên cứu về thuật toán giải quyết các bài toán liên quan tới các đối tượng hình học. Trong toán học và công nghệ hiện đại, hình học tính toán có ứng dụng khá rộng rãi trong các lĩnh vực về đồ hoạ máy tính, thiết kế, mô phỏng...

Do giới hạn nội dung của cuốn sách, chúng ta sẽ chỉ khảo sát một số bài toán và thuật toán căn bản. Để giúp cho người đọc nhanh chóng nắm bắt được ý tưởng và cài đặt thuật toán, đa số công thức hình học được thừa nhận, không chứng minh.

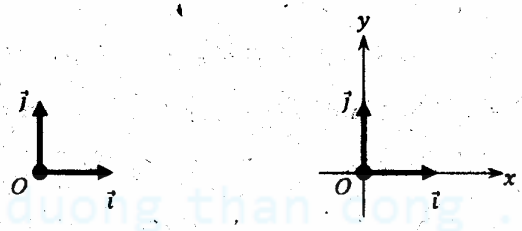
I. Một số khái niệm cơ bản

1. Hệ toạ độ Đề-các

Trong mặt phẳng, chọn một điểm O và hai vectơ đơn vị (vectơ có độ dài 1) $\vec{i}, \vec{j}$ vuông góc với nhau. Khi đó bộ ba $(O, \vec{i}, \vec{j})$ được gọi là hệ toạ độ Đề-các vuông góc (hay còn gọi là một mục tiêu Oclit hai chiều, mục tiêu trực chuẩn).

Đơn vị độ dài là một khái niệm quy ước, có thể là cm, mm, inch... Ràng buộc về hai vectơ $\vec{i}, \vec{j}$ có thể viết dưới dạng biểu thức của tích vô hướng (tích chấm): $\vec{i}^2 = \vec{j}^2 = 1$ và $\vec{i}.\vec{j} = 0$.

Ta cũng kí hiệu mục tiêu đó là Oxy với Ox và Oy là hai tia gốc O có vectơ chỉ phương lần lượt là $\vec{i}$ và $\vec{j}$.



Hình 8.1. Mục tiêu Oclit

Điểm O gọi là gốc toạ độ, đường thẳng Ox được gọi là trục hoành và Oy được gọi là trục tung.

Trong kĩ thuật vẽ hình, hai vectơ $\vec{\imath}$ và $\vec{\jmath}$ thường được vẽ sao cho chiều quay từ $\vec{\imath}$ tới $\vec{\jmath}$ ngược với chiều kim đồng hồ. Ta gọi chiều quay từ vectơ $\vec{\imath}$ tới vectơ $\vec{\jmath}$ là *chiều thuận, chiều quay trái* hay *ngược chiều kim đồng hồ* (*counterclockwise-ccw*). Ngược lại, ta gọi chiều quay từ vectơ $\vec{\jmath}$ tới vectơ $\vec{\imath}$ là *chiều nghịch, chiều quay phải* hay *chiều kim đồng hồ* (*clockwise-cw*).

2. Toạ độ

Xét mặt phẳng trực chuẩn $(O,\vec{\imath},\vec{\jmath})$, với một vectơ $\vec{v}$ bất kì của mặt phẳng, tồn tại duy nhất một cặp số thực $(x;y)$ sao cho $\vec{v}=x\vec{\imath}+y\vec{\jmath}$. Cặp số $(x;y)$ khi đó được gọi là toạ độ của vectơ $\vec{v}$ đối với mục tiêu đã cho, kí hiệu $\vec{v}=(x;y)$.

Các kết quả sau có thể dễ dàng chứng minh bằng định nghĩa toạ độ:

- Vectơ $\vec{\imath}$ có toạ độ $(1;0)$ và vectơ $\vec{\jmath}$ có toạ độ $(0;1)$.
- Nếu $\vec{u}=(x_u;y_u)$ và $\vec{v}=(x_v;y_v)$ thì $\vec{u}=\vec{v}$ khi và chỉ khi $x_u=x_v$ và $y_u=y_v$.
- Nếu $\vec{u}=(x_u;y_u)$ và $\vec{v}=(x_v;y_v)$ thì tổng vectơ $\vec{u}+\vec{v}=(x_u+x_v;y_u+y_v)$ và tích vô hướng $\vec{u}.\vec{v}=x_ux_v+y_uy_v$.
- Nếu $\vec{v}=(x;\ y)$ và $k\in\mathbb{R}$ thì $k\vec{v}=(kx;ky)$.

Trên mặt phẳng trực chuẩn $(O,\vec{\imath},\vec{\jmath})$ ta lấy một điểm M, khi đó toạ độ $(x;y)$ của vectơ $\overrightarrow{OM}$ được gọi là toạ độ của điểm M, kí hiệu $M=(x;y)$.

Ta có mối liên hệ giữa toạ độ của vectơ và toạ độ của điểm:

Nếu $M=(x_M;y_M)$ và $N=(x_N;y_N)$ thì $\overrightarrow{MN}=\overrightarrow{ON}-\overrightarrow{OM}=(x_N-x_M;y_N-y_M)$.

3. Đổi hệ toạ độ

Cho hai hệ toạ độ trực chuẩn trên mặt phẳng: $(O,\vec{\imath},\vec{\jmath})$ và $\left(O',\vec{\imath}',\vec{\jmath}'\right)$. Giả sử điểm M có toạ độ $(x;y)$ đối với hệ toạ độ $\left(O',\vec{\imath}',\vec{\jmath}'\right)$, bài toán đặt ra là xác định toạ độ điểm M đối với hệ toạ độ $(O,\vec{\imath},\vec{\jmath})$.

Giả sử trong hệ toạ độ $(O,\vec{\imath},\vec{\jmath})$ điểm $O'=(p;q)$, vectơ $\vec{\imath}'=(a;b)$, vectơ $\vec{\jmath}'=(c;d)$. Khi đó theo định nghĩa:

$$\begin{aligned}\overrightarrow{OO'} &= p\vec{\imath} + q\vec{\jmath}\\ \vec{\imath}' &= a\vec{\imath} + b\vec{\jmath}\\ \vec{\jmath}' &= c\vec{\imath} + d\vec{\jmath}\end{aligned} \tag{8.1}$$

Điểm M có toạ độ $(x; y)$ đối với hệ toạ độ $\left(O', \vec{\imath}', \vec{\jmath}'\right)$, tức là

$$\overrightarrow{O'M} = x\vec{\imath}' + y\vec{\jmath}'$$

Suy ra

$$\begin{aligned}\overrightarrow{OM} &= \overrightarrow{OO'} + \overrightarrow{O'M}\\ &= p\vec{\imath} + q\vec{\jmath} + x(a\vec{\imath} + b\vec{\jmath}) + y(c\vec{\imath} + d\vec{\jmath})\\ &= \vec{\imath}(ax + cy + p) + \vec{\jmath}(bx + dy + q)\end{aligned} \tag{8.2}$$

Công thức (8.2) cho biết rằng điểm M có toạ độ $(ax + cy + p; bx + dy + q)$ đối với hệ toạ độ $(O, \vec{\imath}, \vec{\jmath})$. Nếu ta gọi $(\bar{x}; \bar{y})$ là toạ độ điểm M đối với hệ toạ độ $(O, \vec{\imath}, \vec{\jmath})$ thì công thức (8.2) có thể viết dưới dạng ma trận:

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} = \begin{pmatrix}a & c\\ b & d\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}p\\ q\end{pmatrix} \tag{8.3}$$

Chú ý: Công thức đổi toạ độ:

$$\begin{cases}\bar{x} = ax + cy + p\\ \bar{y} = bx + dy + q.\end{cases}$$

Ta có thể nhớ như sau cho dễ: các hệ số ở phương trình thứ nhất là hoành độ của ba vectơ $\vec{\imath}', \vec{\jmath}'$ và $\overrightarrow{OO'}$ trong khi các hệ số ở phương trình thứ hai là tung độ của ba vectơ đó.

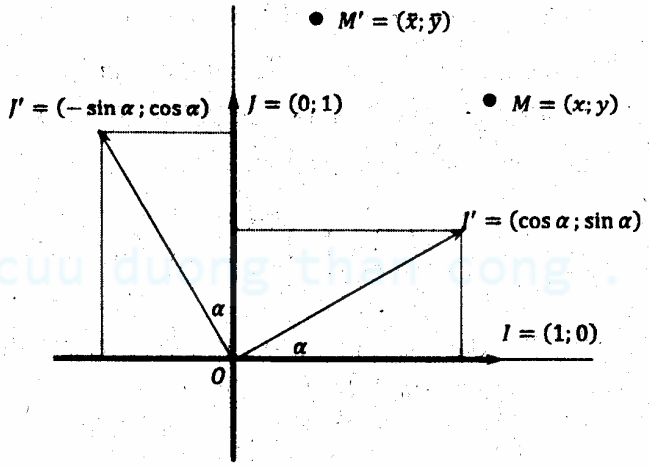
4. Xây dựng công thức biến đổi toạ độ

Trong hình học phẳng, một trong những kĩ thuật quan trọng là xây dựng công thức biến đổi toạ độ cho các phép đồng dạng (tịnh tiến, quay, đối xứng trục, vị tự...). Kĩ thuật chung có thể mô tả như sau:

- Trên hệ toạ độ trực chuẩn ban đầu $(O, \vec{\imath}, \vec{\jmath})$, xác định điểm $I = (1; 0)$ và điểm $J = (0; 1)$.

- Với một phép đồng dạng f, ta thực hiện f trên ba điểm O, I, J để nhận được ba ảnh tương ứng của chúng theo thứ tự là O', I', J'. Từ đó xác định toạ độ điểm $O' = (p; q)$, vectơ $\vec{\imath}' = \overrightarrow{O'I'} = (a; b)$ và vectơ $\vec{\jmath}' = \overrightarrow{O'J'} = (c; d)$.
- Nhận xét rằng nếu điểm M có toạ độ $(x; y)$ đối với hệ toạ độ trực chuẩn đã cho $(O, \vec{\imath}, \vec{\jmath})$ thì sau phép biến đổi f, điểm M sẽ biến thành điểm M' cũng có toạ độ $(x; y)$ nhưng với hệ toạ độ trực chuẩn trực chuẩn $\left(O', \vec{\imath}', \vec{\jmath}'\right)$.
- Áp dụng công thức (8.2) hoặc (8.3) để xây dựng công thức toạ độ của điểm M' theo hệ toạ độ ban đầu.

a) Phép quay



Hình 8.2. Phép quay tâm O góc α

Ta xét phép quay quanh tâm O một góc α. Phép quay này giữ bất biến điểm O, tức là $O' = (0; 0)$, điểm $I = (1; 0)$ biến thành $I' = (\cos\alpha ; \sin\alpha)$, điểm $J = (0; 1)$ biến thành $J' = (-\sin\alpha ; \cos\alpha)$. Như vậy:

$$O' = (0; 0);$$
$$\overrightarrow{O'I'} = (\cos\alpha ; \sin\alpha);$$
$$\overrightarrow{O'J'} = (-\sin\alpha ; \cos\alpha).$$

Áp dụng công thức (8.2) hoặc (8.3), phép quay tâm O góc α sẽ biến điểm $(x; y)$ thành điểm $(\bar{x}; \bar{y})$, trong đó:

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} = \begin{pmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}$$

tức là:

$$\begin{cases}\bar{x} = x\cos\alpha - y\sin\alpha\\ \bar{y} = x\sin\alpha + y\cos\alpha.\end{cases} \quad (8.4)$$

Trường hợp đặc biệt:

Nếu góc $\alpha = 90^o$, ta có $\cos\alpha = 0$ và $\sin\alpha = 1$. Công thức (8.4) trở thành

$$\begin{cases}\bar{x} = -y\\ \bar{y} = x.\end{cases}$$

Nếu góc $\alpha = 180^o$, đây là phép đối xứng tâm O, $\cos\alpha = -1$ và $\sin\alpha = 0$. Công thức (8.4) trở thành:

$$\begin{cases}\bar{x} = -x\\ \bar{y} = -y.\end{cases}$$

b) Phép tịnh tiến

Ta xét phép tịnh tiến theo vectơ $(a; b)$. Phép tịnh tiến này biến điểm $O = (0; 0)$ thành $O' = (a; b)$. Điểm $I = (1; 0)$ biến thành $I'(a + 1; b)$ và điểm $J = (0; 1)$ biến thành $J' = (a; b + 1)$. Như vậy:

$$O' = (a; b);$$
$$\overrightarrow{O'I'} = (1; 0);$$
$$\overrightarrow{O'J'} = (0; 1).$$

Áp dụng công thức (8.2) hoặc (8.3), phép tịnh tiến theo vectơ $(a; b)$ sẽ biến điểm $(x; y)$ thành điểm $(\bar{x}; \bar{y})$, trong đó:

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}a\\ b\end{pmatrix}$$

tức là:

$$\begin{cases}\bar{x} = x + a\\ \bar{y} = y + b.\end{cases} \quad (8.5)$$

c) Phép quay quanh một điểm cho trước

Công thức phép quay góc α quanh một điểm A có thể xây dựng bằng phương pháp tương tự trên, tuy nhiên ta có thể làm theo cách sau:

Thực hiện phép tịnh tiến theo vectơ $\overrightarrow{AO}$, sau đó thực hiện phép quay tâm O góc α, rồi thực hiện tiếp phép tịnh tiến theo vectơ $\overrightarrow{OA}$.

Công thức cụ thể như thế nào xin dành cho bạn đọc.

d) Phép vị tự

Để thực hiện phép vị tự tâm $A = (x_A; y_A)$ tỉ số k, ta có thể dùng phương pháp như sau:

- Tịnh tiến theo vectơ $\overrightarrow{AO}$: Điểm $(x; y)$ sẽ biến thành điểm $(x'; y')$ thoả mãn:

$$\begin{cases} x' = x - x_A \\ y' = y - y_A. \end{cases}$$

- Thực hiện phép vị tự tâm O tỉ số k. Phép vị tự này giữ bất biến điểm O, điểm $I = (1; 0)$ biến thành $I' = (k; 0)$, điểm $J = (0; 1)$ biến thành $J' = (0; k)$. Như vậy:

$$O' = (0; 0);$$
$$\overrightarrow{O'I'} = (k; 0);$$
$$\overrightarrow{O'J'} = (0; k).$$

  Áp dụng công thức (8.2) hoặc (8.3), sau phép vị tự tâm O tỉ số k, điểm $(x'; y')$ sẽ biến thành điểm $(x''; y'')$ thoả mãn:

$$\begin{cases} x'' = kx' \\ y'' = ky'. \end{cases}$$

- Cuối cùng, ta thực hiện phép tịnh tiến theo vectơ $\overrightarrow{OA}$:

$$\begin{cases} \bar{x} = x'' + x_A \\ \bar{y} = y'' + y_A. \end{cases}$$

Tổng hợp lại ta được:

$$\begin{cases} \bar{x} = k(x - x_A) + x_A \\ \bar{y} = k(y - y_A) + y_A. \end{cases} \tag{8.6}$$

5. *Một số khái niệm khác*

a) Tích chấm

Tích chấm (*dot product*) hay tích vô hướng của hai vectơ $\vec{u}$ và $\vec{v}$, kí hiệu $\vec{u}.\vec{v}$ là một số thực được tính bằng tích độ dài hai vectơ $\vec{u}$ và $\vec{v}$ nhân với *cosin* của góc xen giữa hai vectơ đó. Góc xen giữa hai vectơ này là góc không định hướng, có số đo từ 0 tới π.

Hàm số *cosin* là hàm nghịch biến trong khoảng $[0;\pi]$, nó đạt giá trị lớn nhất bằng 1 khi góc giữa hai vectơ bằng 0 (hai vectơ cùng chiều), đạt giá trị nhỏ nhất bằng -1 khi góc giữa hai vectơ bằng π (hai vectơ ngược chiều) và đạt giá trị 0 khi hai vectơ vuông góc (trực giao).

Biểu thức của tích chấm giữa hai vectơ $\vec{u} = (x_u; y_u)$ và $\vec{v} = (x_v; y_v)$ có thể diễn giải như sau:

Ta có:

$$\vec{u} = x_u\vec{i} + y_u\vec{j}$$
$$\vec{v} = x_v\vec{i} + y_v\vec{j}.$$

Vậy:

$$\begin{aligned} \vec{u}.\vec{v} &= (x_u\vec{i} + y_u\vec{j}).(x_v\vec{i} + y_v\vec{j}) \\ &= x_u x_v \underbrace{\vec{i}^2}_{=1} + y_u y_v \underbrace{\vec{j}^2}_{=1} + (x_u y_v + y_u x_v) \underbrace{\vec{i}.\vec{j}}_{=0} \\ &= x_u x_v + y_u y_v. \end{aligned} \tag{8.7}$$

Cũng từ định nghĩa tích chấm, ta suy ra công thức tính cosin của góc α hợp bởi hai vectơ $\vec{u} = (x_u; y_u)$ và $\vec{v} = (x_v; y_v)$:

$$\cos\alpha = \frac{\vec{u}.\vec{v}}{|\vec{u}|.|\vec{v}|} = \frac{x_u x_v + y_u y_v}{\sqrt{(x_u^2 + y_u^2)(x_v^2 + y_v^2)}} \tag{8.8}$$

Tích vô hướng có thể coi như một độ đo về mức độ cùng chiều giữa hai vectơ. Quan trọng hơn, nó là cơ sở để xây dựng các khái niệm về khoảng cách và góc. Ví dụ nếu $A = (x_A; y_A)$ và $B = (x_B; y_B)$ thì độ dài đoạn thẳng AB được tính bằng:

$$\sqrt{\overrightarrow{AB}^2} = \sqrt{(x_B - x_A)^2 (y_B - y_A)^2}$$

b) Tích chéo

Tích chéo (*cross product*) của hai vectơ $\vec{u}$ và $\vec{v}$, kí hiệu $\vec{u} \times \vec{v}$ là một số thực được tính bằng tích độ dài hai vectơ $\vec{u}$ và $\vec{v}$ nhân với *sin* của góc xen giữa hai vectơ đó. Góc xen giữa hai vectơ này là góc định hướng, có số đo từ $-\pi$ tới π, số đo mang dấu dương nếu chiều quay từ $\vec{u}$ tới $\vec{v}$ là chiều thuận (ngược chiều kim đồng hồ) và mang dấu âm nếu chiều quay từ $\vec{u}$ tới $\vec{v}$ là chiều nghịch (theo chiều kim đồng hồ).

Hình 8.3. Góc có hướng

Tích chéo là một khái niệm suy ra từ khái niệm tích có hướng trong không gian vectơ Oclit nhiều chiều. Bằng các công cụ đại số tuyến tính, người ta đã chứng minh được công thức của tích chéo giữa hai vectơ $\vec{u} = (x_u; y_u)$ và $\vec{v} = (x_v; y_v)$:

$$\vec{u} \times \vec{v} = x_u y_v - x_v y_u = \begin{vmatrix} x_u & x_v \\ y_u & y_v \end{vmatrix} \qquad (8.9)$$

tức là giá trị của tích chéo bằng định thức của ma trận $\begin{pmatrix} x_u & x_v \\ y_u & y_v \end{pmatrix}$.

Ta cũng suy ra công thức tính sin của góc định hướng α giữa hai vectơ $\vec{u} = (x_u; y_u)$ và $\vec{v} = (x_v; y_v)$:

$$\sin\alpha = \frac{\vec{u} \times \vec{v}}{|\vec{u}|.|\vec{v}|} = \frac{x_u y_v - x_v y_u}{\sqrt{(x_u^2 + y_u^2)(x_v^2 + y_v^2)}}.$$

Về mặt hình học, giá trị tuyệt đối của tích chéo $\vec{u} \times \vec{v}$ là diện tích hình bình hành $OABC$, trong đó O là gốc toạ độ, $\overrightarrow{OA} = \vec{u}$, $\overrightarrow{OC} = \vec{v}$ và $\overrightarrow{OB} = \vec{u} + \vec{v}$.

Tích chéo có một ứng dụng quan trọng trong việc khảo sát chiều: Giả sử ta đi từ điểm A sang điểm B theo đường thẳng và đi tiếp sang điểm C theo đường thẳng, khi đó:

- Tích chéo $\overrightarrow{AB} \times \overrightarrow{BC}$ sẽ là số dương nếu chỗ rẽ tại B là "rẽ trái" (hay nói đúng hơn là bẻ góc ngược chiều kim đồng hồ);
- Tích chéo $\overrightarrow{AB} \times \overrightarrow{BC}$ là số âm nếu chỗ rẽ tại B là "rẽ phải";
- Tích chéo $\overrightarrow{AB} \times \overrightarrow{BC} = 0$ có nghĩa là ba điểm A, B, C thẳng hàng.

Hình 8.4. Rẽ trái và rẽ phải

Ta lấy ví dụ về một ứng dụng của tích chấm và tích chéo: Trên mặt phẳng cho ba điểm $A = (x_A; y_A)$, $B = (x_B; y_B)$ và $C = (x_C; y_C)$, hãy cho biết điểm C có nằm trên đoạn thẳng AB hay không.

Điều kiện cần và đủ để C nằm trên đoạn thẳng AB là: A, B, C thẳng hàng và hai vectơ $\overrightarrow{AC}$, $\overrightarrow{BC}$ không cùng hướng hoặc một trong số chúng là vectơ $\vec{0}$. Điều kiện này có thể viết bởi

$$\overrightarrow{AC} \times \overrightarrow{BC} = 0 \text{ và } \overrightarrow{AC}.\overrightarrow{BC} \leq 0.$$

c) Đường thẳng

Có nhiều cách biểu diễn một đường thẳng và do đó có nhiều cách biểu diễn đường thẳng trong máy tính. Sau đây là một số cách biểu diễn thường được sử dụng khi giải quyết các bài toán tin học:

- Dạng: $y = ax + b$. Mỗi đường thẳng được đặc trưng bởi một cặp hai hệ số a và b, tuy nhiên dạng biểu diễn này không thể hiện được các đường thẳng song song với trục Oy.

- Dạng tổng quát: $ax + by = c$. Mỗi đường thẳng có thể được biểu diễn bởi bộ ba hệ số (a, b, c). Đây vẫn là cách lưu trữ thông dụng nhất do tính tổng quát và trực quan của nó (gần gũi với đồ thị hàm số). Vectơ pháp tuyến $\vec{n}(a; b)$ có thể được dùng để viết một phương trình đường thẳng tương đương ngắn gọn hơn: $\vec{n}.\overrightarrow{OP} = (a; b).(x; y) = ax + by = c$ với mọi $P(x; y)$ nằm trên đường thẳng. Dạng tổng quát này còn được kí hiệu là $(\vec{n}, c)$.
- Dạng tham số: $(P, \vec{d})$, trong đó P là một điểm trên đường thẳng còn $\vec{d}$ là vectơ chỉ phương của đường thẳng với toạ độ là: $(-b; a)$. Như vậy, một đường thẳng được xác định bởi bộ bốn số x_0, y_0, a và b trong đó $(x_0; y_0)$ là toạ độ điểm P còn $(-b; a)$ là toạ độ của vectơ $\vec{d}$. Biểu diễn này có ý nghĩa: đường thẳng là tập các điểm có toạ độ chính là toạ độ của vectơ $\overrightarrow{OP} + s\vec{d}$ với mọi $s \in \mathbb{R}$.

Chuyển đổi giữa các biểu diễn được thực hiện dựa trên biến đổi tương đương đại số, cùng với công thức chuyển đổi giữa vectơ chỉ phương $\vec{d}(-b; a)$ và vectơ pháp tuyến $\vec{n}(a; b)$.

d) Góc

Góc thể hiện sự khác nhau về hướng của các vectơ và được biểu diễn bởi một số thực. Góc có nhiều đơn vị khác nhau: °, rad... Trong lập trình, để giảm sai số và tăng tốc độ tính toán, người ta tránh dùng *arcsin*, *arcos* mà thay vào đó bằng các đại lượng có cùng tính thứ tự như góc: *sin*/*cos*/*tang* của một góc có thể được tính nhanh hơn bằng toán tử +, –, *, / có sẵn trong các ngôn ngữ lập trình.

e) Đa giác

Đa giác là một đường gấp khúc khép kín. Trong lập trình, một đa giác được lưu bởi một dãy các đỉnh liên tiếp nhau $A_1, A_2, \ldots, A_n$.

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

$$S = \frac{(x_1 - x_2)(y_1 + y_2) + (x_2 - x_3)(y_2 + y_3) + \cdots + (x_n - x_1)(y_n + y_1)}{2}.$$

$|S|$ chính là diện tích của đa giác.

Dãy đỉnh của đa giác có thể được lưu cùng hoặc ngược chiều kim đồng hồ. Ta có thể biết thứ tự lưu đỉnh nhờ dấu của S: $S > 0$ có nghĩa là đỉnh của đa giác được liệt kê ngược chiều kim đồng hồ (và ngược lại).

Để hình dung công thức trên, ta có thể dựng các đường thẳng đứng xuống trục hoành. Mỗi hạng tử sẽ là diện tích đại số của một hình thang thành phần. Diện tích các hình thang phía trên sẽ ngược dấu với các hình thang bên dưới, từ đó cho ta diện tích đại số của đa giác khi lấy tổng của chúng.

Nếu vẫn phân vân về lời giải thiếu tính xây dựng trên, ta có thể xây dựng lại công thức tính diện tích bằng cách khác: tính tổng diện tích các tam giác $A_1A_2A_3$, $A_1A_3A_4$, …, $A_1A_{n-1}A_n$. Diện tích mỗi tam giác được tính bằng tích có hướng của hai vecto $\overrightarrow{A_1A_{i-1}}$ và $\overrightarrow{A_1A_i}$, $i = \overline{3;n}$, do đó đây là diện tích có dấu. Tổng của các diện tích này sau khi rút gọn sẽ đưa về biểu thức trên đây và do đó cũng lí giải tại sao S lại là diện tích đại số.

Dưới đây là chương trình tính diện tích đa giác với khuôn dạng Input/Output như sau:

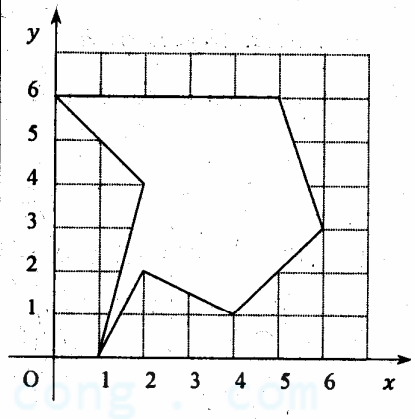
Input

- Dòng đầu tiên chứa số đỉnh n của đa giác ($1 \le n \le 10^6$);
- n dòng tiếp theo, dòng thứ i chứa hai số thực x_i, y_i là toạ độ đỉnh i của đa giác.

Output

Chiều của đa giác (CCW/CW) và diện tích đa giác

Sample Input	Sample Output
7 1 0 2 2 4 1 6 3 5 6 0 6 2 4	Polygon direction: CCW Area = 18.5000



Hình 8.5

💻 POLYGONAREA.PAS ✓ Tính diện tích đa giác

```
{$MODE OBJFPC}
program PolygonArea;
uses Math;
const maxN = 1000000;
var
 x, y: array[0..maxN + 1] of Float;
 i, n: Integer; s: Float;
begin
 ReadLn(n);
 for i := 1 to n do ReadLn(x[i], y[i]);
 x[0] := x[n]; x[n + 1] := x[1];
 s := 0;
 for i := 1 to n do
 s := s + y[i] * (x[i - 1] - x[i + 1]);
 if s > 0 then WriteLn('Polygon direction: CCW')
 else WriteLn('Polygon direction: CW');
 WriteLn('Area = ', Abs(s)/2:0:4);
end.
```

f) Đường tròn

Đường tròn $(O; R)$ là tập hợp các điểm cách đều tâm $O(x; y)$ một khoảng cách R. Đường tròn được hoàn toàn xác định bởi bộ ba số (x, y, R) $(R > 0)$.

II. Một số bài toán cơ bản

1. *Biểu diễn tuyến tính*

Bài toán đầu tiên ta xét đến là cho ba vecto $\vec{a}$, $\vec{b}$ và $\vec{c}$, hãy tìm hai số thực p, q để:

$$\vec{c} = p\vec{a} + q\vec{b}.$$

Hai số p, q có thể tính bằng công thức:

$$p = \frac{\vec{c} \times \vec{b}}{\vec{a} \times \vec{b}} = \frac{Dx}{D};$$

$$q = \frac{\vec{a} \times \vec{c}}{\vec{a} \times \vec{b}} = \frac{Dy}{D}.$$

- Nếu $D = \vec{a} \times \vec{b} \neq 0$ thì có duy nhất một cách biểu diễn tuyến tính vectơ $\vec{c}$ qua hai vectơ $\vec{a}$ và $\vec{b}$ (nghiệm $(p; q)$ là duy nhất).
- Nếu $D = 0$, hai vectơ $\vec{a}$ và $\vec{b}$ song song với nhau, khi đó:
  - $(p; q) = (Nan; Nan)$ nếu $\vec{c}$ song song với cả $\vec{a}$ và $\vec{b}$;
  - $(p; q) = (Inf; Inf)$ nếu $\vec{c}$ không song song với $\vec{a}$ và $\vec{b}$.

Ta viết hàm

```
function SolveSLE(const a, b, c: TVector): TVector;
```

nhận vào ba vectơ $\vec{a}, \vec{b}, \vec{c}$ và trả về một vectơ có toạ độ $(x; y)$ tương ứng với hệ số $(p;, q)$ cần tìm:

```
function SolveSLE(const a, b, c: TVector): TVector;
var
 D: Float;
begin
 D := a >< b;
 Result := Vector((c >< b)/D, (a >< c)/D);
end;
```

2. *Tìm giao điểm của hai đường thẳng*

Trên mặt phẳng với hệ toạ độ Đề-các vuông góc cho hai đường thẳng với phương trình tổng quát:

$$A_1x + B_1y + C_1 = 0;$$
$$A_2x + B_2y + C_2 = 0.$$

Bài toán đặt ra là xác định giao điểm của hai đường thẳng đã cho.

Đặt $\vec{u} = (A_1; A_2)$, $\vec{v} = (B_1; B_2)$ và $\vec{w} = (-C_1; -C_2)$, bài toán trở thành bài toán biểu diễn vectơ $\vec{w}$ qua tổ hợp tuyến tính của hai vectơ $\vec{u}$ và $\vec{w}$

$$\vec{w} = x.\vec{u} + y.\vec{v}.$$

Việc còn lại chỉ là biện luận cho giá trị giao điểm tìm được.

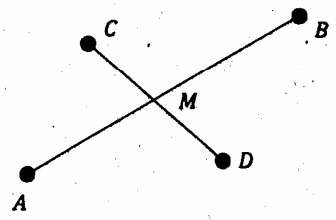
LINEINTERSECT.PAS ✓ Tìm giao điểm của hai đường thẳng

```
{$MODE OBJFPC}
program LineIntersection;
uses math;
type
```

```
 TPoint = record
            x, y: Float;
          end;
 TVector = TPoint;
var
 a1, b1, c1, a2, b2, c2: Float;
 p: TPoint;
function Vector(x, y: Float): Tvector;
begin
 Result.x := x; Result.y := y;
end;
//Tích chéo của hai vecto
operator ><(const u, v: TVector): Float;
begin
 Result := u.x * v.y - u.y * v.x;
end;
//Biểu diễn tuyến tính
function SolveSLE(const a, b, c: TVector): TVector;
var
 D: Float;
begin
 D := a >< b;
 Result := Vector((c >< b)/D, (a >< c)/D);
end;
//Chương trình chính
BEGIN
SetExceptionMask([Low(TFPUExceptionMask)..High(TFPUExceptionMask)]);
ReadLn(a1, b1, c1, a2, b2, c2);
//Tìm giao điểm của hai đường thẳng
p := SolveSLE(Vector(a1,a2),Vector(b1,b2),Vector(c1,c2));
 if IsNan(p.x) then //Hai đường thẳng trùng nhau
 WriteLn('Two lines are coincident')
 else if IsInfinite(p.x) then //Hai đường thẳng song song
 WriteLn('Two lines are parallel')
 else //Hai đường thẳng có giao điểm duy nhất
WriteLn('Intersection point(',p.x:0:4,',', p.y:0:4,')');
END.
```

3. *Tìm giao điểm của hai đoạn thẳng*

Bài toán tiếp theo ta xét là cho bốn điểm A, B, C, D trên mặt phẳng, hãy cho biết hai đoạn thẳng AB và CD có giao điểm duy nhất không, nếu có cho biết toạ độ giao điểm.



Hình 8.6

Để tìm giao điểm của hai đoạn thẳng AB và CD ta có thể:

- Viết phương trình tổng quát của hai đường thẳng AB và CD;
- Tìm giao điểm M của hai đường thẳng;
- Kiểm tra M có nằm trên đoạn thẳng AB và trên đoạn thẳng CD hay không.

Phương trình của đường thẳng đi qua hai điểm phân biệt $A = (x_A; y_A)$ và $B = (x_B; y_B)$ là

$$(x - x_A) * (y_B - y_A) = (y - y_A) * (x_B - x_A).$$

Ta có thể biến đổi phương trình trên về phương trình tổng quát và sử dụng hàm *SolveSLE* để tìm giao điểm M của hai đường thẳng AB và CD, sau đó kiểm tra M có nằm trên hai đoạn thẳng AB và CD hay không.

Cách làm này kéo theo nhiều phép tính không hiệu quả, ta sẽ sử dụng công thức khác.

Nếu M là giao điểm duy nhất của hai đoạn thẳng AB và CD thì sẽ tồn tại duy nhất một cặp số thực $p, q \in [0; 1]$ để

$$\begin{cases} \overrightarrow{AM} = p\,\overrightarrow{AB} \\ \overrightarrow{CM} = q\,\overrightarrow{CD}. \end{cases}$$

Trừ tương ứng hai vế, ta có

$$\overrightarrow{AC} = p\,\overrightarrow{AB} + q\,\overrightarrow{DC}.$$

Như vậy, ta chỉ cần tìm biểu diễn tuyến tính của $\overrightarrow{AC}$ qua $\overrightarrow{AB}$ và $\overrightarrow{DC}$, sau khi có cặp số p, q, ta kiểm tra điều kiện $p, q \in [0; 1]$ và tính toạ độ giao điểm M theo công thức:

$$\overrightarrow{OM} = \overrightarrow{OA} + p\overrightarrow{AB}.$$

Thuật toán như sau:

```
Input → A, B, C, D: TPoint;
r := SolveSLE(B - A, C - D, C - A);
if not InRange(r.x, 0, 1) or not InRange(r.y, 0, 1) then
//Kiểm tra nghiệm ∈[0,1]
Output ← Không có giao điểm duy nhất
else
begin
p := A + (B - A) * r.x;
Output ← Giao điểm duy nhất (p.x, p.y)
end;
```

4. Tìm giao điểm giữa một đoạn thẳng và một tia

Để tìm giao điểm duy nhất giữa đoạn thẳng AB và tia CD, ta có thể sử dụng phương pháp tương tự như trên: Nếu M là giao điểm duy nhất của đoạn thẳng AB và tia CD thì sẽ tồn tại duy nhất một cặp số thực p, q, trong đó $p \in [0; 1]$, $q \in [0; +\infty)$ sao cho

$$\begin{cases} \overrightarrow{AM} = p\,\overrightarrow{AB} \\ \overrightarrow{CM} = q\,\overrightarrow{CD}. \end{cases}$$

Áp dụng thuật toán như trước, chỉ có điều sau khi tìm được cặp số p, q, ta không kiểm tra $p, q \in [0; 1]$ mà kiểm tra $p \in [0; 1]$ và $q > 0$.

Cách tìm giao điểm của đường thẳng AB với đường thẳng CD, tia AB với tia CD, đường thẳng AB với đoạn thẳng CD... có thể thực hiện theo cách tương tự, chỉ cần sửa đổi phạm vi của tham số p, q.

5. Đo góc giữa hai vectơ

Bài toán: Cho hai vectơ khác $\vec{0}$: $\vec{u}$ và $\vec{v}$. Cần tìm số đo góc định hướng từ $\vec{u}$ tới $\vec{v}$. Số đo góc định hướng nằm trong phạm vi $(-\pi; \pi]$.

Thư viện *math* cung cấp hàm $ArcTan2(y, x)$ trả về góc định hướng tạo bởi vectơ $(1; 0)$ với vectơ $(x; y)$. Ta có thể sử dụng hàm này để tính giá trị góc định hướng từ $\vec{u}$ tới $\vec{v}$ bằng cách viết:

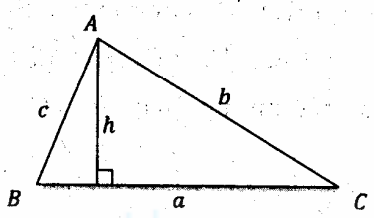
$$\text{ArcTan2}(\vec{u} \times \vec{v}, \vec{u}.\vec{v}).$$

Thủ tục như sau:

```
function Rad(const u, v: TVecto): Float;
begin
 Result := ArcTan2(u >< v, u * v);
end;
```

6. *Tính diện tích*

a) Diện tích tam giác



Hình 8.7

Tuỳ thông tin được cho mà ta sử dụng công thức giải tích thích hợp để tính diện tích tam giác:

- Nếu biết độ dài một cạnh của tam giác là a và chiều cao tương ứng với cạnh đó là h thì diện tích tam giác có thể tính bởi công thức:

$$S = \frac{a + h}{2}.$$

- Nếu biết độ dài hai cạnh tam giác là b và c, đồng thời góc xen giữa hai cạnh đó là A thì diện tích tam giác có thể tính bởi công thức:

$$S = \frac{bc \sin A}{2}.$$

- Nếu ta biết độ dài ba cạnh của tam giác là a, b, c thì diện tích tam giác có thể tính bởi công thức Hê-rông:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

trong đó $p = \frac{a+b+c}{2}$ (nửa chu vi).

- Nếu biết hai vectơ $\overrightarrow{AB}$ và $\overrightarrow{AC}$ thì diện tích tam giác có thể tính bằng công thức tích chéo:

$$S = \left|\frac{\overrightarrow{AB} \times \overrightarrow{AC}}{2}\right| \tag{8.9}$$

Công thức này có thể viết lại khi ta biết toạ độ ba điểm $A = (x_A; y_A)$, $B = (x_B; y_B)$ và $C = (x_C; y_C)$:

$$S = \left|\frac{(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)}{2}\right| \tag{8.10}$$

Chú ý rằng nếu bỏ dấu giá trị tuyệt đối, công thức tính diện tích ở trên sẽ cho giá trị dương nếu hướng quay từ vectơ $\overrightarrow{AB}$ tới vectơ $\overrightarrow{AC}$ là hướng thuận. Ngược lại, công thức sẽ cho giá trị âm nếu hướng quay từ vectơ $\overrightarrow{AB}$ tới vectơ $\overrightarrow{AC}$ là hướng nghịch.

b) Diện tích đa giác

Trên mặt phẳng với hệ toạ độ Đề-các vuông góc, cho đa giác $P = P_1P_2 \dots P_n$, trong đó điểm P_i có toạ độ $(x_i; y_i)$. Bổ sung thêm điểm $P_0 \equiv P_n$ và điểm $P_{n+1} \equiv P_1$, khi đó diện tích đa giác được tính theo công thức (8.11):

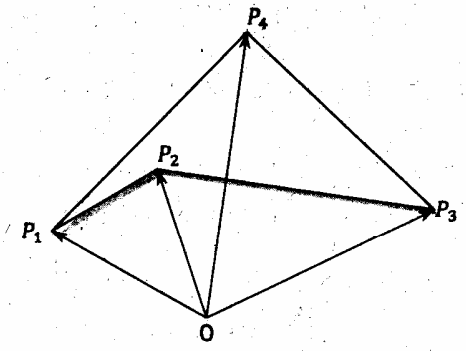
$$\begin{aligned} S &= \frac{1}{2}\left|\sum_{i=1}^{n} \overrightarrow{OP_i} \times \overrightarrow{OP_{i+1}}\right| \\ &= \frac{1}{2}\left|\sum_{i=1}^{n} (x_{i-1} - x_{i+1}) y_i\right| \end{aligned} \tag{8.11}$$

Chứng minh

Giả sử rằng các đỉnh của đa giác được đánh số theo hướng thuận (ngược chiều kim đồng hồ)*, ta nối các điểm $P_1, P_2, \dots, P_n$ với gốc toạ độ O, được n tam giác:

* Cụ thể là nếu ta đi dọc theo các cạnh của đa giác theo thứ tự $P_1, P_2, \dots, P_n, P_1$ thì phía tay trái là miền đa giác.

$P_1OP_2, P_2OP_3, \ldots, P_{n-1}OP_n, P_nOP_1$.



Hình 8.8

Trong hình 8.8, ta xét một tứ giác với bốn đỉnh P_1, P_2, P_3, P_4. Diện tích của tứ giác này có thể tính bằng:

$$S = -S_\Delta(P_1OP_2) - S_\Delta(P_2OP_3) + S_\Delta(P_3OP_4) + S_\Delta(P_4OP_1)$$

tức là diện tích đa giác có thể tính qua các diện tích các tam giác $S_\Delta(P_iOP_{i+1})$. Khi một hạng tử $S_\Delta(P_iOP_{i+1})$ xuất hiện trong tổng, nó sẽ được mang dấu "+" nếu chiều quay từ vectơ $\overrightarrow{OP_i}$ tới vectơ $\overrightarrow{OP_{i+1}}$ là hướng thuận, ngược lại, hạng tử này sẽ mang dấu "-". Thật may mắn, giá trị (có dấu) của các hạng tử $S_\Delta(P_iOP_{i+1})$ có thể tính bằng công thức tích chéo (8.9) nhưng bỏ dấu giá trị tuyệt đối. Tức là diện tích (có dấu) của tam giác P_iOP_{i+1} sẽ được tính bằng:

$$S_\Delta(P_iOP_{i+1}) = \frac{1}{2}\overrightarrow{OP_i} \times \overrightarrow{OP_{i+1}}$$

và như vậy ta có công thức tính diện tích đa giác P:

$$\begin{aligned} S &= \frac{1}{2}\left(\overrightarrow{OP_1} \times \overrightarrow{OP_2} + \overrightarrow{OP_2} \times \overrightarrow{OP_3} + \cdots + \overrightarrow{OP_n} \times \overrightarrow{OP_{n+1}}\right) \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} \overrightarrow{OP_i} \times \overrightarrow{OP_{i+1}}\right) \end{aligned} \tag{8.12}$$

(quy ước $P_{n+1} \equiv P_1$).

Khi các đỉnh của đa giác được đánh số theo chiều nghịch (theo chiều kim đồng hồ), bằng lập luận tương tự, ta có công thức diện tích đa giác là

$$S = -\frac{1}{2}\left(\sum_{i=1}^{n} \overrightarrow{OP_i} \times \overrightarrow{OP_{i+1}}\right). \tag{8.13}$$

Tổng hợp lại, công thức tính diện tích đa giác có thể viết thành:

$$S = \frac{1}{2}\left|\sum_{i=1}^{n} \overrightarrow{OP_i} \times \overrightarrow{OP_{i+1}}\right| \tag{8.14}$$

mà không cần quan tâm tới đa giác được đánh số theo chiều thuận hay chiều nghịch.

Bây giờ ta sẽ sử dụng tới toạ độ để xây dựng công thức tính diện tích đa giác dựa trên các toạ độ đỉnh. Vectơ $\overrightarrow{OP_i}$ có toạ độ $(x_i; y_i)$, dựa vào công thức toạ độ của tích chéo, ta có:

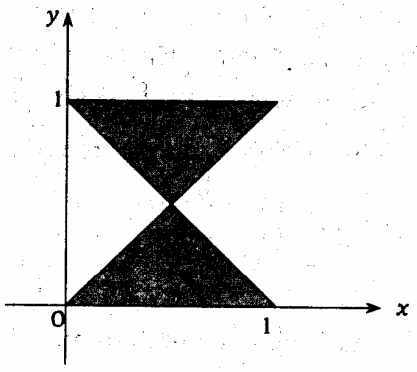
$$\begin{aligned} S &= \frac{1}{2}\left|\sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i)\right| \\ &= \frac{1}{2}\left|\sum_{i=1}^{n} (x_{i-1} - x_{i+1}) y_i\right|. \end{aligned} \tag{8.15}$$

Việc chứng minh đẳng thức thứ hai có thể sử dụng vài phép biến đổi đại số. Ở đây ta có thể hình dung như sau: Do cơ chế quay vòng số thứ tự (đỉnh 0 trùng với đỉnh n và đỉnh $n + 1$ trùng với đỉnh 1), với mỗi đỉnh i sẽ có đúng một hạng tử $+x_{i-1}y_i$ và một hạng tử $-x_{i+1}y_i$ xuất hiện trong tổng $\sum_{i=1}^{n}(x_i y_{i+1} - x_{i+1} y_i)$. Ta có thể ghép hai hạng tử này lại với thừa số chung y_i.

Nếu như việc tính đẳng thức thứ nhất cần $2n$ phép nhân thì ở đẳng thức thứ hai ta chỉ cần n phép nhân mà thôi (số phép cộng và phép trừ không thay đổi). Chính vì vậy, đẳng thức thứ hai hiệu quả hơn khi lập trình tính diện tích đa giác. ■

Chú ý: Khi sử dụng công thức (8.11):

- Ta có thể xác định phép đánh số các đỉnh đa giác là thuận hay nghịch dựa vào công thức tính diện tích: Bỏ dấu giá trị tuyệt đối trong công thức (8.11), khi đó $S > 0$ tương ứng với phép đánh số thuận và $S < 0$ tương ứng với phép đánh số nghịch.
- Đường biên đa giác phải là một đường gấp khúc khép kín, không tự cắt. Công thức (8.11) sẽ sai nếu dữ liệu vào không thoả mãn điều kiện này.
- Hình 8.9 là một ví dụ về một "đa giác" gồm bốn đỉnh $(0; 0)$, $(1; 1)$, $(0; 1)$ và $(1; 0)$.



Hình 8.9

Công thức (8.11) sẽ tính được diện tích miền mặt phẳng giới hạn bởi đường gấp khúc này là 0, tuy nhiên trên thực tế thì diện tích miền này là 0.5.

c) Diện tích đường cong

Trên mặt phẳng cho C là một đường cong khép kín, không tự cắt, trơn từng khúc, có phương trình tham số:

$$\begin{cases} x = x(t) \\ y = y(t) \end{cases}$$

với $t \in [a, b]$, $x(a) = x(b)$ và $y(a) = y(b)$.

Ta có thể hình dung khi t chạy từ a tới b thì các điểm $(x(t); y(t))$ vẽ ra trên mặt phẳng trực chuẩn một đường cong khép kín không tự cắt C. Đường cong C trơn từng khúc tức là ta có thể chia khoảng $[a; b]$ thành một số đếm được các khoảng con mà trên mỗi khoảng con đó, các đạo hàm $x'(t)$ và $y'(t)$ là các hàm liên tục.

Khi đó, diện tích của miền mặt phẳng D giới hạn bởi đường cong C có thể tính qua một hệ quả của công thức Green trong giải tích hàm:

$$\begin{aligned} S &= \frac{1}{2}\int_C x dy - y dx \\ &= \frac{1}{2}\int_a^b \big(x(t)y'(t) - y(t)x'(t)\big)dt. \end{aligned} \tag{8.16}$$

Chú ý: Công thức (8.16) cho kết quả là diện tích có dấu của miền D (là số âm nếu miền D được định hướng nghịch). Có thể thêm vào công thức dấu giá trị tuyệt đối nếu chỉ cần biết diện tích.

Thực ra công thức tính diện tích đa giác mà ta trình bày ở mục trước chỉ là trường hợp riêng của công thức (8.16). Việc chứng minh công thức (8.16) cần phải sử dụng nhiều kiến thức của toán cao cấp, ta chỉ cần nhớ và áp dụng.

Ví dụ: Xét đường tròn tâm O bán kính R, đường tròn này có phương trình tham số

$$\begin{cases} x = R\cos t \\ y = R\sin t \end{cases}$$

với $t \in [0; 2\pi]$.

Áp dụng công thức (8.16), ta có diện tích hình tròn:

$$S = \frac{1}{2}\int_0^{2\pi} (R^2\cos^2 t + R^2\sin^2 t)dt = \frac{1}{2}R^2\int_0^{2\pi} dt = \pi R^2.$$

III. Một số bài toán thông dụng khác

1. Tam giác

Xét một tam giác xác định bởi toạ độ ba đỉnh là ba điểm $A(A.x; A.y)$, $B(B.x; B.y)$, $C(C.x; C.y)$. Kí hiệu a, b, c là độ dài ba cạnh của tam giác ABC.

a) Đường tròn ngoại tiếp tam giác

Tâm đường tròn ngoại tiếp tam giác ABC là giao điểm của hai đường trung trực thuộc hai cạnh của tam giác. Đường trung trực của cạnh AB được xác định thông qua trung điểm $M\left(\frac{A.x+B.x}{2}; \frac{A.y+B.y}{2}\right)$ của cạnh AB và vectơ chỉ phương là vectơ pháp tuyến của đường thẳng AB.

b) Đường phân giác

Đường phân giác của góc $\widehat{ABC}$ được xác định thông qua dạng biểu diễn $(P, \vec{d})$ trong đó P chính là điểm B còn vectơ $\vec{d}$ xác định theo công thức sau:

$$\vec{d} = \frac{\overrightarrow{BA}}{|\overrightarrow{BA}|} + \frac{\overrightarrow{BC}}{|\overrightarrow{BC}|}$$

c) Đường tròn nội tiếp tam giác

Tâm của đường tròn nội tiếp tam giác ABC có thể xác định bằng cách tìm giao điểm của hai đường phân giác. Tuy nhiên, ta có thể sử dụng công thức toán học để xác định toạ độ tâm của đường tròn nội tiếp chính là toạ độ của vectơ:

$$\frac{a.\overrightarrow{OA} + b.\overrightarrow{OB} + c.\overrightarrow{OC}}{a+b+c}$$

trong đó O là gốc toạ độ.

Bán kính đường tròn nội tiếp bằng

$$r = \frac{2S}{a+b+c}$$

với S là diện tích tam giác.

2. *Kiểm tra điểm nằm trong đa giác*

Bài toán. Trên mặt phẳng với hệ toạ độ Đề-các vuông góc, cho đa giác $P = P_1P_2 \dots P_n$ và một điểm A. Hãy cho biết điểm A có nằm trong đa giác hay không.

Thuật toán

Nếu A thuộc một cạnh đa giác thì kết luận ngay A nằm trong đa giác. Nếu không, từ điểm A, xác định một tia gốc A không đi qua đỉnh nào của đa giác, gọi tia này là tia AB. Có nhiều cách chọn tia này, chẳng hạn như chọn ngẫu nhiên $n+1$ tia đôi một khác nhau, khi đó chắc chắn sẽ có một tia không đi qua đỉnh nào của đa giác. Tuy nhiên trên thực tế, phương pháp thực dụng tỏ ra hiệu quả hơn: sinh ngẫu nhiên một tia, nếu tia đó đi qua một đỉnh của đa giác thì sinh ngẫu nhiên một tia khác và thử lại…

Nếu tia AB cắt cạnh đa giác một số lẻ lần thì điểm A nằm trong đa giác, ngược lại thì điểm A nằm ngoài đa giác.

Tất cả những kĩ thuật tìm giao điểm, xác định tia đã được nói đến trong các bài trước, ta chỉ tổng hợp lại các kĩ thuật này để lập trình giải bài toán điểm nằm trong đa giác.

Input

- Dòng thứ nhất chứa số nguyên $n \le 10^6$ là số đỉnh của đa giác và hai số thực xA, yA tương ứng là hoành độ và tung độ điểm A;

- n dòng tiếp theo, dòng thứ i chứa hai số thực x_i, y_i là toạ độ đỉnh P_i của đa giác.

Output

Cho biết điểm A có nằm trong đa giác P hay không?

💻 PTINPOLYGON.PAS ✓ Kiểm tra điểm nằm trong đa giác

```
{$MODE OBJFPC}
program PointInPolygon;
uses Math;
const
 maxN = 1000000; epsilon = 1E-6;
type
 TPoint = record
            x, y: Float;
          end;
 TVector = TPoint;
var
 p: array[1..maxN + 1] of TPoint;
 n: Integer; A, B: TPoint;
 Inside: Boolean;
function Vector(x, y: Float): Tvector;
begin
 Result.x := x; Result.y := y;
end;
//Phép trừ vecto
operator -(const u, v: TVector): TVector;
begin
 Result.x := u.x - v.x;
 Result.y := u.y - v.y;
end;
//Tích chấm của hai vecto
operator *(const u, v: TVector): Float;
begin
 Result := u.x * v.x + u.y * v.y;
end;
//Tích chéo của hai vecto
operator ><(const u, v: TVector): Float;
begin
 Result := u.x * v.y - u.y * v.x;
end;
```

```
procedure Enter;
var i: Integer;
begin
 ReadLn(n, A.x, A.y);
 for i := 1 to n do
   with p[i] do
     begin
       ReadLn(x, y);
     end;
   p[n + 1] := p[1];
end;
//Kiểm tra điểm nằm trên đoạn thẳng
function OnSegment(const P, Q: TPoint): Boolean;
begin
 Result := IsZero(P >< Q, epsilon) and (P * Q <= 0)
end;
//Tạo tia
procedure MakeRay;
var
 OK: Boolean;
 i: Integer;
begin
 repeat
   OK := True; B.x := Random; B.y := Random;
   for i := 1 to n do
     if IsZero((B - A) >< p[i], epsilon) then
       begin
         OK := False;
         Break;
       end;
 until OK;
end;
//Biểu diễn tuyến tính
function SolveSLE(const a, b, c: TVecto): TVecto;
var D: Float;
begin
 D := a >< b;
 Result := Vecto((c >< b)/D, (a >< c)/D);
end;
//Kiểm tra tia cắt cạnh đa giác
function Cut(const C, D: TPoint): Boolean;
var r: TVecto;
```

```
begin
 r := SolveSLE(B - A, C - D, C - A);
 Result := (r.x >= 0) and InRange(r.y, 0, 1);
end;
function Check: Boolean;
var i: Integer;
begin
 for i := 1 to n do
   if OnSegment(p[i], p[i + 1]) then Exit(True);
 MakeRay;
 Result := False;
 for i := 1 to n do
   if Cut(p[i], p[i + 1]) then Result := not Result;
end;
//Chương trình chính
BEGIN
SetExceptionMask([Low(TFPUExceptionMask)..High(TFPUExceptionMask)]);
 Enter;
 if Check then WriteLn('Inside the polygon')
 else WriteLn('Outside the polygon');
END.
```

3. *Tìm hai điểm gần nhất*

Cho một tập điểm Q (có thể trùng nhau). Tìm cặp điểm trong Q có khoảng cách nhỏ nhất.

Nếu kiểm tra tất cả các cặp điểm ta sẽ có một thuật toán với độ phức tạp $O(N^2)$. Thuật toán sau đây có độ phức tạp $O(N\log N)$ giải quyết vấn đề bằng cách chia để trị. Với một bài toán kích thước N, ta đưa về thuật toán giải hai bài toán nhỏ kích thước $\frac{N}{2}$, và kết hợp kết quả trong thời gian $O(N)$.

Q được biểu diễn bởi hai mảng đã sắp xếp X (theo hoành độ) và Y (theo tung độ). Thuật toán như sau:

Bước 1. Dùng một đường thẳng đứng d, ta có thể chia đôi Q thành Q_1 và Q_2. Các điểm nằm đúng trên d có thể phân bố tuỳ ý sao cho Q_1 và Q_2 cân bằng.

Bước 2. Gọi đệ quy tìm khoảng cách ngắn nhất d_1 trong Q_1 và d_2 trong Q_2. Đặt $d_0 = \min(d_1,d_2)$.

Bước 3. Kết quả trả về sẽ là d_0, hoặc một giá trị nhỏ hơn d_0 nếu tồn tại một điểm trong Q_1 và một điểm trong Q_2 có khoảng cách nhỏ hơn d_0. Các điểm như

vậy chỉ có thể nằm trong khoảng cách d_0 từ đường thẳng d. Tìm tập hợp Q_0 các điểm thoả mãn đường thẳng đó, cùng với Y_0 tương ứng của Q_0.

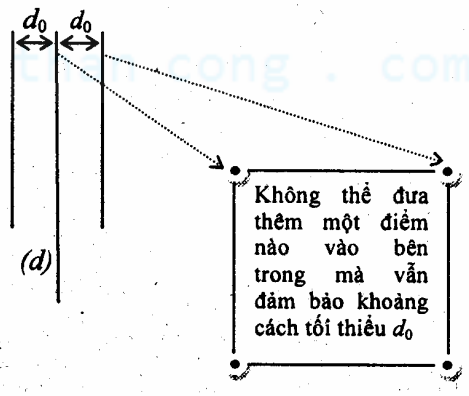
Bước 4. Với mỗi điểm P_i trong Q_0, ta xét khả năng P_i là đầu mút phía trên của cặp điểm gần nhau nhất. Như vậy, điểm còn lại phải nằm trong hình chữ nhật $d_0 \times 2d_0$ có cạnh trên đi qua P_i. Hình chữ nhật này có tối đa bốn điểm trong Q_1 và bốn điểm trong Q_2. Do đó nó có tối đa tám điểm trong Q_0. Như vậy, ta chỉ cần kiểm tra khoảng cách của P_i đến bảy điểm theo sau nó trong Y_0.

Bước 5. Trả về khoảng cách ngắn nhất tìm được.

Tại bước 2, ta cần X_1, X_2, Y_1, Y_2 tương ứng cho Q_1, Q_2. Các mảng này có thể được sinh ra trong thời gian $O(N)$.

Tại bước 3, Y_0 cũng được sinh ra trong $O(N)$ bằng cách đọc tuần tự mảng Y và loại ra các điểm cách xa d.

Các bước 1, 2, 3, 4 đảm bảo chi phí xử lí $O(N)$ tại mỗi lần đệ quy, từ đó cho ta thuật toán có độ phức tạp $O(N\log N)$.



Hình 8.10. Khoảng cách tối thiểu

Minh hoạ. Do các điểm trong Q_1 có khoảng cách tối thiểu d_0, từ P_i trở xuống chỉ có tối đa bốn điểm thuộc về Q_1.

4. *Một số thủ tục cơ bản*

a) Kiểm soát sai số

Khi tính toán các công thức hình học trên máy tính, có rất nhiều phép tính không thể đạt được độ chính xác tuyệt đối như phép chia, phép khai căn bậc 2... Ngay

cả những phép cộng và nhân số thực trên máy tính cũng phải chịu sai số ở chữ số có nghĩa cuối cùng.

Sai số trong các phép tính có thể dẫn tới nhiều phiền toái, vì thế phải có phương pháp để kiểm soát sai số này trong quá trình trính toán. Kinh nghiệm chung là:

- Nếu dữ liệu vào là các số nguyên, ta cố gắng sử dụng các phép tính toán số nguyên đến chừng nào còn có thể được.
- Nếu bắt buộc phải tính toán số thực, trong một số trường hợp ta cần phải dự kiến sai số tính toán. Chẳng hạn, chọn một hằng số ϵ đủ nhỏ (chẳng hạn $\epsilon = 10^{-6}$), sau đó thay vì so sánh $p = q$, ta viết $|p - q| < \epsilon$.

Ví dụ, ta sẽ viết hàm $OnSegment(xA, yA, xB, yB, xC, yC)$ để kiểm tra xem điểm $C = (xC; yC)$ có nằm trên đoạn thẳng AB với $A = (xA; yA)$, $B = (xB; yB)$ hay không:

```
Const epsilon = 1E-6;
function OnSegment(xA, yA, xB, yB, xC, yC: Float): Boolean;
var
 x1, y1, x2, y2: Float;
begin
 x1 := xC - xA; y1 := yC - yA; //(x1, y1) = vectơ AC
 x2 := xC - xB; y2 := yC - yB; //(x2, y2) = vectơ BC
 Result :=
          (Abs(x1 * y2 - x2 * y1) < epsilon) and
                                    (x1 * x2 + y1 * y2 ≤ 0)
end;
```

Việc kiểm soát sai số không phải lúc nào cũng cần thực hiện. Cụ thể trong trường hợp trên, ta chỉ kiểm soát sai số trong phép so sánh $\overrightarrow{AC} \times \overrightarrow{BC} = 0$ mà không cần kiểm soát sai số trong phép so sánh $\overrightarrow{AC}.\overrightarrow{BC} \leq 0$. Lí do là bởi vì hàm *OnSegment* bản thân nó đã có sai số (nếu điểm C nằm rất sát đoạn thẳng AB nhưng không nằm trên đoạn thẳng AB, hàm vẫn có thể trả về giá trị *True*) và sai số của phép so sánh $\overrightarrow{AC}.\overrightarrow{BC} \leq 0$ được tính luôn vào sai số của hàm.

b) Các thủ tục, hàm thường dùng

Một trong những khó khăn gặp phải khi giải quyết các bài toán hình học tính toán là ta phải làm việc với số thực và thường gặp nhiều sai sót trong khi lập trình vì chương trình thường rất dài và cần phải cài đặt tỉ mỉ, chi tiết. Những thủ

tục này giúp bạn tránh được những sai sót đáng tiếc và giúp bạn cài đặt nhanh các thuật toán cho bài toán hình học.

```
Type Real=Extended;
 TPoint=Record
          x,y:real;
        end;
Const_Eps: Real=1e-3;
     ZeroPnt: TPoint=(X:0; Y:0);
/* so sánh hai số thực */
Function RealEq(Const a, b:real): Boolean;
Begin
 RealEq:=Abs(a-b)<=_Eps;
End;
Function RealMax(Const a, b:Real):Real;
Begin
 If RealMore(a,b) Then RealMax:=a Else RealMax:=b;
End;
/* Kiểm tra hai điểm trùng nhau */
Function EqPoint(Const A, B:TPoint):Boolean;
Begin
 EqPoint:=RealEq(A.x, B.x) And RealEq(A.y, B.y);
End;
/* Khoảng cách hai điểm */
Function Dist(Const A, B:TPoint):Real;
Begin
 Dist:=Sqrt(Sqr(A.x-B.x)+Sqr(A.y-B.y));
End;
/* Tổng vectơ */
Type TVecCart=TPoint;
Procedure AssVecCart(Const a, b:TVecCart;Var c:TVecCart);
Begin
 c.x:=a.x+b.x;
c.y:=a.y+b.y;
End;
/* Tích vô hướng */
Function SkMulCart(Const a, b:TVecCart):Real;
Begin
 SkMulCart:=a.x*b.x+a.y*b.y;
End;
Type TVecPol=Record
                rst, angle:Real
              End;
```

```
/* Góc với trục Ox, đo bằng radian */
Function GetAngle(Const x, y:Real):Real;
Var rs, c:Real;
Begin
 rs:=Sqrt(Sqr(x)+Sqr(y));
 If RealEq(rs, 0) Then GetAngle:=0
 Else
   Begin
     c:=x/rs;
     If RealEq(c, 0) Then c:=Pi/2
     Else c:=ArcTan(Sqrt(Abs(1-Sqr(c)))/c);
     If RealLess(c, 0) Then c:=Pi+c;
     If RealLess(y, 0) Then c:=2*Pi-c;
     GetAngle:=c;
  End;
End;
/* toạ độ Đề-các chuyển sang toạ độ cực */
Procedure CartToPol(Const a:TVecCart;Var b:TVecPol);
Begin
 b.rs:=Sqrt(Sqr(a.x)+Sqr(a.y));
 b.angle:=GetAngle(a.x, a.y);
End;
/* Toạ độ cực chuyển sang toạ độ Đề-các*/
Procedure PolToCart(Const a:TVecPol;Var b:TVecCart);
Begin
 b.x:=a.rs*cos(a.angle);
 b.y:=a.rs*sin(a.angle);
End;
Type TLine=Record
             A,B,C:Real;
           End;
/* Viết phương trình đường thẳng đi qua hai điểm */
Procedure Point2ToLine(Const v, w:Tpoint;Var L:TLine);
Begin
 L.A:=v.y -w.y;
 L.B:=w.x- v.y;
 L.C:=-(v.x*L.A+v.y*L.B);
End;
```

```
/* Hai đường thẳng có khác phương hay không */
Function CrossLine(Const L1,L2:TLine):Boolean;
Var st:Real;
Begin
 st:=L1.A*L2.B-L2.A*L1.B;
 CrossLine:=Not RealEq(st,0);
End;
/* Giao điểm hai đường thẳng */
Procedure Line2ToPoint(Const L1,L2:TLine;Var P:TPoint);
Var st:Real;
Begin
 st:=L1.A*L2.B-L2.A*L1.B;
 P.X:=(L1.C*L2.B-L2.C*Lq.B)/st;
 P.Y:=(L1.A*L2.C-L2.A*L1.C)/st;
End;
/* Tìm giao điểm hai đường thẳng */
Procedure FindPointCross(Const fL, fR, sL, sR:TPoint;Var rs:TPoint);
Var L1,L2:TLine;
Begin
 Point2ToLine(fL,fR,L1); Point2ToLine(sL,sR,L2);
 If CrossLine(L1,L2) Then Line2ToPoint(L1,L2,rs)
 Else rs:=ZeroPnt;
End;
/* Input: 2 đoạn thẳng nằm trên cùng đường thẳng
Nhiệm vụ: Kiểm tra vị trí tương đối
Output: 0: có 1 điểm chung; 2: có 1 đoạn chung; 1: không giao nhau
*/
Function SegmLineCross(Const fL,fR,sL,sR:Tpoint):Byte;
Var Minf,Maxf,Mins,Maxs:Real;
Begin
 Minf:=RealMin(Dist(fL,ZeroPnt),Dist(fR,ZeroPnt));
 Maxf:=RealMax(Dist(fL,ZeroPnt),Dist(fR,ZeroPnt));
 Mins:=RealMin(Dist(sL,ZeroPnt),Dist(sR,ZeroPnt));
 Maxs:=RealMax(Dist(sL,ZeroPnt),Dist(sR,ZeroPnt));
If RealEq(Minf,Maxs) or RealEq(Maxf,Mins)
 Then SegmLineCross:=0
 Else
 If RealMore(Mins,Maxf) Or RealMore(Minf,Maxs)
 Then SegmLineCross:=1
 Else SegmLineCross:=2;
End;
```

```
/* Kiểm tra đoạn thẳng cắt nhau.
Thuật toán: tìm giao điểm, sau đó kiểm tra giao điểm nằm trên đoạn thẳng
Trả về một số nguyên đánh số các trường hợp vị trí tương đối đoạn thẳng-đoạn thẳng.
Hai đoạn thẳng cắt nhau sẽ trả về 7
*/
Function SegmCross(Const fL,fR,sL,sR :TPoint);
Var rs:TPoint;
 L1,L2:TLine;
Begin
 Point2ToLine(fL,fR,L1);
 Point2ToLine(sL,sR,L2);
 If CrossLine(L1,L2)Then
 Begin
 Line2ToPoint(L1,L2,rs)
 If EqPoint(rs,fL) Or EqPoint(rs,fR)
     Or EqPoint(rs,sL) Or EqPoint(rs,sR)
 Then SegmCross:=5
 Else
 If AtSegm(fL,fR,rs) And AtSegm(sL,sR,rs) Then SegmCross:=7
 Else
 If AtSegm(fL,fR,rs) Or AtSegm(sL,sR,rs) Then SegmCross:=6
 Else SegmCross:=4;
 End
 Else
 If EqPoint(L1.A*L2.B,L2.A*L1.B)
     And Not (EqPoint(L1.C,L2.C)
 Then SegmCross:=3
 Else SegmCross:=SegmLineCross(fL,fR,sL,sR);
End;
/* Đường thẳng vuông góc */
Procedure PerLine(Const n:TLine;Const P:TPoint;Var L:TLine);
Begin
 L.A:=n.B;
L.B:=n.A;
L.C:=L.A*P.X-L.B*P.Y;
End;
/* Khoảng cách từ một điểm đến một đường thẳng */
Function DistPointToLine(Const P:TPoint;Const L:TLine):Real;
Begin
 DistPointToLine :=
        Abs((L.A*P.x+L.B*P.y+L.C))/Sqrt(Sqr(L.A)+Sqr(L.B));
End;
```

```
/* Kiểm tra điều kiện bất đẳng thức tam giác */
Function IsTrian(Const a,b,c:Real):Boolean;
Begin
 IsTrian:=RealMore(a+b,c) And RealMore(a+c,b)
                          And RealMore(b+c;a);
End;
/* Diện tích tam giác bằng công thức Hê-rông */
Function Sq(a,b,c:Réal):Real;
Var p:Real;
Begin
 p:=(a+b+c)/2; Sq:=Sqrt(p*(p-a)*(p-b)*(p-c));
End;
/* Diện tích tam giác khi biết toạ độ ba đỉnh */
Function SquareTrian(Const p1,p2,p3:TPoint);
Begin
 SquareTrian := Abs(p1.x*(p2.y-p3.y)-p1.y*(p2.x-p3.x)
                                  +(p2.x*p3.y-p3.x*p2.y))/2);
End;
/* Chiều cao tam giác ứng với cạnh a */
Function GetHeight(Const a,b,c:Real):Real;
Var p:Real;
Begin
 p:=(a+b+c)/2; GetHeight:=2*Sqrt(p*(p-a)*(p-b)*(p-c))/a;
End;
/* Tìm độ dài trung tuyến */
Function GetMed(Const a,b,c:Real):Real;
Begin
 GetMed:=Sqrt(2*(b*b+c*c)-a*a)/2;
End;
/* Độ dài đường phân giác xuất phát từ một đỉnh tới cạnh đối diện có độ dài bằng a */
Function GetBis(Const a,b,c:Real):Real;
Var p:Real;
Begin
 p:=(a+b+c)/2; GetBis:=2*Sqrt(b*c*p*(p-a))/(b+c);
End;
/* Bán kính đường tròn nội tiếp */
Function GetRadIns(Const a,b,c:Real):Real;
Var p:Real;
Begin
 p:=(a+b+c)/2; GetRadIns:=Sqrt((p-a)*(p-b)*(p-c)/p);
End;
```

```
/* Bán kính đường tròn ngoại tiếp */
Function GetRadExt(Const a,b,c:Real):Real;
Var p:Real;
Begin
 p:=(a+b+c)/2;
 GetRadExt:=a*b*c/(4*Sqrt(p*(p-a)*(p-b)*(p-c)));
End;
/* Kiểm tra đa giác không tự cắt */
Function IsPoligonSimple
            (Const A:Array Of TPoint;Const N:Word):Boolean;
Var i,j:Integer;
 pp:Boolean;
Begin
 pp:=True;
 i:=1;
 While (i<=N-1) And pp Do
   Begin
     j:=i+1;
     While (j<=N) And pp Do
       Begin
         Case SegmCross(A[i],A[i+1]],A[j],A[j+1]) Mod N
                  Of 0,2,6,7: pp:=False;
         End;
         Inc(j);
       End;
     Inc(i);
   End;
   IsPoligonSimple:=pp;
End;
```

c) Thư viện Math và các hằng số không xác định

Free Pascal cung cấp thư viện Math để hỗ trợ cho các phép tính toán học chuyên dụng. Ngoài một hệ thống phong phú các thủ tục và hàm, thư viện Math cung cấp ba hằng số đặc biệt *Nan*, *Infinite* và *NegInfinite*.

- *Nan* tương ứng với một số thực không xác định (Not a Number), trong thư viện Math, hằng *Nan* được khai báo bằng 0.0/0.0.
- *Infinite* tương ứng với hằng số thực +∞, phép chia một số thực dương cho 0 sẽ cho kết quả bằng *Infinite*.

- *NegInfinite* tương ứng với hằng số thực $-\infty$, phép chia một số thực âm cho 0 sẽ cho kết quả bằng *NegInfinite*.

Ví dụ đoạn chương trình sau:

```
program TestDiv0;
uses math;
var
 x, y, z: Float;
 zero: Float = 0;
begin
SetExceptionMask([Low(TFPUExceptionMask)..High(TFPUExceptionMask)]);
 x := 0/zero;
 y := 1/zero;
 z := -2/zero;
 WriteLn(x); //In ra Nan
 WriteLn(y); //In ra +Inf
 WriteLn(z); //In ra -Inf
end.
```

Ba hằng số này rất tiện dụng trong việc lập trình các bài toán hình học, chẳng hạn khi tìm giao điểm của hai đường thẳng, nếu hai đường thẳng đó trùng nhau, ta có thể gán toạ độ giao điểm là (*Nan*; *Nan*), còn nếu hai đường thẳng song song với nhau, ta có thể gán toạ độ giao điểm là (*Infinite*; *Infinite*). Chương trình chính chỉ việc gọi hàm để lấy toạ độ giao điểm, sau đó kiểm tra các toạ độ có khác *Nan* và khác *Infinite* hay không để xử lí tiếp.

Những chú ý sau rất quan trọng khi sử dụng các hằng *Nan*, *Inifinite* và *NegInfinite*:

Chú ý:

- Việc chia cho 0 để lấy hằng *Nan*, *Infinite*, *NegInfinite*, cũng như việc so sánh trực tiếp với các hằng số này, chỉ thực hiện được trong lúc chạy (run-time) nếu ta đặt lại thiết lập trong bộ đồng xử lí toán học bằng lệnh:

  ```
  SetExceptionMask([Low(TFPUExceptionMask)..High(TFPUExceptionMask)]);
  ```

- Việc chia trực tiếp cho 0, gán, so sánh các hằng *Nan*, *Infinite*, *NegInfinite* với một biến số thực chỉ được phép viết khi dịch chương trình ở chế độ {$*R*-,*Q*-}, tức là tất cả các chế độ kiểm tra tràn phạm vi và tràn số học phải được tắt. Tuy nhiên, việc tắt hoàn toàn chế độ kiểm tra

tràn phạm vi và tràn số học có thể rất nguy hiểm và khó gỡ rối, vì vậy ta nên làm theo cách khác như sau:

- Sử một biến *zero* gán bằng 0 và thực hiện phép chia cho *zero* thay vì chia cho 0, kĩ thuật này nhằm đánh lừa trình biên dịch.
- Không so sánh trực tiếp một biểu thức số thực với các hằng $Nan, Infinite, NegInfinite$ mà sử dụng các hàm cung cấp sẵn trong thư viện Math:
  - Hàm $IsNan(x): Boolean$: Kiểm tra số thực x có phải bằng Nan hay không.
  - Hàm $IsInifinite(x): Boolean$: Kiểm tra số thực x có phải bằng $Infinite$ hay $NegInifinite$ hay không.
  - Hàm $IsZero(x, \varepsilon): Boolean$: Kiểm tra $x \approx 0$, cụ thể là $|x|$ có nhỏ hơn ε hay không, hàm này có thể dùng để xử lí sai số. ε là sai số chấp nhận do người dùng tuỳ chọn.

d) Thư viện Matrix

Trong các bài toán ví dụ, chúng tôi chỉ sử dụng thư viện Math để hỗ trợ cho việc tính toán được ngắn gọn. Nếu muốn thực hiện các tính toán hình học phức tạp, trên không gian nhiều chiều, bạn có thể sử dụng thêm thư viện Matrix. Thư viện này cung cấp các phép toán thao tác trực tiếp trên các vectơ và ma trận: tính tích ma trận, nghịch đảo ma trận, định thức của ma trận v.v...

Hiện tại, các thuật toán trong thư viện Matrix vẫn là những thuật toán tuần tự. Tuy nhiên trong tương lai, sẽ có những phiên bản với thuật toán song song, sử dụng kĩ thuật đa xử lí để tăng tốc quá trình tính toán. Một số dự án đang viết lại thư viện Matrix, dùng sức mạnh của hàng nghìn GPU trong card đồ hoạ để tính toán thay vì sử dụng CPU, các phép toán ma trận được tăng tốc lên hàng trăm lần và các chương trình sử dụng thư viện này theo đó cũng được tăng lên đáng kể.

Bài tập

8.1. Trên mặt phẳng cho một tam giác và một đoạn thẳng. Tính độ dài của đoạn thẳng nằm trong tam giác.

8.2. Trên mặt phẳng cho hai tam giác. Tính diện tích phần chung của hai tam giác.

8.3. Cho hai hình chữ nhật với các cạnh song song với hệ trục toạ độ. Mỗi hình chữ nhật xác định bởi toạ độ hai đỉnh đối. Tìm diện tích phần mặt phẳng được phủ bởi hai hình chữ nhật và diện tích phần chung của chúng.

8.4. Trên mặt phẳng cho N điểm. Tìm hai điểm xa nhau nhất trong N điểm đó.

Gợi ý: Hai điểm cần tìm là hai đỉnh của đa giác bao lồi.

8.5. Trên mặt phẳng cho N điểm. Hãy phân hoạch N điểm thành K tập $S_1, S_2, \ldots, S_K$ sao cho đường kính lớn nhất của K tập là nhỏ nhất, trong đó đường kính của tập S_i là khoảng cách lớn nhất giữa các cặp đỉnh thuộc S_i.

8.6. Trên mặt phẳng cho N điểm đôi một khác nhau. Hãy tìm hai trong N điểm sao cho đường thẳng đi qua hai điểm đó chia mặt phẳng thành hai phần mà số điểm thuộc hai nửa mặt phẳng lệch nhau ít nhất.

8.7. Trên mặt phẳng cho N điểm. Hãy tìm tập ít nhất các đường thẳng sao cho mỗi điểm trong N điểm đã cho thuộc ít nhất một đường thẳng.

8.8. Trên mặt phẳng cho N điểm. Hãy nối N điểm thành một mạng liên thông bằng các đoạn thẳng nối các cặp điểm sao cho tổng độ dài các đoạn thẳng được nối là nhỏ nhất.

8.9. Trên mặt phẳng cho N hình chữ nhật có các cạnh song song với hai trục toạ độ. Hình thứ i xác định bởi hai đỉnh đối $(x_i; y_i)$ và $(z_i; t_i)$. Toạ độ các điểm là nguyên; có giá trị tuyệt đối không vượt quá 10^4. Khi tô các hình chữ nhật này bằng một màu xanh, ta nhận được một số phần mặt phẳng được tô màu xanh. Tính tổng độ dài các đường bao quanh các vùng màu xanh.

8.10. *Tầm nhìn*

Trên mặt phẳng cho N đoạn thẳng ($1 \le N \le 1000$). Toạ độ các đầu mút của các đoạn thẳng là các số nguyên không âm không vượt quá 20000. Các đường thẳng thu được bằng cách kéo dài các đoạn thẳng đã cho luôn cắt hai trục toạ độ và hai giao điểm cùng gốc toạ độ tạo thành một tam giác vuông cân. Không có hai đoạn thẳng nào giao nhau.

Ta nói một đoạn thẳng là nhìn thấy được từ gốc toạ độ O, nếu tìm được điểm X trên nó sao cho đoạn thẳng OX không cắt bất cứ đoạn nào khác trong số các đoạn thẳng đã cho.

Hãy viết chương trình đếm số đoạn thẳng nhìn thấy được từ gốc toạ độ.

Input: Vào từ tệp VPOINT.INP:

- Dòng đầu tiên chứa số đoạn thẳng N;
- Mỗi một trong số N dòng tiếp theo chứa bốn số nguyên không âm X_1, Y_1, X_2 và Y_2, phân cách nhau bởi dấu cách, trong đó (X_1, Y_1) là toạ độ của đầu mút thứ nhất còn (X_2, Y_2) là toạ độ của đầu mút thứ hai của đoạn thẳng tương ứng.

Output: Ghi ra trên một dòng của tệp VPOINT.OUT số đoạn thẳng nhìn thấy được từ gốc toạ độ.

Ví dụ:

VPOINT.INP	VPOINT.OUT
4 3 13 11 5 14 1 10 5 10 14 20 4 5 6 10 1	3

8.11. *Khôi phục đa giác*

Bờm vẽ trên mặt phẳng một hình đa giác tổng quát (đường gấp khúc khép kín không tự cắt) với các cạnh song song với các trục toạ độ và các đỉnh có toạ độ nguyên. Sau đó, vì vô ý Bờm đã xoá mất tất cả các cạnh thẳng đứng (cạnh song song với trục tung) của đa giác. Bạn hãy tìm cách từ những thông tin còn lại trên hình vẽ giúp Bờm tính diện tích của đa giác đã vẽ ban đầu.

Input: Vào từ tệp văn bản POLYGON.INP:

- Dòng đầu tiên chứa N là số cạnh nằm ngang (cạnh song song với trục hoành) của đa giác đã cho ($N \leq 1000$) ;
- Mỗi dòng trong số N dòng tiếp theo chứa thông tin về một cạnh nằm ngang của đa giác bao gồm bốn số nguyên x, y, u, v được ghi cách nhau bởi dấu cách, trong đó (x, y) và (u, v) là hai cặp toạ độ của hai đầu mút

của cạnh nằm ngang. Giả thiết rằng các toạ độ là các số nguyên có giá trị tuyệt đối không vượt quá 100.

Output: Ghi ra tệp văn bản POLYGON.OUT:

- Dòng đầu tiên ghi diện tích của đa giác.
- Dòng thứ i trong số $2*N$ dòng tiếp theo chứa toạ độ đỉnh thứ i của đa giác được liệt kê theo thứ tự đi vòng quanh đa giác theo chiều kim đồng hồ (đỉnh bắt đầu được chọn tuỳ ý).

Ví dụ:

POLYGON.INP	POLYGON.OUT
2 1 1 3 1 1 3 3 3	4 1 1 1 3 3 3 3 1

8.12. *Đỉnh núi*

Đỉnh núi có thể nhìn thấy được ở chân trời chỉ khi nó không bị che khuất bởi một ngọn núi khác hay là đường viền của nó không bị ẩn trên nền một ngọn núi khác. Giả thiết rằng tất cả các dốc núi đều có góc nghiêng là 45^o đồng thời cho biết toạ độ và độ cao đỉnh của tất cả các ngọn núi. Cần xác định số lượng đỉnh núi nhìn thấy được.

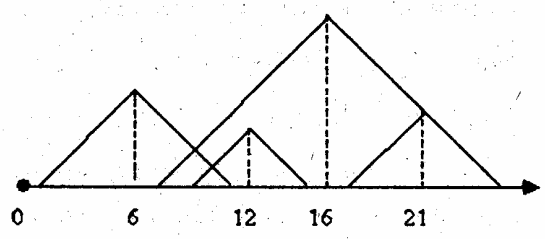
Input: Vào từ tệp văn bản MOUNT.INP:

- Dòng đầu tiên chứa số nguyên N ($1 \le N \le 10000$).
- Mỗi dòng trong N dòng tiếp theo chứa hai số nguyên được ghi cách nhau bởi dấu cách: đó là toạ độ x của đỉnh núi ($0 \le x \le 30000$) và độ cao của đỉnh núi y ($1 \le y \le 10000$).

Output: Ghi ra tệp văn bản MOUNT.OUT số lượng đỉnh núi nhìn thấy được.

Ví dụ:

MOUNT.INP	MOUNT.OUT
4 6 5 12 3 16 9 21 4	2



Hình 8.11

8.13. *Đài phun nước*

Để làm đẹp cảnh quan, Ban giám đốc một Công ti quyết định xây dựng ở sân tiền sảnh trụ sở công ti một đài phun nước. Đài phun nước phải có dạng một hình tròn với kích thước lớn nhất có thể được. Nhà thiết kế được biết là sân tiền sảnh của công ti có dạng một hình chữ nhật kích thước $X \times Y$. Tuy nhiên, khi lựa chọn vị trí cho đài phun nước nhà thiết kế gặp phải một vấn đề phức tạp: Trong sân tiền sảnh có N cột hình trụ tròn xoay không được phép di chuyển. Vì vậy vấn đề đặt ra cho nhà thiết kế là: Cần đặt đài phun nước ở vị trí nào để nó có bán kính lớn nhất có thể được đồng thời không được có diện tích chung khác 0 với các cột. Bạn hãy lập trình giúp nhà thiết kế giải quyết vấn đề trên.

Input: Vào từ tệp văn bản FONTAN.INP:

- Dòng đầu tiên chứa hai số thực X, Y, $1 \le X, Y \le 10^4$. Giả thiết rằng sân tiền sảnh là hình chữ nhật trên mặt phẳng toạ độ có toạ độ các đỉnh là $(0, 0)$, $(X, 0)$, (X, Y) và $(0, Y)$;
- Dòng thứ hai chứa số nguyên N $(0 \le N \le 10)$ là số lượng cột trong sân tiền sảnh;
- Dòng thứ i trong số N dòng tiếp theo chứa ba số thực X_i, Y_i và R_i cho biết toạ độ của tâm và bán kính của cột thứ i ($R_i \le X_i \le X - R_i$, $R_i \le Y_i \le Y - R_i$, $0.1 \le R_i \le \min\left(\frac{X}{2}, \frac{Y}{2}\right)$, $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \ge R_i + R_j$ với mọi $i \ne j$).

  Các số trên một dòng trong tệp dữ liệu được ghi cách nhau bởi dấu cách.

Output: Ghi ra tệp văn bản FONTAN.OUT ba số thực X_F, Y_F, R_F là toạ độ tâm và bán kính của đài phun nước.

Chú ý: Đài phun nước phải được đặt trong sân, được phép tiếp xúc với tường của sân hoặc cột, nhưng không được có diện tích chung khác không với các cột. Nếu có nhiều vị trí cùng cho bán kính lớn nhất chỉ cần đưa ra một trong số chúng.

Ví dụ:

FONTAN.INP	FONTAN.OUT
10 20 0	5.000 5.000 5.000

FONTAN.INP	FONTAN.OUT
20 20 4 2 2 2 18 2 2 2 18 2 18 18 2	10.000 10.000 9.314

FONTAN.INP	FONTAN.OUT
20 20 4 2 2 2 18 2 2 3 17 2 16 16 4	9.510 7.054 7.053

HƯỚNG DẪN GIẢI BÀI TẬP

Chuyên đề 8

8.1. Tìm tập các điểm thoả mãn một trong hai tính chất sau:

a) Là giao điểm của đoạn thẳng và tam giác.

b) Là đầu mút của đoạn thẳng nằm trong tam giác.

Nếu tập trên gồm hai điểm, kết quả là khoảng cách giữa hai điểm đó, ngược lại kết quả bằng 0.

8.2. Tìm tập các điểm thoả mãn một trong hai tính chất sau:

a) Là giao điểm của hai đoạn thẳng, mỗi đoạn thuộc một tam giác.

b) Là đỉnh của một tam giác nằm trong tam giác còn lại.

Tính diện tích bao lồi của tập điểm trên.

8.3. Các đường thẳng chứa các cạnh của các hình chữ nhật chia mặt phẳng toạ độ thành một lưới hình chữ nhật. Ta xét lần lượt các khe ngang của lưới, với mỗi khe, xét các cạnh dọc của các hình chữ nhật theo thứ tự toạ độ x tăng dần. Chỉ quan tâm đến các cạnh phủ hết khe đang xét. Ta có biến đếm s. Nếu gặp cạnh trái của một hình chữ nhật thì tăng s lên 1. Nếu gặp cạnh phải thì giảm s đi 1. Khi s tăng từ 0 lên 1, đánh dấu lại toạ độ x của cạnh đang xét, gọi là x_1. Khi s giảm về 0, đánh dấu toạ độ x của cạnh đang xét, gọi là x_2. Lấy $x_2 - x_1$ nhân với độ rộng khe ngang đang xét rồi cộng vào kết quả.

Độ phức tạp là N^2, với N là số hình chữ nhật.

Để đạt được độ phức tạp $N\log N$ có thể sử dụng cấu trúc Segment tree.

Đề bài và bộ test có tại website http://www.spoj.pl/problems/NKMARS.

8.4. Tìm bao lồi của tập điểm. Hai điểm cần tìm là hai đỉnh xa nhau nhất của bao lồi.

8.5. Giả sử kết quả đang là s. Phân tập điểm thành các tập con sao cho khoảng cách giữa hai điểm bất kì thuộc cùng một tập không vượt quá s. Nếu phân được nhiều hơn K tập thì tăng s. Nếu phân được ít hơn hoặc bằng K tập thì giảm s.

8.6. Xét từng cặp điểm, mỗi cặp điểm viết phương trình đường thẳng đi qua hai điểm. Sau đó đếm số điểm nằm ở hai nửa mặt phẳng chia bởi đường thẳng.

8.7. Áp dụng luồng min-cost.

8.8. Xây dựng đồ thị đầy đủ với các đỉnh là các điểm trong tập, trọng số của một cạnh là khoảng cách giữa hai điểm ứng với hai đầu mút của cạnh. Tìm cây khung của đồ thị.

8.9. Tương tự như bài 8.3 tính diện tích các hình chữ nhật, bài này là tính chu vi.

8.10. Xét tập các tia gồm Ox, Oy, các tia gốc O đi qua một trong các đầu mút của các đoạn thẳng. Sắp xếp các tia trong tập theo góc tạo bởi tia và trục hoành (theo tan). Xét các góc tạo bởi hai tia liên tiếp. Với mỗi góc, xét các đoạn thẳng phủ hết góc và đánh dấu đoạn gần gốc O nhất là nhìn thấy được (lấy khoảng cách gần nhất trong số hai khoảng cách từ O đến hai giao điểm của hai tia và đoạn thẳng). Kết quả là số các đoạn thẳng được đánh dấu.

8.11. Mỗi điểm chỉ có thể nối được với một trong hai điểm sau:

a) Cùng toạ độ x, toạ độ y lớn hơn và gần nhất.

b) Cùng toạ độ x, toạ độ y nhỏ hơn và gần nhất.

Với mỗi điểm, ta đếm xem có bao nhiêu điểm cùng toạ độ x mà toạ độ y lớn hơn toạ độ y của nó. Nếu số lượng đó là lẻ, ta nối điểm đang xét với điểm loại a, nếu không, ta nối với điểm loại b.

8.12. Biểu diễn mỗi đỉnh núi bằng một đoạn thẳng nằm trên trục hoành thể hiện chân núi (cạnh huyền của tam giác vuông). Sắp xếp các đoạn thẳng theo thứ tự toạ độ x của đầu mút bên trái tăng dần. Xét từng đoạn thẳng, nếu đầu mút bên trái của nó không vượt quá đầu mút bên phải xa nhất của các đoạn thẳng đã xét thì đánh dấu ngọn núi ứng với đoạn thẳng đó nhìn thấy được.

8.13. Với mỗi kết quả s, cần kiểm tra xem có tồn tại một hình tròn trong sân sao cho nó không giao với bất cứ hình tròn bán kính R_i nào. Điều này tương đương với việc kiểm tra xem có tồn tại một điểm trong sân sao cho nó không nằm trong bất cứ hình tròn bán kính R_{i+s} nào.

Với hình chữ nhật lớn ban đầu, kiểm tra xem trọng tâm của nó có phải điểm cần tìm không. Nếu không, chia hình chữ nhật thành bốn hình chữ nhật con bằng nhau rồi tiếp tục kiểm tra từng hình bằng cách tương tự.