Зміст

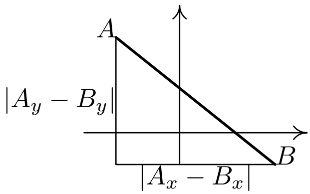
1 Теоретичний матеріал

1.1 Точки та вектори

Точки площини зазвичай задають у декартовій прямокутній системі у вигляді пар координат $(x; y)$. У програмах точки зазвичай подають таким типом записів:

```
type TPoint = record
  x, y: extended
end;
```

Наприклад, може бути змінна `A` типу `TPoint`; тоді до окремих її координат можна звернутися через `A.x` та `A.y`. Відповідно, з метою полегшення переходу між математичним та програмним записами, будемо і в суто математичних формулах використовувати позначення, наприклад, $(A_x; A_y)$ (замість більш вживаного на уроках математики $(x_A; y_A)$).



Ві́дстань між точками A і B з координатами $(A_x; A_y)$ та $(B_x; B_y)$ (позначається $d(A, B)$, або, що те саме, $|AB|$) можна знайти за формулою:

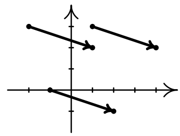
$$d(A, B) = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}. \quad (1)$$

Справді, за теоремою Піфагора: $(d(A, B))^2 = \underbrace{|A_x - B_x|^2}_{=(A_x - B_x)^2} + \underbrace{|A_y - B_y|^2}_{=(A_y - B_y)^2}$.

Вектор $\vec{a}$ можна означити як величину, яка характеризується числовим значенням і напрямком.

У деяких задачах, особливо фізичних, буває важливо, де починається (до якої точки прикладений) вектор. Ми ж будемо розглядати переважно *вільні* вектори, які дозволено переносити в інше місце, важливо лише не змінювати довжину та напрям.

Наприклад, на рис. тричі показано один і той самий (вільний) вектор.



Є два стандартні, альтернативні один одному, способи задати вектор:

- Через довжину (позначатимемо або $|a|$, або l_a, це одне й те ж) і кут нахилу α

  (Кут нахилу вектора рахують згідно означення тригонометричного кута, тобто у радіанах, від осі Ox, проти годинникової стрілки. Кути, що відрізняються один від одного на $\pm 2\pi$, $\pm 4\pi$ і т. д., однакові.)

- Через координати, вони ж проекції $(a_x; a_y)$.

Наприклад, вектор, тричі зображений на рис. вище, можна задати або як «координати (проекції) $a_x{=}3$, $a_y{=}{-1}$», або «довжина $\sqrt{10}$, напрям $\operatorname{arctg} \frac{-1}{3} {\approx} {-0{,}32}$ (радіан)».

Подання вектора парою координат $(a_x; a_y)$ ***фактично однакове з поданням точки; отже, для подання (вільного) вектора можна використати той самий тип*** `TPoint`***, що для точок.***

При потребі, можна переходити від подання довжиною $|a|$ і напрямом α до подання координатами (a_x, a_y), за формулами

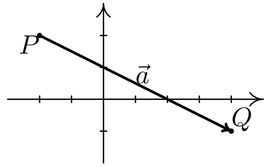
$$a_x = |a| \cdot \cos \alpha; \quad a_y = |a| \cdot \sin \alpha. \tag{2}$$

У зворотньому напряму —

$$|a| = \sqrt{a_x^2 + a_y^2}; \quad \alpha = \mathtt{arctan2}(a_y, a_x). \tag{3}$$

(`Arctan2` — не математичне позначення, а функція мови програмування. Параметри цієї функції — координати вектора (спочатку y, потім x), результат — кут його нахилу ($-\pi{<}\alpha{\leqslant}\pi$). У FreePascal/Delphi ця функція називається `arctan2` і потребує підключення `uses Math`, у C/C++ називається `atan2` і потребує підключення `#include <math.h>`. Величезними перевагами цієї функції над очевидним виразом `arctan(y/x)` є: (1) вміє розрізнити протилежні кути, наприклад, $\texttt{arctan2}(3,3)=\frac{\pi}{4}$, $\texttt{arctan2}(-5,-5)=-\frac{3\pi}{4}$, хоча $\frac{3}{3}=\frac{-5}{-5}=1$; (2) працює при $x{=}0$.)

Для будь-яких точок P та Q з координатами $(P_x;P_y)$ і $(Q_x;Q_y)$ можна побудувати вектор $\overrightarrow{PQ}$ з початком P і кінцем Q. Координати вектора $\overrightarrow{PQ}$ дорівнюють $(Q_x{-}P_x;Q_y{-}P_y)$.



$P_x{=}{-}2,\quad P_y{=}2;$
$Q_x{=}4,\quad Q_y{=}{-}1;$
$a_x{=}6,\quad a_y{=}{-}3.$

Аналогічно, якщо розмістити початок вектора $\vec{a}$ у точку P, то кінець вектора потрапить у точку з координатами $(P_x{+}a_x;P_y{+}a_y)$, де $(a_x;a_y)$ — координати вектора $\vec{a}$, $(P_x;P_y)$ — координати точки P.

Якщо сумістити (перенести в одну й ту ж точку площини) кінець вектора $\vec{a}$ і початок вектора $\vec{b}$, то *сумою векторів* $\vec{a}$ та $\vec{b}$ (позначається $\vec{a}{+}\vec{b}$) буде вектор, який починається у початку $\vec{a}$ і закінчується у кінці $\vec{b}$. Координати вектора-суми рівні $(a_x{+}b_x;a_y{+}b_y)$.

Добуток числа k на вектор $\vec{a}$ (записується $k\cdot\vec{a}$) — це вектор з координатами $(k\cdot a_x;k\cdot a_y)$. Якщо говорити про вектор як про напрямлений відрізок, то цей добуток має довжину $|k|\cdot l_a$, і при $k>0$ вектор $k\cdot\vec{a}$ співнапрямлений з $\vec{a}$, а при $k<0$ — протинапрямлений.

1.1.1 Програмна реалізація матеріалів даного підрозділу

Вищенаведені математичні поняття можна реалізувати підпрограмами:

```
(*Для утворення вектора по кінцю A й початку B.
Крім того, являє собою різницю векторів, де A-B = A + (-1)*B  *)
function minus (const A, B : TPoint) : TPoint;
Begin
  Result.x := A.x - B.x;
  Result.y := A.y - B.y;
End;
```

```
(*Для утворення точки кінця вектора
по точці початку A й вектору B.
Крім того, являє собою суму векторів A+B*)
function plus (const A, B : TPoint) : TPoint;
Begin
  Result.x := A.x + B.x;
  Result.y := A.y + B.y;
End;
```

```
(*Добуток числа на вектор*)
function multiply (k : extended ; const A : TPoint) : TPoint;
Begin
  Result.x := k*A.x;
  Result.y := k*A.y;
End;
```

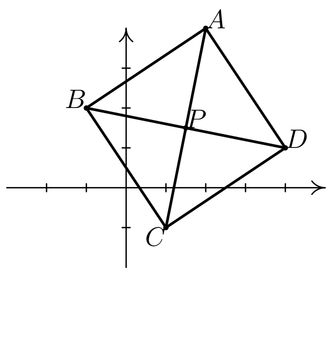
Що всі ці підпрограми дають? Головним чином — можливість оперувати у програмі з точками і векторами як єдиними сутностями. Як наслідок — зручно знаходити потрібні точки/вектори.

1.1.2 Задача «Квадрат по двом вершинам»

Квадрат (довільно орієнтований) заданий координатами двох своїх протилежних вершин A, C. Потрібно знайти координати решти двох вершин B, D.

Зобразимо (на папері) шуканий квадрат, і розглянемо, зокрема, точку перетину його діагоналей (позначимо її P).

Ми *можемо* знайти її: оскільки P належить відрізку AC і відрізки AP та CP мають однакову довжину, то $\overrightarrow{AP} = \frac{1}{2}\overrightarrow{AC}$.



Тобто, достатньо записати у програмі
`AC:=minus(C,A);` {знайшли $\overrightarrow{AC}$}
`AP:=multiply(0.5,AC);` {знайшли $\overrightarrow{AP}$}
`P:=plus(A,AP);` {знайшли точку P}

Раз треба побудувати квадрат — значить, діагоналі одночасно і рівні, і перетинаються під прямим кутом. Тому зараз можна застосувати (досі не розглянуте) вираження повороту вектора на прямий кут. Неважко переконатися, що результатом повороту вектора $(v_x; v_y)$ на $+\frac{\pi}{2}$ (де "+" означає «проти годинникової стрілки») буде вектор $(-v_y; v_x)$: при повороті вісі Oy виходить напрям, протилежний вісі Ox, а при повороті вісі Ox — сама вісь Oy. А ця операція повороту дає можливість отримати, наприклад, вектор $\overrightarrow{PD}$ з вектора $\overrightarrow{PC}$ (він відомий, бо рівний $\overrightarrow{AP}$). А коли є P і є $\overrightarrow{PD}$, то вже геть не важко побудувати D та B.

```
{$mode delphi}

type TPoint = record
  x, y: extended
end;

function minus (const A, B : TPoint) : TPoint;
Begin
  Result.x := A.x - B.x;
  Result.y := A.y - B.y;
End;

function plus (const A, B : TPoint) : TPoint;
Begin
  Result.x := A.x + B.x;
  Result.y := A.y + B.y;
End;

function multiply (k : extended ; const A : TPoint) : TPoint;
Begin
  Result.x := k*A.x;
  Result.y := k*A.y;
End;

function rotate_plus_pi_2 (const A : TPoint) : TPoint;
Begin
  Result.x := -A.y;
  Result.y :=  A.x;
End;

procedure ReadPoint (var A : TPoint);
```

```
Begin
  read(A.x, A.y);
End;

procedure WritelnPoint (const A : TPoint);
Begin
  writeln(A.x, ' ', A.y);
End;

var A,B,C,D,P,AC,AP,PD : TPoint;

BEGIN
  ReadPoint(A);
  ReadPoint(C);
  AC:=minus(C,A);
  AP:=multiply(0.5,AC);
  P:=plus(A,AP);
  PD:=rotate_plus_pi_2(AP);
  D:=plus(P,PD);
  B:=minus(P,PD);
  WritelnPoint(B);
  WritelnPoint(D);
END.
```

1.1.3 Альтернативний розв'язок задачі «Квадрат по двом вершинам» і міркування щодо можливих підходів до алгоритмічного розв'язування геометричних задач

Наведені раніше співвідношення можна не кодувати мовою програмування, а поставитися до них як до системи рівнянь, розв'язати олівцем на папері, отримати прямі аналітичні формули вираження шуканих координат через відомі, й написати у програмі вже їх. Наприклад:

```
var x1,y1,x2,y2:real;
begin
  readln(x1, y1);
  readln(x2, y2);
  writeln((x1+x2)/2+(y2-y1)/2, ' ', (y1+y2)/2+(x1-x2)/2);
  writeln((x1+x2)/2+(y1-y2)/2, ' ', (y1+y2)/2+(x2-x1)/2);
end.
```

Це — *повний* текст (правильної) програми; тут, на відміну від попереднього способу, не треба ніяких підпрограм. Тому може скластися враження, ніби такий розв'язок всебічно кращий за описаний у попередньому підрозділі.

Що ж, для даної конкретної задачі такий розв'язок має повне право на існування, і, *якби* це була єдина геометрична задача — можна було *б* стверджувати, що швидше розв'язати рівняння на папері, вивести аналітичними перетвореннями пряму формулу й програмно реалізувати її, ніж писати у програму всі геометричні функції.

Але коли є також інші геометричні задачі, й підпрограми для роботи з геометрією все одно будуть потрібні ще й у інших місцях, ситуація міняється: вже написані підпрограми принесуть користь також і в інших задачах/підзадачах, а от затрати часу на розв'язування рівнянь, аналітичні перетворення й виведення нестандартних прямих формул рідко коли приносять користь зразу для багатьох різних задач/підзадач.

Інший вагомий аргумент на користь першого, а не другого з розглянутих способів полягає у тому, що довга програмна реалізація з попереднього пункту розщеплюється на *дуже прості* кроки, кожен з яких легко уявити і твердо переконатися у його правильності. Крім того, технічні помилки програми, яка крок за кроком виконує проміжні побудови (знаходить допоміжні вектори/точки) можна шукати засобами налагодження програм (debug-а). А якщо якась помилка трапилася десь невідомо де в аналітичному розв'язуванні системи рівнянь, маємо і складніші міркування/перетворення, і, як правило, повну відсутність технічних засобів, що могли б допомогти розібратися.

1.2 Скалярний добуток

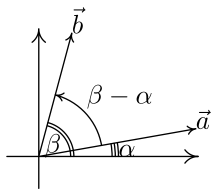
Скалярний добуток векторів $\vec{a}$ та $\vec{b}$ (позначається $\vec{a}\cdot\vec{b}$) можна визначити одним з двох способів:

- $|a| \cdot |b| \cdot \cos(\beta - \alpha)$, де $|a|$ та $|b|$ — довжини, α та β — кути нахилу;
- $a_x b_x + a_y b_y$.

Якщо вважати відомими формули (2) вираження координат вектора через кут та нахил, а також формулу вираження $\cos(\beta-\alpha)$, то доведення рівності двох наведених виразів зводиться до

$$\begin{gathered} a_x b_x + a_y b_y = \\ = (|a|\cos\alpha)(|b|\cos\beta) + (|a|\sin\alpha)(|b|\sin\beta) = \\ = |a|\cdot|b|\cdot\big(\cos\alpha\cos\beta+\sin\alpha\sin\beta\big) = \\ = |a|\cdot|b|\cdot\big(\cos\beta\cos\alpha+\sin\beta\sin\alpha\big) = \\ = |a|\cdot|b|\cdot\cos(\beta-\alpha). \end{gathered}$$

Втім, розуміння наведених перетворень корисне, але не є абсолютно обов'язковим.



Обов'язкові: (1) розуміння самого факту рівності $a_x b_x + a_y b_y = |a|\cdot|b|\cdot\cos(\beta-\alpha)$; (2) розуміння, що $\beta-\alpha$ являє собою *кут між векторами* $\vec{a}$ та $\vec{b}$.

З цього (і факту, що косинус додатний для гострих кутів і від'ємний для тупих) слідує, що скалярний добуток…

- додатний тоді й тільки тоді, коли кут між векторами гострий;
- дорівнює нулю тоді й тільки тоді, коли вектори перпендикулярні;
- від'ємний тоді й тільки тоді, коли кут між векторами тупий.

Надалі, будемо вважати готовою функцію

```
(*Скалярний добуток*)
function scal_prod (const A, B : TPoint) : extended;
Begin
  Result := A.x*B.x + A.y*B.y;
End;
```

1.3 Орієнтована площа (псевдоскалярний добуток, векторний добуток для площини)

Дуже важливу роль у обчислювальній геометрії на площині відіграє вираз

$$a_x b_y - a_y b_x. \tag{4}$$

(який зазвичай не згадують у шкільному курсі алгебри/геометрії). На жаль, для нього нема єдиної загальноприйнятої назви; *усі* написані у підзаголовку варіанти «орієнтована площа», «псевдоскалярний добуток» та «векторний добуток для площини» відповідають саме йому. Будемо позначати дану величину як $\vec{a} \times \vec{b}$ (зверніть увагу: для чисел «$6 \cdot 7$» та «6×7» позначають абсолютно один і той самий добуток, а для векторів «$\vec{a} \cdot \vec{b}$» та «$\vec{a} \times \vec{b}$» як правило, дають *різні* результати). Вважатимемо надалі, що у програмі є функція

```
(*Псевдоскалярний добуток, він же орієнтована площа,
він же <<векторний добуток для площини>>  *)
function cross_prod (const A, B : TPoint) : extended;
Begin
  Result := A.x*B.y - A.y*B.x;
End;
```

Провівши перетворення

$$\begin{gathered} a_x b_y - a_y b_x = \\ = (|a| \cos\alpha)(|b| \sin\beta) - (|a| \sin\alpha)(|b| \cos\beta) = \\ = |a| \cdot |b| \cdot \big(\cos\alpha \sin\beta - \sin\alpha \cos\beta\big) = \\ = |a| \cdot |b| \cdot \big(\sin\beta \cos\alpha - \sin\alpha \cos\beta\big) = \\ = |a| \cdot |b| \cdot \sin(\beta - \alpha) \end{gathered}$$

(аналогічні перетворенням для скалярного добутку), отримаємо добуток довжин векторів на *синус* кута між ними.

З цього, та факту, що синус додатний для кутів $0<\varphi<\pi$ і від'ємний для кутів $(-\pi)<\varphi<0$, слідує, що значення виразу $a_x b_y - a_y b_x$ ...

- додатнє тоді й тільки тоді, коли напрям найкоротшого повороту від $\vec{a}$ до $\vec{b}$ додатний (проти годинникової стрілки);
- дорівнює нулю тоді й тільки тоді, коли вектори колінеарні (співнапрямлені або протинапрямлені);
- від'ємне тоді й тільки тоді, коли напрям найкоротшого повороту від $\vec{a}$ до $\vec{b}$ від'ємний (за годинниковою стрілкою).

Крім можливості класифікувати повороти за знаком, дана величина має зв'язок ще й з площею: добуток довжин сторін на синус кута між ними — це площа параллелограма, або ж подвоєна площа трикутника. Щоправда, площа не буває від'ємною, а досліджувана величина $a_x b_y - a_y b_x$ буває. Тож, щоб отримати площу $\triangle ABC$, в якому вершини A, B, C задані координатами, достатньо виконати такі кроки:

1. знайти вектор $\overrightarrow{AB}$ як $B{-}A$;
2. знайти вектор $\overrightarrow{AC}$ як $C{-}A$;
3. обчислити власне площу як

$$S_{\triangle ABC} = \left| \frac{\overrightarrow{AB} \times \overrightarrow{AC}}{2} \right|. \tag{5}$$

Цей спосіб знаходження площі дійсно працює, і (якщо відомі координати вершин трикутника, а не самі лише довжини сторін) має значні переваги над більш відомою формулою Герона $\sqrt{p(p{-}a)(p{-}b)(p{-}c)}$ — формула (5) і швидше обчислюється, і з нею значно рідше виникає проблема накопичення похибки (неточності обчислень).

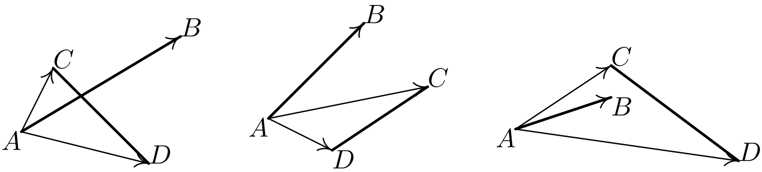
Ще один цікавий факт, пов'язаний з (5), розглянемо у розд. 1.3.3.

1.3.1 Чи перетинаються відрізки—1

Задано (координатами) чотири точки A, B, C, D, *щодо яких гарантовано, що ніякі три не лежать на одній прямій.* Чи перетинаються відрізки AB і CD?

Розглянемо вектори $\overrightarrow{AB}$, $\overrightarrow{AC}$ та $\overrightarrow{AD}$. Оскільки ніякі три точки не лежать на одній прямій, то $\overrightarrow{AB}$ не колінеарний ні $\overrightarrow{AC}$, ні $\overrightarrow{AD}$, і тому $\overrightarrow{AB} \times \overrightarrow{AC} \neq 0$, $\overrightarrow{AB} \times \overrightarrow{AD} \neq 0$. З рис. (a) та (b) видно, що коли точки C та D у різних півплощинах від прямої AB, то $\overrightarrow{AB} \times \overrightarrow{AC}$ та $\overrightarrow{AB} \times \overrightarrow{AD}$ різних знаків, а коли у одній — однакових.

При цьому не можна стверджувати, наприклад, $\overrightarrow{AB} \times \overrightarrow{AC} > 0$ `and` $\overrightarrow{AB} \times \overrightarrow{AD} < 0$, бо це так лише для конкретного випадку з малюнка; а от стверджувати «$\overrightarrow{AB} \times \overrightarrow{AC}$ та $\overrightarrow{AB} \times \overrightarrow{AD}$ різних знаків» — можна. Завдяки тому, що ми оголосили координати



(a) (b) (c)

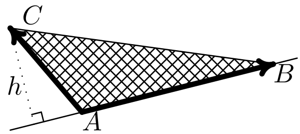
(поля `TPoint`) та результат функції `cross_prod` в типі `extended`, тут зручно застосувати математичну умову «два ненульові числа a, b різних знаків тоді й тільки тоді, коли $a \cdot b < 0$» (якби намагалися зробити те саме, маючи цілочисельні типи, це було б неправильно, бо дуже часто траплялися б переповнення, внаслідок яких додатність чи від'ємність визначалася б неправильно.

Рис. (c) показує, що можлива ситуація, коли C та D по різні боки від *прямої* AB, але *відрізки* AB і CD все ж не перетинаються. Тому умову перетину відрізків слід остаточно сформулювати, наприклад, так:

$$\left(\left(\overrightarrow{AB} \times \overrightarrow{AC}\right) \cdot \left(\overrightarrow{AB} \times \overrightarrow{AD}\right) < 0\right) \texttt{and} \left(\left(\overrightarrow{CD} \times \overrightarrow{CA}\right) \cdot \left(\overrightarrow{CD} \times \overrightarrow{CB}\right) < 0\right) \tag{6}$$

(одночасно і C та D по різні боки від AB, і A та B по різні боки від CD).

1.3.2 Відстань від точки до прямої



Щоб знайти відстань між точкою C і прямою AB, розглянемо $\triangle ABC$, і виразимо (аналітичними формулами на папері, а не у програмі) його площу двома різними способами. З одного боку, вона згідно (5) дорівнює $\left|\frac{\overrightarrow{AB} \times \overrightarrow{AC}}{2}\right|$. З іншого, вона дорівнює $\frac{1}{2} \cdot d(A, B) \cdot h$ (де $d(A, B) = |AB|$ — довжина відрізка AB), тому що можна вважати AB основою, h висотою. Маємо

$$\left|\frac{\overrightarrow{AB} \times \overrightarrow{AC}}{2}\right| = \frac{1}{2} \cdot d(A, B) \cdot h;$$

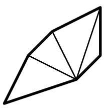
домноживши на 2 (це додатня константа, тому її можна проносити крізь модуль) і поділивши на $d(A,B)$, остаточно отримуємо

$$h = \frac{\left|\overrightarrow{AB} \times \overrightarrow{AC}\right|}{d(A,B)}. \tag{7}$$

1.3.3 Площа простого многокутника

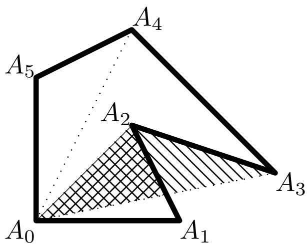
Многокутник на площині задано цілочисельними координатами N вершин. Потрібно знайти його площу. Многокутник простий, тобто його сторони не перетинаються і не дотикаються (за винятком сусідніх, у вершинах), але він *не обов'язково опуклий.*

(Приклад *не*опуклого простого многокутника див. на рис.)



Спочатку подумаємо, що можна було б зробити, якби многокутник був опуклий. Наприклад, його можна було б розбити на трикутники. Наприклад, провівши всі можливі діагоналі з однієї й тієї ж вершини. Рахувати площу окремо взятого трикутника ми вміємо, і лишається тільки пододавати площі цих трикутників.

Спробуємо зробити так само (додати площі $A_0A_1A_2$, $A_0A_2A_3$, $\ldots$, $A_0A_{n-2}A_{n-1}$) в неопуклому многокутнику. На перший погляд, усе сумно: заштриховані області включаються по кілька разів, хоча заштрихована навхрест повинна потрапляти лише один раз, а заштрихована у смужку повинна взагалі не потрапляти.



Але виявляється, що при використанні *орієнтованих* площ трикутників усе гаразд! Наприклад, на рис. область, заштрихована у смужку, враховується спочатку як частина орплощі $\triangle A_0A_2A_3$ зі знаком "$-$", а потім як частина орплощі $\triangle A_0A_3A_4$ зі знаком "$+$", і в результаті не враховується (як і треба). Аналогічно, область, заштрихована у клітинку, додається двічі (як частина $\triangle A_0A_1A_2$ і $\triangle A_0A_3A_4$ і віднімається один раз (як частина $\triangle A_0A_2A_3$, і в результаті враховується один раз (знов-таки, як і треба).

Тобто, треба просто додати $N-2$ штук *орієнтованих* площ трикутників $A_0A_1A_2$, $A_0A_2A_3$, $\ldots$, $A_0A_{n-2}A_{n-1}$, а *наприкінці* все-таки взяти модуль, бо сума орієнтованих площ — орієнтована площа; якщо обхід многокутника відбувається у від'ємному напрямку (за годинниковою стрілкою), то вона виявляється від'ємною.

(Зрозуміло, що наведений конкретний приклад не може доводити правильність для всіх можливих випадків; але тепер твердження стало більш-менш зрозумілим, а формальне доведення спирається на акуратне застосування аналогічних міркувань.)

Насамкінець, розглянутий спосіб дозволяє знайти площу, обмежену будь-яким простим многокутником, але *не* будь-якою замкненою ламаною: якби не було навіть гарантій про відсутність самоперетинів, цей метод не був би гарантовано правильним.

2 Література

1. Порублёв И. Н., Ставровский А. Б., «Алгоритмы и программы. Решение олимпиадных задач», М.–СПБ–К., Диалектика, 2007 — глава 6 «Вычислительная геометрия на плоскости».

2. Ласло М., «Вычислительная геометрия и компьютерная графика на C++», М., Binom, 1997.

3. Андреева Е. В., Егоров Ю. Е., «Вычислительная геометрия на плоскости», газета "Информатика", № 39–44, 2004 г.

4. `http://codeforces.ru/blog/entry/6642`

3 Задачі

Даний комплект задач доступний для on-line перевірки за такими посиланнями (будь-яке з них, хоч там, хоч там):

- `ejude.ckipo.edu.ua`, змагання № 13.
- `www.e-olimp.com.ua/competitions/3336`

Задача A. «Квадрат по двом вершинам»

Вхідні дані:	`Клавіатура (стандартний вхід)`
Результати:	`Екран (стандартний вихід)`

На площині задано квадрат координатами двох своїх протилежних вершин.

Знайти координати решти двох вершин квадрата.

Вхідні дані

Програма повинна прочитати зі стандартного входу (клавіатури) два рядки. У першому рядку вводяться координати однієї з вершин квадрата Ax і Ay, у другому — координати протилежної вершини квадрата Cx і Cy. Всі числа по модулю не перевищують 10^6.

Результати

Програма має вивести на стандартний вихід (екран) два рядки по два розділених пробілами числа Bx By, Dx Dy — координати решти двох вершин квадрата. Порядок виведення повинен бути таким, щоб A, B, C, D відповідало додатному порядку обходу (проти годинникової стрілки).

Приклад

Клавіатура (стандартний вхід)	Екран (стандартний вихід)
5 6 4 1	2 4 7 3

Задача повністю розібрана у розд. 1.1.2, 1.1.3.

Задача B. «Чи перетинаються відрізки—1»

Вхідні дані:	`Клавіатура (стандартний вхід)`
Результати:	`Екран (стандартний вихід)`

Дано чотири точки A, B, C, D, щодо яких гарантовано, що ніякі три не лежать на одній прямій. Чи перетинаються відрізки AB і CD?

Вхідні дані

слід прочитати зі стандартного входу (клавіатури). Це будуть рівно три групи по два рядки у кожній (сумарно шість рядків). Перший з рядків кожної групи містить відрізок AB (у вигляді чотирьох чисел Ax Ay Bx By), другий і останній рядок кожної групи — CD як Cx Cy Dx Dy. Усі координати є цілими числами, не перевищують по модулю мільйон.

Результати

Для кожної з груп вивести у окремому рядку `YES` (якщо відрізки перетинаються) або `NO` (якщо ні).

Приклад

Клавіатура (стандартний вхід)	Екран (стандартний вихід)

`0 0 1 0`
`100 -100 100 100`
`-5 0 5 0`
`0 5 0 -5`
`592741 76372 273343 724795`
`408678 74450 197154 3779`
	`NO`
	`YES`
	`NO`

Задача повністю розібрана у розд. 1.3.1.

Задача C. «Чи перетинаються відрізки—2»

Вхідні дані: `Клавіатура (стандартний вхід)`
Результати: `Екран (стандартний вихід)`

Дано чотири точки A, B, C, D. Чи мають відрізки AB і CD хоча б одну спільну точку? Програма повинна працювати в усіх випадках, включаючи в тому числі й ситуації, коли відрізки накладаються, а також $A = B$ або $C = D$, тобто один чи обидва відрізки вироджені в точку.

Вхідні дані

слід прочитати зі стандартного входу (клавіатури). У першому рядку задано N ($1 \leqslant N \leqslant 5$) — кількість пар відрізків у даному тесті, далі йдуть N груп по два рядки у кожній. Перший з рядків кожної групи містить відрізок AB (у вигляді чотирьох чисел Ax Ay Bx By), другий і останній рядок кожної групи — CD як Cx Cy Dx Dy. Усі координати є цілими числами, не перевищують по модулю мільйон.

Результати

Для кожної з груп вивести у окремому рядку `YES` (якщо відрізки перетинаються) або `NO` (якщо ні).

Приклади

Клавіатура (стандартний вхід)	Екран (стандартний вихід)
3 0 0 1 0 100 -100 100 100 -5 0 5 0 0 5 0 -5 592741 76372 273343 724795 408678 74450 197154 3779	NO YES NO
1 0 0 3 6 1 2 1 2	YES

Розбір

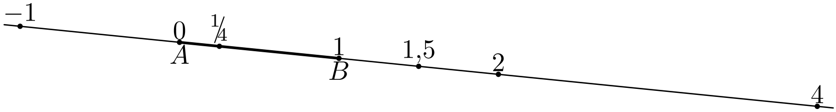
При бажанні можна пододавати розгляд особливих випадків у розв'язок попередньої задачі (і об'єм коду для тих випадків *значно* перевищить об'єм коду для «основного» випадку). Але розглянемо інший підхід. Його перевага, зокрема, в тому, що він дозволяє знайти не лише факт наявності/відсутності перетину, а ще й конкретні координати.

Розглянемо пару функцій від одного й того ж дійсного аргументу

$$\begin{cases} x_1(t_1) = A_x + AB_x \cdot t_1, \\ y_1(t_1) = A_y + AB_y \cdot t_1 \end{cases} \tag{8}$$

(де $(A_x; A_y)$ та $(B_x; B_y)$ — координати точок A та B, а $(AB_x; AB_y) = (B_x - A_x;\ B_y - A_y)$ — координати вектора $\overrightarrow{AB}$.)

(8) називають *поданням прямої через початкову точку й напрямляючий вектор.* Можна вважати, що це одна функція, яка приймає один дійсний параметр t_1 і вертає значення типу «координати точки». І це якраз і будуть точки, які належать прямій AB (на рис. показані розміщення точок при різних t):



Аналогічно можна записати й пряму CD, як $\begin{cases} x_2(t_2) = C_x + CD_x \cdot t_2, \\ y_2(t_2) = C_y + CD_y \cdot t_2. \end{cases}$
Причому, для різних координат однієї прямої користуємося тією ж змінною (щоб зміна координат була узгодженною). А для різних прямих — різними змінними, щоб рух поточної точки по одній з прямих не впливав на поточну точку на іншій прямій.

Тоді твердження «прямі AB та CD перетнулися у точці $(x^\star; y^\star)$» можна записати такою системою:

$$\begin{cases} x^\star = A_x + AB_x \cdot t_1, \\ y^\star = A_y + AB_y \cdot t_1, \\ x^\star = C_x + CD_x \cdot t_2, \\ y^\star = C_y + CD_y \cdot t_2. \end{cases}$$

Прирівнявши праві частини першого і третього, а також другого і четвертого рівнянь, отримуємо $\begin{cases} A_x + AB_x \cdot t_1 = C_x + CD_x \cdot t_2, \\ A_y + AB_y \cdot t_1 = C_y + CD_y \cdot t_2, \end{cases}$ що перетворюється до системи двох лінійних рівнянь з двома невідомими

$$\begin{cases} AB_x \cdot t_1 - CD_x \cdot t_2 = C_x - A_x, \\ AB_y \cdot t_1 - CD_y \cdot t_2 = C_y - A_y \end{cases} \tag{9}$$

(невідомі, відносно яких потрібно розв'язати систему — t_1 і t_2; усі значення AB_x, AB_y, CD_x, CD_y, $C_x - A_x$, $C_y - A_y$ відомі, бо легко обчислюються на основі вхідних даних).

З одного боку, задача розв'язування системи лінійних рівнянь загальновідома, і це добре. З іншого:

- а що, власне, робити зі знайденими значеннями t_1 та t_2?
- що робити, якщо виявиться, що система не має розв'язків? а якщо має нескінченно багато розв'язків?

Почнемо з другого питання.

Якщо система взагалі не має розв'язків — відповідь "`NO`". У невироджених випадках це значить, що прямі AB та CD не перетинаються (паралельні), отже відрізки тим паче не перетинаються. Оскільки A, B, C, D абсолютно які завгодно, «система не має розв'язків» може означати і щось інше, наприклад ситуацію «$A{=}B$, $C{\neq}D$, точка A (вона ж B) не лежить на прямій CD». Але в усіх випадках справедливо: якщо система взагалі не має розв'язків — відповідь "`NO`".

Якщо розв'язків нескінченно багато — прямі накладаються (або один з відрізків вироджений у точку, і ця точка лежить на прямій, що містить інший відрізок; або $A = B = C = D$). Але що це значить — треба розбиратися: відрізки на одній прямій можуть і не перетинатися, і мати єдину спільну точку (один «починається» там, де «закінчився» інший), і мати спільну частину ненульової довжини.

Якщо відомо, що має місце саме цей випадок, *то* відрізки AB та CD (тут і далі у межах цього випадку, «відрізок» може означати як «справжній» відрізок, так і вироджений у точку) мають хоча б одну спільну точку тоді й тільки тоді, коли виконується *хоча б одне* з чотирьох тверджень:

$$\left[\begin{array}{l} \text{точка } A \text{ належить відрізку } CD; \\ \text{точка } B \text{ належить відрізку } CD; \\ \text{точка } C \text{ належить відрізку } AB; \\ \text{точка } D \text{ належить відрізку } AB. \end{array}\right. \tag{10}$$

Факт «точка P належить відрізку AB» можна виражати різними способами, зокрема:

- $\overrightarrow{PA} \cdot \overrightarrow{PB} \leqslant 0$, де "$\cdot$" — скалярний добуток; взагалі-то ця умова лише необхідна, але *в рамках випадку* «(9) має нескінченно багато розв'язків» вона також і достатня.
- $|AP| + |PB| = |AB|$ — ця умова може здатися простішою, але насправді відповідний фрагмент програми не простіший, і при цьому спосіб має істотний недолік: на нього *значно* суттєвіше, ніж на попередній спосіб, впливають похибки (неточності) обчислень.

Звісно, цю умову варто записати функцією, наприклад,
`function point_on_segm(const P,  A, B : Point) : boolean`
і викликати чотири рази з різними параметрами, наприклад,
`point_on_segm(C,A,B) or point_on_segm(D,A,B) or`
`point_on_segm(A,C,D) or point_on_segm(B,C,D)`.

Нарешті, якщо система (9) має єдиний розв'язок $t_1{=}t_1^\star$, $t_2{=}t_2^\star$ — обидва відрізки не вироджені, а прямі мають (єдину) точку перетину. (При потребі, її координати можна було б знайти, підставивши $t_1^\star$ у (8).) Раз нам треба не прямі, а відрізки — слід перевірити, чи знайдена точка перетину належить відрізку AB (тоді й тільки тоді, коли $0{\leqslant}t_1^\star{\leqslant}1$) та чи належить відрізку CD ($0{\leqslant}t_2^\star{\leqslant}1$).

Розглянемо (без доведень; бажаючі можуть, ознайшовшись із методом Крамера та його доведенням, провести аналіз особливих випадків), як розв'язувати систему двох лінійних рівнянь з двома невідомими

$$\left\{\begin{array}{l} a_{11}x_1 + a_{12}x_2 = b_1, \\ a_{21}x_1 + a_{22}x_2 = b_2, \end{array}\right. \tag{11}$$

коли наперед не відомо, чи існує і чи єдиний розв'язок.

Спочатку обчислимо три величини

$$\Delta = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}, \quad \Delta_1 = b_1 \cdot a_{22} - a_{12} \cdot b_2, \quad \Delta_2 = a_{11} \cdot b_2 - b_1 \cdot a_{21}.$$

Якщо $\Delta \neq 0$, то розв'язок існує, єдиний, і дорівнює: $x_1 = \frac{\Delta_1}{\Delta}$, $x_2 = \frac{\Delta_2}{\Delta}$ (які б не були Δ_1, Δ_2).

Якщо $\Delta = 0$ *та* $\Delta_1 = \Delta_2 = 0$, то розв'язків нескінченно багато.

(Насправді з цього твердження є виключення: випадок «$a_{11} = a_{12} = a_{21} = a_{22} = 0$, і при цьому $(b_1 \neq 0)$ `or` $(b_2 \neq 0)$», у початкових геометричних термінах це значить $A{=}B$, $C{=}D$, $A{\neq}C$, тобто обидва відрізки вироджені у точки, й ці точки різні. Але у даній конкретній програмі особливість цього випадку можна й не враховувати, бо та сама перевірка (10) дасть правильну відповідь `false`.)

Якщо $\Delta = 0$, але хоча б одне зі значень Δ_1, Δ_2 не нульове, то розв'язків не існує.

Задача D. «Яка частина прямої у крузі?»

Вхідні дані:	`Клавіатура (стандартний вхід)`
Результати:	`Екран (стандартний вихід)`

Є коло (задане радіусом і координатами центра) і пряма (задана координатами двох своїх точок).

Якої довжини відрізок прямої лежить у крузі (всередині кола)?

Вхідні дані

слід прочитати зі стандартного входу (клавіатури). У першому рядку задано три числа: спочатку радіус кола R, потім координати його центра Cx Cy. У другому та третьому задано по два числа — x- та y-координати точок (гарантовано двох різних), через які проходить пряма. Всі числа цілі, за абсолютним значенням не перевищують 10000.

Результати

Вивести єдине число: якщо пряма і коло мають хоча б одну спільну точку — довжину відрізка цієї прямої, що лежить у крузі (всередині кола); якщо не мають жодної спільної точки — замість цієї довжини вивести число `-1`.

У випадку торкання прямої до кола, спільна точка є, але відрізка ненульової довжини нема; отже, при торканні слід виводити `0`.

Результат при виведенні *не можна* заокруглювати (а виводити в експоненційній формі, як-то `6.0000000000000000E+0000` замість `6`, можна).

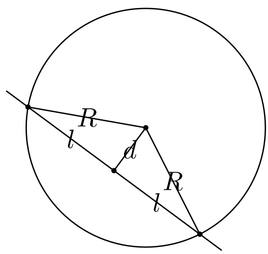
Приклад

Клавіатура (стандартний вхід)	Екран (стандартний вихід)
5 0 0 4 1 4 2	6

Розбір

Взагалі-то, одна з найпростіших задач.

Перш за все, знайдемо (згідно розд. 1.3.2) відстань d від центра кола до прямої. Якщо ця відстань строго більша за радіус кола, то пряма проходить ззовні кола, і, згідно умови, треба вивести `-1`.



Інакше неважко переконатися, що шукана довжина дорівнює $2l = 2\sqrt{R^2 - d^2}$ (справді, на рис. бачимо два рівні прямокутні трикутники, R заданий у вхідних даних, d знайшли на попередньому кроці).

Випадок, коли пряма лише дотикається до кола, не заходячи всере́дину нього, теж описується цією самою формулою (вона дає 0, як і треба).

Задача E. «Відстань від точки до відрізка»

Вхідні дані: `Клавіатура (стандартний вхід)`
Результати: `Екран (стандартний вихід)`

Дано точку P з координатами Px Py та відрізок AB, кінці якого мають координати Ax Ay та Bx By. Відрізок гарантовано не вироджений, тобто A та B — різні точки.

Напишіть програму, яка знаходитиме відстань між точкою P та відрізком AB.

Примітка. Відстань між точкою та відрізком слід трактувати згідно зі стандартним означенням відстані між точкою та складним геометричним об'єктом: якщо точка належить цьому об'єкту, відстань рівна нулю; якщо не належить, відстань рівна довжині найкоротшого з можливих відрізків, для яких одним з кінців є дана точка, а інший кінець належить цьому об'єкту.

Вхідні дані

слід прочитати зі стандартного входу (клавіатури), у форматі Px Py Ax Ay Bx By (в одному рядку). Всі координати цілі й не перевищують по модулю 10000.

Результати

Вивести єдине число — знайдену відстань від точки до відрізка. Виводити можна хоч у експоненційній формі, хоч стандартним десятковим дробом. Результат зараховується, коли похибка не перевищує 10^{-4}.

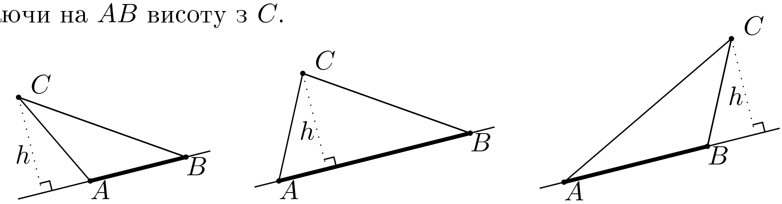
Приклад

Клавіатура (стандартний вхід)	Екран (стандартний вихід)
0 4 2 3 2 5	2.0

Розбір

Задача у значній мірі спирається на розд. 1.3.2: там розказано, як шукати відстань між точкою C і *прямою* AB.

Розглянемо (як і у розд. 1.3.2) $\triangle ABC$, виділяючи AB як основу і опускаючи на AB висоту з C.



З цих рисунків, а також наведеного в умові задачі означення відстані між точкою і протяжним об'єктом, ясно, що можуть бути три випадки:

- якщо $\angle CAB$ тупий, то найкоротшим з можливих відрізків від точки C до відрізка AB є відрізок CA;
- якщо $\angle CBA$ тупий, то найкоротшим з можливих відрізків від точки C до відрізка AB є відрізок CB;
- якщо жоден з кутів кутів $\angle CAB$ чи $\angle CBA$ не є тупим, то найкоротшим з можливих відрізків від точки C до відрізка AB є висота, і її слід обчислити згідно формули (7) з розд. 1.3.2.

Тобто, лишилося тільки з'ясувати, який з цих випадків має місце. Наприклад, це можна зробити за допомогою скалярних (розд. 1.2) добутків: $\angle CAB$ тупий $\Leftrightarrow \overrightarrow{AB} \cdot \overrightarrow{AC} < 0$, $\angle CBA$ тупий $\Leftrightarrow \overrightarrow{BA} \cdot \overrightarrow{BC} < 0$.

(В принципі можливі й альтернативні способи перевірки, наприклад: $\angle CAB$ тупий $\Leftrightarrow |BC|^2 > |AC|^2 + |AB|^2$. Але навряд чи вони об'єктивно кращі й простіші за перевірку скалярних добутків.)

Зауважимо, що тип $\angle ACB$ (гострий/прямий/тупий) ніяк не впливає на розв'язок. Ще зауважимо, що випадок прямого $\angle CAB$ нема потреби розглядати окремо: його можна віднести хоч до випадку тупого кута, хоч до випадку гострого, і так і так буде правильно, бо при прямому $\angle CAB$ висота дорівнює катету. З $\angle CBA$ аналогічно.

Задача F. «Площа простого многокутника»

Вхідні дані:	`Клавіатура (стандартний вхід)`
Результати:	`Екран (стандартний вихід)`

Многокутник на площині задано цілочисельними координатами N вершин. Потрібно знайти його площу.

Многокутник простий, тобто його сторони не перетинаються і не дотикаються (за винятком сусідніх, у вершинах), але він не обов'язково опуклий.

Вхідні дані

слід прочитати зі стандартного входу (клавіатури). У першому рядку задано кількість вершин N ($1 \leqslant N \leqslant 50000$). У наступних N рядках записані пари чисел — координати вершин. Сторони многокутника — відрізки між 1-ою і 2-ою, 2-ою і 3-ьою, ..., $(N-1)$-ою і N-ою, N-ою і 1-ою вершинами. Значення координат — цілі числа, не перевищують по модулю мільйон.

Результати

Вивести єдине число — знайдену площу многокутника. Виводити можна хоч у експоненційній формі, хоч стандартним десятковим дробом. Результат зараховується, коли похибка не перевищує 0,1.

Приклад

Клавіатура (стандартний вхід)	Екран (стандартний вихід)
4 0 4 0 0 3 0 1 1	3.5

Задача повністю розібрана у розд. 1.3.3.

Задача G. «Трикутник і точка 2»

Вхідні дані:	`Клавіатура (стандартний вхід)`
Результати:	`Екран (стандартний вихід)`

Задано трикутник ABC координатами вершин і точка O. Визначити місце розміщення точки.

Вивести "`In`", якщо точка лежить сторого всере́дині трикутника, "`Edge`", якщо точка лежить на стороні, "`Vertex`", якщо точка лежить на вершині і "`Out`", якщо точка лежить поза трикутником.

Вхідні дані

Чотири рядки по 2 цілих числа, що не перевищують по модулю 10 000 000.

Перші три рядки — координати вершин трикутника A, B, C. Четвертий — точка O.

Результати

Відповідь до задачі ("`In`", "`Edge`", "`Vertex`" або "`Out`").

Приклад

Клавіатура (стандартний вхід)	Екран (стандартний вихід)
-2 -2 3 1 0 1 0 0	In

Розбір

Якщо сума площ $S_{\triangle OAB}$, $S_{\triangle OBC}$ і $S_{\triangle OCA}$ перевищує $S_{\triangle ABC}$, точка лежить поза трикутником. Якщо ж сума перших трьох площ дорівнює четвертій, то перевіримо, чи не дорівнюють нулю деякі з перших трьох площ. Якщо жодна не дорівнює — точка строго всередині трикутника, якщо нулю дорівнює одна площа — точка на стороні, якщо дві — точка O співпадає з однією з вершин A або B або C.

Примітка. Порівнювати треба суму «звичайних» площ, а не орієнтованих. Сума орплощ завжди дорівнює орплощі ABC, незалежно від розміщення точки O.