**★★☆☆☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10263: Railway**

**解題者：杜詠聖**

**解題日期：2019年3月8日**

**題意：**

有一組連續的鐵道(線段)，找到此鐵道離火車站(點)最近的點。

先輸入點M，再輸入N(代表N條鐵路)，再輸入N+1個點(2N+2行)。重複上述步驟，直到EOF結束。

**題意範例：**

6 -3 3 0 1 5 5 9 -5 15 3 → 7.8966 -2.2414

0 0 1 1 0 2 0 → 1.0000 0.0000

**解法：**

1.利用外積(|X1Y2-Y1X2|)求點到此直線的距離

兩向量外積=兩向量圍成的平行四邊形面積

距離=外積/線段長度

2.利用內積(X1X2+Y1Y2)(內積/線段長度^2 =|𝑎|cos𝜃/|𝑏| , b為鐵路的線段)

1.判斷投影點是否在此線段上。

if (0 ≤ |𝑎|cos𝜃 / |𝑏| ≤ 1) 在線段上

else 在線段外

2.求投影點和線段兩點之間長度的比例。

比例 = |𝑎|cos𝜃 / |𝑏|

**解法範例：**

上述第一組測資

第1條線段向量=(5, 4), 點與此線段第一點向量=(6,-4)

內積/線段長度^2=14/41 距離=44/ 投影點：(70/41, 97/41)

☑ 第2條線段向量=(4, -10), 點與此線段第一點向量=(1,-8)

內積/線段長度^2=21/29 距離=11/ 投影點：(229/29, -65/29)

第3條線段向量=(6, 8), 點與線段第一點向量=(-3,-2)

內積/線段長度^2=-34/100 距離=點到此線段第一點的距離= 投影點：(9, -5)

第二條與點最近，故答案為(229/29, -65/29) 輸出為7.8966 -2.2414

**討論：**

其他做法：點在某直線的垂線與某直線聯立解交點

**程式：**

#include<iostream>

#include<cstdint>

#include<cmath>

#include<vector>

#include<cfloat>

using namespace std;

class point;

static const double length(const point &a, const point &b);

static const double dot(const point &a, const point &b);

static const double cross(const point &a, const point &b);

static ostream & operator<<(ostream &out, const point &ans);

class point {

public:

point();

point(const double x, const double y);

point(const point &p);

const point operator+(const point &b)const;

const point operator-(const point &b)const;

const point operator\*(const double b)const;

point & operator=(const point &p);

friend const double length(const point &a, const point &b);

friend const double dot(const point &a, const point &b);

friend const double cross(const point &a, const point &b);

friend ostream & operator<<(ostream &out, const point &ans);

private:

double x;

double y;

};

int main() {

cout.setf(ios::fixed);

cout.precision(4);

double x, y;

while (cin >> x >> y) {

point m(x, y);

uint64\_t n;

cin >> n;

cin >> x >> y;

point ans(x, y);

if (n) {

vector<point> v;

v.emplace\_back(ans);

double min = DBL\_MAX;

while (n--) {

cin >> x >> y;

v.emplace\_back(point(x, y));

double len\_01 = length(v[0], v[1]), len\_m0 = length(m, v[0]), len\_m1 = length(m, v[1]);

double dis = fabs(cross(m - v[0], v[1] - v[0])) / len\_01;

double rate = dot(m - v[0], v[1] - v[0]) / pow(len\_01, 2);

if (rate >= 1)

dis = len\_m1;

else if (rate <= 0)

dis = len\_m0;

if (dis < min) {

min = dis;

if(rate <= 1 && rate >= 0)

if (dis)

ans = v[0] \* (1 - rate) + v[1] \* rate;

else

ans = m;

else

ans = len\_m0 <= len\_m1 ? v[0] : v[1];

}

v.erase(v.begin());

}

}

cout << ans << endl;

}

return 0;

}

inline point::point() {}

inline point::point(const double x, const double y) :x(x), y(y) {}

inline point::point(const point &p) : x(p.x), y(p.y) {}

inline const point point::operator+(const point &b)const {

return point(x + b.x, y + b.y);

}

inline const point point::operator-(const point &b)const {

return point(x - b.x, y - b.y);

}

inline const point point::operator\*(const double b)const {

return point(b\*x, b\*y);

}

inline point & point::operator=(const point &p) {

if (this != &p) {

x = p.x;

y = p.y;

}

return \*this;

}

static inline const double length(const point &a, const point &b) {

return sqrt(pow(a.x - b.x, 2) + pow(a.y - b.y, 2));

}

static inline const double dot(const point &a, const point &b) {

return a.x\*b.x + a.y\*b.y;

}

static inline const double cross(const point &a, const point &b) {

return a.x\*b.y - a.y\*b.x;

}

static inline ostream & operator<<(ostream &out, const point &ans) {

out << ans.x << endl << ans.y;

return out;

}