Геометрия. Часть 1.

Вычислительная геометрия - тема, в рамках которой необходимо решать задачи связнные со взаминым расположением геометрических объектов друг относительно друга. Например, задача проверки пересекаются ли 2 окружности - классическая задача вычислительной геометрии. Так же часто надо отвечать на запросы про характеристики фигур, например, чему равны их площади.

Стандартные объекты, которыми занимается вычислительная геометрия - точка, прямая, отрезок, окружность, многоугольник.

Самая простой объект - точка. В двумерном пространстве он задается 2 координатами x и y. В зависимости от задачи они могут быть целыми или вещественными. Если задачу можно решить в целых числах **не нужно** создавать вещественные координаты, так как можно потерять в точности.

В любой задаче вычислительной геометрии **обязательно** должна встречаться структура $pt(point)$ для работы с классом точка. Связано это с тем, что невозможно называя переменные $x1, y1, x2, y2$ и прочее не запутаться. А операции, которые над ними выполняются одинаковые.

Пример объявления данной структуры:

```
struct pt
{
        double x, y;
};
```

Надо понимать, что pt может работать и с вектором (в геометрическом смысле слова). То есть вектор также задается 2 координатами. Вектора можно складывать, вычитать и умножать на число. Доработаем нашу струкуру с учетом этого:

```
struct pt
{
        double x, y;
        pt(double x = 0, double y = 0):x(x), y(y)//pt a; <=> pt a(0, 0);
        {}
        pt operator + (pt b)
        {
                pt res;
                res.x = x + b.x;
                res.y = y + b.y;
                return res;
        }
        pt operator - (pt b)
        {
                pt res;
                res.x = x - b.x;
                res.y = y - b.y;
                return res;
```

```
        }
        pt operator * (double k)
        {
                pt res;
                res.x = x * k;
                res.y = y * k;
                return res;
        }
};
```

Теперь если надо по двум точкам A и B получить вектор AB, то это делается так:

```
    pt A, B, vec;
        cin >> A.x >> A.y;
        cin >> B.x >> B.y;
        vec = B - A;
```

Над точками существует операция расстояния между точками (или же длины вектора меду ними). Ее обязательно также всегда писать функцией. Лучше писать ее как квадрат длины вектора, чтобы не брать лишний раз корень (например, для сравнения этой длины с чем либо этого обычно достатоточно).

```
double dist2(pt a)
{
        return a.x * a.x + a.y * a.y;
}
```

Так же очень важными операциями являются скалярное и псевдовекторное произведение. Их также обязательно писать отдельной функцией.

Формула скалярного произведения: $(a, b) = |a||b|cos\alpha = a.x * b.x + a.y * b.y$

Формула векторного произведения: $[a, b] = |a||b|sin\alpha = a.x * b.y - a.y * b.x$

Теперь несложно заметить, что скалярного и вектрного произведения, нам хватит чтобы вычислить угол между двумя векторами (при использовании $asin$ и $acos$). Однако, так как он может варьироваться от 0 до 2π, это связано с разбором случаев. На помощь в такой ситуации приходит функция $atan2(y, x)$, которая по координатам точки возвращает угол меду осью OX и точкой. Примечание: внимательно смотрите область определения функции.

Следующим ключевым объектом является прямая. Ее можно задавать 2 точками или с помощью коффициентов A, B и . Разберем второй случай:

```
struct line
{
        double a, b, c;
};
```

Конструктор, который по двум точкам делает прямую с коеффициентами a, b и вам предстоит написать самостоятельно.

Для хранения струкутры отрезок можно использоватьконтейнер $pair$. Для хранения структуры многоугольник можно использовать $vector$.

Замечание. Сравнение двух вещественных чисел. Часто бывает нужно странить 2 вещественных числа. Следует понимать, что вещественные чилса хранятся в памяти компьютера приближенно, и компьютер не всегда понимает с какой точность сранивать числа. Поэтому, вместо $a == b$ следует писать $fabs(a - b) < eps$, где eps - глобальная перменная заданная в начале программы (например: $double\ eps = 1e - 7$). Запись $1e - 2$ соответсвует 0.01, а $1e3$ - 1000. Если надо проверить, что $a < b$, то правильно будет написать $a < b - eps$.