Занятие 21. Применения векторного произведения

Вспомним из математики, что такое многоугольник. Его определяют по-разному.

Определение 1. Многоугольник — замкнутая ломаная линия, именно: если A_1, A_2, …, A_n — различные точки, никакие последовательные три из которых не лежат на одной прямой, то совокупность отрезков A_1A_2, A_2A_3, …, A_nA_1 называется многоугольником. При этом точки A_1, …, A_n называются вершинами многоугольника, а отрезки A_1A_2, A_2A_3, …, A_nA_1 — его сторонами.

Определение 2. Многоугольником называют область, ограниченную замкнутой ломаной линией.

Многоугольники могут быть пространственными или плоскими.

Если ломаная, о которой говорится в определении 1, не имеет самокасаний и самопересечений, то многоугольник называется простым.

Далее мы будем говорить *только* о простых плоских многоугольниках.

Задача 1. Направление обхода

Многоугольник $A_1A_2…A_n$ задан координатами своих вершин, перечисленных в порядке их обхода по или против часовой стрелки. Определить это направление обхода: по часовой стрелке или против.

Будем считать, что все координаты — целые числа, по модулю не превосходящие 10^4, и заданы они попарно (x,y) после указания количества вершин многоугольника $n \le 1000$. Примем также, что система координат такова, ось x направлена вправо, а ось y — вверх.

Решить эту задачу можно так.

1. Найдём самую левую вершину многоугольника, а если таких окажется несколько, то самую нижнюю из них. Пусть это будет вершина A.
2. Проведём векторы из вершины D, предшествующей вершине A в порядке обхода в саму вершину A и из вершины A в следующую вершину B (векторы $\overrightarrow{DA}$ и $\overrightarrow{AB}$, см. рис. 1). При этом вершиной, следующей за последней, будем считать первую.
3. Вычислим $\vec{c} = \overrightarrow{DA} \times \overrightarrow{AB}$
4. Если $\vec{c}$ направлен от нас, то обход многоугольника осуществлялся по часовой стрелке, иначе — против часовой стрелки.

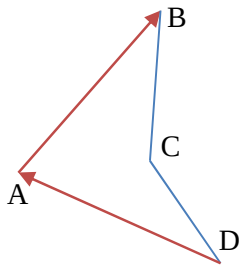

Рисунок 1

Заметим, что случай $\vec{c} = \vec{0}$ невозможен, т.к. никакие 3 соседние вершины многоугольника в силу определения 1 не лежат на одной прямой.

Может показаться, что п.п. 2 – 4 этого алгоритма можно применить для любой вершины, а не только самой левой. Однако это неверно для невыпуклых вершин[1]. Убедитесь в этом, рассмотрев векторы $\overrightarrow{BC}$ и $\overrightarrow{CD}$ на рис. 1. Выбор же самой левой вершины гарантирует, что эта вершина будет выпуклой. Выпуклыми будут также самая правая, самая верхняя и самая нижняя вершины. Поэтому в качестве вершины A в приведённом алгоритме можно использовать любую из них. На рис. 1 это были бы вершины A, B или D.

[1] Невыпуклой вершиной многоугольника будет вершина, внутренний угол при которой больше 180°

Координаты точек будем хранить в массиве записей TPoint таких же, как и на предыдущем занятии:

```
Type TPoint = record
                x, y : Integer
              end;
```

Реализация пункта 1 намеченного алгоритма будет напоминать поиск минимального элемента массива (см. занятие 6). Это делает функция LeftPoint программы Obhod.

Само же направление обхода вычисляется в функции Orient. Для этого сначала вычисляется индекс левой нижней точки и сохраняется в переменной $left$, причём нумерация вершин многоугольника ведётся с нуля. Тогда индекс i предыдущей в процессе обхода точки может быть вычислен так:

$$i = \begin{cases} left - 1 \text{ при } left \geq 1 \\ n - 1 \text{ при } left = 0 \end{cases}$$

То же самое можно сделать иначе: $i = (left - 1 + n) \bmod n$.

```
{$mode delphi}
Program Obhod;

Const MAXN=1000; //Максимальное количество вершин многоугольника

Type TPoint = record
                x, y : Integer
              end;
     TVector = TPoint;
//Описание вершин многоугольника
     TPolygon = array[0..MAXN-1] of TPoint;

//Вычисляем проекцию векторного произведения a x b на ось z
Function CrossProduct(const a,b : TVector) : Integer;
Begin
  Result := a.x * b.y - a.y * b.x
End;

//Вычисление проекций вектора AB по координатам его концов
Procedure MakeVector(const A,B : TPoint; out AB : TVector);
Begin
  AB.x := B.x - A.x;
  AB.y := B.y - A.y
End;

//Вычисление номера левой нижней вершины многоугольника
Function LeftPoint(const p : TPolygon; n : Integer) : Integer;
Var i, num : Integer;
    A      : TPoint; //Искомая точка
Begin
  num:=0;//Предположим, что самая первая вершина и есть левая нижняя
  A := p[0];
  For i:=1 To n-1 Do
  If (p[i].x < A.x) //Найдена точка левее
     or ((p[i].x = A.x) and (p[i].y < A.y))
//... или с той же абсциссой, что A, но ниже
  Then Begin
         num:=i;
         A:=p[i]
       End;
  Result := num
End;
```

```
//Вычисление направления обхода многоугольника.
//Возвращаемое значение:
//         положительное, если обход против часовой стрелки,
//         отицательное, если обход по часовой стрелке.
Function Orient(const p : TPolygon; n : Integer) : Integer;
Var left   : Integer;
    AB, DA : TVector;
Begin
  left:=LeftPoint(p,n);
  MakeVector(p[(left-1+n) mod n],p[left],DA);
  MakeVector(p[left],p[(left+1) mod n],AB);
  Orient := CrossProduct(DA,AB); //Никогда не ноль
End;

//-------------------------------------------------------

Var p    : TPolygon;
    f    : Text;
    n    : Integer; //Количество вершин многоугольника
    i    : Integer;
Begin
  Assign(f,'input.txt');
  Reset(f);
  Read(f,n);
  For i:=0 To n-1 Do
  Read(f,p[i].x,p[i].y);
  Close(f);

  i:=Orient(p,n);

  Assign(f,'output.txt');
  Rewrite(f);
  If i<0 Then
    Write(f,'По часовой стрелке')
  Else
    Write(f,'Против часовой стрелки');
  Close(f)
End.
```

Задача 2. Площадь многоугольника

Определить площадь многоугольника, не обязательно выпуклого.
Входной файл input.txt в первой строке содержит натуральное число N — количество вершин многоугольника. Следующие N строк содержат по 2 целых числа x_i и y_i — координаты вершины номер i. ($1 \le |x_i|, |y_i| \le 10^5,\ N \le 10^5$)
В выходной файл output.txt выведем единственное число — площадь многоугольника.

Эту задачу можно свести к задаче вычисления площадей треугольников. Для этого из одной из вершин многоугольника проведём векторы во все остальные вершины.

Если многоугольник выпуклый (см. рис. 2), то весь он будет разделён на треугольники, сумма площадей которых равна площади многоугольника. Площадь же каждого треугольника равна половине длины (модуля) векторного произведения векторов, на которых он построен (см. занятие 20). Например, $S_{\Delta A_0A_1A_2} = \frac{1}{2}\left|\overrightarrow{A_0A_1} \times \overrightarrow{A_0A_2}\right|$.

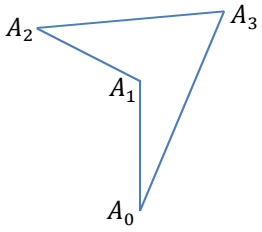

Рисунок 3

Пусть теперь многоугольник не является выпуклым (см. рис. 3). Если мы теперь проведём векторы из одной из вершин так

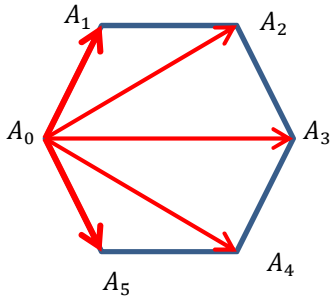

Рисунок 2

же, как и в случае выпуклого многоугольника, то сумма площадей полученных треугольников может не совпасть с площадью многоугольника. В примере на рис. 4

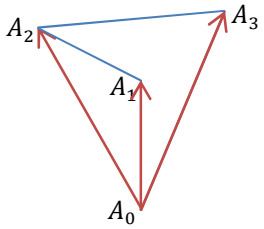

Рисунок 4

$$S_{A_0A_1A_2A_3} < S_{\Delta A_0A_1A_2} + S_{\Delta A_0A_2A_3}$$

Заметим, что $S_{A_0A_1A_2A_3} = S_{\Delta A_0A_2A_3} - S_{\Delta A_0A_1A_2}$, а также что векторы

$$\vec{S}_{\Delta A_0A_1A_2} = \frac{1}{2}\overrightarrow{A_0A_1} \times \overrightarrow{A_0A_2} \qquad (1)$$

и

$$\vec{S}_{\Delta A_0A_2A_3} = \frac{1}{2}\overrightarrow{A_0A_2} \times \overrightarrow{A_0A_3}$$

равны по длине площадям соответствующих треугольников и направлены в противоположные стороны. Назовём $\vec{S}_{\Delta A_0A_1A_2}$ и $\vec{S}_{\Delta A_0A_2A_3}$ векторами ориентированной площади треугольников. Складывая эти векторы, получим вектор ориентированной площади многоугольника:

$$\vec{S} = \vec{S}_{\Delta A_0A_1A_2} + \vec{S}_{\Delta A_0A_2A_3}$$

Длина вектора $\vec{S}$ и есть ответ на вопрос задачи.

Можно доказать, что площадь любого -угольника, вершины которого пронумерованы с нуля, равна

$$S = \frac{1}{2}\left|\sum_{i=1}^{n-2} \overrightarrow{A_0A_i} \times \overrightarrow{A_0A_{i+1}}\right| \qquad (2)$$

Знаком

$$\sum_{i=1}^{m} a_i$$

в математике и физике обозначают сумму $a_1 + a_2 + \ldots + a_m$

Обратите внимание, что в формуле (2) сначала вычисляется сумма векторов, а только потом находится длина полученного вектора. Если сделать наоборот (считать сумму длин векторов), то формула (2) будет верна только для выпуклых многоугольников.

Теперь примем меры для повышения точности вычислений. Все координаты в этой задаче — целые числа. Как следует из формулы (1), площади всех треугольников в этом случае будут целыми или полуцелыми. Чтобы избежать использования переменных вещественного типа, в (2) будем выполнять деление на 2 в самую последнюю очередь, уже после суммирования.

Какой из целочисленных типов данных выбрать? Проекция на ось z векторного произведения двух векторов, равная $x_1y_2 - x_2y_1$, учитывая ограничения задачи на исходные данные, не может превосходить по модулю $10^5 \cdot 10^5 - (-10^5) \cdot 10^5 = 2 \cdot 10^{10}$ (любая из координат может быть от -10^5 до $+10^5$). Но она может достигать этой величины. Поэтому для её хранения нужен 64-битный тип данных. Модуль суммы этих проекций может достигать величины в N раз большей, т.е. $2 \cdot 10^{15}$. Это тоже соответствует 64-битному типу данных.

Попробуйте сначала сами написать программу, и только потом читайте текст ниже.

```
{$mode delphi}
Program area_polyg;

Const MAXN=100000; //Максимальное количество вершин многоугольника

Type TPoint = record
                 x, y : Integer
              end;
     TVector = TPoint;
//Описание вершин многоугольника
     TPolygon = array[0..MAXN-1] of TPoint;
```

```
//Вычисляем проекцию векторного произведения a x b на ось z
Function CrossProduct(const a,b : TVector) : Int64;
Begin
  Result := a.x * Int64(b.y) - a.y * Int64(b.x)
End;

//Вычисление проекций вектора AB по координатам его концов
Procedure MakeVector(const A,B : TPoint; out AB : TVector);
Begin
  AB.x := B.x - A.x;
  AB.y := B.y - A.y
End;

//Вычисление площади многоугольника
//n - количество углов
Function Area(const a : TPolygon; n : Integer) : Real;
Var v0, v : TVector;
    i     : Integer;
    s     : Int64;
Begin
  MakeVector(a[0],a[1],v0);
  s:=0;
  For i:=1 To n-2 Do
  Begin
    MakeVector(a[0],a[i+1],v);
    s:=s+CrossProduct(v0,v);
    v0:=v
  End;
  Area := abs(s)/2
End;

//-------------------------------------------------------------

Var p    : TPolygon;
    f    : Text;
    n    : Integer; //Количество вершин многоугольника
    i    : Integer;
    s    : Real;
Begin
  Assign(f,'input.txt');
  Reset(f);
  Read(f,n);
  For i:=0 To n-1 Do
  Read(f,p[i].x,p[i].y);
  Close(f);

  s:=Area(p,n);

  Assign(f,'output.txt');
  Rewrite(f);
  Write(f,s:0:1);
  Close(f)
End.
```

Заметим, что, используя ориентированную площадь многоугольника, можно ответить и на вопрос задачи 1 о направлении обхода. Если вектор ориентированной площади направлен к нам, то обход вершин многоугольника осуществлялся против часовой стрелки.

Задания для самостоятельного решения

1. Чем объяснить различие в реализации функции CrossProduct в решениях задач 1 и 2?

2. В задаче 2 из текста занятия проводить векторы к вершинам многоугольника, образуя треугольники, можно не обязательно из какой-либо вершины этого многоугольника. Для этого достаточно использовать любую точку плоскости. Решите задачу 2, проводя векторы из начала координат.
3. В решении задачи 2 из текста занятия используется массив вершин многоугольника. Он занимает довольно много памяти. Напишите программу решения той же задачи без использования массивов.
4. Координаты вершин простого многоугольника заданы в порядке их обхода в одном из двух возможных направлений: по или против часовой стрелки. Определить, является ли он выпуклым. Все координаты вершин — целые числа, по модулю не превышающие 10^4. Количество вершин многоугольника $3 \leq n \leq 10^6$ задано в первой строке входного файла. В следующих n строках этого файла заданы через пробел координаты вершин x и y по одной вершине в каждой строке.
   Указание. Если многоугольник выпуклый, то при движении вдоль его границы мы каждый раз будем поворачивать в одном направлении: либо направо, либо налево.