Занятие 23. Задача «Треугольники»

На этом занятии мы разберём решение олимпиадной задачи по геометрии. Однако, сначала вспомним некоторые сведения из математики.

Определение 1. Средней линией треугольника называется отрезок, соединяющий середины двух его сторон.

Теорема 1. *Средняя линия трегольника, соединяющая середины двух сторон треугольника, параллельна его третьей стороне и равна её половине.*

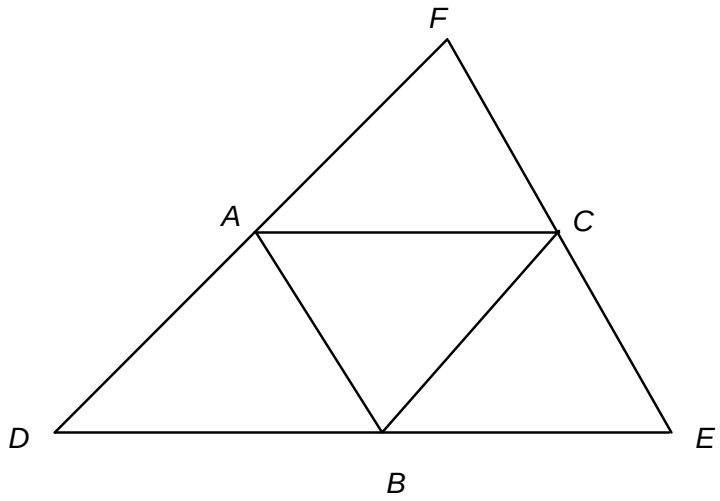
Определение 2. Радиусом-вектором точки называется вектор, проведённый в эту точку из начала координат.

Теорема 2. *Проекции радиуса-вектора точки на оси координат равны координатам этой точки.*

1 Условие задачи «Треугольники»

На седьмой индивидуальной олимпиаде цикла Интернет-олимпиад для школьников сезона 2009–2010 года была предложена следующая

Задача. Недавно Вася объяснил своему младшему брату Пете основы работы с циркулем и линейкой. В качестве примера он показал, как по данному треугольнику построить такой, что середины его сторон совпадают с вершинами первого. Такое действие он назвал «расширением» треугольника.



Например, на рисунке треугольник DEF является расширением треугольника ABC.

Для закрепления материала Вася решил дать брату какое-нибудь простое задание на точность построения геометрических фигур. Немного поразмыслив, он нарисовал на листе бумаги n треугольников и назвал «шагом» однократное расширение всех треугольников. После этого он попросил Петю узнать, через какое число шагов найдется пара пересекающихся треугольников.

Два треугольника пересекаются, если существует точка, лежащая внутри или на границе каждого треугольника. В частности, если один треугольник находится внутри другого, то такие треугольники пересекаются.

Петя слушал умного брата не очень внимательно, поэтому с первого раза у него ничего не вышло. Тогда Вася заготовил ему еще несколько примеров, а для того, чтобы ему было проще проверять, попросил вас написать программу, которая по заданным треугольникам выводит ответ на задачу.

Формат входного файла

Первая строка входного файла triangles.in содержит натуральное число n — количество треугольников ($2 \le n \le 50$). Следующие n строк содержат по шесть целых чисел — $x_1, y_1, x_2, y_2, x_3, y_3$, разделенных пробелами. Эти величины задают треугольник с вершинами в точках (x_1, y_1), (x_2, y_2) и (x_3, y_3). Все числа во входном файле не превышают по модулю 10^8.

Формат выходного файла

В первой строке выходного файла triangles.out выведите одно целое число — через какое число шагов найдется пара пересекающихся треугольников.

Примеры

triangles.in	triangles.out
2 0 0 1 0 0 1 0 0 -1 0 0 -1	0
2 0 0 1 0 0 1 -1 -1 -1 3 3 -1	0
2 10 2 5 9 3 5 -9 0 0 -9 -9 -5	3

2 Решение

2.1 Расширение треугольника

Заметим, что согласно определению 1 стороны исходного треугольника являются средними линиями сторон расширенного треугольника. Из теоремы 1 следует существование трёх пар параллельных отрезков: $AB \parallel EF$, $BC \parallel DF$ и $AC \parallel DE$. Кроме того, согласно той же теореме, длины отрезков в каждой паре различаются ровно в 2 раза:

$$EF = 2AB$$
$$DF = 2BC$$
$$DE = 2AC$$

Это значит, что точку E можно получить, прикладывая вектор $\overrightarrow{AB}$ к точке C, точку F — вектор $\overrightarrow{BA}$ к точке C и т.д.

Чтобы определить координаты вершин расширенного треугольника, будем рассматривать каждую точку как конец вектора, проведённого из начала координат, т.е. радиуса-вектора. Обозначим начало координат через O. Тогда

$$\overrightarrow{OE} = \overrightarrow{OC} + \overrightarrow{AB} \quad (1)$$

$$\overrightarrow{OF} = \overrightarrow{OC} - \overrightarrow{AB} \quad (2)$$

$$\overrightarrow{OD} = \overrightarrow{OB} + \overrightarrow{CA} \tag{3}$$

Далее по теореме 2 можно найти координаты точек D, E, F.

2.2 Определение пересечения треугольников

2.2.1 Достаточные условия пересечения

Рассмотрим множество T из n треугольников, каждый из которых характеризуется координатами точек A, B и C.

Треугольники T_i и T_j пересекаются, если выполняется одно из условий:

1. одна из сторон треугольника T_i пересекает какую-либо сторону треугольника T_j;
2. одна из вершин треугольника T_i лежит внутри треугольника T_j;
3. одна из вершин треугольника T_j лежит внутри треугольника T_i.

Если условие 1 ложно, то условие 2 говорит о том, что треугольник T_i лежит внутри T_j, а условие 3 — что T_j находится внутри T_i.

Проверка условия 1 сводится к задаче 3 из текста занятия 20.

2.2.2 Определение принадлежности точки треугольнику

Проверка условий 2 и 3 может быть выполнена с использованием решения задачи 6 к занятию 20, т.к. треугольник есть частный случай выпуклого многоугольника.

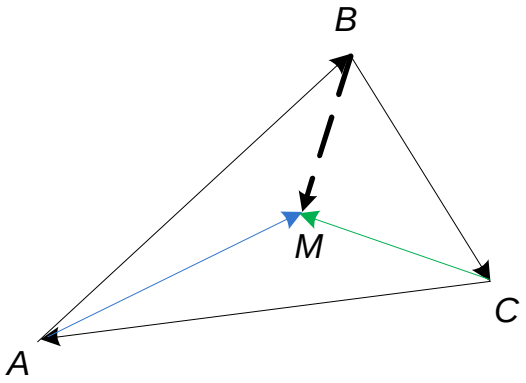


Рис. 1: Точка внутри треугольника

Если точка M принадлежит внутренней области треугольника, то при обходе треугольника вдоль границы по часовой стрелке точка M будет оставаться или всё время справа от границы (см. рис. 1), а при обходе против часовой стрелки — всё время слева. Это означает сонаправленность векторов $\overrightarrow{AB} \times \overrightarrow{AM}$, $\overrightarrow{BC} \times \overrightarrow{BM}$, $\overrightarrow{CA} \times \overrightarrow{CM}$. В принципе этого достаточно, если учесть, что случай, когда точка попадает на границу треугольника будет учтён при обработке условия 1.

Небольшая доработка этого алгоритма позволяет учесть и случай, когда точка попадает на границу треугольника. Нужно просто проследить, чтобы среди упомянутых выше векторных произведений не было противоположно направленных. Тогда точка принадлежит треугольнику: его внутренней области или границе.

2.3 Набросок алгоритма решения задачи «Треугольники»

1. Количество расширений $cnt := -1$
2. $IsInt := false$ //Пересечений треугольников не найдено
3. Пока $IsInt = false$, повторять
   (a) $i := 1$
   (b) Пока $i < n$ и $IsInt = false$, повторять
      i. $j = i + 1$
      ii. Пока $j \leq n$ и $IsInt = false$, повторять
         A. $IsInt$:=треугольник T_i пересекается с треугольником T_j
         B. $j := j + 1$
      iii. $i := i + 1$
   (c) $cnt := cnt + 1$
   (d) Расширить все треугольники
4. Вывести cnt

Уточним пункт 3d. После расширения сведения о «старых» треугольниках нам будут уже не нужны. Поэтому «расширяться» будем на месте. То есть расширенный треугольник будет храниться на том же самом месте в оперативной памяти, что и исходный. Этого можно добиться, если после расширения треугольника ABC в треугольник DEF выполнить замену $A := E$, $B := F$, $C := D$

2.4 Программа

Программа решения задачи представлена в приложении. Но лучше сейчас остановиться и не читать текст приложения до тех пор, пока не напишете программу сами. Или, по меньшей мере, не приложите максимум усилий к её написанию.

Задания для самостоятельной работы

1. Напишите программу решения задачи «Треугольники». В этом вам помогут подпрограммы, написанные на предыдущих занятиях.
2. Проверьте вашу программу при помощи тестов, скачанных с сервера ИТМО (См. [2] в списке литературы). Тесты к этой задаче находятся в папке triangles. Файлы без расширений — это входные файлы, которые можно переименовывать в triangles.in при тестировании. Файл с тем же именем (номером) и расширением *a* содержит правильный ответ.

Список литературы

[1] Условия задач седьмой индивидуальной олимпиады цикла интернет-олимпиад для школьников сезона 2009–2010 года (20 марта 2010 года) http://neerc.ifmo.ru/school/io/archive/20100320/problems-individual-20100320.pdf

[2] Тесты и решения жюри задач седьмой интернет олимпиады школьников http://neerc.ifmo.ru/school/io/archive/20100320/archive-individual-20100320.rar

[3] Michaël Van Canneyt. Free Pascal version 2.4.2: Reference guide for RTL units. Document version 2.1. November 2010

Приложение. Текст программы решения задачи «Треугольники» на FreePascal

```
{$mode delphi}
Program Triangles;
Uses math;
Const MAXN=50; //Максимальное кол-во треугольников

Type TPoint = record //Точка
                x, y : Integer
              end;
     TVector = TPoint; //Вектор

     Triangle = record //Треугольник
                  A,B,C : TPoint
                end;

     //Массив треугольников
     TrArray = array[1..MAXN] Of Triangle;

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

//Сложение векторов: AC = AB + BC
Procedure SummVector(const AB, BC : TVector; out AC : TVector);
Begin
  AC.x:=AB.x+BC.x;
  AC.y:=AB.y+BC.y
End;

//Вычитание векторов: c = a - b
Procedure DiffVector(const a,b : TVector; out c : TVector);
Begin
  c.x:=a.x-b.x;
  c.y:=a.y-b.y
End;

//Пересекает ли прямая AB отрезок CD?
Function LineAndSegment(const A,B,C,D : TPoint) : Boolean;
```

```
Var pz,qz : Int64;
    AB,AC,AD : TVector;
Begin
  MakeVector(A,B,AB);
  MakeVector(A,C,AC);
  pz:=CrossProduct(AB,AC);
  MakeVector(A,D,AD);
  qz:=CrossProduct(AB,AD);
  Result:=sign(pz)*sign(qz)<=0
End;

//Поиск минимального и максимального из 2 чисел
Procedure Sort2(a,b : Integer; out min,max : Integer);
Begin
  min:=a;
  max:=b;

  If a>b Then
  Begin
    max:=a;
    min:=b
  End
End;

//Пересекаются ли ограничивающие прямоугольники отрезков AB и CD?
Function RectIntersect(const A,B,C,D : TPoint) : Boolean;
Var xmin1,ymin1,xmax1,ymax1,xmin2,ymin2,xmax2,ymax2 : Integer;
Begin
  Sort2(A.x,B.x,xmin1,xmax1);
  Sort2(A.y,B.y,ymin1,ymax1);
  Sort2(C.x,D.x,xmin2,xmax2);
  Sort2(C.y,D.y,ymin2,ymax2);
  Result := not((xmin2 > xmax1) or (xmax2 < xmin1) or
            (ymin2 > ymax1) or (ymax2 < ymin1))
end;

//Пересекаются ли отрезки AB и CD?
Function SegmentsIntersect(const A,B,C,D : TPoint) : Boolean;
Begin
  Result := RectIntersect(A,B,C,D) and
            LineAndSegment(A,B,C,D) and
            LineAndSegment(C,D,A,B)
End;

//Принадлежит ли точка M треугольнику T?
//Работает и для случая точки на границе
Function PointInTriangle(const M : TPoint; const T : Triangle) : Boolean;
Var
    //Количество векторных произведений отрицательных и положительных
    c       : array[1..2] of ShortInt = (0,0);
```

```
    vt, toM : array[1..3] of TVector;
    i       : ShortInt;
Begin
  MakeVector(T.A,T.B,vt[1]);
  MakeVector(T.B,T.C,vt[2]);
  MakeVector(T.C,T.A,vt[3]);
  MakeVector(T.A,M,toM[1]);
  MakeVector(T.B,M,toM[2]);
  MakeVector(T.C,M,toM[3]);

  For i:=1 To 3 Do
  If CrossProduct(vt[i],toM[i]) < 0 Then
    c[1]:=c[1]+1
  Else
    c[2]:=c[2]+1;

  Result := c[1]*c[2] = 0
End;

//Пересекает ли отрезок p1p2 стороны треугольника T
Function TrIntersect(const p1,p2 : Tpoint; const T : Triangle) : Boolean;
Begin
  Result := SegmentsIntersect(T.A,T.B,p1,p2) or
            SegmentsIntersect(T.A,T.C,p1,p2) or
            SegmentsIntersect(T.B,T.C,p1,p2)
End;

//Расширение треугольника
Procedure Expansion(Var t : Triangle);
Var AB,CA : TVector;
    F     : TPoint;
Begin
  MakeVector(t.A,t.B,AB);
  MakeVector(t.C,t.A,CA);
  SummVector(t.C,AB,t.A);
  DiffVector(t.C,AB,F);
  SummVector(t.B,CA,t.C);
  t.B:=F
End;

//Собственно решение. Подсчитываем количество расширений
Function Solve(Var T : TrArray; n : Integer) : Integer;
Var cnt : Integer = -1;
    i,j : Integer;
    IsInt : Boolean = false;
Begin
  While not IsInt Do
  Begin
    i:=1;
    While (i<n) and not IsInt Do
```

```
    Begin
      j:=i+1;
      While (j<=n) and not IsInt Do
      Begin
        IsInt := TrIntersect(T[j].A,T[j].B,T[i]) or
                 TrIntersect(T[j].B,T[j].C,T[i]) or
                 TrIntersect(T[j].A,T[j].C,T[i]) or
                 PointInTriangle(T[i].A,T[j]) or
                 PointInTriangle(T[j].A,T[i]);
        j:=j+1
      End;
      i:=i+1
    End;
    cnt:=cnt+1;
    For i:=1 To n Do
    Expansion(T[i]);
  End;
  Result:=cnt
End;


//-------------------------------------------------------------
Var f          : Text;
    i,n        : ShortInt;
    t          : TrArray; //Массив треугольников
    cnt        : Integer; //Количество расширений
Begin
  Assign(f,'triangles.in');
  Reset(f);
  Read(f,n);
  For i:=1 To n Do
   Read(f,t[i].A.x,t[i].A.y,t[i].B.x,
          t[i].B.y,t[i].C.x,t[i].C.y);
  Close(f);

  cnt:=Solve(t,n);

  Assign(f,'triangles.out');
  Rewrite(f);
  Write(f,cnt);
  Close(f)
End.
```

В тексте программы используется функция $\mathrm{sign}(x)$ (читается «сигнум икс»). Эта математическая функция показывает знак числа x

$$\mathrm{sign}(x) = \begin{cases} 1 & \text{при } x > 0 \\ 0 & \text{при } x = 0 \\ -1 & \text{при } x < 0 \end{cases}$$

Подпрограмму-функцию для вычисления $\mathrm{sign}(x)$ легко реализовать самостоятельно. Но она есть в стандартной библиотеке Delphi и FreePascal в модуле math[1]. Чтобы подключить к своей программе этот модуль после имени программы пишем

```
Uses math;
```

Казалось бы, что в функции LineAndSegment можно было вместо

```
Result:=sign(pz)*sign(qz)<0
```

написать просто

```
Result:=pz*qz<0
```

Однако это может привести к ошибке. Учтём, что pz и qz — проекции векторных произведений векторов, координаты которых могут быть достаточно велики. Тогда величина pz*qz может стать столь большой, что для её хранения не подойдёт даже тип Int64.

Действительно, координаты точек по условию задачи могут быть от -10^8 до 10^8. Значит, проекции векторов $(x_2 - x_1, y_2 - y_1)$ могут быть до $2 \cdot 10^8$. Тогда проекция векторного произведения $a_x b_y - a_y b_x$ может превысить $\left(2 \cdot 10^8\right)^2 = 4 \cdot 10^{16}$, а произведение проекций стать больше квадрата этого числа, т.е. $1,6 \cdot 10^{33}$.

Применение функции $\mathrm{sign}(x)$ позволяет избежать ошибки переполнения в этом случае.

И ещё одно пояснение. В этой программе использовалась инициализация переменных при описании. Например, запись

```
Var cnt : Integer = -1;
```

значит, что объявляется переменная целого типа cnt и ей присваивается начальное значение -1. С аналогичным приёмом для массивов мы уже встречались в занятии 6.

Обратите внимание, что в Delphi инициализировать при описании можно только глобальные переменные или переменные основного алгоритма. FreePascal допускает инициализацию при описании и для локальных переменных. Если вы хотите скомпилировать текст этой программы на Delphi, вам надо его немного подправить, заменив в нескольких местах инициализацию при описании обычным описанием с последующей инициализацией. Например, это может выглядеть так:

```
Var cnt : Integer;
Begin
  cnt:=-1;
  ...
End;
```

[1] Смотрите [3, стр. 771] в списке литературы