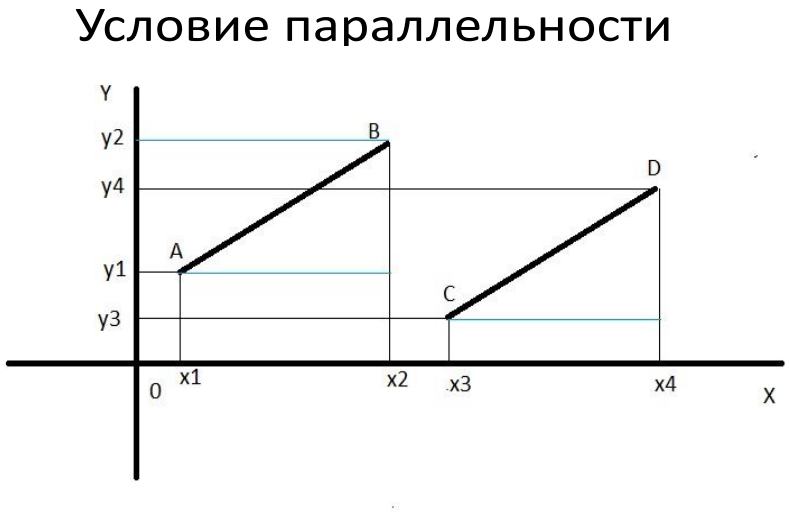
Алгоритм решения задачи фактически изложен в условии задачи, но не совсем понятно, что значит прямые параллельны и взаимно перпендикулярны. Мало кто из учеников знает условия коллинеарности и ортогональности векторов и кажется задача для них неразрешима. Однако посмотрим на это с точки зрения ученика знакомого с подобием треугольников

Пусть координаты точек A {x1,y1},. B{x2,y2}, C{x3,y3}, D{x4,y4}.

Две прямые параллельны, если у них одинаковые углы с осью OX или, что почти то же самое, одинаковые тангенсы этих углов, или же, что то же самое, треугольники образованные отрезками прямых и их проекциями на оси координат подобны. Последнее понятно даже семикласснику, только начавшему изучать планиметрию.



Условие параллельности

k1=(Y2-y1)/(x2-x1) ;

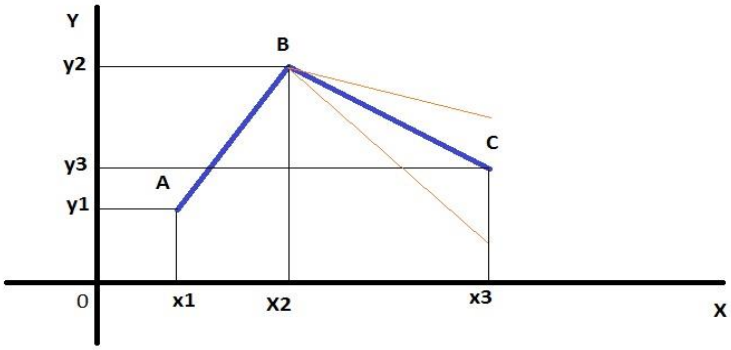
k2=(y4-y3)/(x4-x3);

K1=k2;

(Y2-y1)/(x2-x1) =(y4-y3)/(x4-x3)

Чтобы избежать деления, лучше последнее равенство привести к общему знаменателю:

(Y2-Y1)(X4-X3)-(Y4-Y3)*(X2-X1)=0.*



Если два отрезка выходят из одной точки B(x2;y2), то,

k1=(Y2-y1)/(x2-x1);

k2=(y3-y2)/(x3-x2).

Если K1>k2, то

(Y2-y1)/(x3-x2) >(y3-y2)/(x2-x1) или

(Y2-Y1)(X3-X1)-(Y3-Y2)*(X2-X1)>0.*

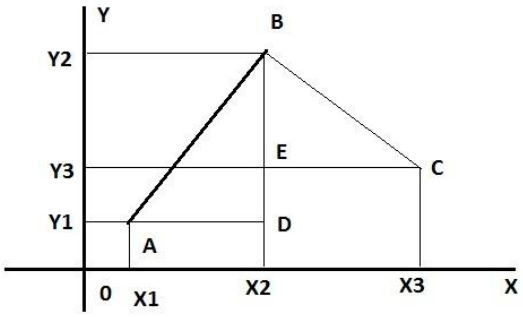
Т.е. первая прямая лежит левее (выше) второй.

Для выпуклости многоугольника необходимо, чтобы все k были одного знака при обходе вершин многоугольника подряд в одну сторону.

Выпуклость можно проверить и другими способами.

Например, проверить лежат ли все точки многоугольника по одну сторону от прямых, проходящих через стороны многоугольника. Но это существенно сложнее.

Условие перпендикулярности (ортогональности)



Пусть даны два взаимно-перпендикулярных отрезка прямых AB и BC. из подобия треугольников ABD и BCE получаем,

что BD/AD=CE/BE или (y2-y1)/(x2-x1)=(x3-x2)/(y2-y3).

Это соотношение не совсем очевидное и можно запутатться в выборе начальных и конечных точек

(С точки зрения аналитической геометрии вывод проще, но ек все его знают:

y=k1*x+b1; и y=k2*x+b2.

Тогда k2 = -1/k1 или

(y2-y1)/(x2-X1)=-(x3-x2)/(y3-y2));

(y2-y1)*(y3-y1)+(x2-X1)*(x3-x2)=0;

Анализируя теорему Пифагора получаем. Если выражение $L=Sqrt((x2-x1)^2+(x3-x2))$ меньше нуля, то угол между прямыми AB и BC острый, больше – тупой.

Если стороны взаимнопересекаются, то тогда для одной из двух смежных сторон, две другие вершины должны быть по разные стороны, т.е.

(((y2-y1)*(y3-y1)-(x2-x1)*(x3-x1))*((y2-y1)*(y4-y1)-(x2-x1)*(x4-x1))<0) and

(((y4-y3)*(y1-y3)-(x4-x3)*(x1-x3))*((y4-y3)*(y2-y3)-(x4-x3)*(x2-x3))<0)

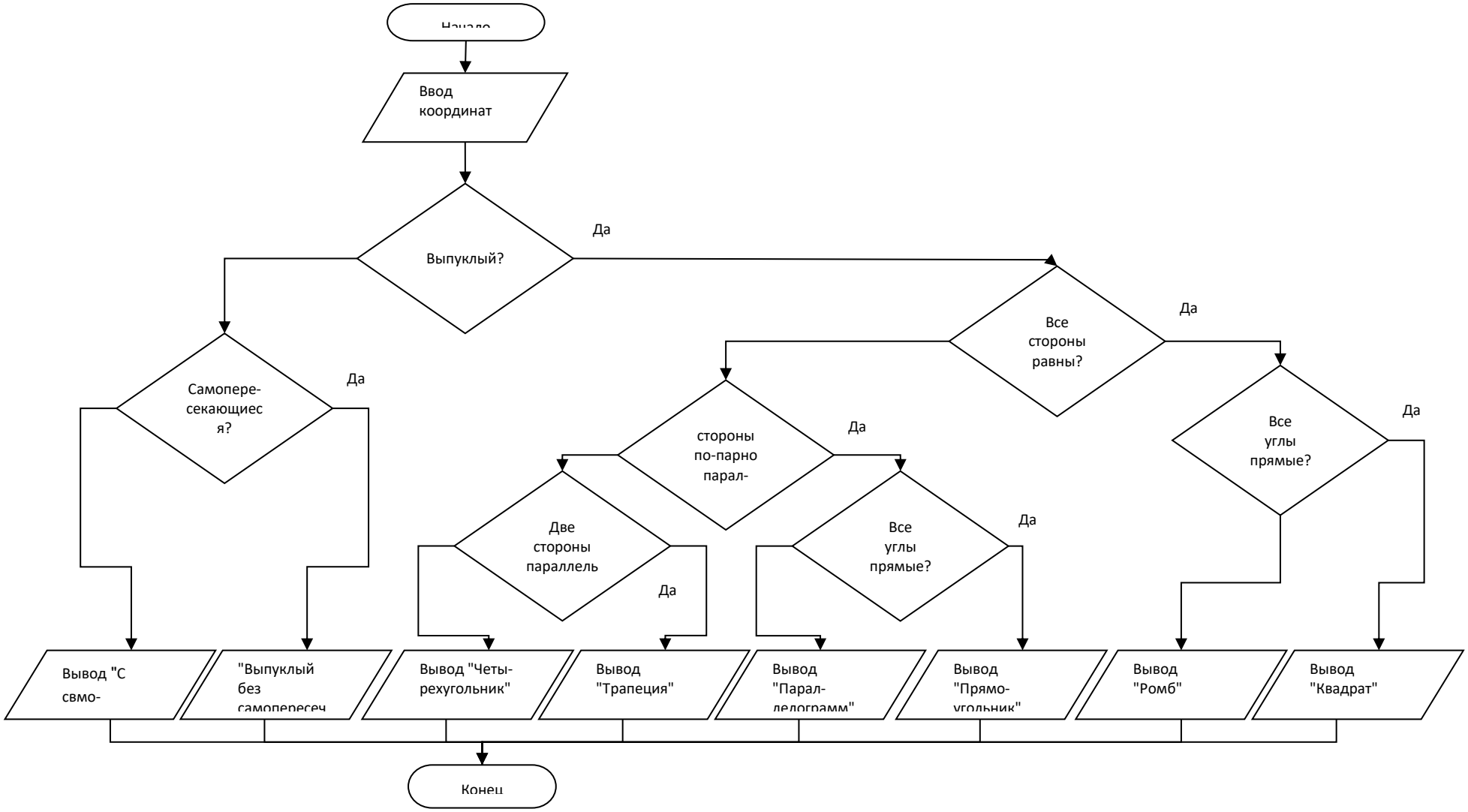
Изложенного вполне достаточно чтобы решить задачу.

Один из вариантов алгоритма решения этой задачи представлен на блок-схеме и приведен текст программы на языке Pascal. Программы представлены на языке Pascal, в расчете на то, что большинство школ выбрало этот язык в качестве базового для обучения.

На основании изложенного можно сделать следующие выводы.

- Не всякий первый найденные алгоритм решения может удовлетворять заданным ограничениям.
- Чем проще алгоритм, тем, в общем случае, дольше решение.
- Нужно помнить, что, как правило, лучший алгоритм - это компромисс между сложностью, объемом требуемой памяти и быстродействием.

Блок-схема


Начало
Ввод координат
Выпуклый?
Да
Самопере-секающиеся?
Да
Все стороны равны?
Да
стороны по-парно парал-
Да
Две стороны параллель
Да
Все углы прямые?
Да
Все углы прямые?
Да
Вывод "С свмо-
"Выпуклый без самопересеч
Вывод "Четы-рехугольник"
Вывод "Трапеция"
Вывод "Парал-делограмм"
Вывод "Прямо-угольник"
Вывод "Ромб"
Вывод "Квадрат"
Конец

```
uses CRT;
  var
     X1,Y1,X2,Y2,X3,Y3,X4,Y4        :Real;
  Function ravni(X1,Y1,X2,Y2,X3,Y3,X4,Y4:real):Boolean;
   begin   {длины равны}
    ravni :=False;
    If (y2-y1)*(y2-y1)+(x2-x1)*(x2-x1)=(y4-y3)*(y4-y3)+(x4-x3)*(x4-
x3)
      then  ravni :=True;
   end;
  function Parall (X1,Y1,X2,Y2,X3,Y3,X4,Y4:real):Boolean;
   begin          { стороны парал лл ены}
    Parall :=False;
    If (y2-y1)*(x4-x3)=(y4-y3)*(x2-x1)
     then  Parall :=True;
   end;
  function  Gr90  (X1,Y1,X2,Y2,X3,Y3,X4,Y4:real):Boolean;  {Все
углы прямые}
   begin
    Gr90:=True;
    If (y2-y1)*(y3-y2)<>(x2-x1)*(x3-x2)  then Gr90:=False;
    If (y3-y2)*(y4-y3)<>(x3-x2)*(x4-x3)  then Gr90:=False;
    If (y4-y3)*(y1-y4)<>(x4-x3)*(x1-x4)  then Gr90:=False;
{     If (y2-y1)*(y1-y4)<>(x2-x1)*(x1-x4)  then Gr90:=False;}
   end;
  function       Vipuk       (X1,Y1,X2,Y2,X3,Y3,X4,Y4:real):Boolean;
{Выпуклый}
   const   np:integer=0;
           no:integer=0;
   begin
    Vipuk:=False;
    If (y2-y1)*(x3-x2)>(y3-y2)*(x2-x1)  then inc(no) else inc(np);
    If (y3-y2)*(x4-x3)>(y4-y3)*(x3-x2)  then inc(no) else inc(np);
    If (y4-y3)*(x1-x4)>(y1-y4)*(x4-x3)  then inc(no) else inc(np);
    If (y1-y4)*(x2-x1)>(y2-y1)*(x1-x4)  then inc(no) else inc(np);
    if (np=4) or (no=4) then Vipuk:=True
   end;
  function       Peres       (X1,Y1,X2,Y2,X3,Y3,X4,Y4:real):Boolean;
{Пересечечение}
   begin
    peres:=False;
     If       (((y2-y1)*(y3-y1)-(x2-x1)*(x3-x1))*((y2-y1)*(y4-y1)-(x2-
x1)*(x4-x1))<0) and
       (((y4-y3)*(y1-y3)-(x4-x3)*(x1-x3))*((y4-y3)*(y2-y3)-(x4-
x3)*(x2-x3))<0)
```

```
      then peres:= true;
    end;
(*************************************)
begin
  clrscr;
 { readln(X1,Y1,X2,Y2,X3,Y3,X4,Y4);}
 x1:=0;
 y1:=0;
 x2:=3;
 y2:=3;
 x3:=9;
 y3:=3;
 x4:=18;
 y4:=0;
  if Vipuk (X1,Y1,X2,Y2,X3,Y3,X4,Y4)
    then
     begin
       if (ravni(X1,Y1,X2,Y2,X3,Y3,X4,Y4)) and
          (ravni(X1,Y1,X4,Y4,X2,Y2,X3,Y3)) and
          (ravni(X1,Y1,X2,Y2,X3,Y3,X2,Y2)) {and
          (ravni(X1,Y1,X2,Y2,X3,Y3,X4,Y4))}
         then
           if Gr90 (X1,Y1,X2,Y2,X3,Y3,X4,Y4)
             then  writeln('square')
             else  writeln('rhombus')
         else
           if Parall (X1,Y1,X2,Y2,X3,Y3,X4,Y4) and
                Parall (X2,Y2,X3,Y3,X4,Y4,X1,Y1)
              then
                if Gr90 (X1,Y1,X2,Y2,X3,Y3,X4,Y4)
                  then  writeln('rectangle')
                  else  writeln('parallelogram')
              else
           if
             Parall (X1,Y1,X2,Y2,X3,Y3,X4,Y4) or
                Parall (X2,Y2,X3,Y3,X4,Y4,X1,Y1)
             then  writeln('trapezoid')
             else  writeln('convex polygon')
     end
    else
     if Peres (X1,Y1,X2,Y2,X3,Y3,X4,Y4)
        then
          writeln('selfintersectingpolyline')
        else
           writeln('non-convex polygon');
```

```
  readkey;
end.
```