

```
1  {
2    Ractangle.pas
3
4    Необходимо по заданным пользователем координатам 4х точек на плоскости
5    * определить, являются ли они вершинами прямоугольника.
6
7    Замечания
8    * Будем считать, что порядок ввода точек A1, A2, A3, A4 определяет
9    * порядок их соединения отрезками.
10   * Коль пользователь перепутал пару вершин,
11   * будем считать, что он ввёл "песочные часы" ,а не 4х-угольник.
12
13   Вспоминаем свойства (признаки) прямоугольника.
14   * Будем проверять углы при вершинах. Точнее, пользуясь определением
15   * скалярного произведения проверим косинусы углов при вершинах
16   * на равенство 0. Если он 0, то угол равен +-90 градусов
17
18   А также проверяем равенство двух пар противоположных сторон,
19   * хотя это и необязательно.
20
21   Это некий шаблон без изысков в оформлении и
22   * дополнительных проверок при вводе данных
23 }
24
25
26 program Rectangle_Simple;
27 {Прямоугольник}
28
29 uses math, crt;
30
31 {координаты на плоскости точек и векторов}
32 type PlaneCoord=Array[1..2] of Real;
33
34 var
35     Ak: Array [1..4] of PlaneCoord; {вершины}
36     fi_cos: Array[1..4] of Real; {косинусы углов при вершинах}
37
38     v1, v2 :PlaneCoord; {Вспомогательные величины координаты векторов,
39     исходящих из обрабатываемой вершины. В принципе можно и без них
40     * всё считать через координаты концов отрезков}
41
42     i      :Integer; {переменная счетчик вершин}
43     pk, nk :Integer; {переменные номера предыдущей и следующих вершин}
44
45     crit   :boolean; {Флаг выполнения признаков прямоугольника
46                       * выполнены все (true)
47                       * Хотя бы один не выполнен (false)}
```

```
52
53 { процедура вычисления координат вектора на плоскости по координатам
54 * точек начала и конца }
55
56 procedure VecCoord(B1, B2:PlaneCoord; var RES: PlaneCoord );
57
58     begin {Тело процедуры}
59         RES[1]:=B2[1]-B1[1];
60         RES[2]:=B2[2]-B1[2];
61     end ;
62
63
64 { Функция вычисления скалярного произведения векторов на плоскости
65 * по их координатам }
66
67 function ScalMult(B1, B2:PlaneCoord): Real;
68
69     begin {Тело функции }
70         ScalMult:=B1[1]*B2[1]+B1[2]*B2[2];
71     end ;
72
73
74 { функция вычисления модуля вектора на плоскости}
75 function AbsVecPlane(X:PlaneCoord ) : Real ;
76
77     begin {Тело функции}
78         AbsVecPlane:=SQRT(X[1]*X[1]+X[2]*X[2]);
79     end;
80
81 {Функции определения номеров вершин соседних с данной k}
82 {Следующая вершина}
83 function next_vertex(k: integer):integer;
84     begin
85         if k=4
86         then next_vertex:=1
87         else next_vertex:=k+1;
88     end;
89
90 {предыдущая вершина}
91 function prev_vertex(k: integer):integer;
92     begin
93         if k=1
94         then prev_vertex:=4
95         else prev_vertex:=k-1;
96     end;
97
98 BEGIN
99     {-- Запрос данных от пользователя -----}
100     ClrScr(); {чистим экран}
101
102     For i:=1 to 4 DO
```

```
103     begin
104         Writeln('Задайте X координату точки',i);
105         Readln(Ak[i,1]);
106         Writeln('Задайте Y координату точки',i);
107         Readln(Ak[i,2]);
108     end;
109     {^    конец запроса данных от пользователя -----}
110     Writeln(); {пустая строка для отделения результатов}
111
112
113     {-    Подготовка основного цикла-----}
114
115
116     {устанавливаем флаг предполагаем прямоугольник}
117     crit:=true;
118
119     {^    Подготовка основного цикла-----}
120
121
122     {--    Основная часть -----}
123     {переходим от вершины к вершине}
124     For i:=1 to 4 DO
125         begin
126             {находим номера соседних вершин}
127             pk:=prev_vertex(i);
128             nk:=next_vertex(i);
129
130             {"протягиваем" вектора из текущей вершины в соседние}
131             {в предыдущую}
132             VecCoord(Ak[i], Ak[pk], v1 );
133             {в следующую}
134             VecCoord(Ak[i], Ak[nk], v2 );
135
136             {Вычисляем косинус угла между ними}
137             {Фактически, так как нас интересуют 0 можно было}
138             * просто вычислить скалярное произведение векторов и проверить его
139             * на равенство нулю. }
140             fi_cos[i]:=ScalMult(v1, v2)/(AbsVecPlane(v1)*AbsVecPlane(v2));
141
142             {если косинус <>0, то выводим соотв. сообщение }
143             if (fi_cos[i]<>0)
144                 then
145                     begin
146                         Writeln('угол при вершине ', i, ' не прямой!');
147                         crit:=false
148                     end;
149             end;
150
151     {Проверяем равенство противоположных сторон}
152     {A1 A2=A3 A4 ?}
153     VecCoord(Ak[1], Ak[2], v1 );
```

```
154   VecCoord(Ak[3], Ak[4], v2 );
155   if AbsVecPlane(v1)<>AbsVecPlane(v1)
156   then
157       begin
158           Writeln('стороны A1 A2 и A3 A4 не равны');
159           crit:=false
160       end;
161
162   {A2 A3=A1 A4 ?}
163   VecCoord(Ak[2], Ak[3], v1 );
164   VecCoord(Ak[1], Ak[4], v2 );
165   if AbsVecPlane(v1)<>AbsVecPlane(v1)
166   then
167       begin
168           Writeln('стороны A2 A3 и A1 A4 не равны');
169           crit:=false
170       end;
171
172   {подводим итог если все углы прямые и "противоположные стороны равны"
173   * (crit=true), то объявляем прямоугольник }
174   if crit
175   then writeln('Прямоугольник')
176   else writeln('НЕ Прямоугольник!');
177
178   {^ конец основной части -----}
179
180   {v-- Ожидание нажатой клавиши задержка -----v-}
181   Writeln('Работа программы окончена. Нажмите любую клавишу');
182   ReadKey();
183   {^ Ожидание нажатой клавиши -----}
184 END.
185
```