ГРАФИКИ

1. $y = F(x)$
2. Полярные координаты
3. Изолинии поля $z = F(x, y)$

БЛОК-СХЕМА ПОСТРОЕНИЯ КУСОЧНО-ЛИНЕЙНОГО ГРАФИКА ТАБЛИЧНОЙ ФУНКЦИИ

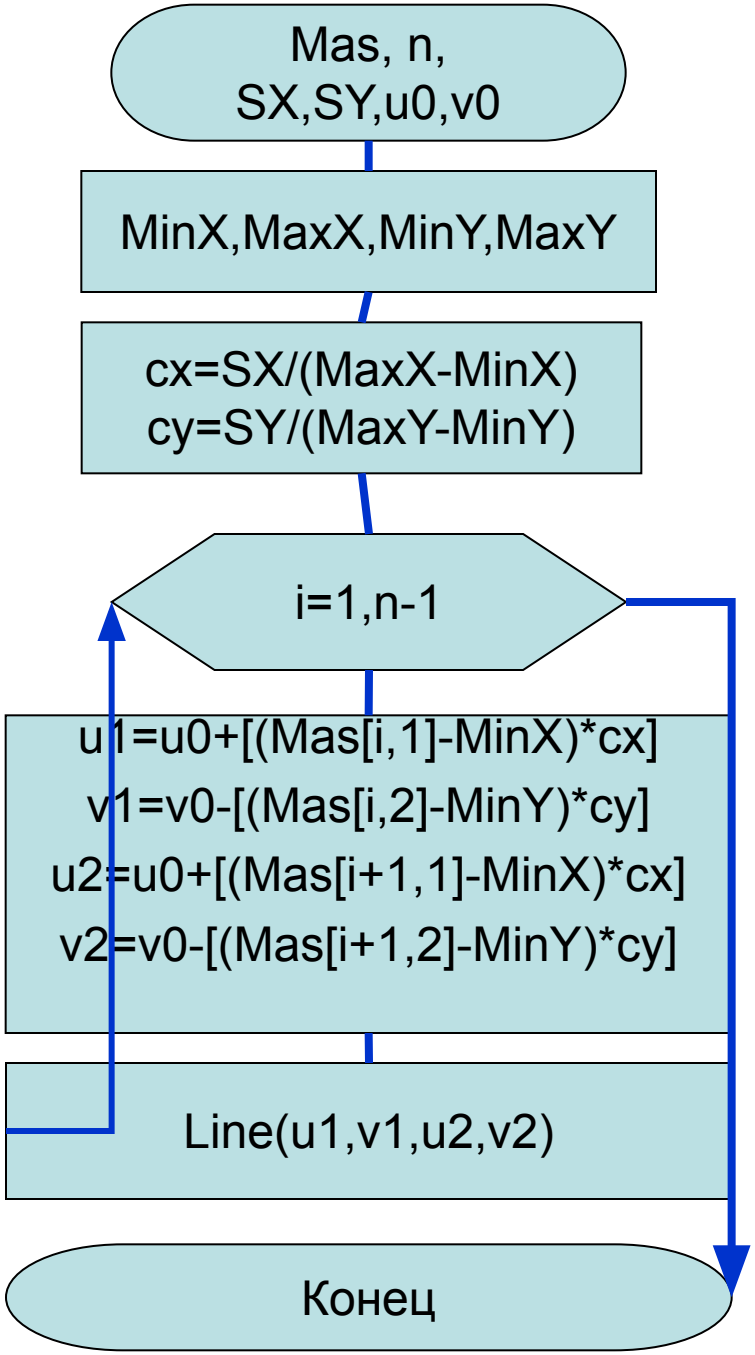


Таблица (массив): Mas[N,2] y = f(x)

Окно построения: (SX, SY, u0, v0)

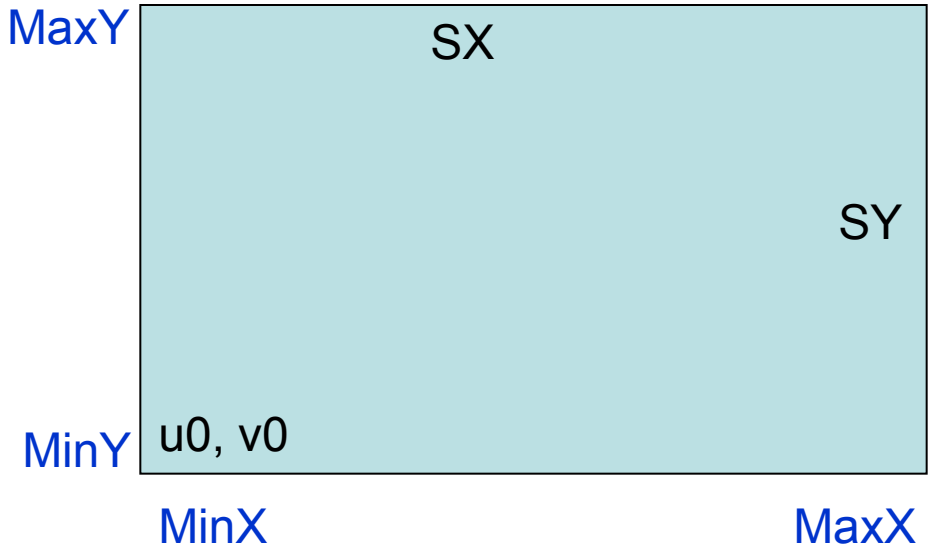
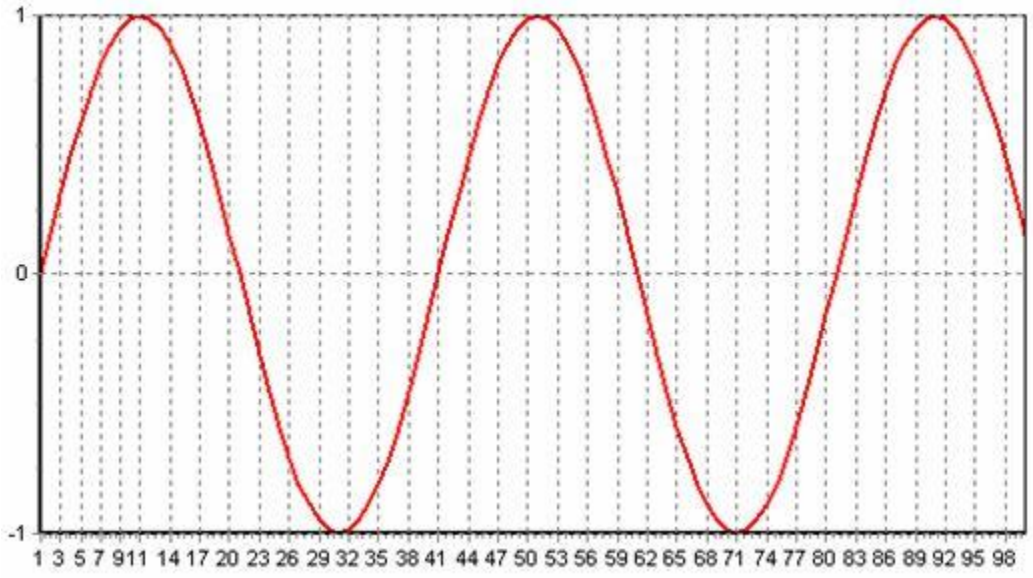
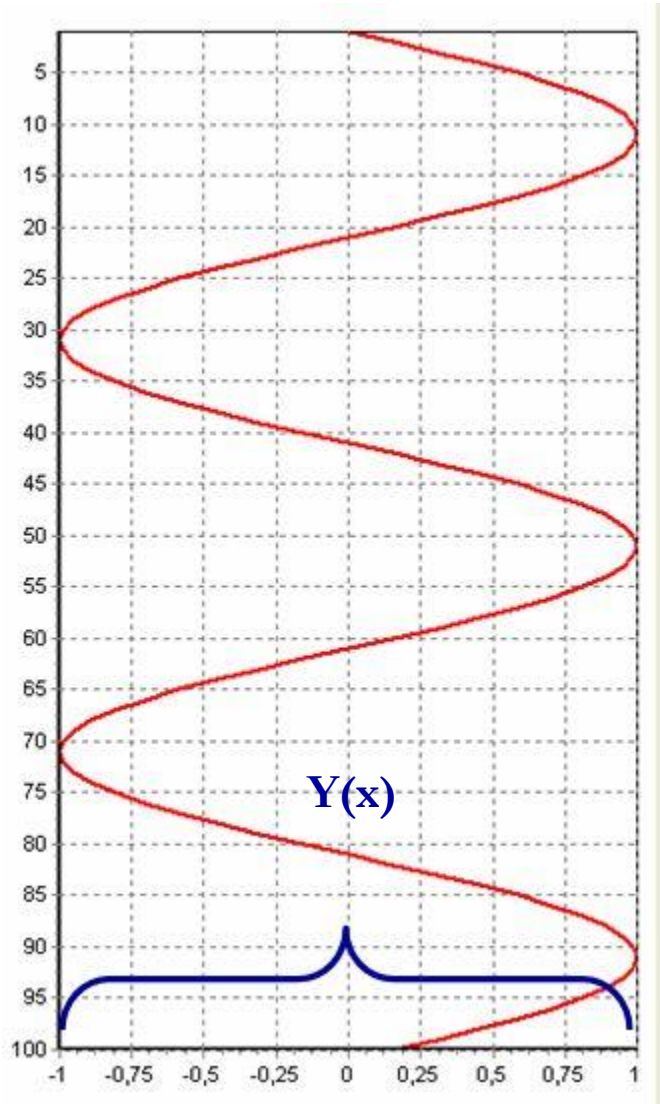


График неоднозначной функции?
?: Поворот графика на 90°



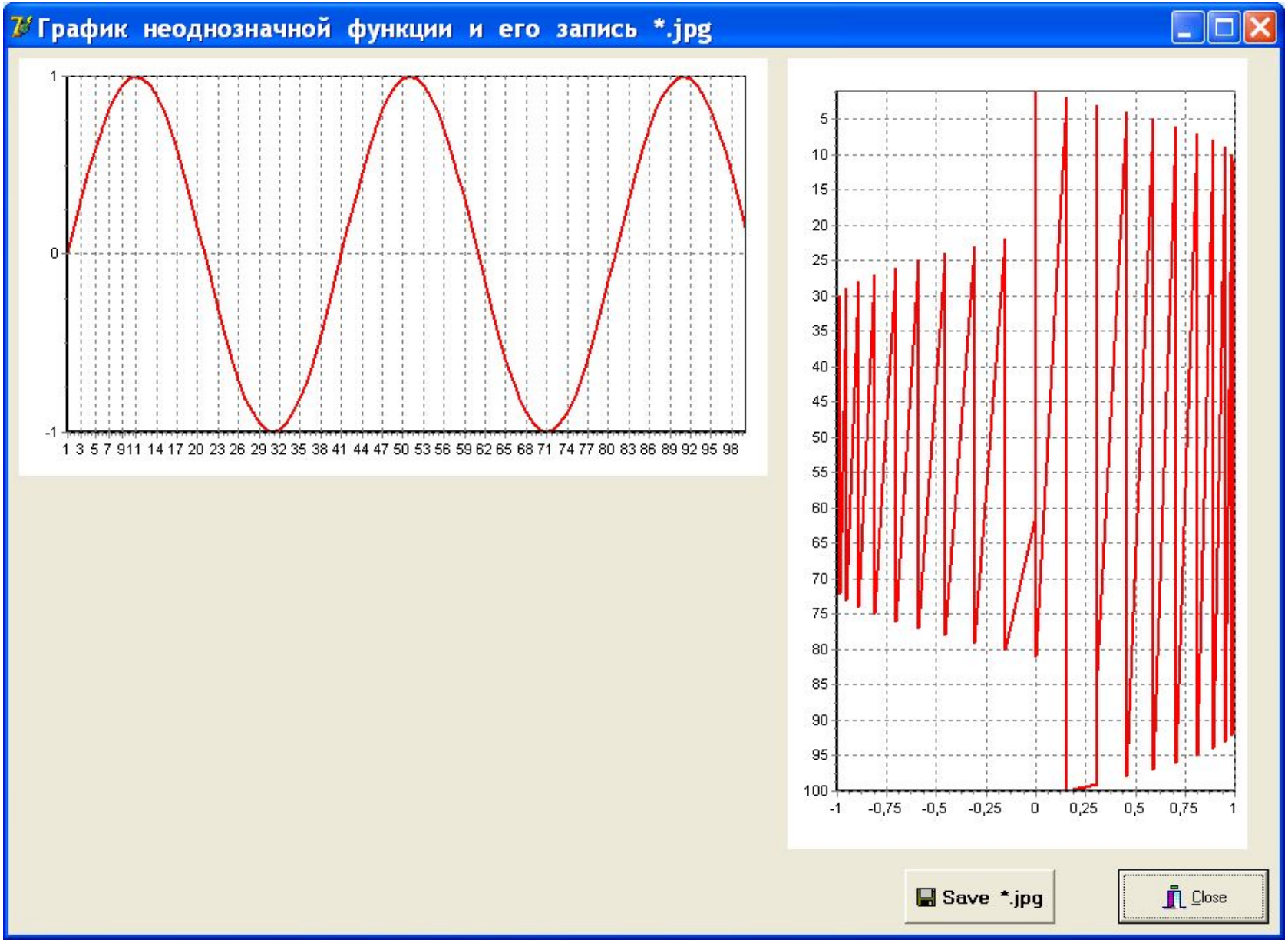


Процедура построения

```
procedure TForm1.FormCreate(Sender: TObject);
   var i:integer;
Begin       FillMas;
   Series1.Clear;   Series2.Clear;
   for i:=1 to 100 do begin
     Series1.Add(Mas[i],IntToStr(i));
     Series2.AddXY(Mas[i],i);
   end;
   with  Series2  do begin
    YValues.Order:=loAscending;
    YValues.Sort;
    Repaint;
  end;
end;
```

Сортировка неоднозначной функции
ЗАЧЕМ?

Выполнение без сортировки



Полярные координаты

Полярные координаты определяются полярным радиусом R и полярным углом F и связаны с прямоугольными координатами X и Y формулами: X=R*CosF, Y=R*SinF. В этих координатах удобно задавать ряд функций, например, таких как параболическая спираль, плоская улитка, розы, синусоидальная спираль, эллипс, гиперболическая спираль, логарифмическая спираль, окружность и т. п.

ИТАК: Полярные координаты

Имеется единственная ось и некоторая точка на ней, называемая ПОЛЮСОМ.

Любую точку на плоскости теперь можно определить парой чисел (R, ϕ),

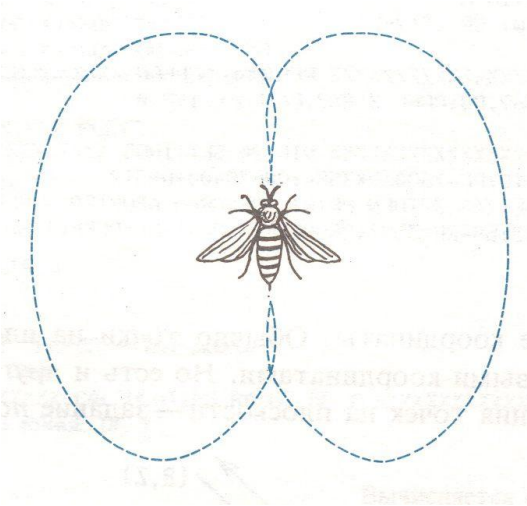
где R – расстояние от полюса и

ϕ - угол между осью и прямой, соединяющей полюс и данную точку (угол изменяется в направлении против часовой стрелки от оси).

Полярные координаты пчелы

Используют для обмена информацией об источниках пищи. Найдя новый источник пищи (клумбу, цветущее дерево,…), пчела-разведчица возвращается в улей, приносит образец и исполняет танец, на языке которого рассказывает, где находится пища. Танец состоит в том, что пчела, покачиваясь с боку на бок, прочерчивает прямую и затем, описав плавную кривую, возвращается в начальную точку. Участок прямой повторяется, но на этот раз она возвращается по кривой в другом направлении.

Весь процесс повторяется несколько раз. Длина отрезка прямой даёт расстояние до пищи (в «пчелиных» единицах), а направление прямой – направление, в котором надо лететь. Таким образом пчела- разведчица сообщает другим пчёлам полярные координаты нового источника пищи.



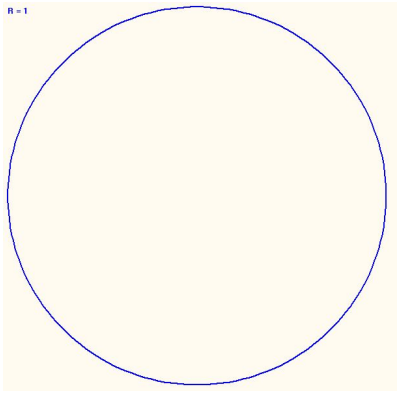
АЛГОРИТМ ПОСТРОЕНИЯ

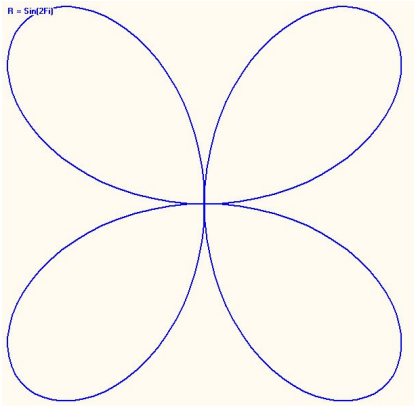
Точка (R, ϕ) в ПК – это то же самое что и точка (RCos(ϕ), RSin(ϕ)) в декартовых координатах.
Задаётся интервал изменения угла [ϕ1, ϕ2]
Задаётся шаг (приращение) угла Δϕ
В цикле для ϕ от ϕ1 до ϕ2 шагом Δϕ вычисляются значения R = f(ϕ).
Определяются декартовые составляющие, которые собираются в массивы для последующего построения кусочно-линейного графика в заданном окне (SX, SY, u0, v0).
При масштабировании изображения для сохранения угловых соотношений следует выбрать одинаковые коэффициенты перехода от физических величин к экранным
(Если cx = SX/(Max_X-Min_X); cy = SY/(Max_Y – Min_Y), то выбирается Min(cx, cy) для обеих осей)

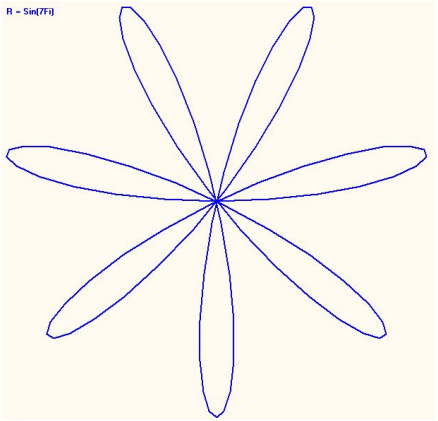
ФУНКЦИИ В ПОЛЯРНЫХ КООРДИНАТАХ

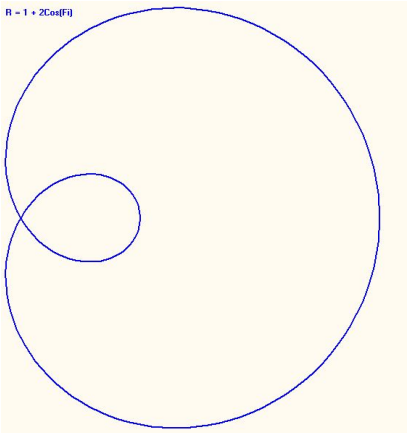
$R = 1$	Окружность
$R = \sin(2\phi)$	Четырёхлепестковая роза
$R = \sin(7\phi)$	Семилепестковая роза
$R = 1 + 2\cos(\phi)$	Улитка Паскаля
$R = 1 + \cos(\phi)$	Кардиоида
$R = \phi/4$	Спираль
$R = 1 + \sin(2\phi)$	Двухлепестковая роза
$R = 1 + 2\cos(2\phi)$	Петельное сцепление

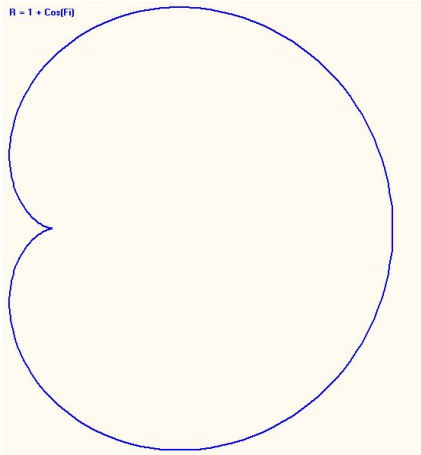
Результаты построения

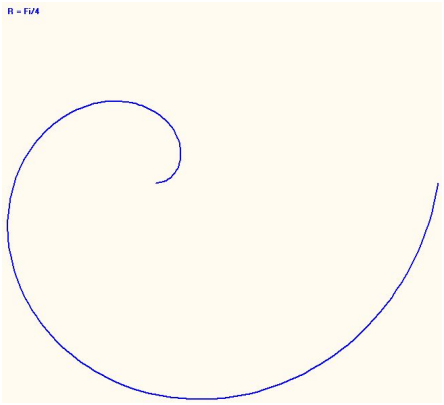


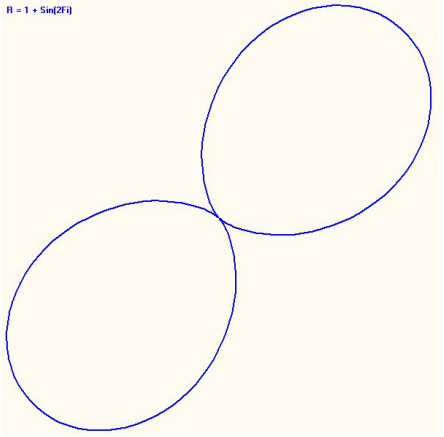


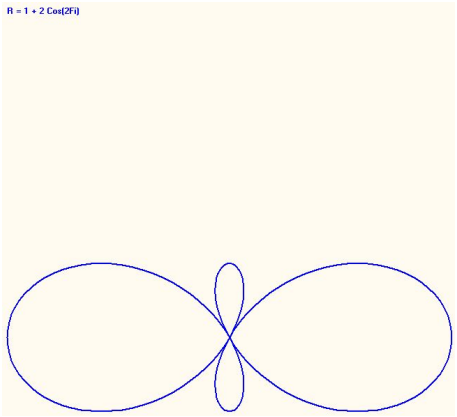










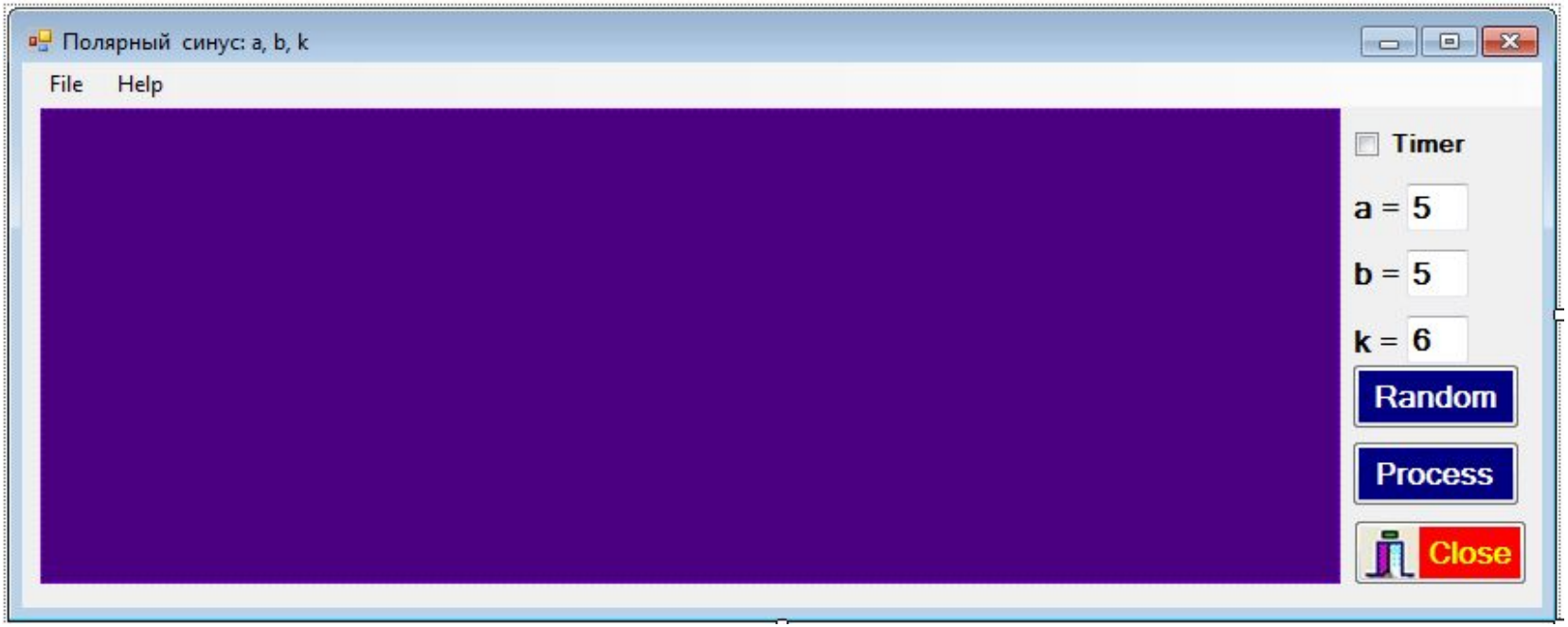


Demo: PolSinus
Полярный синус

```
r = 0.8*sin(k*fi);
x = 0.5*r*cos(a*fi)+0.5;
y = 0.5*r*sin(b*fi)+0.5;
```

Параметры: a, b, k

«Волшебные картинки»: Ч. Коснёвски*). Занимательная математика и персональный компьютер. М.Мир.1987



*) Czes Kosniowski. Fun mathematics on your microcomputer

Событие кнопки Random

```
button2_Click(System::Object^  sender, System::EventArgs^  e) {
     //Random a, b, k
    a = 2 + rand()%(4); this -> textBox1 -> Text =  Convert::ToString(a);
    b = 2 + rand()%(5); this -> textBox2 -> Text =  Convert::ToString(b);
    k = 2 + rand()%(5); this -> textBox3 -> Text =  Convert::ToString(k);
    this ->pictureBox1-> Refresh();
}
```

Событие pictureBox1_Paint

```
pictureBox1_Paint(System::Object^  sender,
System::Windows::Forms::PaintEventArgs^  e) {
            Color ^col = gcnew Color();
        Pen ^pen = gcnew Pen(col -> Yellow);
        pen -> Width=1;
    // «ОЧИСТКА» ПОЛОТНА
        e -> Graphics -> Clear (col -> Indigo); //e - аргумент!!!
         float cx, cy, x0, y0,x, y, u0, v0, u1, v1, fi, r;
        cx= Lx-20; cy=Ly-20; if (cy<cx)  cx=cy;
e-> Graphics -> DrawRectangle(pen,10,10,int(cx),int(cx)); // FRAME
```

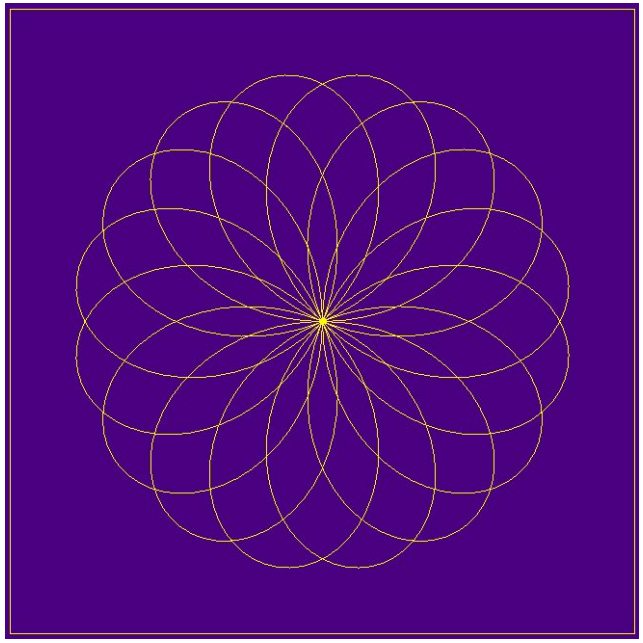
Событие pictureBox1_Paint (продолжение)

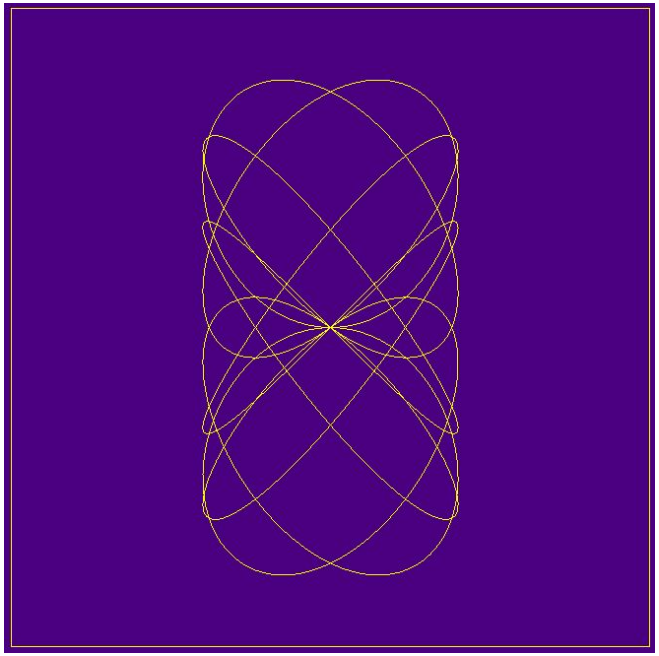
```
    x0=0.5;y0=0.5; // ЦЕНТР ПОЛОТНА
    u0=10+(cx*x0);v0=10+(cx*y0);
    fi=0;
    while(fi<=2*3.141593 ) {
     r = 0.8*sin(k*fi);
    x = 0.5*r*cos(a*fi)+0.5;
    y = 0.5*r*sin(b*fi)+0.5;
         u1=10+(cx*x);v1=10+(cx*y);
    e  -> Graphics-> DrawLine(pen, u0,v0,u1,v1);
    u0 = u1;  v0 = v1;
    fi = fi + 0.01;
    } //while fi
}
```

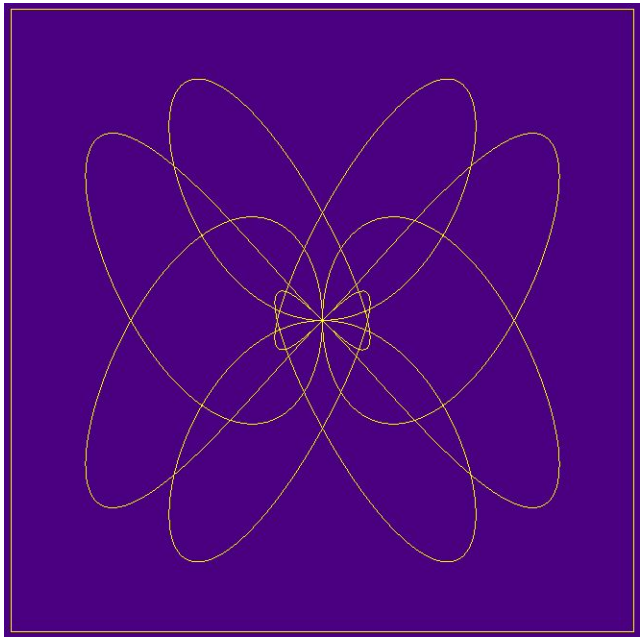
Подготовка счёта

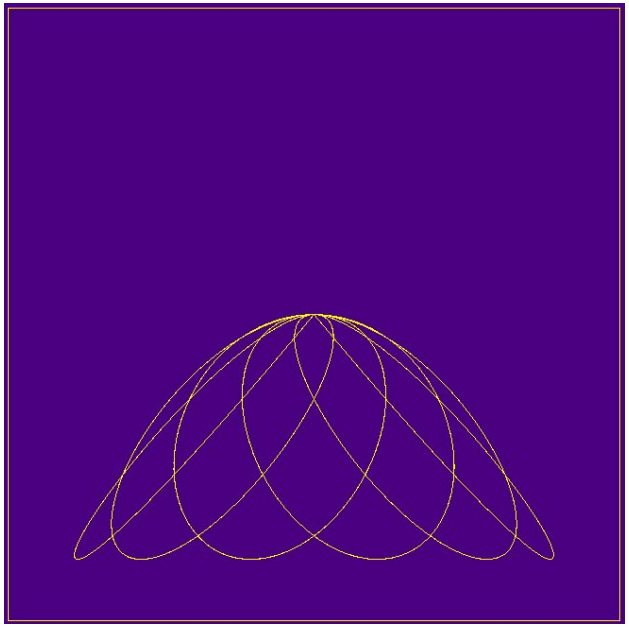
```
Form1_Load(System::Object^  sender,
        System::EventArgs^  e) {
         a = 5;
         b = 5;
         k = 8;
         Lx = this-> pictureBox1->Width;
         Ly = this-> pictureBox1->Height;
 }
```

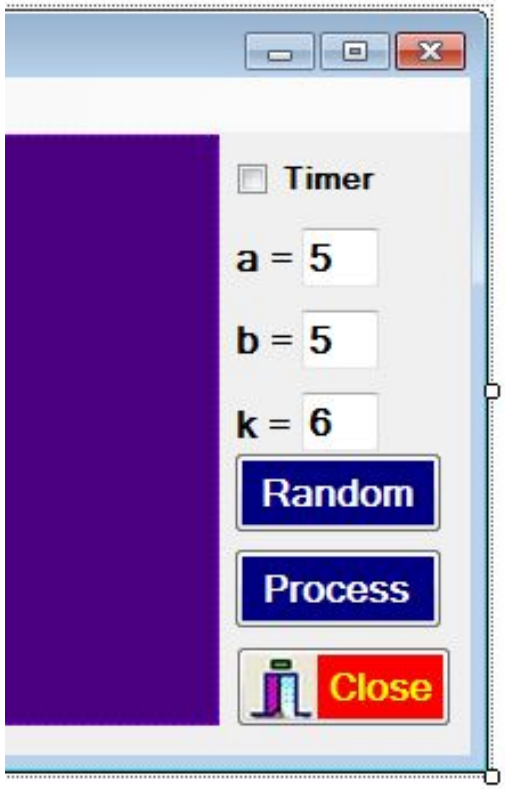
Примеры картинок











Timer?

Компонент 


```
checkBox1_Click(System::Object^  sender,
     System::EventArgs^  e) {
 if(this->checkBox1->Checked)
          this-> timer1->Enabled = true;
    else this->timer1-> Enabled = false;
 }
```

СВОЙСТВА Timer

Enabled false

Interval 1000

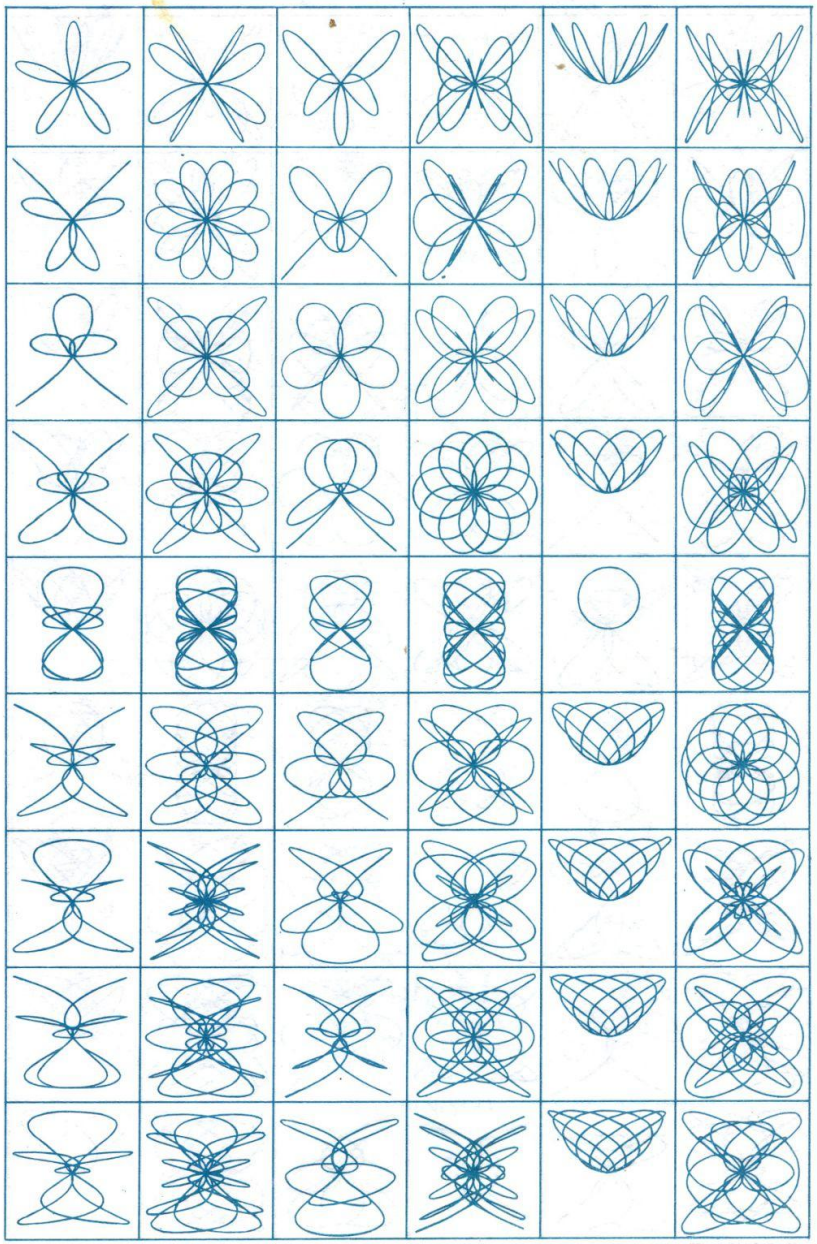
СОБЫТИЕ Timer

Tick button2_Click

[Кнопка Random]

Автоматическая смена параметров!

K = 5

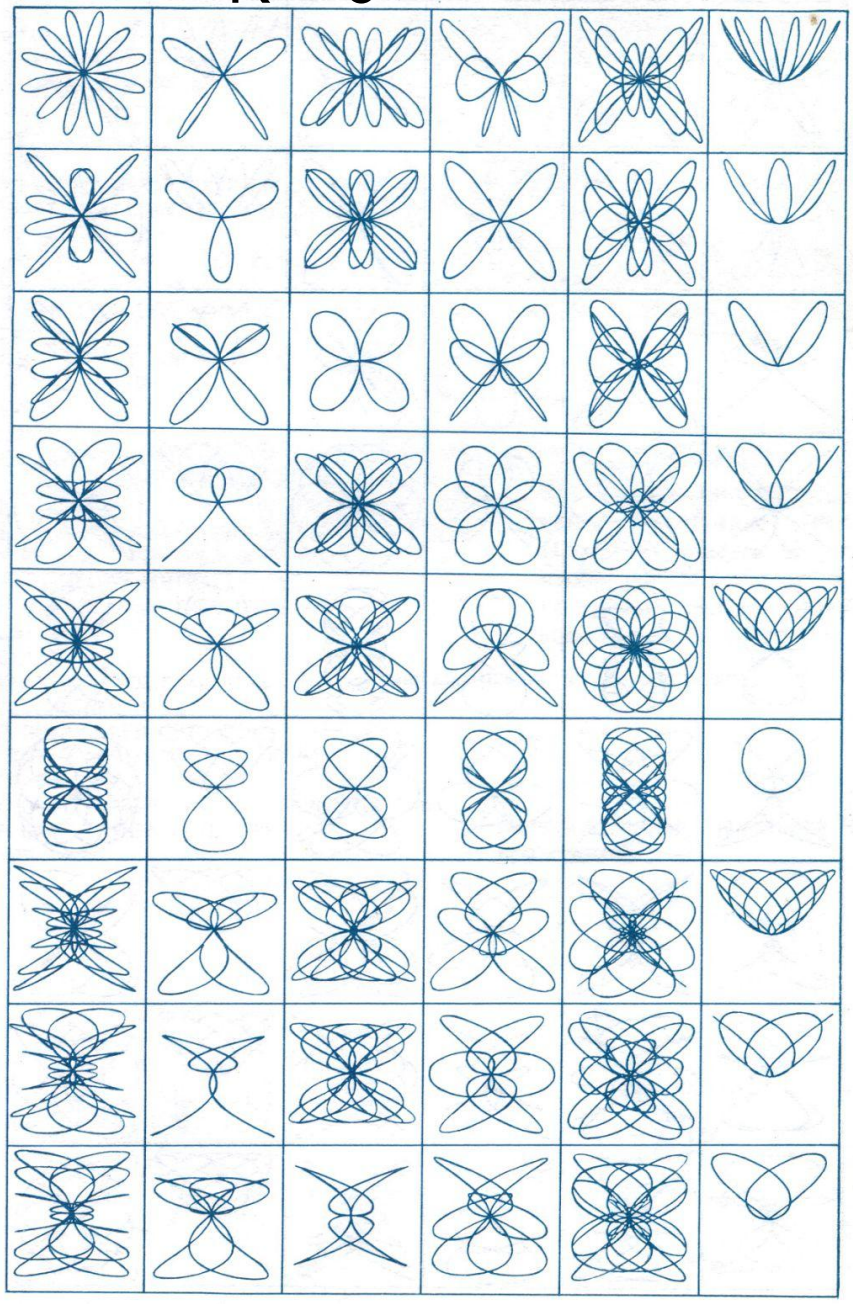


Изменения параметров:

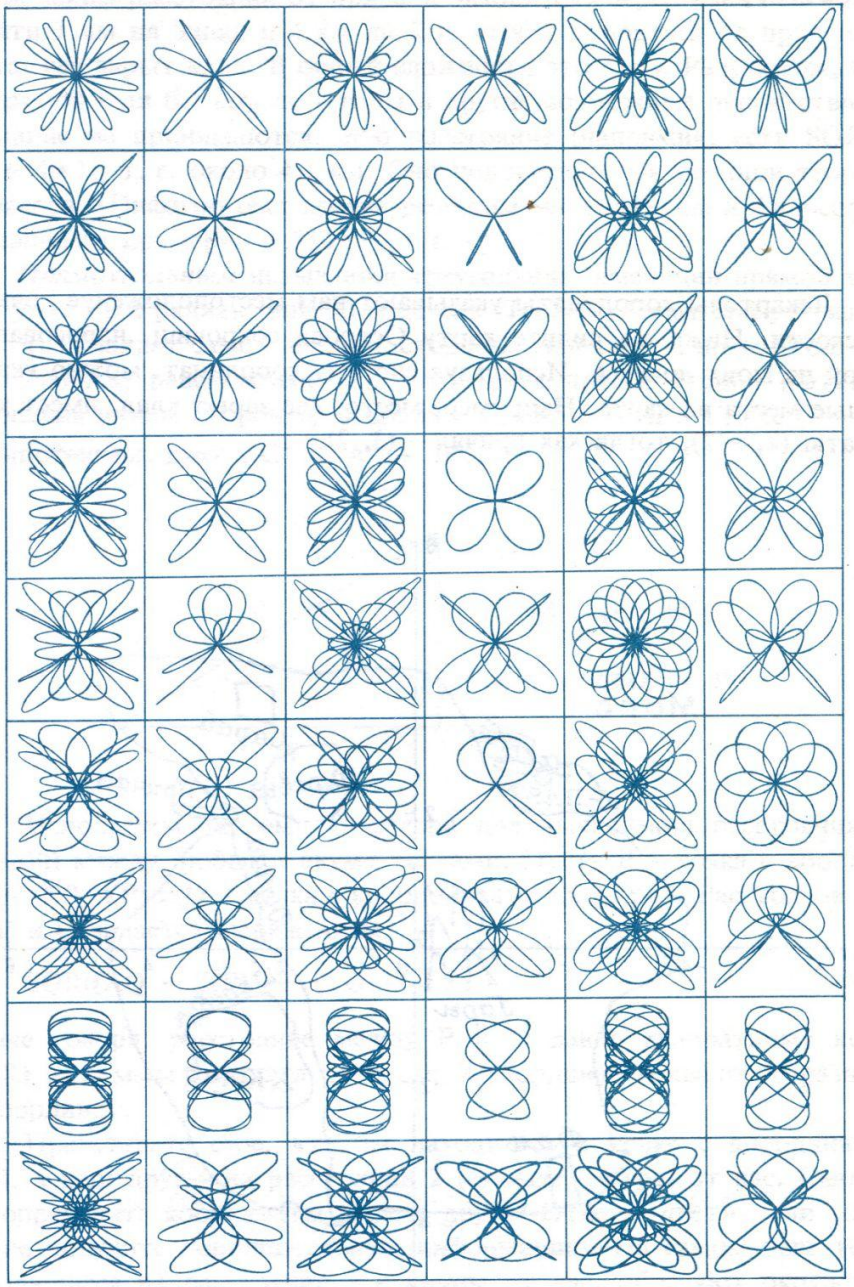
a – вниз;

b – слева направо

K = 6



K = 8

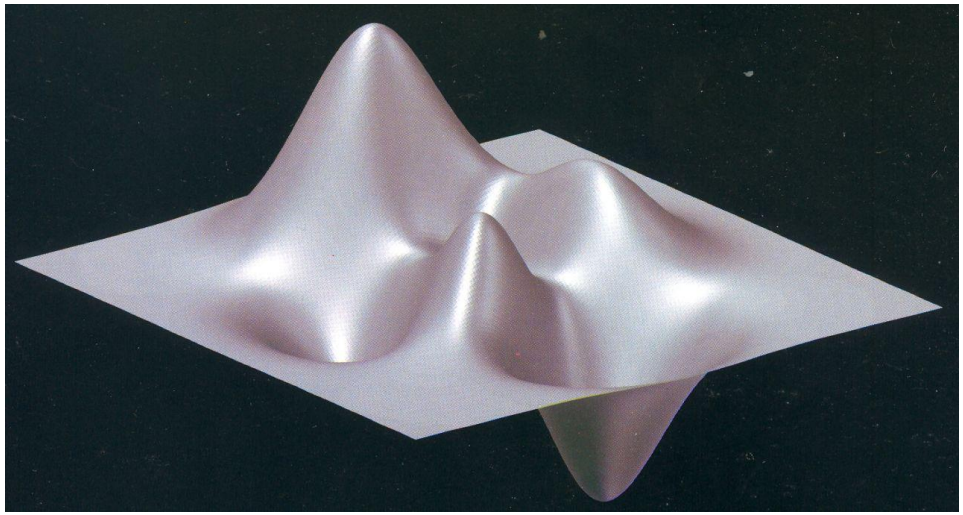


ИЗОЛИНИИ ПОЛЯ
$z = F(x, y)$

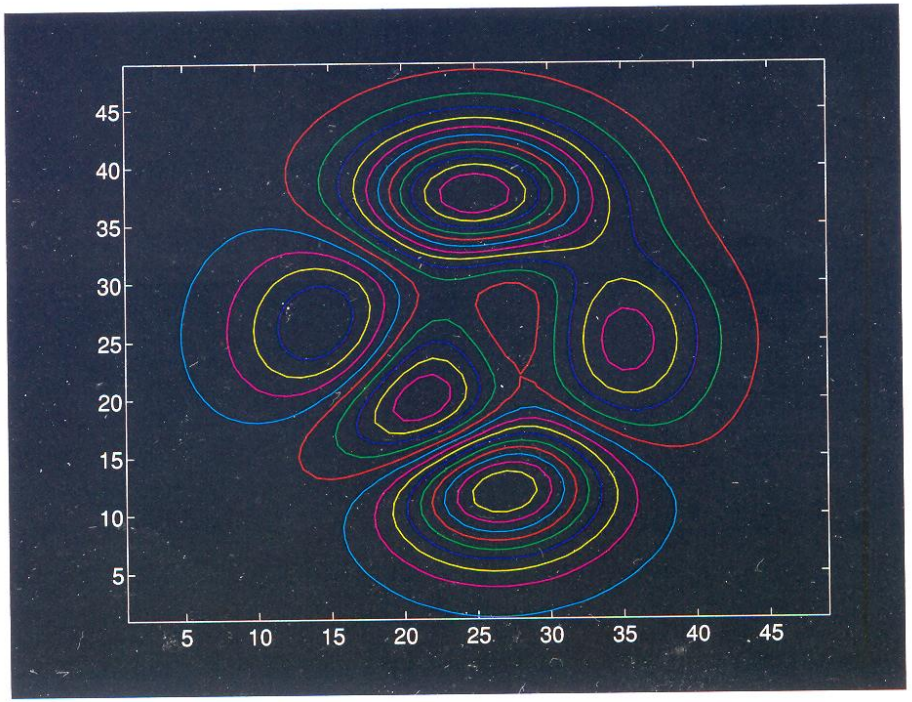
ПРИМЕР ПОЛЯ

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(x/5 - x^3 - y^5)e^{-x^2-y^2} - (1/3)e^{-(x+1)^2-y^2}$$

3D

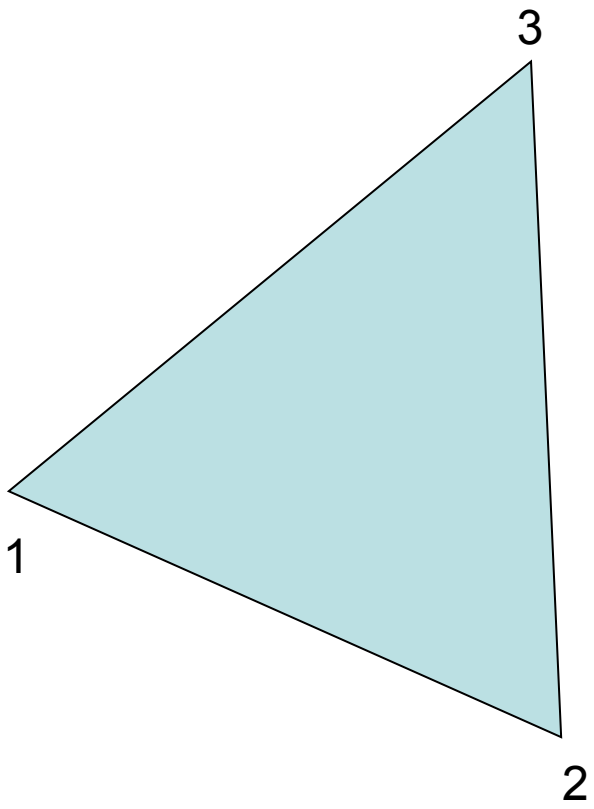


2D



from: An Introduction to MATLAB

ИНТЕРПОЛЯЦИЯ ПОЛЯ Z(x,y) ВНУТРИ ТРЕУГОЛЬНИКА



ВЕРШИНЫ ТРЕУГОЛЬНИКА

1 (x1, y1, z1)
2 (x2, y2, z2)
3 (x3, y3, z3)

Для линейной модели поля:

$$Z(x,y) = a + b*x + c*y$$

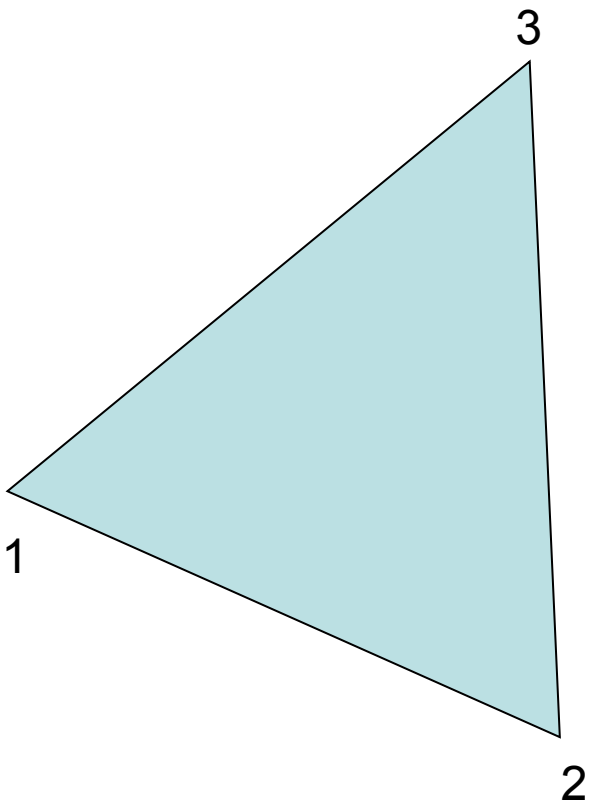
коэффициенты a, b, c определяются значениями координат и поля в 3-х вершинах:

$$\begin{cases} Z1 = a + b*x1 + c*y1 \\ Z2 = a + b*x2 + c*y2 \\ Z3 = a + b*x3 + c*y3 \end{cases}$$

$$\text{Det} = \begin{vmatrix} 1 & x1 & y1 \\ 1 & x2 & y2 \\ 1 & x3 & y3 \end{vmatrix}$$

Система имеет единственное решение a, b, c, т.к. определитель системы отличен от нуля (равен удвоенной площади треугольника)

ВИЗУАЛИЗАЦИЯ ПОЛЯ *Z(X,Y)* ВНУТРИ ТРЕУГОЛЬНИКА



ПРОВЕДЕНИЕ ИЗОЛИНИЙ ПОЛЯ

(изолиния – линия -> *Z = const)*

АЛГОРИТМ

Среди значений Z1, Z2, Z3 выберем Max и Min

(предполагается, что Min ≠Max, иначе задача теряет смысл)

Зададим нужное число изолиний K_Izo

Определим шаг изолиний h = (Max-Min)/(K_Izo-1)

Набор изолиний:

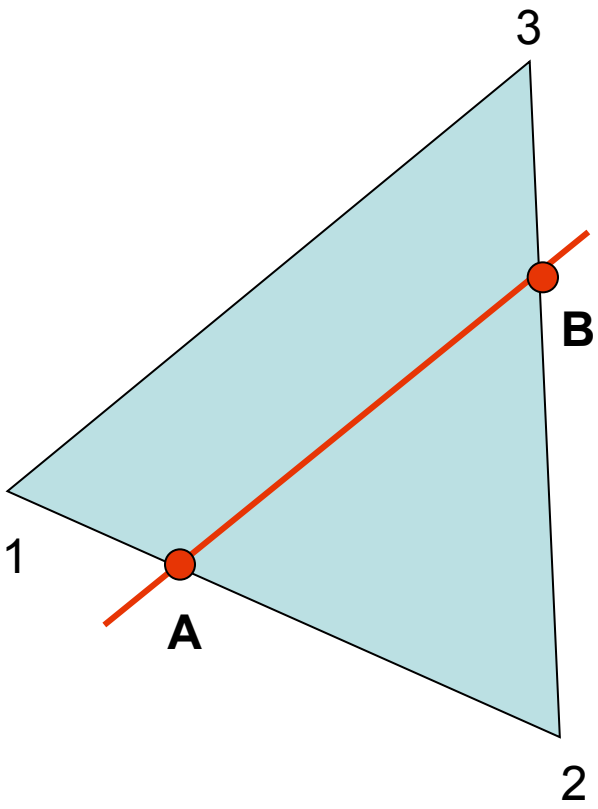
$Z^*(i) = Min +(i-1)*h;\ \ i = 1 \ldots K\_Izo$

при i=1: $Z^*(i) = Min;$

при i=K_Izo: $Z^*(i) = Min +(K\_Izo-1)*h = Max.$

ХОД ИЗОЛИНИИ ВНУТРИ ТРЕУГОЛЬНИКА?

ХОД ИЗОЛИНИИ ВНУТРИ ТРЕУГОЛЬНИКА



Изолиния внутри – ПРЯМАЯ ЛИНИЯ

(при линейной модели поля).

Задача: найти точки пересечения изолинии со сторонами треугольника и соединить их

На рисунке изолиния Z^* пересекает стороны (1,2) и (2,3) в точках A и B. Условие пересечения изолинии со стороной в точке A:

$$(Z^* - Z1)*(Z^* - Z2) \le 0$$

В точке B: $(Z^* - Z2)*(Z^* - Z3) \le 0$

Координаты точки A (интерполяцией):

$Xa = X1+(X2-X1)*(Z^* - Z1)/(Z2-Z1)$

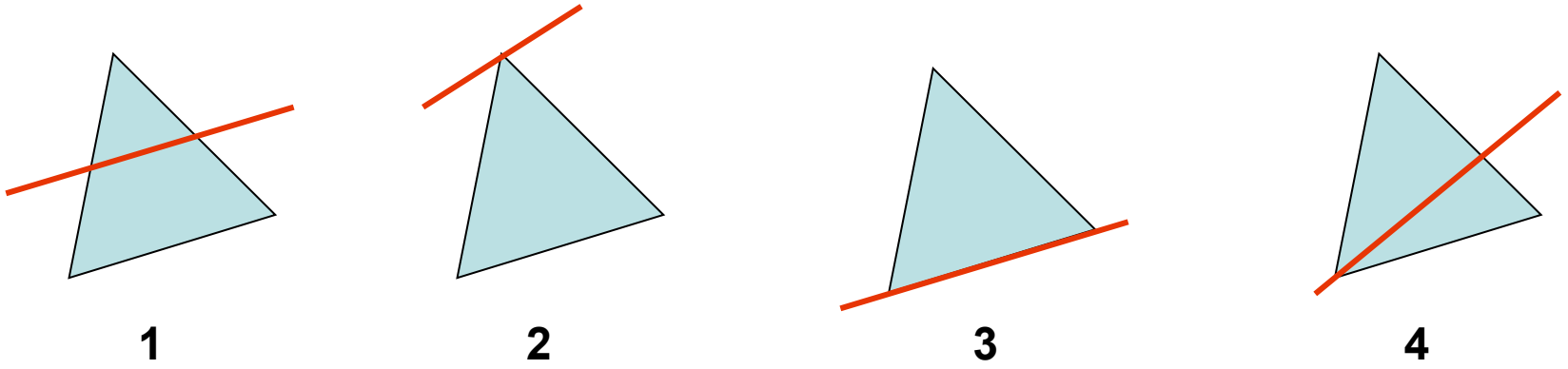
$Ya=Y1+(Y2-Y1)*(Z^* - Z1)/(Z2-Z1)$

ТЕСТ: *если* $Z^*=Z1$, *то* $Xa = X1; Ya = Y1.$

если $Z^*=Z2$, *то* $Xa = X2; Ya = Y2.$

АНАЛОГИЧНО ОПРЕДЕЛЯЮТСЯ КООРДИНАТЫ *Xb, Yb*

ВАРИАНТЫ РАСПОЛОЖЕНИЯ ИЗОЛИНИИ И СТОРОН ТРЕУГОЛЬНИКА



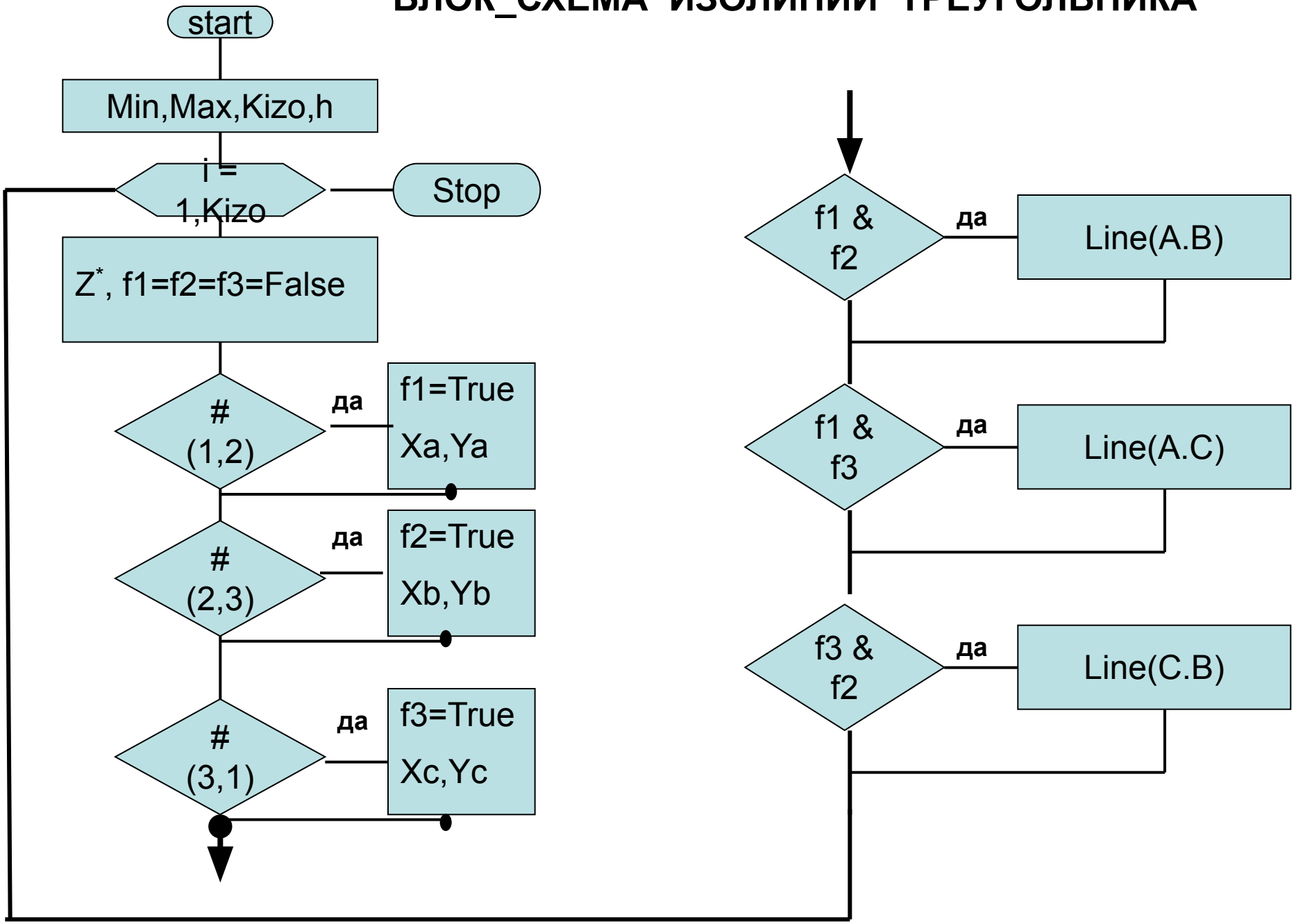
1 2 3 4

Для вариантов 1,2 изолиния имеет общие точки с двумя сторонами треугольника

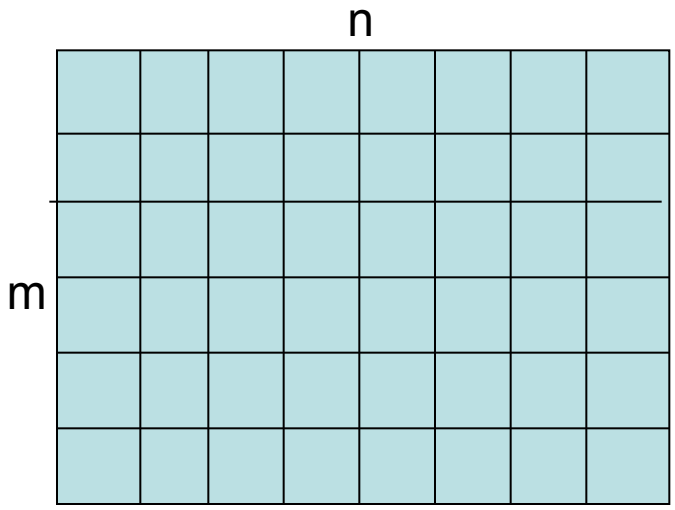
Для вариантов 3,4 изолиния имеет общие точки с тремя сторонами треугольника

В алгоритме необходимо предусмотреть соединение ВСЕХ общих точек

БЛОК_СХЕМА ИЗОЛИНИЙ ТРЕУГОЛЬНИКА



ИЗОЛИНИИ ПОЛЯ Z(x,y), ЗАДАННОГО ТАБЛИЦЕЙ [m x n]



Min, Max, K_Izo, h
Z(i) = Min +(i-1)*h; i = 1 ..K_Izo*

Прямоугольник таблицы – область построения

Координаты узлов (нормированы [0..1]):

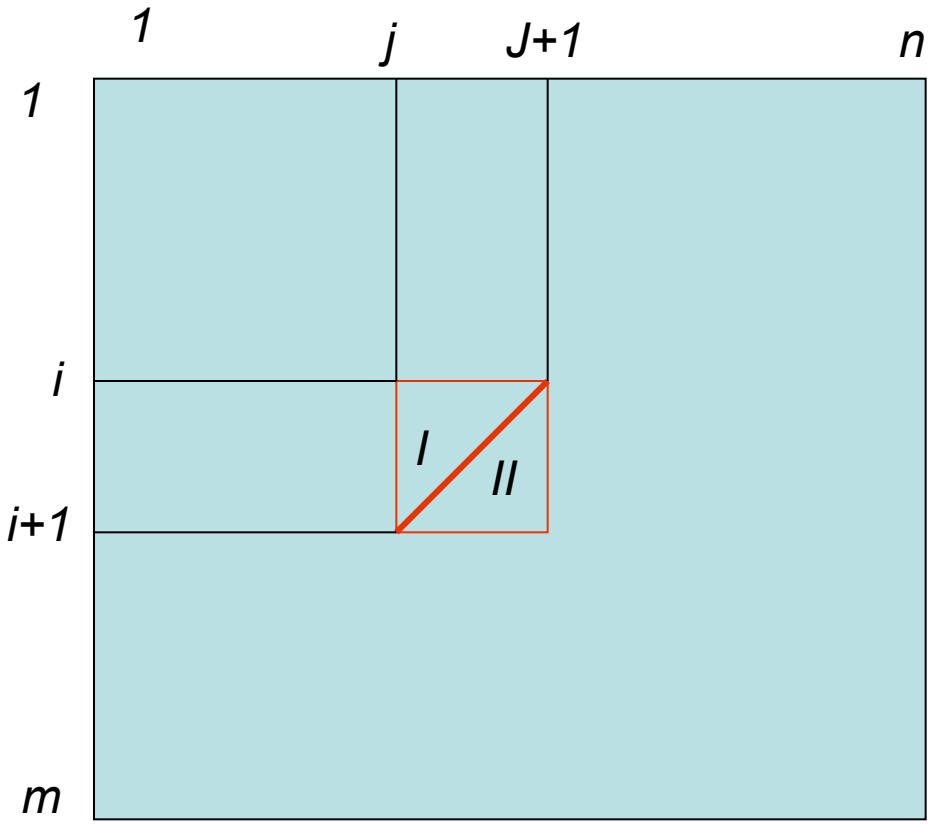
Xj – (горизонтальная) = *(j-1)/(n-1); j=1..n*

Yi- (вертикальная)= *1-(i-1)/(m-1); i=1..m*

(Вертикальная координата имеет направление обратное нумерации строк таблицы)

ХОД ИЗОЛИНИИ ВНУТРИ ПРЯМОУГОЛЬНИКА?

ЯЧЕЙКА СЕТКИ, РАЗДЕЛЕННАЯ НА 2 ТРЕУГОЛЬНИКА



УЗЛЫ ТРЕУГОЛЬНИКОВ

(обход против часовой стрелки)

I

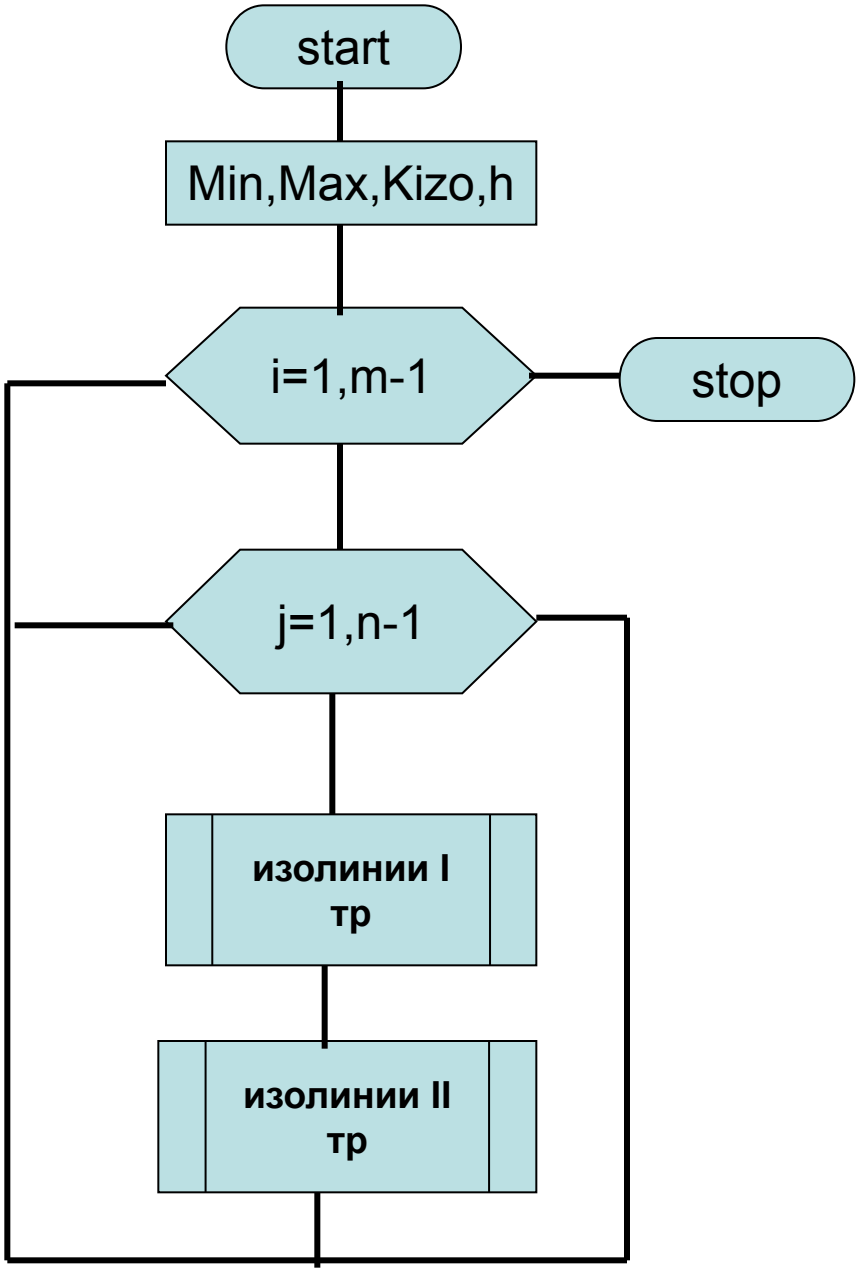
(i,j) (i+1,j) (i,j+1)

II

(i+1,j) (i+1,j+1) (i,j+1)

Остается построить изолинии в каждом треугольнике

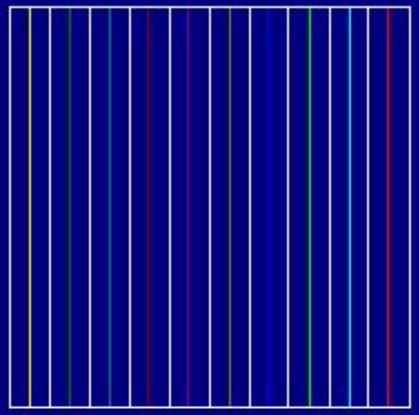
БЛОК-СХЕМА ПОСТРОЕНИЯ ИЗОЛИНИЙ ТАБЛИЧНОЙ ФУНКЦИИ



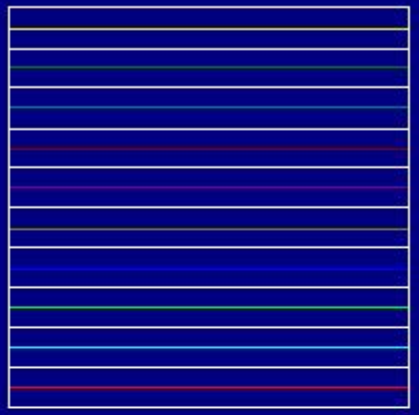
РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ ПРОГРАММЫ: ИЗОЛИНИИ ТАБЛИЦЫ

Центральное поле

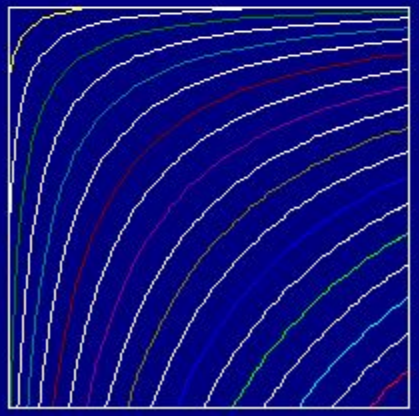
Z = F(x)



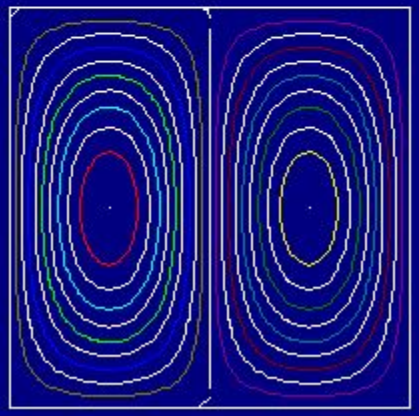
Z = F(y)



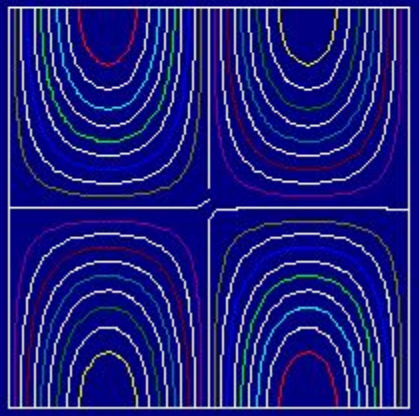
Z = SQRT(X*Y)



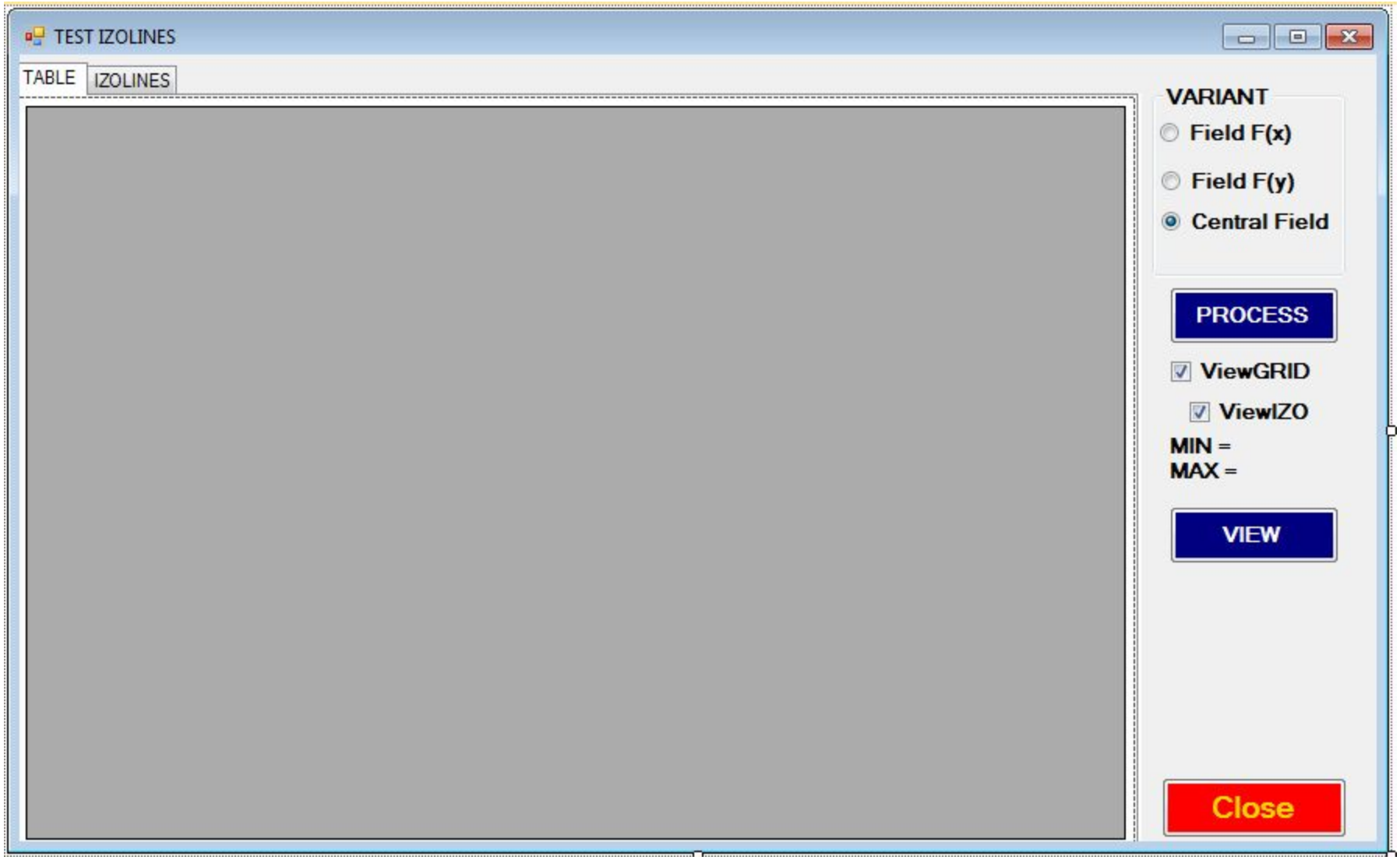
Z = Sin * COS



Z = Sin * Sin



Demo: _IZO


TEST IZOLINES
TABLE
IZOLINES
VARIANT
Field F(x)
Field F(y)
Central Field
PROCESS
ViewGRID
ViewIZO
MIN =
MAX =
VIEW
Close

ЗАПОЛНЕНИЕ МАССИВА

```
if(this -> radioButton1-> Checked)VARIANT = 1;
if(this -> radioButton2-> Checked)VARIANT = 2;
if(this -> radioButton3-> Checked)VARIANT = 3;
switch (VARIANT){
case 1: {
for(int i = 0; i< NNN; i++)  for(int j = 0; j< NNN; j++)
    MAS[i][j] = j +1; break; }
case 2: {
for(int i = 0; i< NNN; i++)  for(int j = 0; j< NNN; j++)
    MAS[i][j] = i +1; break; }
case 3: {
for(int i = 0; i< NNN; i++)  for(int j = 0; j< NNN; j++){
double r =  ((i - NNN/2)*(i - NNN/2)+(j-NNN/2)*(j-NNN/2));
    r = sqrt(r);
    MAS[i][j] = r; break;}
} // switch
}
```

ТАБЛИЦА

```
this -> dataGridView1 -> RowCount = NNN;
this -> dataGridView1 -> ColumnCount = NNN;
 // Columns  Numbers
     for (int i = 0; i < NNN; i++)
this ->dataGridView1->Columns[i]->HeaderText = "  " + (i + 1).ToString() + "  ";
 //  Row Numbers
 for (int i = 0; i < NNN; i++)
       this ->dataGridView1->Rows[i]->HeaderCell->Value  = (i + 1).ToString();
// Value [i][j]
for(int i = 0; i< NNN; i++)  for(int j = 0; j< NNN; j++){
 double r = MAS[i][j];
 String^ st =  Convert::ToString(Math::Round(r, 1));
 // Clear Cell
     this -> dataGridView1 ->Rows[i]->Cells[j] -> Style -> BackColor =
           System::Drawing::Color::White;
     this -> dataGridView1 ->Rows[i]->Cells[j]-> Value = st;
 }           //Cross Aqua;
for (int i = 0; i < NNN; i++)
this -> dataGridView1 ->Rows[i]->Cells[NNN/2] -> Style -> BackColor =
      System::Drawing::Color::Aqua;
for (int j = 0; j < NNN; j++)
this -> dataGridView1 ->Rows[NNN/2]->Cells[j] -> Style -> BackColor =
      System::Drawing::Color::Aqua;
```

ВЫВОД ТАБЛИЦЫ

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	14.1	13.5	12.8	12.2	11.7	11.2	10.8	10.4	10.2	10	10	10	10.2	10.4	10.8	11.2	11.7	12.2	12.8	13.5	14.1
2	13.5	12.7	12	11.4	10.8	10.3	9.8	9.5	9.2	9.1	9	9.1	9.2	9.5	9.8	10.3	10.8	11.4	12	12.7	13.5
3	12.8	12	11.3	10.6	10	9.4	8.9	8.5	8.2	8.1	8	8.1	8.2	8.5	8.9	9.4	10	10.6	11.3	12	12.8
4	12.2	11.4	10.6	9.9	9.2	8.6	8.1	7.6	7.3	7.1	7	7.1	7.3	7.6	8.1	8.6	9.2	9.9	10.6	11.4	12.2
5	11.7	10.8	10	9.2	8.5	7.8	7.2	6.7	6.3	6.1	6	6.1	6.3	6.7	7.2	7.8	8.5	9.2	10	10.8	11.7
6	11.2	10.3	9.4	8.6	7.8	7.1	6.4	5.8	5.4	5.1	5	5.1	5.4	5.8	6.4	7.1	7.8	8.6	9.4	10.3	11.2
7	10.8	9.8	8.9	8.1	7.2	6.4	5.7	5	4.5	4.1	4	4.1	4.5	5	5.7	6.4	7.2	8.1	8.9	9.8	10.8
8	10.4	9.5	8.5	7.6	6.7	5.8	5	4.2	3.6	3.2	3	3.2	3.6	4.2	5	5.8	6.7	7.6	8.5	9.5	10.4
9	10.2	9.2	8.2	7.3	6.3	5.4	4.5	3.6	2.8	2.2	2	2.2	2.8	3.6	4.5	5.4	6.3	7.3	8.2	9.2	10.2
10	10	9.1	8.1	7.1	6.1	5.1	4.1	3.2	2.2	1.4	1	1.4	2.2	3.2	4.1	5.1	6.1	7.1	8.1	9.1	10
11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10
12	10	9.1	8.1	7.1	6.1	5.1	4.1	3.2	2.2	1.4	1	1.4	2.2	3.2	4.1	5.1	6.1	7.1	8.1	9.1	10
13	10.2	9.2	8.2	7.3	6.3	5.4	4.5	3.6	2.8	2.2	2	2.2	2.8	3.6	4.5	5.4	6.3	7.3	8.2	9.2	10.2
14	10.4	9.5	8.5	7.6	6.7	5.8	5	4.2	3.6	3.2	3	3.2	3.6	4.2	5	5.8	6.7	7.6	8.5	9.5	10.4
15	10.8	9.8	8.9	8.1	7.2	6.4	5.7	5	4.5	4.1	4	4.1	4.5	5	5.7	6.4	7.2	8.1	8.9	9.8	10.8
16	11.2	10.3	9.4	8.6	7.8	7.1	6.4	5.8	5.4	5.1	5	5.1	5.4	5.8	6.4	7.1	7.8	8.6	9.4	10.3	11.2
17	11.7	10.8	10	9.2	8.5	7.8	7.2	6.7	6.3	6.1	6	6.1	6.3	6.7	7.2	7.8	8.5	9.2	10	10.8	11.7
18	12.2	11.4	10.6	9.9	9.2	8.6	8.1	7.6	7.3	7.1	7	7.1	7.3	7.6	8.1	8.6	9.2	9.9	10.6	11.4	12.2
19	12.8	12	11.3	10.6	10	9.4	8.9	8.5	8.2	8.1	8	8.1	8.2	8.5	8.9	9.4	10	10.6	11.3	12	12.8
20	13.5	12.7	12	11.4	10.8	10.3	9.8	9.5	9.2	9.1	9	9.1	9.2	9.5	9.8	10.3	10.8	11.4	12	12.7	13.5
21	14.1	13.5	12.8	12.2	11.7	11.2	10.8	10.4	10.2	10	10	10	10.2	10.4	10.8	11.2	11.7	12.2	12.8	13.5	14.1

Изолинии поля

VARIANT

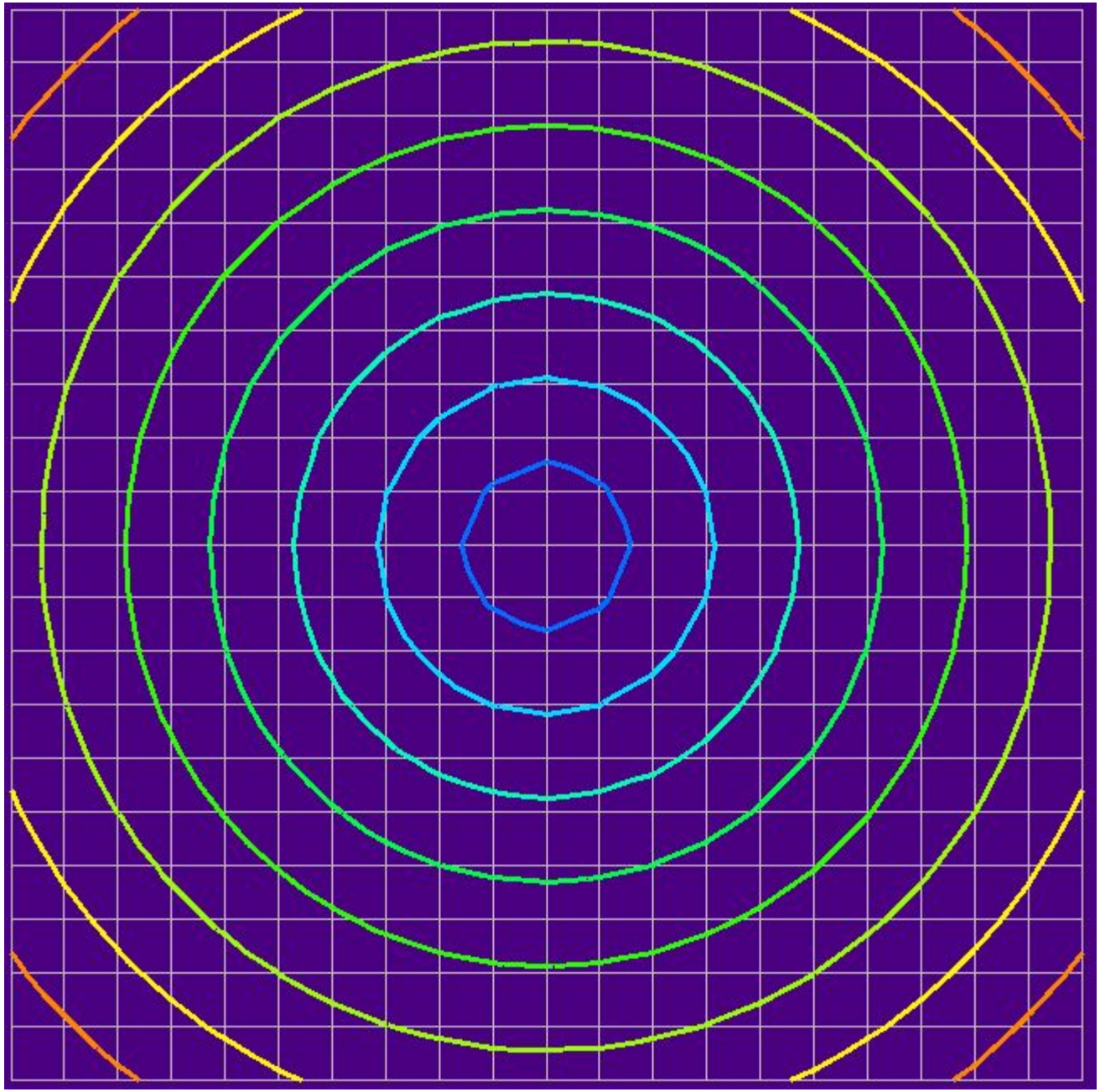
- Field F(x)
- Field F(y)
- Central Field

PROCESS

- ViewGRID
- ViewIZO

MIN = 0
MAX = 14.1

VIEW



СЕТКА

```
if(flViewGRID){
 // GRID x, y
 x1 = 10; x2 = 10 + SY;
 for(int i = 1; i < NNN -1; i++){
     y1 = 10 + i * hy; y2 = y1;
      im -> DrawLine(pen, int(x1), int(y1), int(x2), int(y2));
 }
 y1 = 10; y2 = 10 + SY;
 for(int i = 1; i < NNN -1; i++){
     x1 = 10 + i * hy; x2 = x1;
     im -> DrawLine(pen, int(x1), int(y1), int(x2), int(y2));
 }
```

ИЗОЛИНИИ

```
if(flViewIZO){
 double hIZO = (MAX - MIN) / (kIZO - 1);
 int dm = (1023) / (kIZO - 1);
          // Цикл изолиний
 for(int ii = 0; ii < kIZO; ii++){
     double zIZO = MIN + hIZO * ii;
         // Color zIZO FromArgb !!!
          int  m_rab =  ii*dm;  GetRGB(m_rab);
  // Изменение цвета в цикле
         pen -> Color = col -> FromArgb(rR, rG, rB);
         pen -> Width = 3;
```

```
// ЦИКЛ по ПРЯМОУГОЛЬНИКАМ  СЕТКИ
 for(int i = 0; i < NNN -1; i++)  for(int j = 0; j < NNN -1; j++){
 // Triangle 1
     double xi = x(j); double xj = x(j); double xk = x(j + 1);
     double yi = y(i); double yj = y(i + 1); double yk = y(i);
     double zi = MAS[i][j]; double zj = MAS[i + 1][j];
     double zk = MAS[i][j + 1];
     bool fl1=false,fl2=false,fl3=false;
     double xr1, xr2, xr3, yr1, yr2, yr3;
     int u1, u2, u3, v1, v2, v3;
//First side
 if ((fabs(zi-zj)>1e-3)&&((zi-zIZO)*(zj-zIZO)<=0)){
              fl1=true;
              xr1=xi+(xj-xi)/(zj-zi)*(zIZO-zi);
              yr1=yi+(yj-yi)/(zj-zi)*(zIZO-zi);
              u1=10+int(xr1*cx);    v1=Bottom-int(yr1*cy);
 }
```

```
//Second side
 if((fabs(zj-zk)>1e-3)&&((zk-zIZO)*(zj-zIZO)<=0)){
                        fl2=true;
                        xr2=xj+(xk-xj)/(zk-zj)*(zIZO-zj);
                        yr2=yj+(yk-yj)/(zk-zj)*(zIZO-zj);
                        u2=10+int(xr2*cx);
                        v2=Bottom-int(yr2*cy);
 }
 //Third side
 if((fabs(zk-zi)>1e-3)&&((zi-zIZO)*(zk-zIZO)<=0)){
                        fl3=true;
                        xr3=xk+(xi-xk)/(zi-zk)*(zIZO-zk);
                        yr3=yk+(yi-yk)/(zi-zk)*(zIZO-zk);
                        u3=10+int(xr3*cx);
                        v3=Bottom-int(yr3*cy);
 }
```

```
      if(fl1 && fl2) im -> DrawLine(pen, u1,v1,u2,v2);
      if(fl1 && fl3) im -> DrawLine(pen, u1,v1,u3,v3);
      if(fl2 && fl3) im -> DrawLine(pen, u2,v2,u3,v3);
 } // Triangle 1
// ЦИКЛ по ПРЯМОУГОЛЬНИКАМ  СЕТКИ
for(int i = 0; i < NNN -1; i++)  for(int j = 0; j < NNN -1; j++){
          // Triangle 2
      double xi = x(j); double xj = x(j+1); double xk = x(j + 1);
      double yi = y(i+1); double yj = y(i + 1); double yk = y(i);
      double zi = MAS[i+1][j];  double zj = MAS[i + 1][j+1];
      double zk = MAS[i][j + 1];
     bool fl1=false,fl2=false,fl3=false;
     double xr1, xr2, xr3, yr1, yr2, yr3;
     int u1, u2, u3, v1, v2, v3;
```

```
//First side
 if ((fabs(zi-zj)>1e-3)&&((zi-zIZO)*(zj-zIZO)<=0)){
                          fl1=true;
                         xr1=xi+(xj-xi)/(zj-zi)*(zIZO-zi);
                         yr1=yi+(yj-yi)/(zj-zi)*(zIZO-zi);

                         u1=10+int(xr1*cx);
                         v1=Bottom-int(yr1*cy);
 }
 //Second side
 if((fabs(zj-zk)>1e-3)&&((zk-zIZO)*(zj-zIZO)<=0)){
                         fl2=true;
                         xr2=xj+(xk-xj)/(zk-zj)*(zIZO-zj);
                         yr2=yj+(yk-yj)/(zk-zj)*(zIZO-zj);
                         u2=10+int(xr2*cx);
                         v2=Bottom-int(yr2*cy);
 }
```

```
//Third side
 if((fabs(zk-zi)>1e-3)&&((zi-zIZO)*(zk-zIZO)<=0)){
                        fl3=true;
                        xr3=xk+(xi-xk)/(zi-zk)*(zIZO-zk);
                        yr3=yk+(yi-yk)/(zi-zk)*(zIZO-zk);
                        u3=10+int(xr3*cx);
                        v3=Bottom-int(yr3*cy);
 }
 if(fl1 && fl2) im -> DrawLine(pen, u1,v1,u2,v2);

 if(fl1 && fl3) im -> DrawLine(pen, u1,v1,u3,v3);

 if(fl2 && fl3) im -> DrawLine(pen, u2,v2,u3,v3);

 } // Triangle 2
 } // ii IZO
 } // if(flViewIZO)
```

```
// СОСТАВЛЯЮЩИЕ  R G B
void GetRGB(int m )
        {
            int di = m / 256;   int mo = m % 256;
            switch (di)
            {
                case 0: rR = 0; rG = mo; rB = 255; break;
                case 1: rR = 0; rG = 255; rB = 255 - mo; break;
                case 2: rR = mo; rG = 255; rB = 0; break;
                case 3: rR = 255; rG = 255 - mo; rB = 0; break;
}
        }
```

ПРАКТИКУМ 04 «полярные координаты»

ПОСТРОЕНИЕ ФУНКЦИЙ В ПОЛЯРНЫХ КООРДИНАТАХ

Во всех приведенных ниже уравнениях [ ] - знак, обозначает целую часть числа, т. е. от результата вычисления выражения, заключенного в скобки [ ] следует отбросить дробную часть. R – радиус, F - угол.
Варианты задания: _04_ПРАКТИКУМ…doc

The End