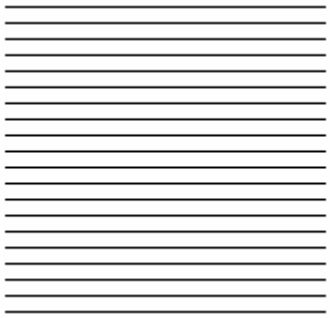
GRAPHIC 2D

Example 1 Horizontal sections

The episode has no beginning and no end, it has two ends! "("each stick has two ends"), but for our use, the first of the given points will be called the starting point and the second end point, which is also consistent with the way we draw. Of course, the order of these points is irrelevant.
In order to obtain horizontal segments for the beginning and the end point, we leave the coordinates x (cut off) unchanged, and the coordinates y (ordinates) are increased with a fixed increment, e.g. every 8 pixels.
The drawing in relation to the beginning of the screen coordinate system is shifted (translation) by the vector [px; py], in this example it is [100; 100].
The value of 160 is determined by the number of loop repetitions: 20 * 8 = 160.

Translated with www.DeepL.com/Translator

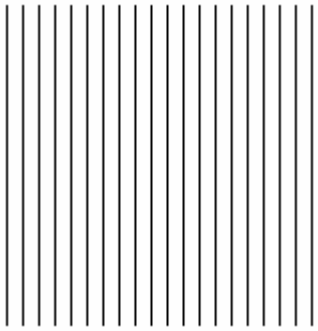
```
int i, px=100, py=100;
for (i=0; i<20; i++)
  line(px,py+i*8,px+160,py+i*8);
```



Example 2 Vertical sections

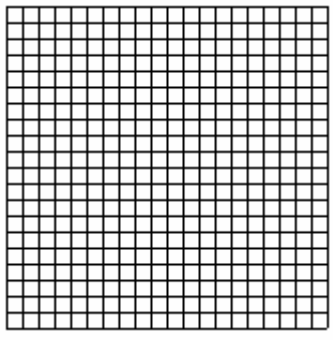
In order to obtain horizontal segments for the beginning and the end point, leave the coordinates y (ordinates) unchanged, and the coordinates x (cut off) are increased with a fixed increment, e.g. every 8 pixels.

```
int i, px=100, py=100;
for (i=0; i<20; i++)
  line(px+i*8,py,px+i*8,py+160);
```



After connecting the horizontal and vertical sections we obtain a grid.

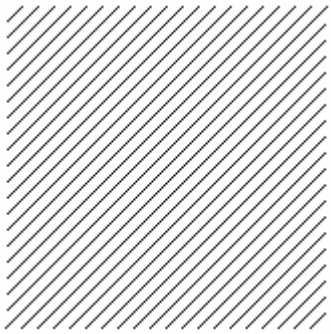
```
int i, px=100, py=100;
for (i=0; i<20; i++)
{
  line(px,py+i*5,px+100,py+i*5);
  line(px+i*5,py,px+i*5,py+100);
}
```



Example 3 Right-angled diagonal sections

The diagonal segments drawn at 45o are obtained by increasing the coordinates x of the starting point (constant row) and y of the end point (constant cut off) with the same interval.
The "line" instruction draws only half of the planned drawing, so you need to use it twice.

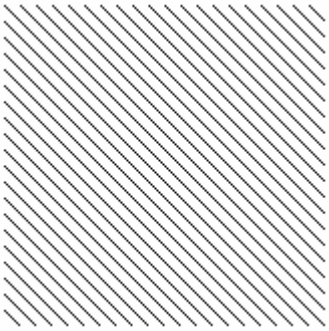
```
int i, px=100, py=100;
for (i=0; i<20; i++)
{
  line(px+i*8,py+160,px+160,py+i*8);
  line(px+i*8,py,px,py+i*8);
}
```



Example 4 Left-angled diagonal sections

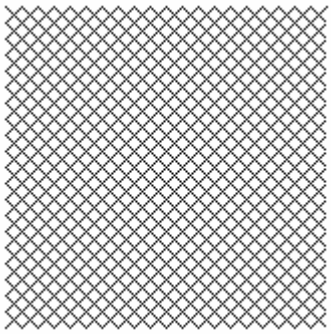
Left-angled segments drawn at an angle of 45o are obtained similarly to the previous example, but by increasing the coordinates y of the starting point (cut off constant) and x of the end point (fixed ordinate). As before, the "line" instruction must be applied twice.

```
int i, px=100, py=100;
for (i=0; i<20; i++)
{
  line(px,py+i*8,px+160-i*8,py+160);
  line(px+i*8,py,px+160,py+160-i*8);
}
```



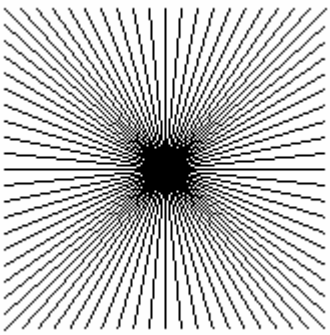
After combining both types of sections we get a grid.

```
int i, px=100, py=100;
for (i=0; i<20; i++)
{
  line(px+i*8,py+160,px+160,py+i*8);
  line(px+i*8,py,px,py+i*8);

  line(px,py+i*8,px+160-i*8,py+160);
  line(px+i*8,py,px+160,py+160-i*8);
}
```



And after a slight modification:

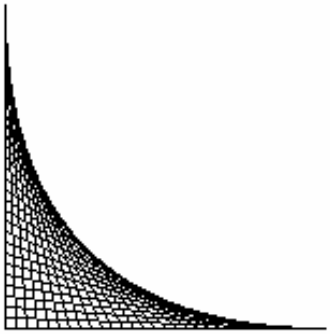
```
int i, px=100, py=100;
for (i=0; i<21; i++)
{
  line(px,py+i*8,px+160,py+160-i*8);
  line(px+i*8,py,px+160-i*8,py+160);
}
```



Example 5 „Curve of sections”

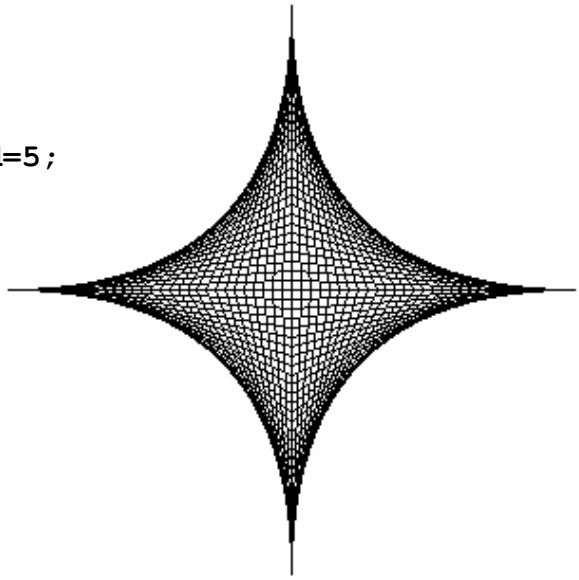
Drawing only the sections themselves will give the impression of an arc.
The segments are drawn alternately for starting points lying on the vertical line and ending points lying on the horizontal line with a fixed spacing, e.g. every 5 pixels.
The value of 150 is determined on the basis of the number of loop repetitions: 30 * 5 = 150.

```
void draw()
{
int px=400, py=100, ile=30, od=5;
for (int k=0; k<=ile; k++)
  line(px, py+k*od, px+k*od, py+150);
}
```



We are now modifying this code to the other three coordinate system orientations on the plane:

```
int px=400, py=100, ile=30, w=150, od=5;
for (int k=0; k<=ile; k++)
{
  line(px, py+k*5, px+k*5, py+w);
  line(px, py+k*5, px-k*5, py+w);
  line(px, w+py+k*5,w+px-k*5, py+w);
  line(px, w+py+k*5, px+k*5-w,py+w);
 }
```



Example 6 Chessboard

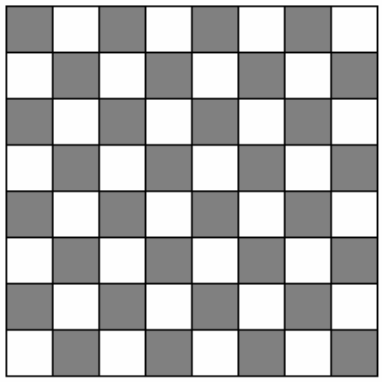
Checking the parity of the expression line+column we can determine the color of the rectangle's fill.
To shorten the entry in the explanation, 'w' will denote the variable 'row' and 'k' the variable 'column'.

w = 0 ⇒ w+1=1 (odd), w+2=2 (even) , w+3=3 (odd) , w+4=2 (even),
w+5=5 (odd), w+6=6 (even), w+7=7 (odd);

w = 1 ⇒ w+1=2 (even), w+2=3 (odd) , w+3=4 (even) , w+4=5 (odd),
w+5=6 (even), w+6=7 (odd), w+7=8 (even);

fill(150, 150, 150) - determines the colour grey, fill(255, 255, 255) - white,

```
int px=100, py=100, ile=8, bok=30;

for (int wiersz=0;  wiersz<8; wiersz++)
for (int kolumna=0; kolumna<8;kolumna++)
{
 if ((wiersz+kolumna)%2==0) fill(150,150,150);
 else  fill(255,255,255);
 rect(px+wiersz*bok,py+kolumna*bok,bok,bok);
}
```



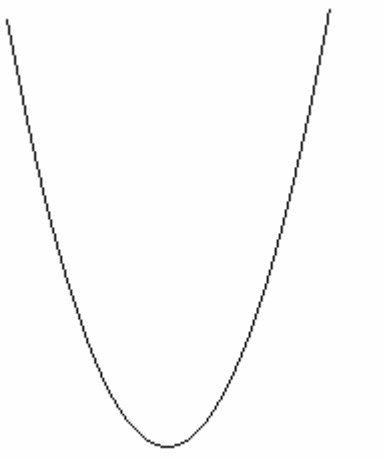
Example 7 Cosine function diagram with segments

```
int i,j, px=100, py=100, hy=20;
float o=32;
for (i=0; i<=500; i++)
{
  j=i+1;
  line (i+px, round(py+hy*cos(i*PI/o)),j+px,
                          round(py+hy*cos(j*PI/o)));
}
```

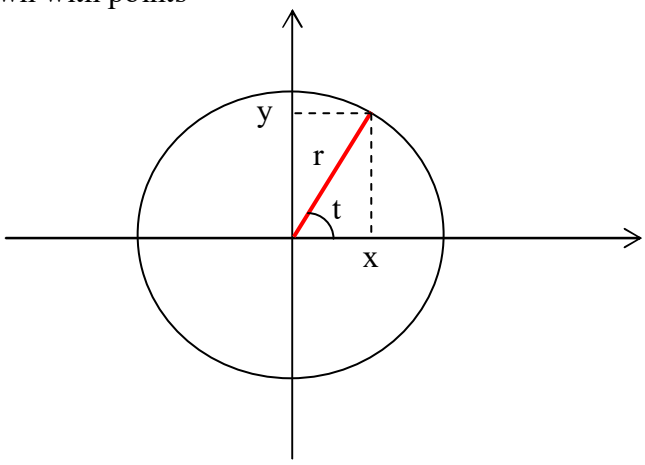
Example 8 Parabola drawn by means of sections

A parabola with a vertex at the beginning of a coordinate system whose axis of symmetry is the y axis is defined by the formula $y = x^2$

```
int j, px=200, py=300, o=30, s=80;
for ( int i=-s; i<s; i++)
{
  j=i+1;
  line (px+i, py-round(i*i/o),px+j, py-round(j*j/o));
}
```

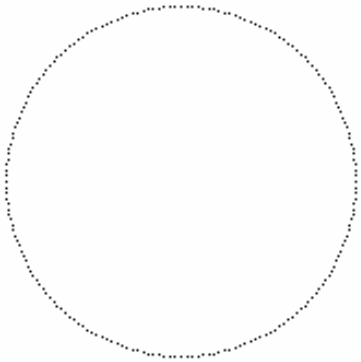


Example 9 A circle drawn with points



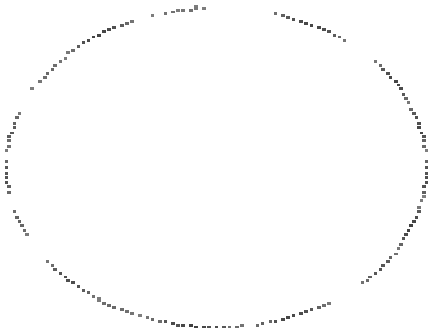
Parametric equations of a circle:

$$x = r \cdot \cos(t)$$
$$y = r \cdot \sin(t) \qquad t \in [0;2\pi]$$



```
int px=200, py=200, r=80;
float t, x, y;
for (int k=0; k<=200; k++)
{
  t=k*0.01*PI;
  x=r*cos(t);
  y=r*sin(t);
  point(px+round(x), py+round(y));
}
```

An ellipse is obtained for different R ≠ r factors in parametric equations.



```
int px=200, py=200, R=120, r=80;
float t, x, y;
for (int k=0; k<=200; k++)
{
  t=k*0.01*PI;
  x=R*cos(t);
  y=r*sin(t);
  point(px+round(x), py+round(y));
}
```

Example 10 Circle drawn by segments

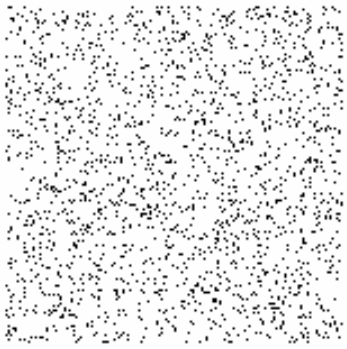
```
int px=200, py=200, r=80;
float t, x1, y1, x2, y2;
for (int k=0; k<100; k++)
{
  x1=r*cos(k*0.01*2*PI);       y1=r*sin(k*0.01*2*PI);
  x2=r*cos((k+1)*0.01*2*PI);   y2=r*sin((k+1)*0.01*2*PI);
  line(px+round(x1), py+round(y1),px+round(x2), py+round(y2));
}
```

Example 11 Square filled with randomly selected points

The random(a, b) function draws a real number from the range [a, b).
random(10) returns a float value between 0 and 5 (starting from zero, but without 5).

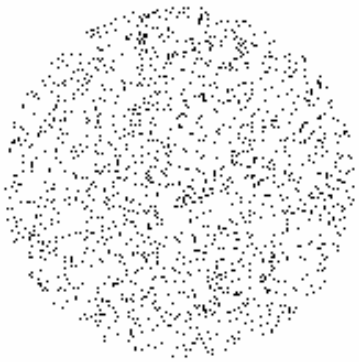
To obtain integer values, either the int() or round() function should be used.
For example, int(random(10)) will return a range from 0 to 9 and round(random(10)) from 0 to 10.
If you want to draw integers from the [n, m] range, where n, m∈**C** and n < m, you should use the "random" function as either **round(random(n-m))+m** or **int(random(n-m-1)+m**.

```
int x, y, px=100, py=100;
for (int i=0; i<3000; i++)
{
  x=round(random(150))+px;
  y=round(random(150))+py;
  point(x,y);
}
```



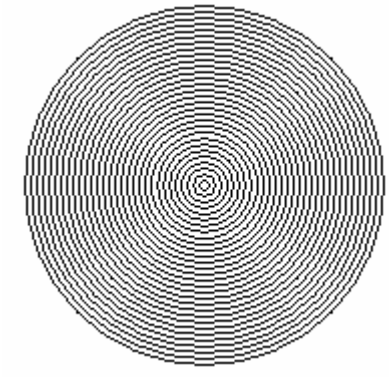
Example 12 A circle filled with randomly selected points

```
int a, b, x, y, k;
int px=200, py=200, r=80;
a=r; b=r;
for (k=1; k<3000; k++)
{
  x=round(random(160));
  y=round(random(160));

  if ((x-a)*(x-a)+(y-b)*(y-b)<=r*r)
  point(x+px,y+py);
}
```



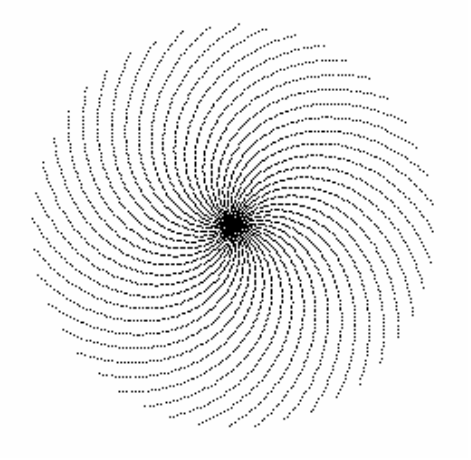
Example 13 Co-centric circles

```
int px=300, py=300, co=5;
for (int i=50; i>1; i--) circle(px,py,i*co);

void circle(int a, int b, int r)
{
  ellipse(a,b,r,r);
}
```



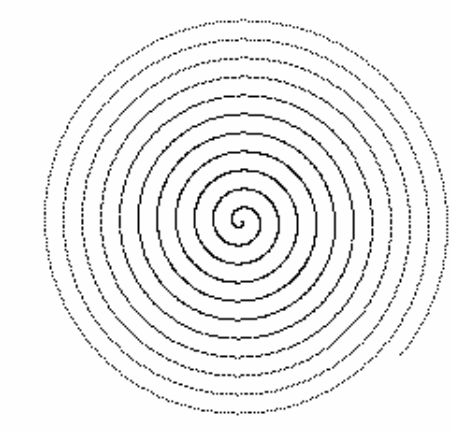
Example 14 „Spiral I”

```
int px=200, py=200;
for(float i=0; i<4000; ++i)
point(px+cos(i)*i/36, py+sin(i)*i/36);
```



Example 15 „Spiral II” (after switching from steps to radians)

```
int px=200,py=200;
for(float i=0; i<4000; ++i)
point(px+cos(radians(i))*i/30,py+sin(radians(i))*i/30);
```



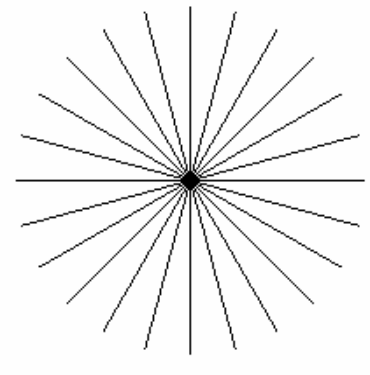
Example 16 Rosette

We will use the parametric equations of the elispa, as in example 9.

$$\begin{aligned} x &= hx \cdot \cos(kat) \\ y &= hy \cdot \sin(kat) \end{aligned} \qquad t \in [0;2\pi]$$

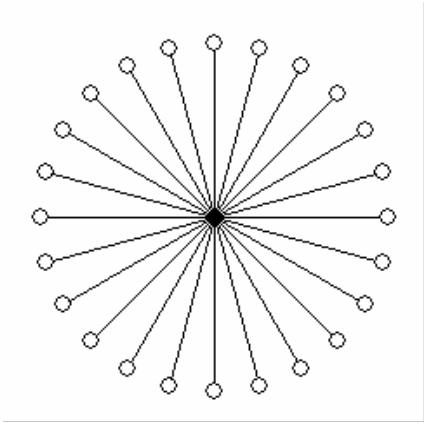
For hx = hy this is the equation of the circle.

```
int ile=12;
int px=300,py=300, hx=100, hy=100;
float kat,x,y;

kat=2*PI/ile;

for (int i=0; i<=ile; i++)
{
  x=hx*cos(kat*i);
  y=hy*sin(kat*i);
  line(px,py,px+round(x),py+round(y));
}
```



After adding:

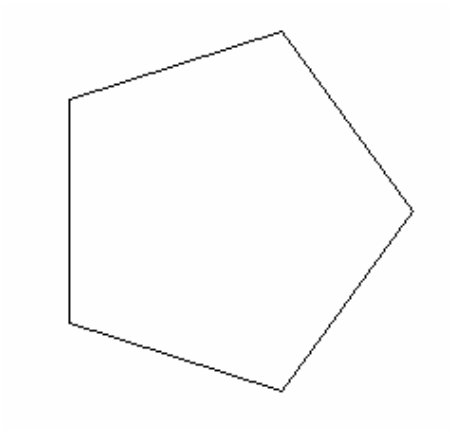
```
circle(px+round(x),py+round(y),5);
```



Example 17 Modifying the "Rosette" project we can easily obtain a drawing of a regular polygon.

Coordinates of vertices of a polygon are written in tables **tx[]** and **ty[]** respectively.

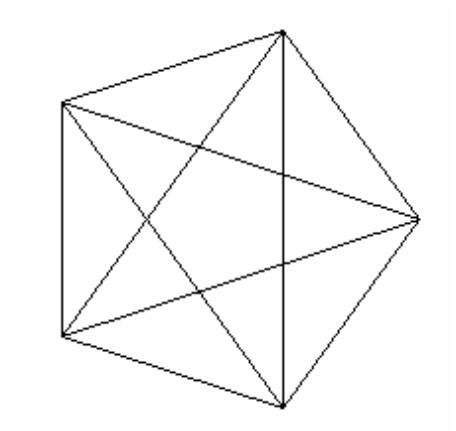
```
int ile=5;   //ile - ilość wierzchołków
int px=300,py=300, R=100;  // R - promień okręgu opisanego
int tx[]=new int[100], ty[]=new int[100];
float kat,x,y;

kat=2*PI/ile;

 for (int i=0; i<=ile; i++)
  {
    x=R*cos(kat*i);
    y=R*sin(kat*i);
    tx[i]=px+round(x);
    ty[i]=py+round(y);
}

   for (int i=0; i<ile; i++)
      line(tx[i],ty[i],tx[i+1],ty[i+1]);
```



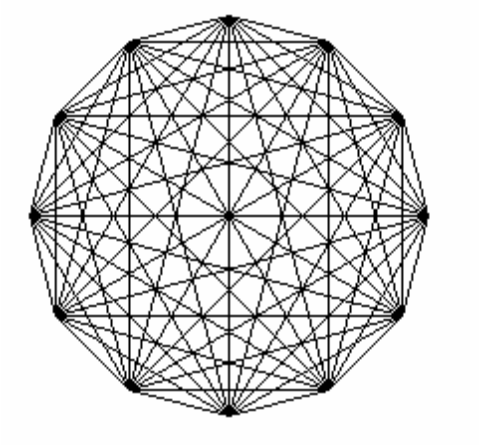
The length of the side of the polygon can be calculated from the formula **bok=2*R*sin(PI/ile).**

By remembering the coordinates in the arrays, we can connect vertices to each other.
with everyone.

```
for (int i=0; i<=ile; i++)
for (int j=i+1; j<=ile; j++)
    line(tx[i],ty[i],tx[j],ty[j]);
```



ile=5

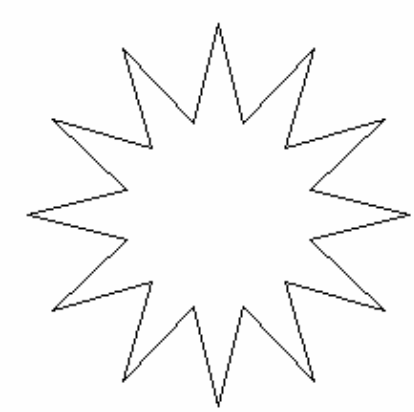


ile=12

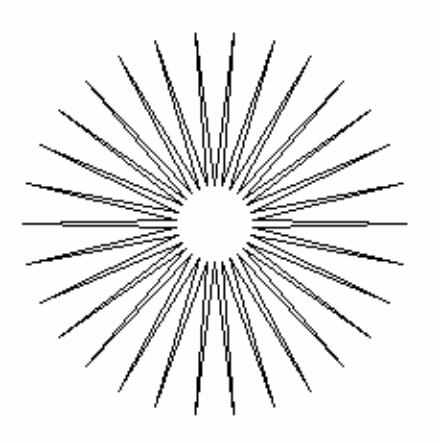
Example 18 n-armed star

```
int k,n=12;
  int px=300,py=300, hx=100, hy=100, rx=50, ry=50;
  int tx[]=new int[100], ty[]=new int[100];
  int tx2[]=new int[100], ty2[]=new int[100];
  float kat,x,y;

  kat=2*PI/n;

// Remembering the coordinates of the vertices of a "large  polygon".
     for (k=0; k<=n; k++)
     {
         x=hx*cos(kat*k);
         y=hy*sin(kat*k);
         tx[k]=px+round(x);
         ty[k]=py+round(y);
     }
// Remembering the coordinates of the vertices of the "small polygon".
     kat=0.5*kat;

    for (k=0;k<=2*n;k++)
     {
         x=rx*cos(kat*(k));
         y=ry*sin(kat*(k));
         tx2[k]=px+round(x);
         ty2[k]=py+round(y);
     }
```

```
// drawing a star
for (k=0; k<n; k++)
{
  line(tx[k],ty[k],tx2[2*k+1],ty2[2*k+1]);
  line(tx2[2*k+1],ty2[2*k+1],tx[k+1],ty[k+1]);
}
```



n=12, rx=50, ry=50



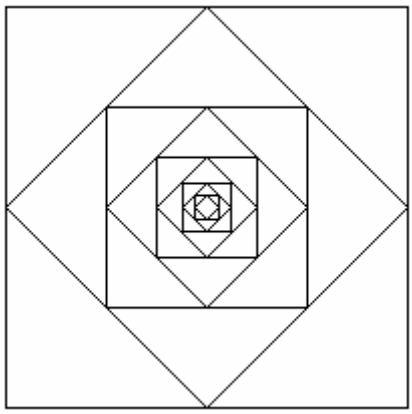
n=30, rx=20, ry=20

hx=hy - act as the radius of the circle described on the 'large polygon'
rx=ry - act as the radius of the circle described on the 'samall polygon'

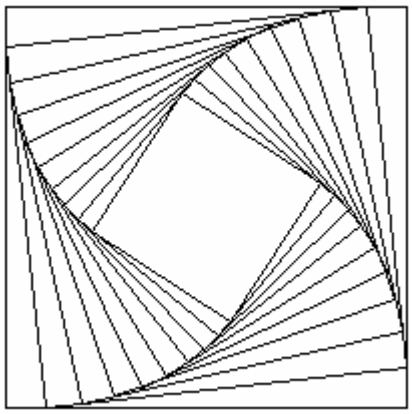
Example 19 Squares

The squares inscribed one on top of the other at the locations indicated by the variable 'skok'.Two wx[5] and wy[5] arrays should be prepared to store x and y coordinates of vertices of subsequent squares respectively. The array contains 5 elements, because it is necessary to connect vertex no. 1 with no. 2, no. 2 with no. 3, no. 3 with no. 4 and at the end of no. 4 with no. 1 (its role is fulfilled by no. 5, because the coordinates of point no. 1 are changed earlier). (x, y) is the left, upper point of the first, largest square.

```
int bok=200, x=100, y=100, ile=10;
int i,j,px=50,py=50;
float wx[]=new float[6], wy[]=new float[6];
float skok=0.1;
wx[1]=x; wy[1]=y;
wx[2]=x; wy[2]=y+bok;
wx[3]=x+bok; wy[3]=y+bok;
wx[4]=x+bok; wy[4]=y;
wx[5]=x;  wy[5]=y;

for (i=1;i<=ile;i++)
{
  for (j=1;j<=4;j++)
     line(px+round(wx[j]),py+round(wy[j]),
                    px+round(wx[j+1]),py+round(wy[j+1]));

  for (j=1;j<=4;j++)
  {
    wx[j]=(1-skok)*wx[j]+skok*wx[j+1];
    wy[j]=(1-skok)*wy[j]+skok*wy[j+1];
  }
 wx[5]=wx[1]; wy[5]=wy[1];
 }
```



`skok=0.5`



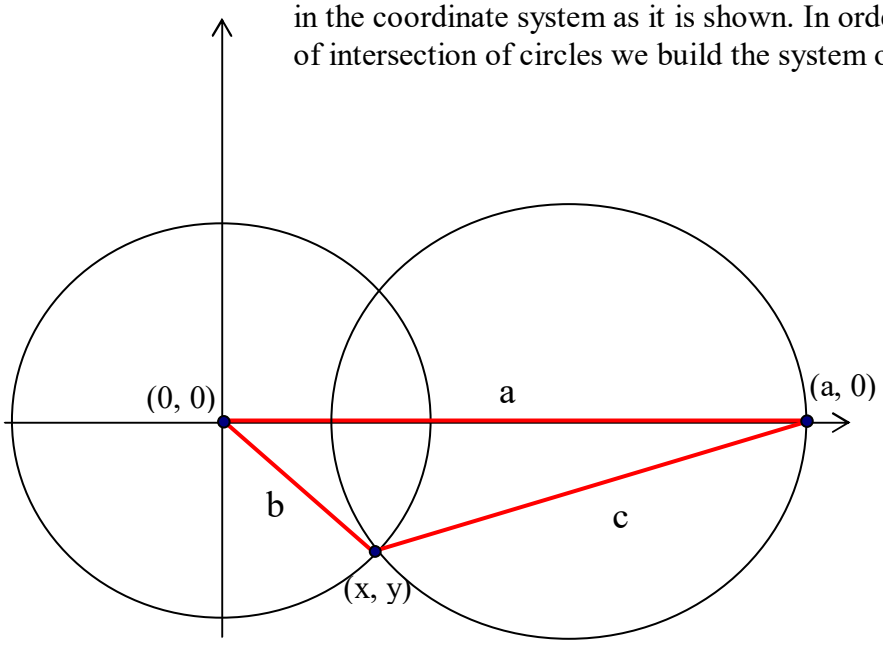
`skok=0.1`

Example 20 Triangle

The **`triangle(x1, y1, x2, y2, y2, x3, y3)`** procedure allows to draw a triangle.
Its parameters are the coordinates x, y of the vertices of the triangle.

How to draw a triangle if you only know the length of the sides?

We are looking for the intersection of the respective circles.
We can greatly simplify the calculation, which will have practically no impact on the resulting figure, because we will translate (px, py), placing the circles in the coordinate system as it is shown. In order to find the coordinates of the point of intersection of circles we build the system of equations.



$$\begin{cases} x^2 + y^2 = b^2 \\ (x - a)^2 + y^2 = c^2 \end{cases}$$

$$\begin{cases} x^2 + y^2 = b^2 \\ x^2 - 2ax + a^2 + y^2 = c^2 \end{cases}$$

After subtracting the equations pages:

$$2ax - a^2 = b^2 - c^2$$
$$2ax = a^2 + b^2 - c^2$$
$$x = (a^2 + b^2 - c^2) / (2a)$$

For data a=5, b=4, c=3 we will get x=3,2 and y= 2,4 (+2,4)

For a=1, b=1, c=1 we get x=0.5 and y= 0.866025 (+0.866025)

ALGORITHM

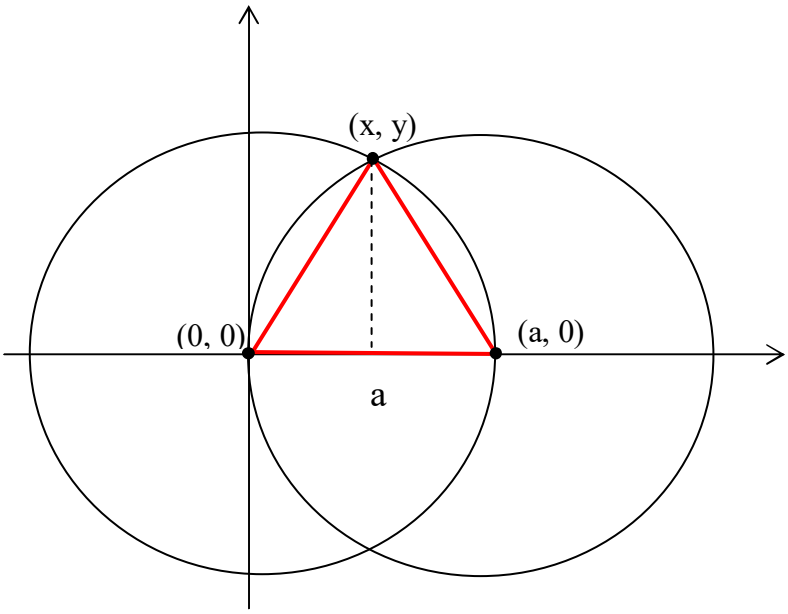
1) Check whether the three numbers may be the lengths of the sides of a triangle.
If not, give the message and finish the program.

2) We find the longest side. From it we start drawing a triangle by drawing this side in orientation.
and assuming appropriate scaling of the drawing.

3) Draw the other two sides of the triangle by calculating the coordinates of the apex connecting them.

```
float a,b,c, d, maks, x, y, sa, sb, sc, sx, sy;
int px, py;

// data
a=3; b=4; c=5;

// the condition of the triangle
if ((a>=b+c) || (b>=a+c) || (c>=a+c))
println("This is not a triangle!");
else
{

// translation
px=200; py=200;

// max
if (a>=b && a>=c) maks=a;
else
if (b>=a && b>=c) maks=b;
else maks=c;
if (maks==b) {d=a; a=b; b=d;}
else
if (maks==c) {d=a; a=c; c=d;};

// the point of intersection of the circles
x=(a*a+b*b-c*c)/(2*a);
y=sqrt(abs(b*b-x*x));

// scaling
sa=300; sb=sa*b/a; sc=sa*c/a;
sx=sa*x/a; sy=sa*y/a;

// drawing
line(px,py, px+round(sa), py);
line(px,py, px+round(sx), py+round(sy));
line(px+round(sa), py, px+round(sx), py+round(sy));
}
```

Example 21 Equilateral triangle



$x = a/2$

$x^2 + y^2 = a^2$

$y^2 = a^2 - x^2 = a^2 - a^2/4 = 3a^2/4$

$y = \text{sqrt}(3a^2/4) = a \cdot \text{sqrt}(3)/2$

```
int px, py;
float a;
px=200; py=300; a=200;
line(px,py,px+round(a),py);
line(px,py,px+round(a/2),py-round(a*sqrt(3)/2));
line(px+round(a),py,px+round(a/2),py-round(a*sqrt(3)/2));
```

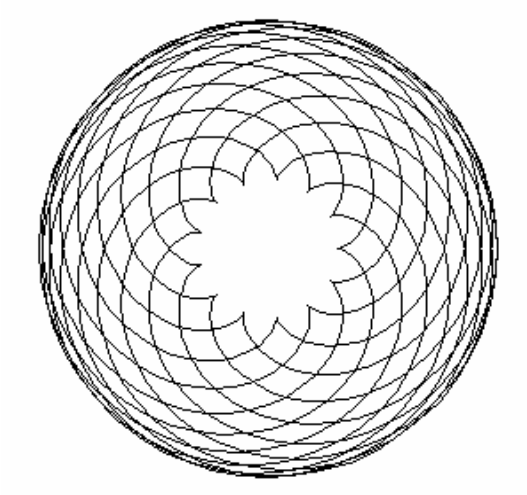
Example 22 Epicykloid I

The epicloid is delineated by a fixed point in a circle running outside another circle, which is stationary. The shape of the epicycloid depends on the ratio R/y of the radius of the circles, stationary to rolling.
Parametric equations of the epicycloid:

$$x = (R+r)\cos(t) - r\cos\left(\frac{R+r}{r}t\right)$$

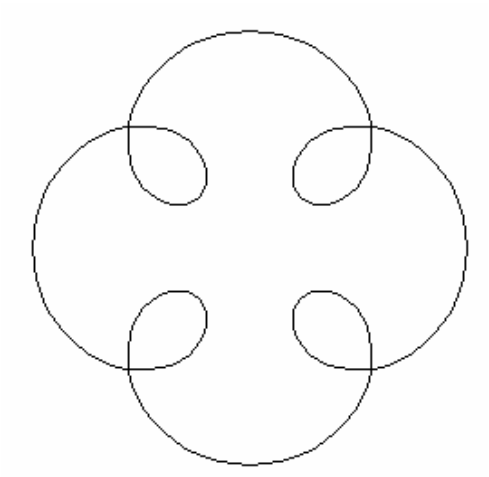
$$y = (R+r)\sin(t) - r\sin\left(\frac{R+r}{r}t\right)$$

```
float t,r1=26*1.5, r2=30*1.5, epsilon=0.1, krok=PI/36;
float x1, x2, y1, y2;
int px=300, py=200;

t=krok;

 x1= (r1+r2)*cos(r2*t/r1)-r2*cos((r1+r2)*t/r1);
 y2= (r1+r2)*sin(r2*t/r1)-r2*sin((r1+r2)*t/r1);
while (sqrt((x1-r1)*(x1-r1)+y1*y1)>=epsilon)
{
 x2= (r1+r2)*cos(r2*t/r1)-r2*cos((r1+r2)*t/r1);
 y2= (r1+r2)*sin(r2*t/r1)-r2*sin((r1+r2)*t/r1);
 line(px+round(x1), py+round(y1),px+round(x2),py+round(y2));
 t=t+krok;x1=x2; y1=y2;
}
```



Example 23 Epicykloid II

```
int px=200, py=200, r=80;
float t, x1, y1, x2, y2;
for (int k=0; k<100; k++)
{
  x1=r*cos(k*0.01*2*PI);       y1=r*sin(k*0.01*2*PI);
  x2=r*cos((k+1)*0.01*2*PI);   y2=r*sin((k+1)*0.01*2*PI);

  x1=x1+r/2*cos(5*k*0.01*2*PI);
  y1=y1+r/2*sin(5*k*0.01*2*PI);

  x2=x2+r/2*cos(5*(k+1)*0.01*2*PI);
  y2=y2+r/2*sin(5*(k+1)*0.01*2*PI);
   line(px+round(x1), py+round(y1),px+round(x2), py+round(y2));
}
```



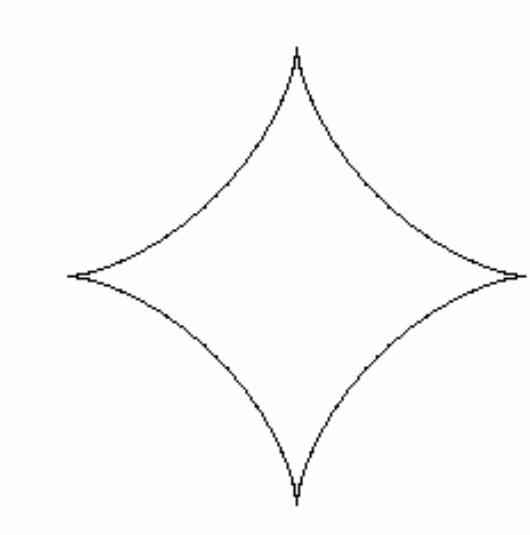
Example 24 Asteroid

Asteroid is an example of a hypocycloid, i.e. a curve that is outlined by a fixed point of a rolling circle. without slippage inside a circle with a larger radius, as for an epicycloid.

Example of parametric equations:

$$x = a \cdot \cos^3(t)$$
$$y = a \cdot \sin^3(t)$$

```
int k, px=200, py=200, a=100;
float t1, t2, x1, y1, x2, y2;
for (k=0; k<800; k++)
{
  t1=k*0.01*PI;
  x1=a*cos(t1/4)*cos(t1/4)*cos(t1/4);
  y1=a*sin(t1/4)*sin(t1/4)*sin(t1/4);

  t2=(k+1)*0.01*PI;
  x2=a*cos(t2/4)*cos(t2/4)*cos(t2/4);
  y2=a*sin(t2/4)*sin(t2/4)*sin(t2/4);

  line(px+round(x1),py+round(y1),px+round(x2),py+round(y2));
```



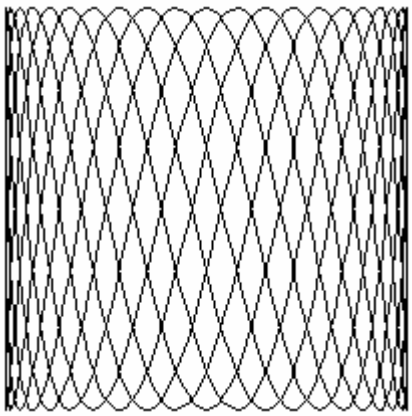
Example 25 Curve Lissajous

Lissajous or Bowditch curve - a parametric curve drawn by a material point that performs harmonic vibrations in two mutually perpendicular directions.

Example of parametric equations:

$$x = a \cdot \sin(t)$$
$$y = a \cdot \sin(bt + c)$$

```
int k, px=200, py=200;
float a, b, c, t1, t2, x1, y1, x2, y2;

a=100; b=4.2; c=PI/2;
for (k=0; k<1000; k++)
{
 t1=k*0.01*PI;
 x1=a*sin(t1);
 y1=a*sin(b*t1+c);

 t2=(k+1)*0.01*PI;
 x2=a*sin(t2);
 y2=a*sin(b*t2+c);

line(px+round(x1),py+round(y1),px+round(x2),py+round(y2));
}
```



Example 26 Butterfly curve (transcendent) Criminal flat curve discovered by Temple H. Faya.

$$x = \sin(t)\left(e^{\cos(t)} - 2\cos(4t) - \sin^5\left(\frac{t}{12}\right)\right)$$
$$y = \cos(t)\left(e^{\cos(t)} - 2\cos(4t) - \sin^5\left(\frac{t}{12}\right)\right)$$

.

```
int k, px, py, hx, hy;
float x1, y1, t1, x2, y2, t2;
px=200; py=200; hx=30; hy=30;

for (k=0; k<1500; k++)
{
 t1=k*0.01*PI;
 x1=hx*sin(t1)*(exp(cos(t1))-2*cos(4*t1)-pow(sin(t1/12),5));
 y1=hy*cos(t1)*(exp(cos(t1))-2*cos(4*t1)-pow(sin(t1/12),5));

 t2=(k+1)*0.01*PI;
 x2=hx*sin(t2)*(exp(cos(t2))-2*cos(4*t2)-pow(sin(t2/12),5));
 y2=hy*cos(t2)*(exp(cos(t2))-2*cos(4*t2)-pow(sin(t2/12),5));

 line(px+round(x1),py+round(y1),px+round(x2),py+round(y2));
}
```

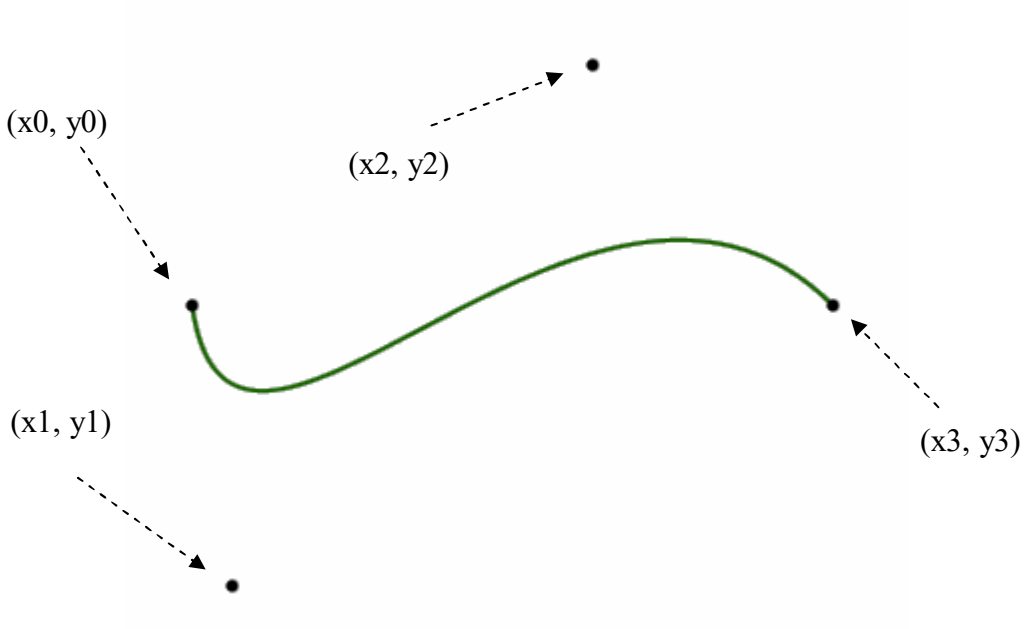
Example 27 Curve Bezier.

The name "Bézier curve" of degree n means the representation of a parametric curve in the form

of a sequence of points $p_0,....,p_n$, so called checkpoints, which should be substituted for the formula $p(t)=\sum_{i=0}^{n} p_i B_i^n(t)$ Its functions B_i^n are Bernstein polynomials of degree n, defined by the formula $B_i^n(t)=\binom{n}{i} t^i (1-t)^{n-i}$ There is also a Catmull-Roma curie. Curve determination algorithm performs the **curve()** function. and analogously to the **beziert()** function requires the determination of the coordinates

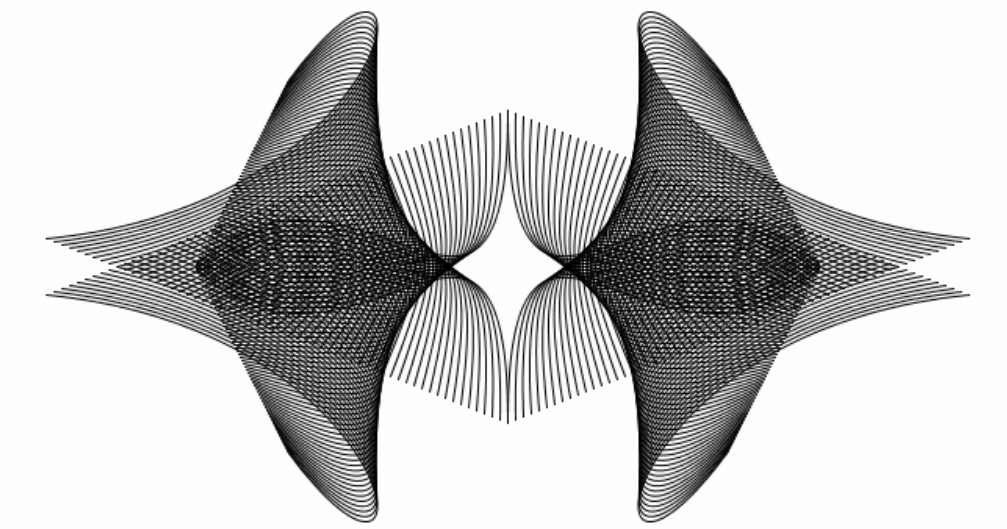
of four points by which a smooth interpolating curve is conducted.

```
bezier(x0, y0, x1, y1, x2, y2, x3, y3)
```




```
int px=200, py=200;
int x0, y0, x1, y1, x2, y2, x3, y3;

     x0=0; y0=20;
     x1=20; y1=160;
     x2=200; y2=-100;
     x3=320; y3=20;

  bezier(px+x0,py+y0,px+x1,py+y1,px+x2,py+y2,px+x3,py+y3);

  // dodatkowo zaznaczono punkty będące parametrami funkcji
     point(px+x0,py+y0);
     point(px+x1,py+y1);
     point(px+x2,py+y2);
     point(px+x3,py+y3);
```

After the modification, the following drawing can be obtained:



```
int k, px=400, py=200;
 int x0, y0, x1, y1, x2, y2, x3, y3;

 for (k=0; k<60; k++)
 {
    x0=5*k;   y0=-100+2*k;
    x1=k;     y1=100-k;
    x2=100-k; y2=-100+k*6;
    x3=200-k; y3=2*k;

    bezier(px+x0,py+y0,px+x1,py+y1,px+x2,py+y2,px+x3,py+y3);
    bezier(px+x0,py-y0,px+x1,py-y1,px+x2,py-y2,px+x3,py-y3);
    bezier(px-x0,py+y0,px-x1,py+y1,px-x2,py+y2,px-x3,py+y3);
    bezier(px-x0,py-y0,px-x1,py-y1,px-x2,py-y2,px-x3,py-y3);
 }
```

At the end an interesting example of graphics made in text mode using the "**x**" sign.