GRAFIKA 2D

Przykład 1. Odcinki poziome

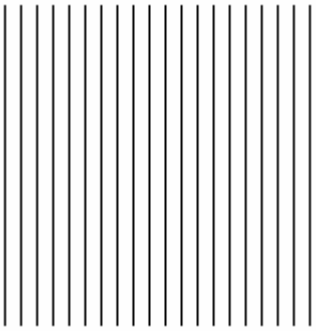
Odcinek nie ma początku i końca, ma dwa końce! („każdy kij ma dwa końce"), ale na nasz użytek pierwszy z podawanych punktów będziemy nazywać punktem początkowym, a drugi punktem końcowym, co jest zresztą zgodne ze sposobem kreślenia. Oczywiście kolejność tych punktów jest nieistotna.
W celu uzyskania odcinków poziomych dla punktu początkowego i końcowego pozostawiamy niezmienione współrzędne x (odcięte), a współrzędne y (rzędne) zwiększamy z ustalonym przyrostem np. co 8 pikseli.
Rysunek w stosunku do początku układu współrzędnych ekranowych jest przesunięty (translacja) o wektor [px; py], w tym przykładzie to [100; 100].
Wartość 160 jest ustalona na podstawie ilości powtórzeń pętli: 20 * 8 = 160.

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

Przykład 2. Odcinki pionowe

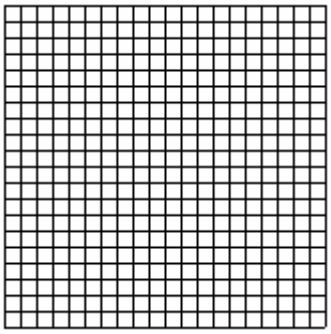
W celu uzyskania odcinków poziomych dla punktu początkowego i końcowego pozostawiamy niezmienione współrzędne y (rzędne), a współrzędne x (odcięte) zwiększamy z ustalonym przyrostem np. co 8 pikseli.

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



Po połączeniu odcinków poziomych z pionowymi otrzymamy siatkę.

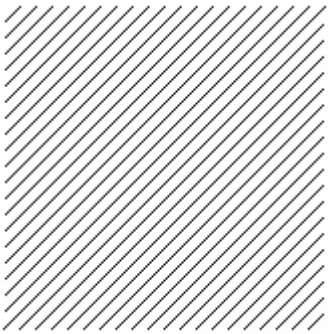
```
int i, px=100, py=100;
for (i=0; i<20; i++)
{
  line(px,py+i*5,px+100,py+i*5);
  line(px+i*5,py,px+i*5,py+100);
}
```



Przykład 3. Odcinki ukośne prawe

Odcinki ukośne rysowane pod kątem 45° uzyskamy zwiększając współrzędne x punktu początkowego (rzędna stała) i y punktu końcowego (odcięta stała) z tym samym odstępem.
Instrukcja „**line**” kreśli jedynie połowę zaplanowanego rysunku, dlatego trzeba zastosować ją dwukrotnie.

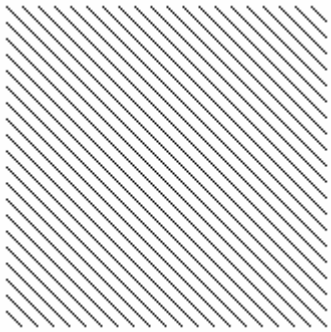
```
int i, px=100, py=100;
for (i=0; i<20; i++)
{
  line(px+i*8,py+160,px+160,py+i*8);
  line(px+i*8,py,px,py+i*8);
}
```



Przykład 4. Odcinki ukośne lewe

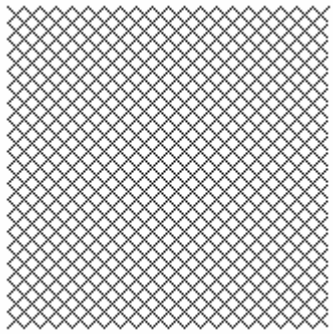
Odcinki ukośne lewe rysowane pod kątem 45° uzyskamy podobnie jak w przykładzie poprzednim, ale zwiększając współrzędne y punktu początkowego (odcięta stała) i x punktu końcowego (rzędna stała). Tak, jak poprzednio, instrukcję „**line**” trzeba zastosować dwukrotnie.

```
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);
}
```



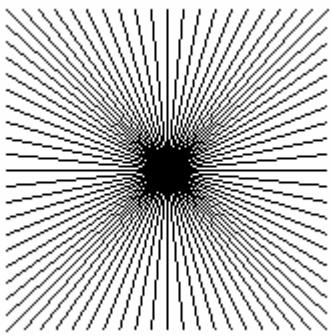
Po połączeniu obu rodzajów odcinków otrzymamy siatkę.

```
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);
}
```



A po niewielkiej modyfikacji:

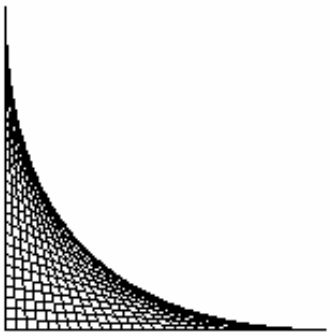
```
int i, px=100, py=100;
for (i=0; i<21; i++)
{
  line(px,py+i*8,px+160,py+160-i*8);
  line(px+i*8,py,px+160-i*8,py+160);
}
```



Przykład 5. „Krzywa z prostych”

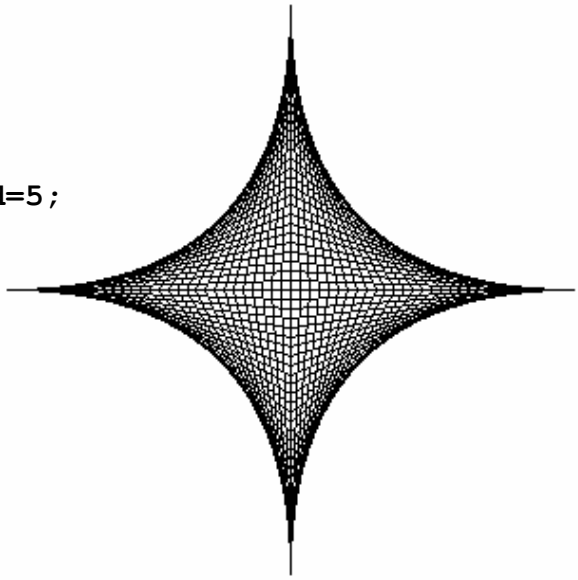
Rysując odpowiednio tylko same odcinki uzyskamy wrażenie powstania łuku.
Odcinki kreślimy naprzemienne dla punktów początkowych leżących na linii pionowej i końcowych leżących na linii poziomej z ustalonym odstępem np. co 5 pikseli.
Wartość 150 jest ustalona na podstawie ilości powtórzeń pętli: 30 * 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);
}
```



Modyfikujemy teraz ten kod na trzy pozostałe orientacje układu współrzędnych na płaszczyźnie:

```
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);
 }
```



Przykład 6. Szachownica

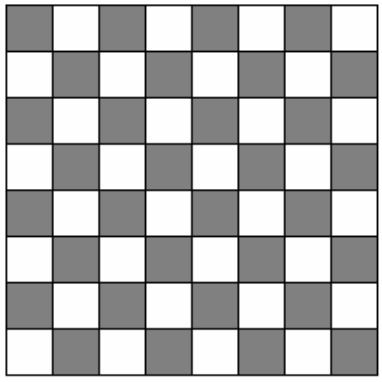
Sprawdzając parzystość wyrażenia **wiersz+kolumna** możemy ustalić kolor wypełnienia prostokąta.
Dla skrócenia zapisu w wyjaśnieniu „**w**” będzie oznaczać zmienną „wiersz”, a „**k**” zmienna „kolumna”.

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

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

fill(150, 150, 150) – ustala kolor szary, fill(255, 255, 255) – kolor biały,

```
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);
}
```



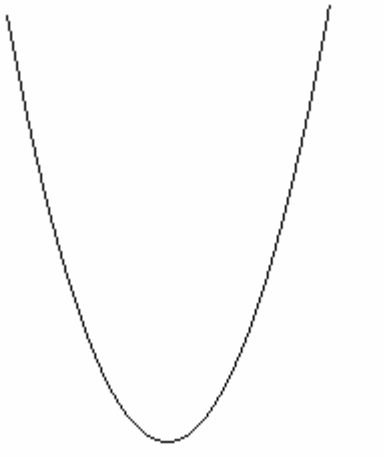
Przykład 7. Wykres funkcji cosinus wykonany przy pomocy odcinków

```
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)));
}
```

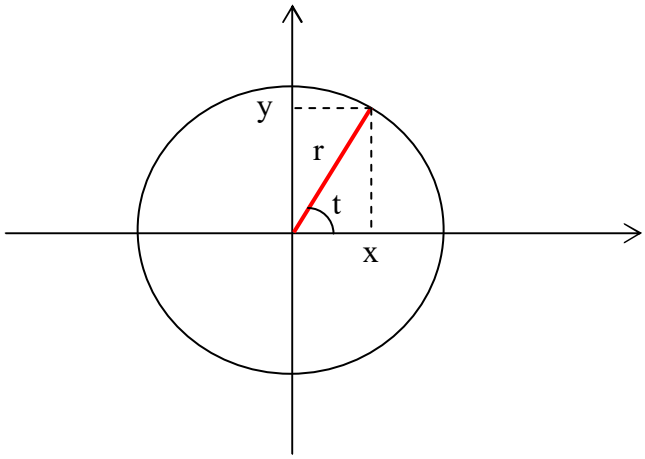
Przykład 8. Parabola narysowana przy pomocy odcinków

Parabola o wierzchołku w początku układu współrzędnych, której osią symetrii jest oś y, określona jest wzorem $y = x^2$.

```
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));
}
```



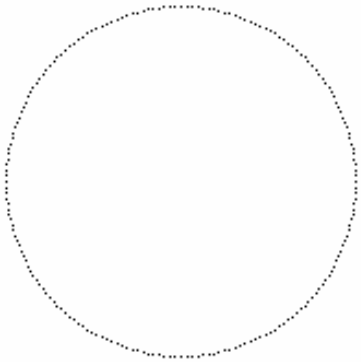
Przykład 9. Okrąg narysowany przy pomocy punktów



Równania parametryczne okręgu:

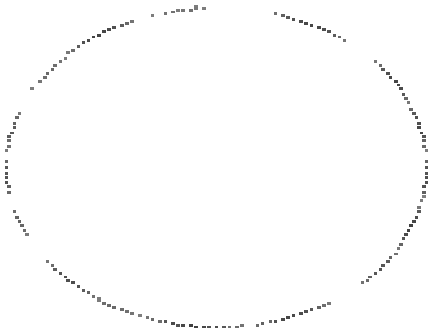
$$x = r \cdot \cos(t)$$
$$y = r \cdot \sin(t) \qquad t \in [0;2\pi]$$

```
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));
}
```



Dla różnych współczynników R ≠ r w równaniach parametrycznych otrzymamy elipsę.

```
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));
}
```



Przykład 10. Okrąg narysowany przy pomocy odcinków

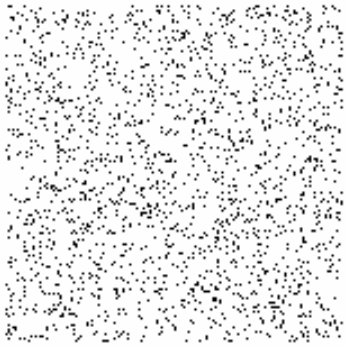
```
int px=200, py=200, r=80;
float t, x1, y1, x2, y2;
for (int k=0; k<100; k++)
{
  x1=r*cos(k*0.01*2*PI);       y1=r*sin(k*0.01*2*PI);
  x2=r*cos((k+1)*0.01*2*PI);   y2=r*sin((k+1)*0.01*2*PI);
  line(px+round(x1), py+round(y1),px+round(x2), py+round(y2));
}
```

Przykład 11. Kwadrat wypełniony losowo wybranymi punktami

Funkcja random(a, b) losuje liczbę rzeczywistą z zakresu [a, b).
random(10) zwróci wartość „float" zawartą pomiędzy 0 i 5 (startując od zera, ale bez 5).

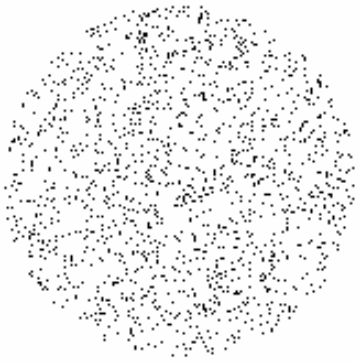
W celu uzyskania wartości całkowitych należy zastosować funkcję int() albo round().
Np. int(random(10)) zwróci wynik z zakresu od 0 do 9, natomiast round(random(10)) od 0 do 10.
Gdybyśmy chcieli losować liczby całkowite z zakresu [n, m], gdzie n, m$\in$**C** i n < m, należałoby użyć funkcji „random" w postaci **round(random(n-m))+m** albo **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);
}
```



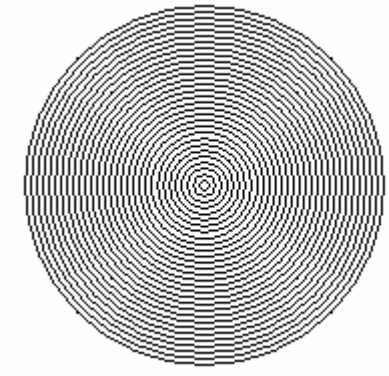
Przykład 12. Koło wypełnione losowo wybranymi punktami

```
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);
}
```



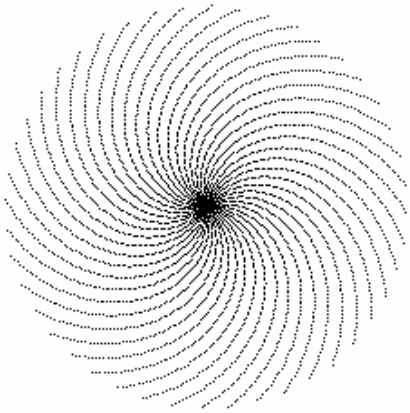
Przykład 13. Okręgi koncentryczne

```
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);
}
```



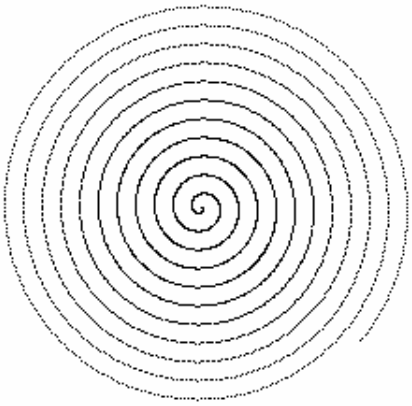
Przykład 14. „Spirala I”

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



Przykład 15. „Spirala II” (po zamianie stopni na radiany)

```
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);
```



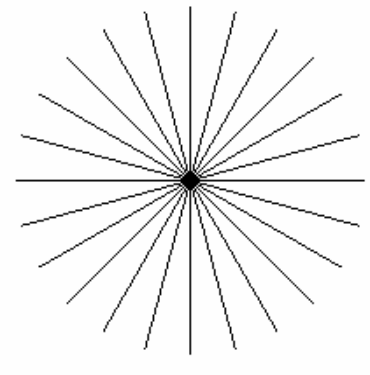
Przykład 16. Rozeta

Wykorzystamy równania parametryczne elispy, tak jak w przykładzie 9.

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

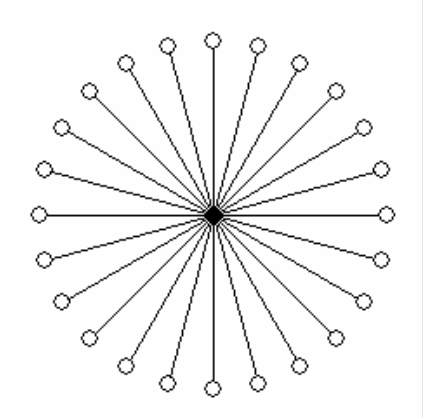
Dla hx = hy jest to równanie okręgu.

```
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));
}
```



Po dopisaniu:

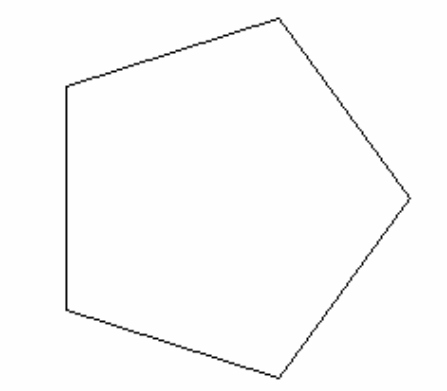
```
circle(px+round(x),py+round(y),5);
```



Przykład 17. Modyfikując projekt „Rozeta” możemy w łatwy sposób uzyskać rysunek wielokąta foremnego.

Współrzędne wierzchołków wielokąta zapisujemy odpowiednio w tablicach **tx[]** i **ty[]**.

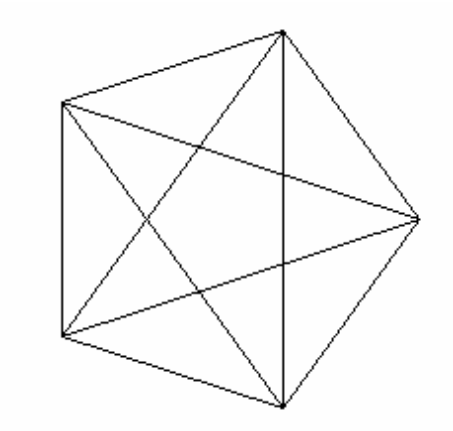
```
int ile=5;   //ile - ilość wierzchołków
int px=300,py=300, R=100;  // R - promień okręgu opisanego
int tx[]=new int[100], ty[]=new int[100];
float kat,x,y;

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]);
```



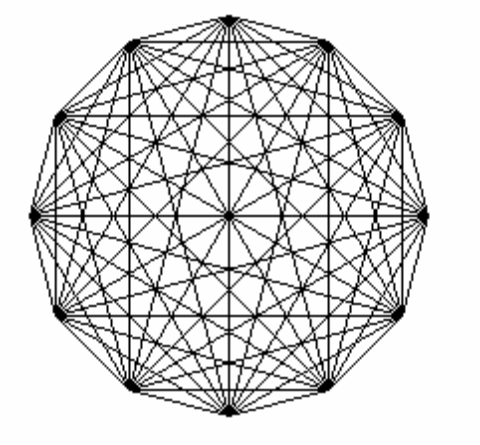
Długość boku wielokąta możemy obliczyć ze wzoru **bok=2*R*sin(PI/ile).**

Dzięki zapamiętaniu współrzędnych w tablicach, możemy wykonać połączenie wierzchołków każdy z każdym.

```
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

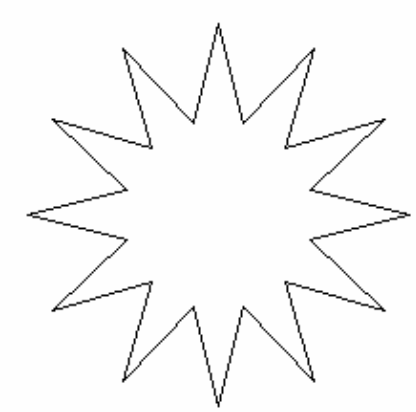


ile=12

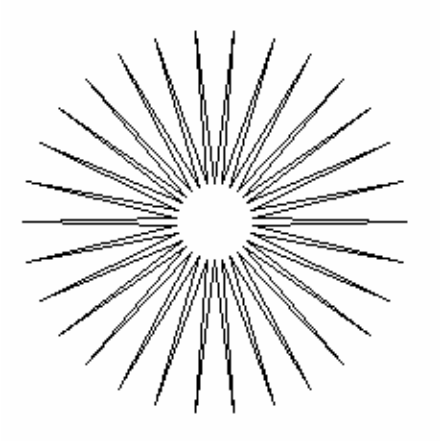
Przykład 18. Gwiazda n-ramienna

```
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;

    // zapamiętanie współrzędnych wierzchołków „dużego wielokąta"
    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);
    }
    // zapamiętanie współrzędnych wierzchołków „małego wielokąta"
    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);
    }
```

```
// rysowanie gwiazdy
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



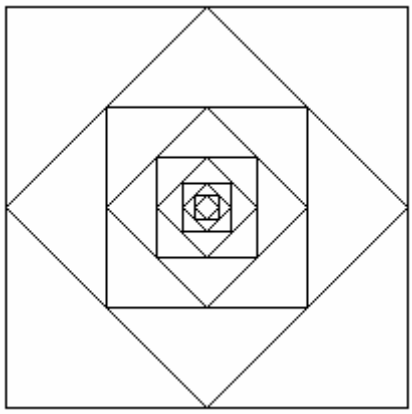
n=30, rx=20, ry=20

hx=hy - spełniają rolę promienia okręgu opisanego na „dużym wielokącie”;
rx=ry - spełniają rolę promienia okręgu opisanego na „małym wielokącie”.

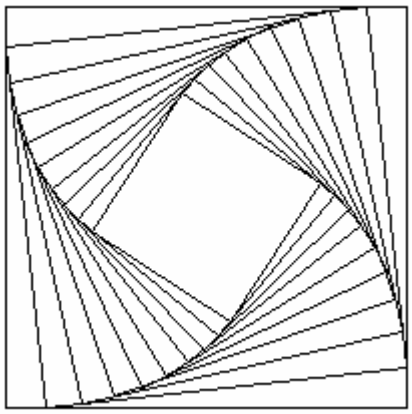
Przykład 19. Kwadraty

Kwadraty wpisane jeden w drugi w miejscach wskazanych przez zmienną „skok”.
Należy przygotować dwie tablice wx[5] i wy[5] przechowujące odpowiednio współrzędne x i y wierzchołków kolejnych kwadratów. Tablica zawiera 5 elementów, gdyż trzeba połączyć wierzchołek nr 1 z nr 2, nr 2 z nr 3, nr 3 z nr 4 i na koniec nr 4 z nr 1 (jego rolę spełnia właśnie nr 5, bo współrzędne punktu nr 1 ulegają wcześniej zmianie). (x, y) to lewy, górny wierzchołek pierwszego, największego kwadratu.

```
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`



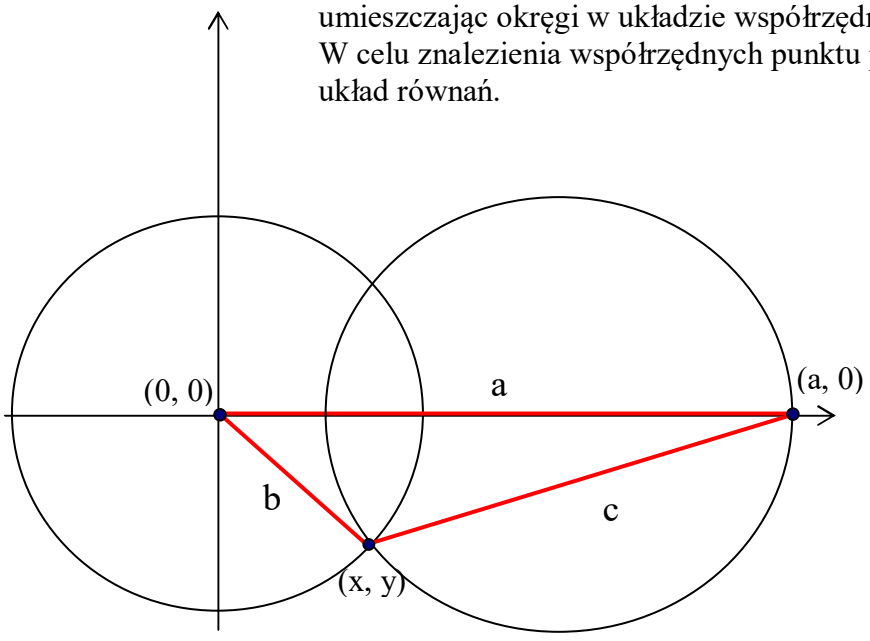
`skok=0.1`

Przykład 20. Trójkąt

Procedura `triangle(x1, y1, x2, y2, x3, y3)` pozwala na narysowanie trójkąta. Jej parametrami są współrzędne x, y wierzchołków trójkąta.

Jak narysować trójkąt, jeśli znamy tylko długości boków?

Szukamy punktu przecięcia się odpowiednich okręgów.
Możemy znacznie uprościć obliczenia, co nie będzie miało praktycznie żadnego wpływu na powstający rysunek, gdyż dokonamy translacji (px, py), umieszczając okręgi w układzie współrzędnych tak, jak jest to przedstawione.
W celu znalezienia współrzędnych punktu przecięcia się okręgów budujemy układ równań.



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

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

Po odjęciu równań stronami:

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

Dla danych a=5, b=4, c=3 otrzymamy x=3,2 i y= –2,4 (+2,4)

Dla a=1, b=1, c=1 otrzymamy x=0,5 i y= –0,866025 (+0,866025)

ALGORYTM

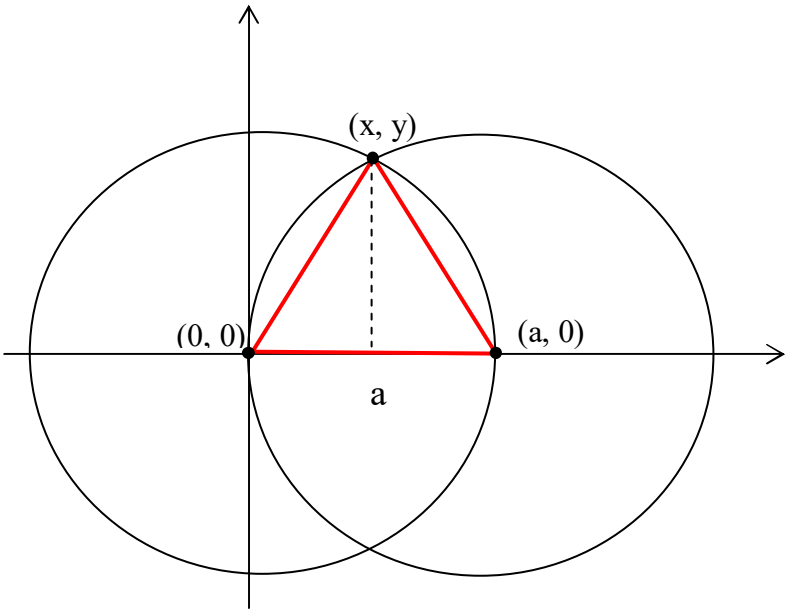
1) Sprawdzamy, czy podane trzy liczby mogą być długościami boków jakiegoś trójkąta.
   Jeśli nie, to podajemy komunikat i kończymy program.

2) Znajdujemy najdłuższy bok. Od niego rozpoczniemy kreślenie trójkąta rysując ten bok w orientacji poziomej i przyjmując odpowiednie skalowanie rysunku.

3) Dorysowujemy pozostałe dwa boki trójkąta obliczając współrzędne wierzchołka je łączącego.

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

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

// warunek trójkąta
if ((a>=b+c) || (b>=a+c) || (c>=a+c))
println("to nie jest trójkąt!");
else
{

// przesunięcie
px=200; py=200;

// maks
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;};

// punkt przecięcia okręgów
x=(a*a+b*b-c*c)/(2*a);
y=sqrt(abs(b*b-x*x));

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

// kreślenie
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));
}
```

Przykład 21. Trójkąt równoboczny



$x = a/2$

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

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

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

```
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));
```

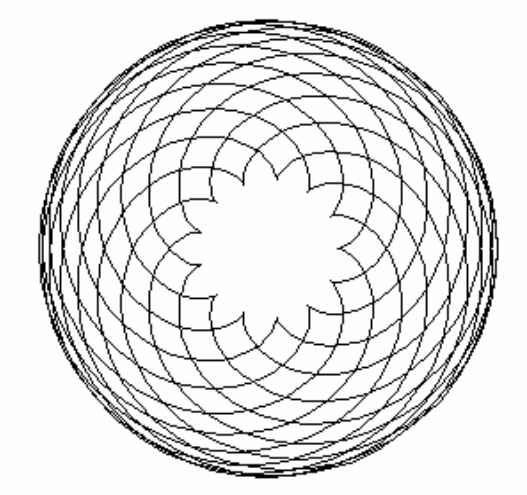
Przykład 22. Epicykloida I

Epicykloidę zakreśla ustalony punkt okręgu toczącego się na zewnątrz innego okręgu, który jest nieruchomy. Kształt epicykloidy zależy od stosunku R/r promieni okręgów, nieruchomego do toczącego się. Równania parametryczne epicykloidy:

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

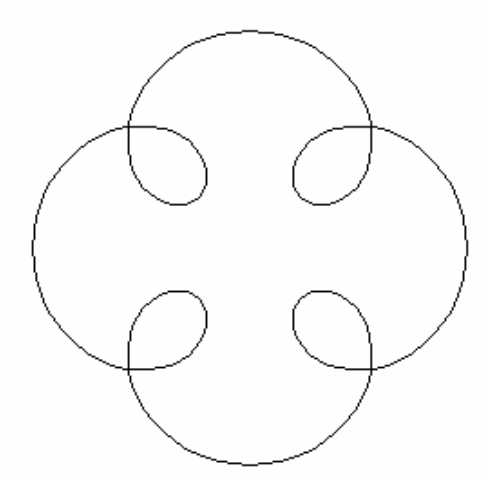
$$y = (R+r)\sin(t) - r\sin\left(\frac{R+r}{r}t\right)$$

```
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;
}
```



Przykład 23. Epicykloida 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));
}
```



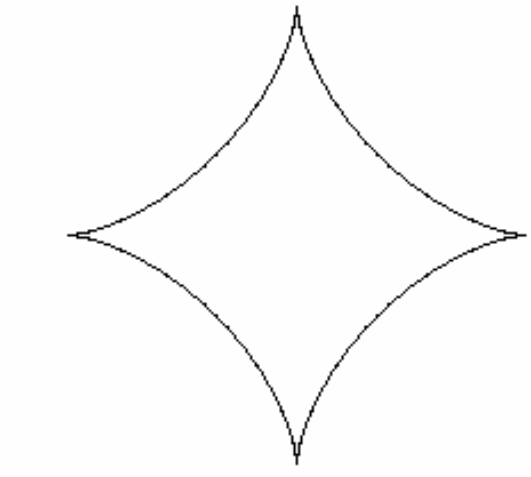
Przykład 24. Asteroida

Asteroida jest przykładem hipocykloidy czyli krzywej, jaką zakreśla ustalony punkt okręgu toczącego się bez poślizgu wewnątrz okręgu o większym promieniu, podobnie jak dla epicykloidy.

Przykład równań parametrycznych:

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

```
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));
```



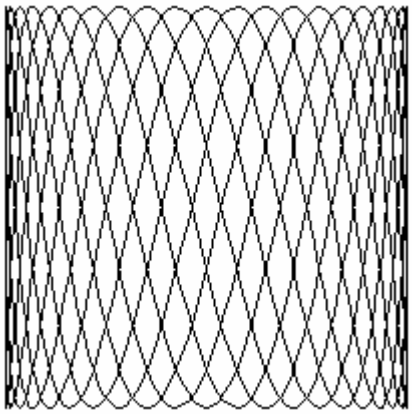
Przykład 25. Krzywa Lissajous

Krzywa Lissajous bądź Bowditcha – krzywa parametryczna wykreślona przez punkt materialny wykonujący drgania harmoniczne w dwóch wzajemnie prostopadłych kierunkach.

Przykład równań parametrycznych:

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

```
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));
}
```



Przykład 26. Krzywa motylkowa (transcendentalna)
Przestępna krzywa płaska odkryta przez 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));
}
```

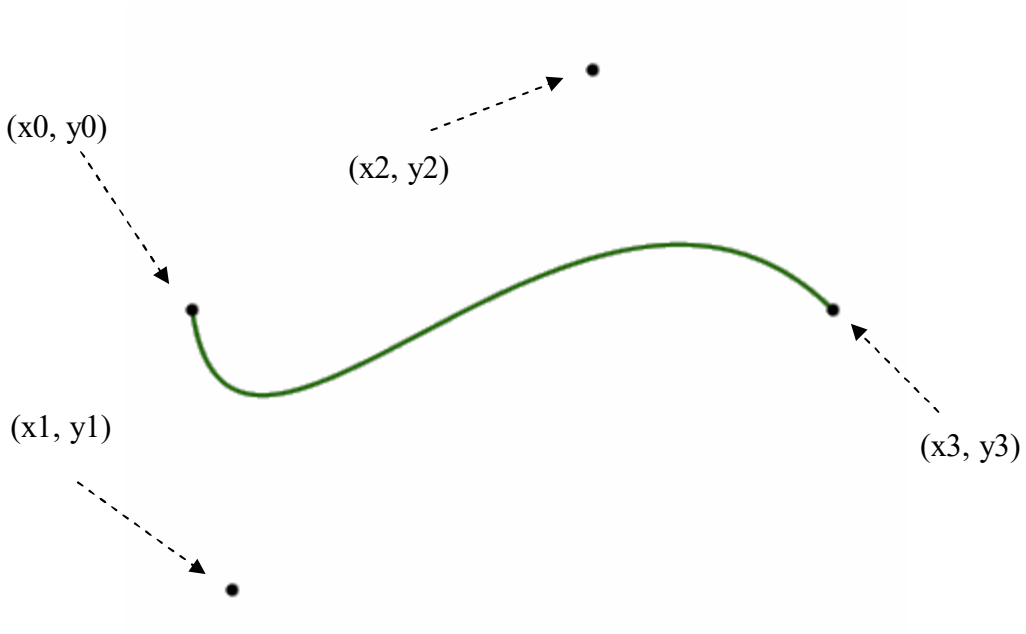
Przykład 27. Krzywa Beziera.

Nazwa ,,krzywa Béziera" stopnia n oznacza reprezentację krzywej parametrycznej w postaci ciągu punktów $p_0,\ldots,p_n$, tzw. punktów kontrolnych, które należy podstawić do wzoru $p(t)=\sum_{i=0}^{n} p_i B_i^n(t)$. Występujące w nim funkcje B_i^n to wielomiany Bernsteina stopnia n, określone wzorem $B_i^n(t)=\binom{n}{i}t^i(1-t)^{n-i}$

Istnieje też krzywa Catmull-Roma Algorytm wyznaczania krzywej realizuje funkcja **curve()**

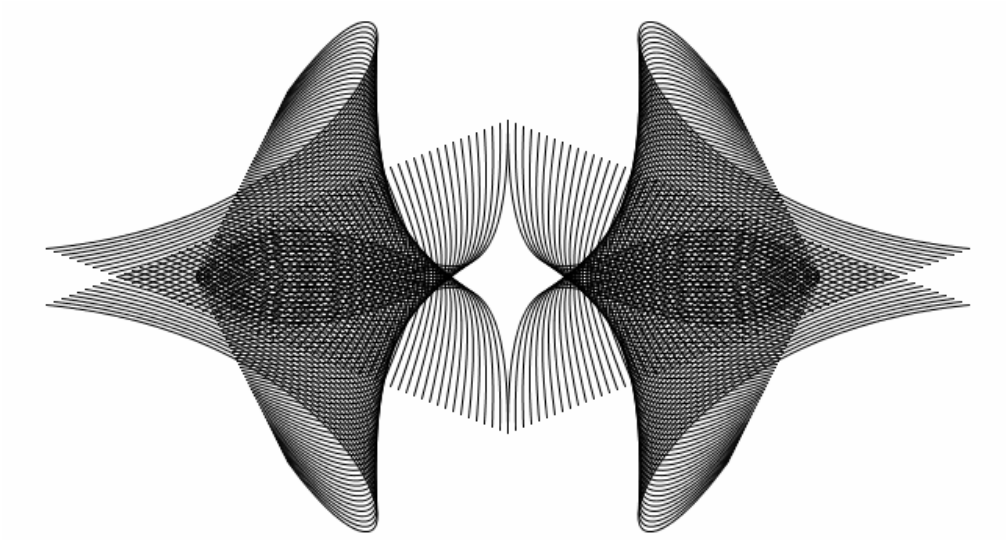
i analogicznie jak dla funkcja **bezier()** wymaga określenia współrzędnych czterech punktów przez które prowadzona jest gładka krzywa interpolująca.

```
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);
```

Po modyfikacji można otrzymać taki oto rysunek:



```
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);
 }
```

Na koniec ciekawy przykład grafiki wykonanej w trybie tekstowym z użyciem znaku „x”.