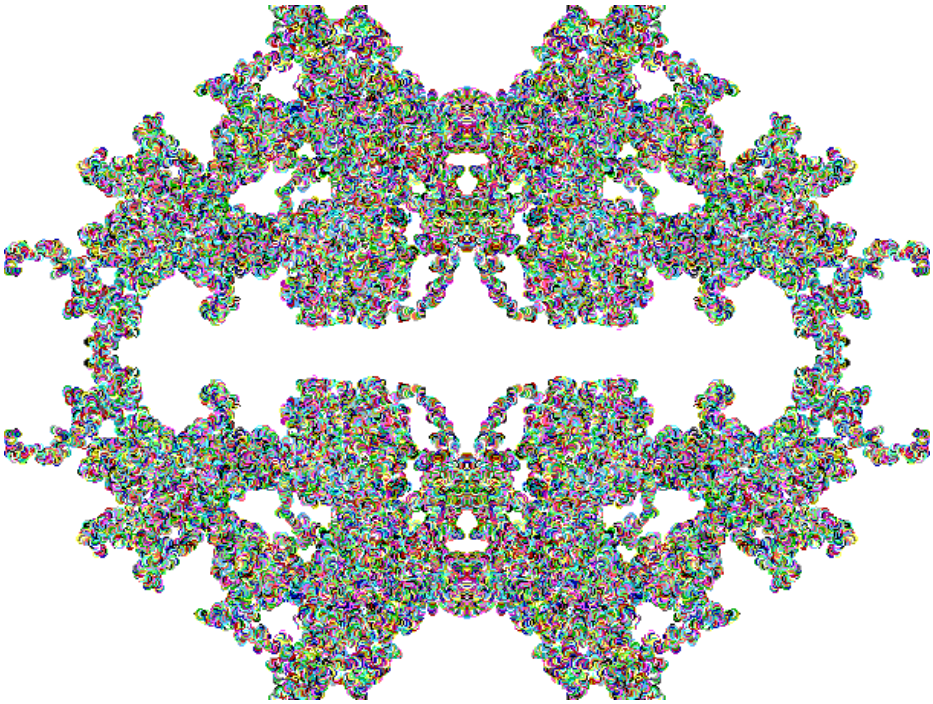
Programozási gyakorlatok- C++

1. **példa**

Rajzoljunk egy véletlenszerű, szimmetrikus ábrát a képernyőre! Rajzoljunk ki négy pontot a képernyő négy negyedének közepére. Változtassuk a pontok koordinátáit véletlenszerűen –1, 0, 1 értékekkel. (A pont véletlenszerűen mozog jobbra, balra, le, fel vagy átlósan.) Rajzoljuk ki az új pontot, és mindezt ismételjük billentyű leütéséig!

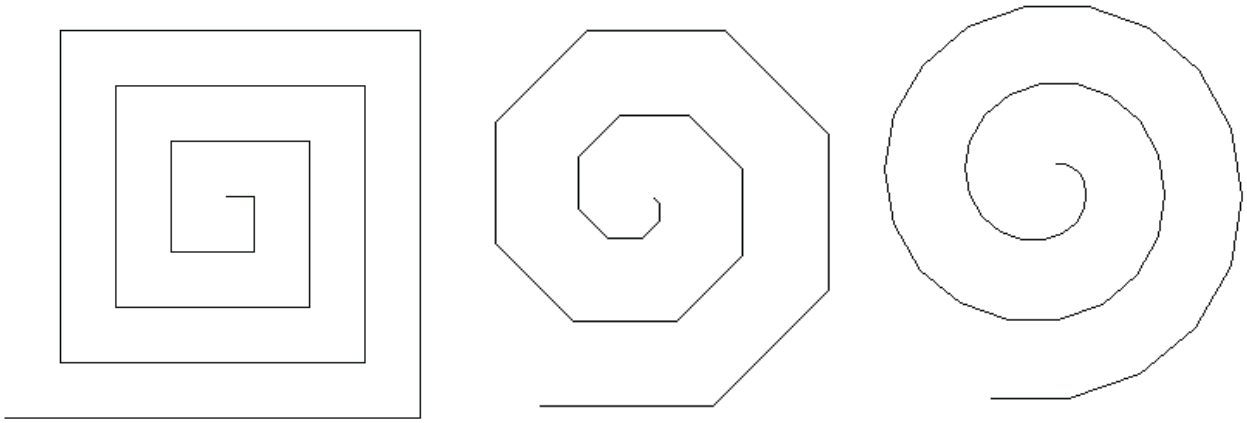


```
/* Grafika */

#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

void main()
{
   int gdriver = DETECT, gmode, x,y;
   char c; long i;
   initgraph(&gdriver, &gmode, "");
   randomize();
   x=getmaxx()/4; y=getmaxy()/4;
   int r=2, d=1;
   while(!kbhit()){
      setcolor(random(16));
      circle(x,y,r);
      circle(getmaxx()-x,y,r);
      circle(x,getmaxy()-y,r);
      circle(getmaxx()-x,getmaxy()-y,r);
      x+=d*(random(3)-1);
      if(x<0) x=d; if(x>getmaxx()) x=getmaxx()-d;
      y+=d*(random(3)-1);
      if(y<0) y=d; if(y>getmaxy()) y=getmaxy()-d;
```

```
      for(i=0;i<(r*r*1000);i++);  // lassítás
   }
   while(getch()!=27);
   closegraph();
}
```

2. példa

Írjunk függvényt, amely rekurzívan megrajzol egy csigát! A függvény rajzoljon ki egy (x,y) kezdőpontú, ’a’ hosszú vonalat ’alfa’ irányba, amennyiben ’a’ elég nagy. Ezután hívja meg saját magát a most meghúzott vonal végét megadva kezdőpontként, megrövidítve az ’a’ hosszt és elfordulva az adott irányba. Próbáljuk ki az alábbi



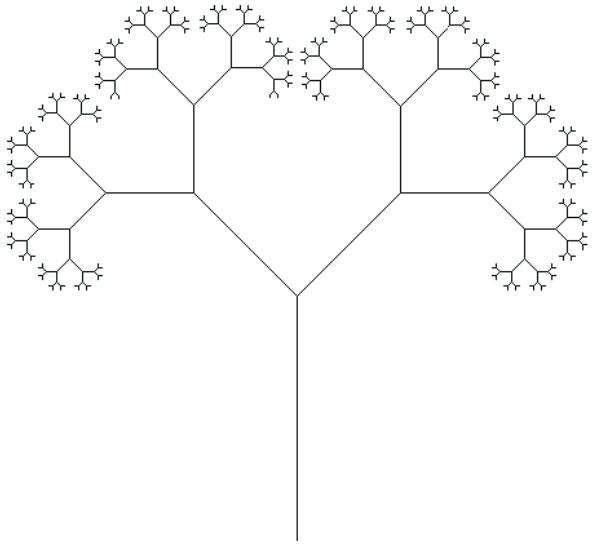
programot változtatva a kezdőhossz, a rovidulés és a fordulas változók definiált értékeit: (300;20;90), (100;5;45), (80;3;30), (50;1;20)!

```
#include <graphics.h>
#include <conio.h>
#include <math.h>

#define kezdohossz 50
#define rovidules 1
#define fordulas 20

void csiga(int x, int y, int a, int alfa) {
     if(a>4) {
       double dalfa=alfa*M_PI/180.;
       int x1=x+(int)(a*cos(dalfa)), y1=y-
(int)(a*sin(dalfa));
       line(x,y,x1,y1);
       csiga(x1,y1,a-rovidules,(alfa+fordulas)%360);
     }
}

void main()
{
   int gdriver = DETECT, gmode;
   initgraph(&gdriver, &gmode, "");
   csiga(150,350,kezdohossz,0);
   getch();
   closegraph();
}
```

3. példa

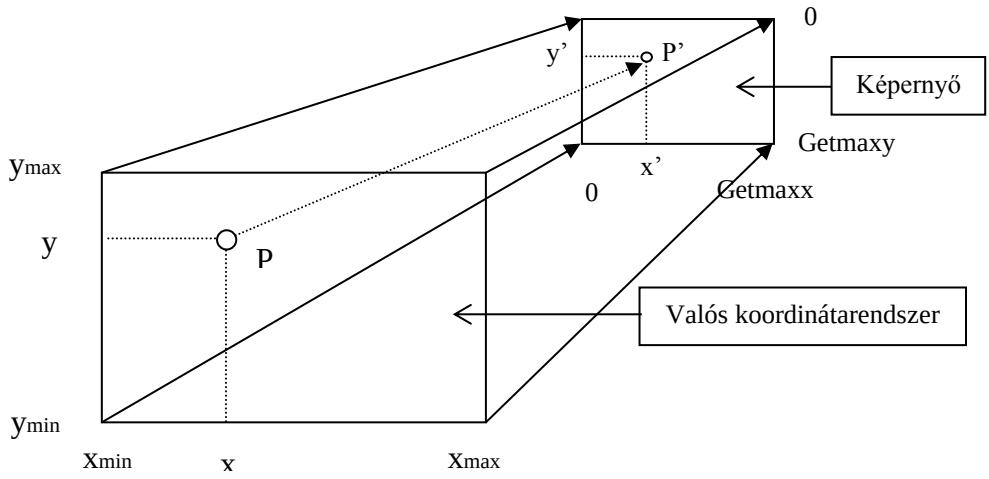
Rajzoljunk egy bináris fát! Ehhez készítsünk függvényt, amely kirajzol egy (x,y) kezdőpontú, ’a’ hosszú és alfa szögű ágat, majd kétfelé ágazva 45 fokkal balra és jobbra húz egy-egy 3/5-szeresére rövidített ágat. A két ág megrajzolásához rekurzívan hívja meg saját magát.

```
/* Bináris fa */

#include <graphics.h>
#include <conio.h>
#include <math.h>

void fa(int x, int y, int a, int alfa) {
     if(a>2) {
       double dalfa=alfa*M_PI/180.;
       int x1=x+(int)(a*cos(dalfa)), y1=y-
(int)(a*sin(dalfa));
       line(x,y,x1,y1);
       fa(x1,y1,3*a/5,alfa+45);
       fa(x1,y1,3*a/5,alfa-45);
     }
}

void main()
{
   int gdriver = DETECT, gmode;
   initgraph(&gdriver, &gmode, "");
   fa(getmaxx()/2,getmaxy()-10,200,90);
   getch();
   closegraph();
}
```



4. példa:

Készítsünk egy VIEW.H nevű C rutingyűjteményt, amelynek segítségével egy tetszőleges koordinátarendszert könnyen leképezhetünk a képernyőre! A view függvénynek paraméterként átadjuk a valós koordinátarendszer bal alsó és jobb felső sarkának koordinátáit. A függvény inicializálja a grafikus képernyőt, beállítja az xmin, xmax, ymin, ymax, dx és dy globális változókat, bekeretezi a képernyőt és berajzolja a tengelyeket. A dx, dy két képernyőpont x és y irányú távolságát jelenti a valós rendszerben. A vx, vy függvények a valós x, y koordinátákat képezik le a képernyőre, azaz x' = vx(x), y' = vy(y). Az alábbi ábra alapján megérthetjük a leképezést, és ellenőrizhetjük a képleteket.



$$\frac{x - x_{\min}}{x'} = \frac{x_{\max} - x_{\min}}{get\max x} = dx \cdots \Rightarrow \cdots x' = \frac{x - x_{\min}}{dx}$$

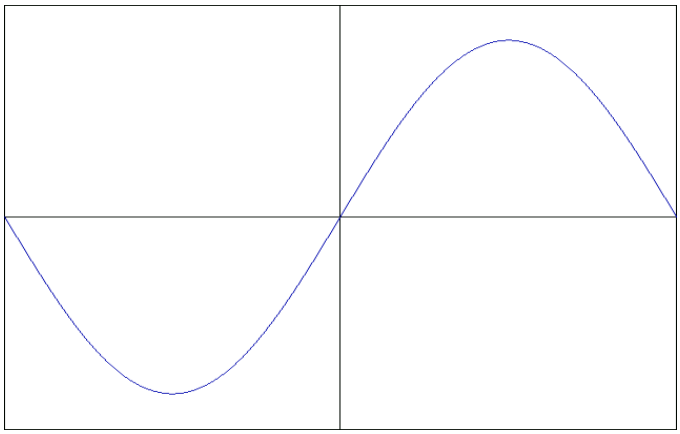
$$\frac{y - y_{\min}}{get\max y - y'} = \frac{y_{\max} - x_{\min}}{get\max y} = dy \cdots \Rightarrow \cdots y' = get\max y - \frac{y - y_{\min}}{dy}$$

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
void view(float x1, float y1, float x2, float y2);
int vx(float x);
int vy(float y);
float xmin,xmax,ymin,ymax,dx,dy;
void view(float x1, float y1, float x2, float y2) {
   int gdriver = DETECT, gmode; initgraph(&gdriver, &gmode,
"");
   xmin=x1; ymin=y1; xmax=x2; ymax=y2;
   dx=(xmax-xmin)/getmaxx(); dy=(ymax-ymin)/getmaxy();
   rectangle(0,0,getmaxx(),getmaxy());
   line(0,vy(0),getmaxx(),vy(0));
line(vx(0),0,vx(0),getmaxy());
}
int vx(float x) {
```

```
    return((int)((x-xmin)/dx));
}
int vy(float y) {
    return((int)(getmaxy()-(y-ymin)/dy));
}
```

5. példa

Rajzoljunk egy sinus hullámot a képernyőre, felhasználva az előző példában elkészített VIEW.H forrásállományt, amelyet előzőleg a Borlandc\Include könyvtárba másolunk.



```
#include <view.h>
void main()
{
   view(-M_PI,-1.2,M_PI,1.2);
   float x,x0;
   x=x0=xmin;
   setcolor(14);
   while(x<=xmax) {
      x+=dx;
      line(vx(x0),vy(sin(x0)),vx(x),vy(sin(x)));
      x0=x;
   }
   getch();
   closegraph();
}
```

6. példa:

Készítsünk programot, amely a Pontok.txt szövegfájlban tárolt koordinátájú pontokra egyenest illeszt a lineáris regresszió segítségével! Számítsuk ki az f(x) = ax + b alakú egyenes együtthatóit, az 'r' korrelációs együtthatót, amely a lineáris kapcsolat erősségét mutatja, végül pedig rajzoljuk ki a megoldást!

A képletek a következők:

$$b = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$a = \bar{y} - b \cdot \bar{x}$$

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

```
/* Regresszió */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <view.h>

void main()
{
   int i,n; char str[20];
   float x[99],y[99],sx,sy,sxy,xa,ya,x1,x2,y1,y2,a,b,r;
   FILE *f=fopen("c:\\Borlandc\\Bin\\Cpp\\Pontok.txt","r");
   clrscr();
   n=0; sx=0.; sy=0.;
   while(!feof(f)){
      n++;
      fscanf(f,"%f%f",&x[n],&y[n]);
      sx+=x[n]; sy+=y[n];
   }
   fclose(f);
   xa=sx/n; ya=sy/n;
   x1=x2=x[1]; y1=y2=y[1];
   for(i=1,sx=sy=sxy=0.;i<=n;i++) {
      sx+=(x[i]-xa)*(x[i]-xa);
      sy+=(y[i]-ya)*(y[i]-ya);
      sxy+=(x[i]-xa)*(y[i]-ya);
      if(x[i]<x1) x1=x[i];
      if(x[i]>x2) x2=x[i];
      if(y[i]<y1) y1=y[i];
      if(y[i]>y2) y2=y[i];
   }
   a=sxy/sx; b=ya-a*xa; r=sxy/(sqrt(sx)*sqrt(sy));
   //Rajzolas
   view(x1-(x2-x1)/10,y1-(y2-y1)/10,x2+(x2-x1)/10,y2+(y2-
y1)/10);
   setcolor(14);
   for(i=1;i<=n;i++) {
```

```
      circle(vx(x[i]),vy(y[i]),5);
   }
   setcolor(2);
   line(vx(xmin),vy(a*xmin+b),vx(xmax),vy(a*xmax+b));
   sprintf(str,"f(x)=%5.3f*x%c%5.3f",a,(b>0)?'+':'-
',fabs(b));
   outtextxy(20,20,str);
   sprintf(str,"r = %5.3f",r);
   outtextxy(20,50,str);   getch();
   closegraph();
}
```