7. Grafikos elementai

7.1. Grafinis ekranas.

Kompiuterio monitoriaus grafinį ekraną sudaro atskirų stačiakampės formos taškų aibė. Taškai ekrane išdėstyti horizontaliomis eilutėmis. Taškų skaičius priklauso nuo grafinės sistemos ir nusakomas dviem skaičiais m*n, kur m- taškų skaičius eilutėje, n- eilučių skaičius. Kompiuterio ekrano koordinačių pradžios taškas yra kairys viršutinis kampas ir jo koordinatės yra (0,0). Kiekvienam taškui skiriamas tam tikras bitų skaičius kompiuterio centrinio mikroprocesoriaus adresų srityje. Ta sritis vadinama videoatmintimi ir, kaip taisyklė, yra specialaus įrenginio – videoadapterio, valdančio monitoriaus darbą, dalis. Galimas dviejų ir daugiau vienodos struktūros visų ekrano taškų sričių egzistavimas. Tokios sritys vadinamos ekraniniais puslapiais. Bet kuriuo laiko momentu bet kuri sritis gali būti parodyta ekrane. Puslapių egzistavimas leidžia akimirksniu pakeisti vaizdą ekrane. Grafinis piešinys gali būti kuriamas nematomame puslapyje, o po to parodomas ekrane. Tai leidžia kurti grafinius paveikslus išvengiant nepageidaujamo piešimo proceso rodymo ekrane.

Grafinis vaizdas ekrane atkuriamas tokiais etapais:

- taškų charakteristikų užrašymas į atmintį; tai padaro specialios DOS ar BIOS funkcijos, nors yra galimybė tiesiogiai dirbti su videoatmintimi;
- speciali videoadapterio schema periodiškai skaito videoatmintį ir taškų charakteristikas verčia į signalų, valdančių ekraną, seką.

Spalvotam monitoriui kiekvieną tašką atitinka trijų spalvų komponentės, kurių intensyvumas ekrane duoda matomo taško sumarinę spalvą.

Taškų spalvinės charakteristikos saugomos spalvų lentelėje, kuri ne visuomet prieinama programuotojui. Pakeitus spalvų lentalę, ekrane akimirksniu atsispindi visi pokyčiai.

C++ grafinė sistema DOS variantui susideda iš išorinės funkcijų bibliotekos graphics.lib, kuri saugoma faile graphics.h, kuris prijungiamas prie programos. Bibliotekoje yra apie 80 funkcijų, kurias galima suskirstyti į grupes:

- grafinės sistemos ir grafinio darbo režimo paruošimas;
- darbas su taškais;
- spalvų valdymas;
- ryšys su videoatmintimi;
- grafiniai primityvai;
- teksto išvedimas;
- klaidų apdorojimas.

Skyrelyje bus pateiktos ne visos funkcijos.

Firmos Bordland International grafinį interfeisą BGI (Bordland Graphics Inerface) sudaro dvi dalys: pastovios grafinės sistemos branduolio (GSB) ir grafinių draiverių rinkinio.

GSB vykdo programų užklausas grafinėms funkcijoms ir nepriklauso nuo įjungto adapterio tipo (VGA, CGA, EGA ir pan.). Tai programos, atliekančios tarpininkų vaidmenį tarp GSB ir konkretaus displėjaus adapterio. Jos leidžia programuotojui nežinoti aparatūros ypatumų ir leidžia rašyti programas, nepriklausomas nuo adapterio tipo.

Grafinių draiverių failai saugomi kataloge BGI su plėtiniu *.bgi.

7.2. Ekrano grafinis darbo režimas.

Kompiuterio grafinis ekrano darbo režimas nustatomas naudojant funkciją

```
void far initgraph( int far * graphdriver,
                    int far * graphmode,
                    int far * pathtodriver );
```

Pirmasis parametras `graphdriver` yra rodyklė į kinamąjį, nurodantį grafinio draiverio numerį. Draiverio parinkimui galima naudoti funkciją

```
void far detectgraph( int far * graphdriver,
                      int far * graphmode);
```

Ši funkcija testuoja videoadapterio aparatūrą ir nustato optimaliausią draiverį ir grafinį darbo režimą. Pirmuoju parametru grąžinamas grafinio draiverio, aptarnaujančio surastą displėjinį adapterį, numerį. Antruoju parametru grąžina grafinio režimo, suteikiančio maksimalias galimybes parinktam adapteriui (daugiausiai taškų), numerį.

Jeigu testuojant aparatūrą funkcija `detectgraph` nerado displėjinio adapterio , tai grąžinamo parametro reikšmė –2. Tokią pat reikšmę grąžina ir ekrano paruošimo darbui rezultatų analizės funkcija `graphresult`. Galima funkcijos `detectgraph` nenaudoti. Pakanka funkcijos `initgraph` pirmam parametrui prieš funkcijos iškvietimą priskirti konstantos `DETECT` (arba 0) reikšmę. Tuomet `initgraph` pirmiausiai paleidžia automatinį aparatūros testavimą ir parenka displėjinio adapterio tipą ir tuo pačiu nusako `graphmode` reikšmę. Šiuo atveju parametro `graphmode` pradinė reikšmė ignoruojama. Jeigu funkcijos `initgraph` darbas nesėkmingas, tai formuojamas klaidos kodas –2 ir funkcija nutraukia darbą.

Prijungto grafinio draiverio vardą galima sužinoti panaudojus funkciją

```
char * far getdrivername( void );
```

Paskutinis funkcijos `initgraph` parametras yra kelias į katalogą, kuriame yra draiverio `*.bgi` failas.

Funkcija

```
int far  graphresult( void );
```

grąžina paskutinio veiksmo su grafiniu ekranu klaidos požymį.

Galima padaryti tokią grafinio ekrano paruošimo darbui funkciją:

```
void Grafika(){
 int A = DETECT, B ;
 initgraph( &A, &B, "c:\programs\bc5\bin" );
 if( graphresult() != grOk ){
   printf("Klaida: %i %i\n ", A, B);
   exit(1);}
}
```

Čia konstanta grOk (kodas 0) atitinka sėkmingo darbo reikšmę.

Grafinis darbo režimas uždaromas funkcija

```
void far closegraph( void );
```

Dirbant grafiniame režime galima laikinai pereiti į tekstinį displėjaus darbo režimą. Tam naudojama funkcija

```
void far rectorecrtmode( void );
```

Iš tekstinio režimo atgal į grafinį režimą sugrįžtama naudojant funkciją

```
void far setgraphmode( void );
```

7.3. Darbas su ekranu.

Ekrane pagal nutylėjimą rodomas nulinis puslapis. Aktyvus puslapis, kuriame piešiama, taip pat nulinis. Puslapiai numeruojami pradedant nuliu. Maksimalus galimas grafinių puslapių skaičius kiekvienam draiveriui yra skirtingas. Jis dar priklauso nuo darbo režimo modifikacijos. Tik EGA, VGA ir Hercules draiveriai turi daugiau kaip vieną puslapį.

Rodomo ir aktyvaus puslapių numerius galima nurodyti (keisti) naudojant tokias funkcijas:

```
void far setactivepage( int page );

void far setvisualpage( int page );
```

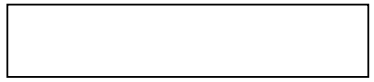
Grafinio lango dydis pagal nutylėjimą sutampa su kompiuterio ekranu. Maksimalus galimas taškų skaičius (koordinatės x ir y) gaunamas naudojant funkcijas:

```
int far getmaxx( void );

int far getmaxy( void );
```

Aktyvaus lango matmenis ir dydį galima valdyti naudojant funkciją

```
void far setviewport( int x1,  int y1,
                      int x2,  inr y2,
                      int  raktas );
```



Kai raktas = 0, tai leidžiama piešti figūras ir už lango ribų, kai ≠0, tai tik lango viduje.

Lango viduje koordinatės yra lokalios. Atskaitos tašku yra kairio viršutinio lango kampo taškas, kuris lange gauna lokalias koordinates (0,0).

Žymeklis ekrane nematomas. Jo vietą lange galime keisti. Tam

skirtos dvi funkcijos:

```
void far moveto ( int x,   int y);
void far moverel( int dx,  int dy);
```

Pirmoji funkcija leidžia perkelti žymeklį į nurodytą lango vietą. Antroji leidžia perkelti žymeklį į lango tašką, kurio koordinatės gaunamos prie buvusios žymeklio vietos pridėjus pokyčius dx ir dy.

Žymeklio padėtį ekrane galima sužinoti naudojant funkcijas:

```
int far getx( void);
int far gety( void);
```

Lango valymui skirta funkcija

```
void far clearviewport( void);
```

Visam aktyviam puslapiui valyti skirta funkcija

```
void far cleardevice( void);
```

Ji valo aktyvų puslapį aktyvia fono spalva.

7.4. Spalvos.

Spalvos saugomos matricos tipo lentele, vadinama palete. Nulinėje eilutėje yra fono spalva. Paletės eilučių skaičius sužinomas naudojant funkciją

```
int far getpalettesize( void);
```

Vieną paletės spalvą galima pakeisti naudojant funkciją

```
void far setpalette( int colornum;
                     int  color);
```

Visą spalvų paletę galima pakeisti naudojant funkciją

```
void far setallpalette( struct palettetype
                          far * palette );
```

Fono spalvą (nulinė paletės eilutė) galima keisti rašant `setpalette( 0, spalva);` , tačiau dažniausiai naudojama kita funkcija:

```
void far setbkcolor( int color);
```

Sužinoti fono spalvą galima kreipiantis į funkciją:

```
void far getbkcolor( void);
```

Aktyvią naudojamą piešimui spalvą galima nurodyti funkcija

```
 void far setcolor( int color);
```

Sužinoti esamą aktyvią spalvą galima naudojant funkciją

```
int far getcolor( void );
```

7.5. Taškai.

Darbui su taškais grafiniame ekrane yra dvi pagrindinės funkcijos:

```
unsigned far getpixel( int x,  int y);
void far putpixel( int x,  int y,  int color);
```

Pirmoji grąžina nurodyto koordinatėmis (x,y) taško spalvą, antroji taškui, kurio koordinatės (x,y) suteikia nurodytą spalvą `color`. Būtina neužmiršti, kad visos koordinatės nurodomos lango koordinačių sistemoje.

Galima sudaryti taškų masyvus. Keitimai ekrane vykdomi stačiakampiais taškų masyvais. Koordinatės nurodomos aktyvaus lango atžvilgiu. Norint išsaugoti atmintyje puslapio fragmentą, reikia išskirti atmintyje vietą. Kartu su taškų fragmentu bus išsaugomas ir jo dydis.

Fragmento dydis gaunamas naudojant funkciją

```
unsigned far imagesize( int x1,  int y1,
                        int x2,  int y2);
```

Funkcijos argumentais yra stačiakampio fragmento kairio viršutinio kampo koordinatės (x1,y1) ir dešiniojo apatinio kampo koordinatės (x2,y2).

Jeigu fragmento dydis viršija 64Kb –1, tai vistiek bus grąžinama reikšmė `FFFF`, tačiau `graphresult` grąžina kodą –11.

Gavus reikalingą atminties kiekį, galima išsaugoti fragmentą naudojant funkciją

```
void far getimage( int x1,  int y1,
                   int x2,  int y2,
                  void far * bitmap);
```

Pirmieji keturi parametrai nurodo išsaugomos taškų masyvo srities koordinates, o paskutinis nurodo operatyvinės atminties sritį, kur bus

išsaugotas ekrano taškų fragmentas. Atmintį galima išskirti su `malloc` funkcija.

Padėti į aktyvų puslapį iš atminties taškų masyvą galima naudojant funkciją

```
void far putimage( int x,  int y,
                void far * bitmap,
                      int  raktas );
```

Čia koordinatėmis (x,y) nurodomas padedamo fragmento kairio viršutinio kampo vieta aktyviame lange, rodykle `bitmap` nurodoma fragmento išsaugojimo sritis, o `raktas` – fragmento vaizdavimo formą.

rakto pavadinimas ir kodas		**Paaiškinimas**
COPY_PUT	0	Naujas fragmentas užrašomas.
XOR_PUT	1	Puslapio ir fragmento taškai sujungiami taikant XOR operaciją.
OR_PUT	2	Puslapio ir fragmento taškai sujungiami taikant OR operaciją.
AND_PUT	3	Puslapio ir fragmento taškai sujungiami taikant AND operaciją.
NOT_PUT	4	Naujas fragmentas rašomas invertuotas.

7.1 pratimas. Demonstruojamos grafiniame ekrane iš taškų formuojamos spalvotos figūros. Padaryta funkcija `Apskritimas`, kuri užpildo nurodytos spalvos taškais apskritimo vidų: nurodomos centro koordinatės (x,y), spindulys, taškų spalva ir jų skaičius. Funkcija `Keturkampis` valdoma nurodant centro koordinates (x,y), plotį a, aukštį b, pradinę spalvą ir taškų skaičių. Keturkampis piešiamas keturių simetriškų sričių, kurių spalvos skirtingos.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

void Grafika();
void Apskritimas( int, int, int, int, int  );
void Keturkampis( int, int, int, int, int, int  );

void main(){
 Grafika();
 setcolor( RED );
 Apskritimas( 200, 150, 100, RED,     3000 );
 Apskritimas( 100, 100,  50, MAGENTA, 2500 );
 Apskritimas( 400, 300,  50, BLUE,    3000 );
 Apskritimas( 500, 100, 100, GREEN,   5000 );
 Apskritimas( 500, 300,  50, 11,      6000 );
 Keturkampis( 140, 350, 100, 100, RED, 8000 );
 getch();
 closegraph();
 }//-----------------------------------------
void Grafika(){
  int A = DETECT, B ;
 initgraph( &A, &B, "" );
 if( graphresult() != grOk ){
   printf("Klaida: %i %i\n ", A, B);
   exit(1);}
}//-----------------------------------------
void Apskritimas( int  x, int y, int r,
                  int sp, int Kiek ){
 int xx, yy ;
 randomize();
 for( int i=0; i<Kiek; i++){
   xx= random( r ); yy = random( r );
   if( xx*xx + yy*yy < r*r) {
     putpixel( x+xx, y+yy, sp);
     putpixel( x+xx, y-yy, sp);
     putpixel( x-xx, y-yy, sp);
     putpixel( x-xx, y+yy, sp);  }}
}//-----------------------------------------
void Keturkampis( int  x, int y, int a, int b,
                  int sp, int Kiek  ){
  int xx, yy;
  randomize();
  for( int i=0; i<Kiek; i++){
    xx= random( a ); yy = random( b );
   putpixel( x+xx, y+yy, sp  );
```

```
   putpixel( x+xx, y-yy, sp+1);
   putpixel( x-xx, y-yy, sp+2);
   putpixel( x-xx, y+yy, sp+3);
}}
```

7.6. Geometrinės figūros.

Grafiniame pakete yra pagrindinių geometrinių figūrų piešimo funkcijos. Jas galima skirstyti į dvi grupes:

- kontūrines, - piešia figūras iš linijų;
- sritines, - pilnavidurių figūrų piešimo ir plotų užpildymo nurodytos spalvos nurodytu raštu.

Tiesės atkarpoms piešti skirtos funkcijos:

```
void far line( int x1, int y1,
               int x2, int y2);
```

Brėžiama atkarpa tarp taškų (x1,y1) ir (x2,y2).

```
void far linerel( int dx, int dy );
```

Brėžiama atkarpa nuo taško, kuriame stovi žymeklis (aktyvus taškas) iki taško, kurio koordinatės gaunamos, pridėjus dx ir dy atitinkamai.

```
void far lineto( int x, int y);
```

Brėžiama atkarpa nuo aktyvaus taško iki nurodyto taško (x,y).

Stačiakampiui brėžti skirta funkcija

```
void far rectangle( int x1, int y1,
                    int x2, int y2);
```

Apskritimui brėžti skirta funkcija

```
void far circle( int xc, int yc, int r );
```

Apskritimo lankui brėžti skirta funkcija

```
void far arc( int x, int y,
              int startangle, int endangle,
              int radius);
```

Elipsės lankui brėžti skirta funkcija

```
void far ellipse( int x, int y,
                  int startangle, int endangle,
                  int xradius, int yradius);
```

Yra grupė funkcijų, skirtų brėžiamų linijų charakteristikoms sužinoti arba nurodyti. Būtiniausios yra funkcijos, skirtos linijų spalvai ir jų pobūdžiui nurodyti.

```
void far setlinestyle( int stilius,
                       unsigned raštas, int storis);
```

Linijų stilius turi standartines keturias formas, kurios parodytos lentelėje 7.1. Stilių galima nurodyti konstantės vardu arba reikšme. Nurodžius 4-ą tipą, linija bus brėžiama pagal programuotojo sukurtą raštą.

Lentelė 7.1.

linijos stilius		Paaiškinimas (rašto forma)
SOLID_LINE	0	Ištisinė linija
DOTTED_LINE	1	Taškinė linija
CENTER_LINE	2	Ašinė linija
DASHED_LINE	3	Punktyrinė linija
USERBIT_LINE	4	Vartotojo sukurto stiliaus linija

Antrasis parametras `raštas` ignoruojamas, jeigu nurodomas standartinis stilius. Esant stiliui nr. 4, šis parametras turi turėti programuotojo sukurtą raštą. Jis užrašomas kaip keturženklis šešioliktainis skaičius, kurio dvejetainėje formoje 1 reiškia matomą tašką, o 0 nematomą.

Linijos storis (trečias parametras) nurodomas vieno taško (1) arba trijų taškų (3).

7.2 pratimas. Demonstruojamas linijų brėžimas.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void Grafika();

void main(){
 Grafika();
//                Žalia atkarpa per visą ekraną.
```

```
 setcolor( RED );
 line( 0, 0, getmaxx(), getmaxy());
//               Žalia stora punktyrinė atkarpa.
 setlinestyle( 3, 0, 3 );  setcolor( GREEN);
 line( 100, 100, 400, 250);
//               Žalia plona ašinė atkarpa.
 setlinestyle( 2, 0, 1);
 line( 200, 50, 450, 400);
//     Violetinė stora savo sukurto rašto atkarpa.
 setlinestyle( 4, 0x1A1A, 3); setcolor( MAGENTA );
 line( 200, 200, 500, 200);
//     Figūrų linija kaip paskutinės atkarpos.
 rectangle( 50, 50, 300, 300 );
 ellipse( 300, 300, 0, 360, 100, 50 );
 getch();

 closegraph();
 }//---------------------------------------
void Grafika(){
  int A = DETECT, B ;
 initgraph( &A, &B, "" );
 if( graphresult() != grOk ){
   printf("Klaida: %i %i\n ", A, B);
   exit(1);}
}//---------------------------------------
```

Sritinės figūros brėžiamos naudojant tokias funkcijas.

```
void far bar( int x1, int y1, int x2, int y2);
void far bar3d( int x1, int y1,
                int x2, int y2,
                int z, int topflag);
```

Trimatis gretasienis, kurio stačiakampį nusako pirmosios keturios koordinatės, z- gylis, t.y. trečiasis matavimas, `topflag = 0`, kai nebrėžiamos viršaus linijos ir `topflag = 1`, kai brėžiamas viršus. Linijų tipas ir spalva nurodoma kaip 7.2. pavyzdyje buvo parodyta.

Pilnavidurei elipsei brėžti skirta funkcija

```
void far fillellipse( int x1 int y,
                int xradius, inr yradius);
```

Lankui brėžti skirta funkcija

```
void far pieslice( int x, int y,
                   int startangle, int endangle,
                   int radius);
```

Figūrų užpildymas nurodomas funkcija

```
void far setfillstyle( int raštas,
                       int spalva);
```

Rašto yra 12 šablonų, kurių numeriai yra 0..11. Jeigu programuotojas nori sukurti savo, tuomet nurodo savo, naudodamas funkciją

```
void far setfillpattern( char far * raštas,
                         int        spalva);
```

Čia `raštas` yra stačiakampė sritis, kuri aprašoma 8 baitais. Kiekvieno bito nulinė reikšmė reiškia nematomą tašką, o vienetas – matomą. Baitų reikšmės užrašomos šešioliktainiais skaičiais ir saugomos kaip iš eilės einančių skaičių seka (masyvas).

Figūros užpildymas vykdomas funkcija

```
void far floodfill( int x, int y, int spalva);
```

Čia (x,y) yra bet kurio taško, esančio figūros viduje, koordinatės, o spalva yra figūros kontūro spalva. Užpildymo metu nuo duoto taško plinta banga, kuri nurodytu raštu viską pakeliui padengia, išskyrus nurodytos spalvos vietas. Jeigu kontūra uždaras, užpildomas tik figūros vidus (kitokiomis spalvomis pieštos vidinės figūros dings). Jeigu kontūras atviras, užpildymas vyks ir kontūro išorėje.

7.3 pratimas. Demonstruojamas figūrų piešimas ir užpildymas.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void Grafika();

void main(){
 Grafika();

 setcolor( RED );
 setlinestyle( 0, 0, 3 );  setcolor( GREEN);
 setfillstyle( 3, RED );
//             Raudonai štrichuotas stačiakampis.
 bar( 100, 100, 200, 200 );
```

```
//       Raudonai štrichuotas priekinis vaizdas.
//       Linijos Žalios. Viršus nenupieštas.
 bar3d( 250, 100, 400, 200, 50, 0);
// Toks pat gretasienis, tik su nupieštu viršumi.
 bar3d( 100, 300, 300, 400, 50, 1);
 getch();

 cleardevice();
// Du koncentriški apskritimai.
 setcolor( RED );  circle( 200, 200, 100 );
                   circle( 200, 200,  50 );
// Žiedas tarp apskritimų užštrichuojamas žaliai.
 setfillstyle( 5, GREEN );
 floodfill( 260, 260, RED );
// Vidinis apskritimas užpildomas savo sukurtu
// rašu violetine spalva.
 char rastas[]= { 0x20, 0xAA, 0xFF, 0x5A,
                  0x55, 0xC3, 0x04, 0x66 };
 setfillpattern(  rastas, MAGENTA );
 floodfill( 200, 200, RED );

// Pakartojamas vaizdas su kitais apskritimais.
 setcolor( BROWN ); circle( 400, 200, 100 );
                    circle( 400, 200,  50 );
 setfillstyle( 5, GREEN );
 floodfill( 460, 260, BROWN );
   setfillpattern(  rastas, BLUE );
   floodfill( 400, 200, BROWN );
 getch();
 closegraph();
 }//-----------------------------------------
void Grafika(){
  int A = DETECT, B ;
 initgraph( &A, &B, "" );
 if( graphresult() != grOk ){
   printf("Klaida: %i %i\n ", A, B);
   exit(1);}
}//-----------------------------------------
```

7.7. Tekstas ekrane.

Grafiniai šriftai būna dviejų tipų: matriciniai ir vektoriniai. Matricinių C++ yra vienas ir jis yra automatiškai užkraunamas į kompiuterio atmintį. Antrojo tipo būna keletas ir jie saugomi failuose su plėtiniu **`*.chr`**. Funkcija

```
void far settextstyle( int font,
                       int direction,
                       int charsize);
```

paruošia teksto ženklų generatorių darbui. Čia pirmuoju parametru nurodomas šrifto numeris, kuris atitinka vienam iš turimų failų. Antrasis parametras nurodo rašymo kryptį: 0 – horizontaliai, 1 – vertikaliai. Trečiasis skirtas simbolių dydžiui nurodyti. Galimos reišmės nuo 1 iki 10.

Teksto išvedimo padėtis pradžios taško atžvilgiu nurodomas funkcija

```
void far settextjustify( int horizontaliai,
                         int vertikaliai);
```

Pirmasis parametras nurodo x koordinatės atžvilgiu teksto padėtį. Antrasis parametras rodo y koordinatės atžvilgiu teksto padėtį. Reikšmės parodytos lentelėje 7.2.

Lentelė 7.2.

Parmetras `horizontaliai`	teksto vieta	Parametras `vertikaliai`	teksto vieta
LEFT_TEXT 0	pradžia	BOTTOM_TEXT 0	apačia
CENTER_TEXT 1	centras	CENTER_TEXT 1	centras
RIGHT_TEXT 2	galas	TOP_TEXT 2	viršus

Tekstas ekrane išvedamas naudojant funkcijas:

```
void far outtext( char far eilutė );
```

Išvedimo taškas yra aktyvus taškas (žymeklio vieta).

```
void far outtextxy( int x, int y,
                    char far eilutė );
```

🖬 7.4 pratimas. Demonstruojamos figūros ir tekstas ekrane.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
void Apskritimas( int, int, int, int  );
void Grafika();
//-----------------------------------------
void main(){
 Grafika();
 setcolor( RED );

 line( 0, 0, getmaxx(), getmaxy());

 setcolor( MAGENTA );
 outtextxy(    80,  40, "Apskritime 1000 tasku");
 Apskritimas( 150, 150, 100, MAGENTA);

 settextstyle( 2, 1, 3);   setcolor( GREEN );
 outtextxy(   200, 300, "Apskritime 1000 tasku");
 Apskritimas( 300, 350, 80, GREEN  );

 settextstyle( 1, 0, 9);  setcolor( WHITE );
 outtextxy(   400, 80, "Apskritime 1000 tasku");
 Apskritimas( 500, 200, 100, WHITE  );
 getch();
 setviewport( 100, 100, 250, 250, 0);
 setbkcolor( RED );

 setcolor( WHITE ); line( 3,3, 40, 40);
 Apskritimas( 50, 50, 20, RED  );

 getch();
 closegraph();
}//-----------------------------------------
void Grafika(){
  int A = DETECT, B ;
 initgraph( &A, &B, "" );
 if( graphresult() != grOk ){
   printf("Klaida: %i %i\n ", A, B);
   exit(1);}
}//-----------------------------------------
void Apskritimas( int x, int y, int r,
                  int spalva ){
 int xx, yy, sp;
 setcolor( spalva );
 circle( x, y, r );
 randomize();
 for( int i=0; i<1000; i++){
   xx= random( r ); yy = random( r );
   sp = random( 15 );
   if( xx*xx + yy*yy < r*r) {
     putpixel( x+xx, y+yy, sp);
     putpixel( x+xx, y-yy, sp);
     putpixel( x-xx, y-yy, sp);
     putpixel( x-xx, y+yy, sp); }
}}//-----------------------------------------
```

7.8. Pavyzdžiai.

7.5 pratimas. Laboratorinio darbo grafinė užsklanda. Imituojamas linijų judėjimas. Pašalinus iš programos vieno žingsnio piešinio valymą, gaunamas figūrinis spalvotas kintantis vaizdas. Uždavinio sprendimo programinė dalis pašalinta.

```
// FMM-8/1 grupės studento Tomo Miliausko
//      laboratorinio darbo užsklanda
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <dos.h>
void Grafika();
void Paruosimas();
// -----------------------------------------
void main() {
  clrscr();
  Paruosimas();
  Grafika();
}// --- Grafinių efektų pateikimas ekrane ------
void Grafika() {
  const int n= 10;  // Taškų skaičius
  int X[n], Y[n],
      DX[n], DY[n], Sp[n]; // Taškų parametrai
  int i, xmax, ymax;
```

```
  xmax= getmaxx(); ymax= getmaxy();
  randomize();
  // ----- Pradinių parametrų suteikimas -------
  for ( i= 0; i < n; i++ )     {
    X[i]= random (xmax-5) + 1;
    Y[i]= random (ymax-5) + 1;
    DX[i]= random (2) * 2 - 1;
    DY[i]= random (2) * 2 - 1;
    Sp[i]= random (15) + 1;     }
 while( !kbhit() ){// Kol nebus paspaustas klavišas.
  for ( i= 0; i < n; i++ ){   // Taškų judėjimas
    X[i]+= DX[i];
    if (( X[i] < 1 ) || ( X[i] > xmax - 1 ))
                                DX[i]= -DX[i];
    Y[i]+= DY[i];
    if (( Y[i] < 1 ) || ( Y[i] > ymax - 1 ))
                                DY[i]= -DY[i];
  } // ----- Taškų sujungimas linijomis --------
  for ( i= 0; i < (n - 1); i++ )
    { setcolor ( Sp[i] );
      line ( X[i], Y[i], X[i+1], Y[i+1] ); }
  setcolor ( Sp[n-1]  );
  line ( X[n-1], Y[n-1], X[0], Y[0] );
  delay( 50 );
  cleardevice();  //  Ekrano valymas.
 }
  getch();  closegraph();
}// ------- Grafinio režimo paruošimas --------
void Paruosimas ()
 { int A = DETECT, B ;
   initgraph( &A, &B, "" );
   if( graphresult() != grOk ){
     printf("Klaida: %i %i\n ", A, B);
   exit(1);}
  }//-----------------------------------------
```

7.6 pratimas. Laboratorinio darbo grafinė užsklanda. Tekstiniai pranešimai apie programos darbą. Taškiniai įvariaspalviai spinduliai. Uždavinio sprendimo programinė dalis pašalinta.

```
// Darbas 3.  Užklanda.
// Atliko : IF 8/10 gr. studente
// Agnė Sviderskaitė
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
// ---- Grafikai skirtos funkcijos ---------
void Ijungimas();
void Taskai();
void Taskas(int,int,int,int,int,int);
//-----------------------------------------
void main()  {
  int x,y;
 Ijungimas();
 settextstyle(9,0,10);
 x = getmaxx()/2;       y = getmaxy()/2;
 outtextxy(x-150,y,"Programa darbą        pradėjo. ");
 Taskai();
 outtextxy(x-90,y+20,"Programa darbą baigė.");
 outtextxy(x-110,y+40,"Rezultatai faile rez3.dat.");
 outtextxy(x-110,y+60,"Spauskite bet kokį klavišą ");
 getch();
 closegraph();
 }//-------- Grafikos prijungimas ------------
 void Ijungimas() {
  int A = DETECT, B, error;
  initgraph(&A, &B, "");
  error = graphresult();
  if (error != grOk)  {
   printf("Grafikos klaida: ");
   printf("Spauskite bet kokį klavišą išeiti.");
   getch();      exit(1);
}}//------ Vieno taško judėjimas -------------
 void Taskas (int k, int x1, int y1,
                     int x2, int y2,int sp)
 { int y, x = x2;
 while ((x>=0)&&(x<=getmaxx())){
  x +=k;   y = (x-x1)*(y2-y1)/(x2-x1)  + y1;
 putpixel(x,y, sp);   delay(2);}
}//----- Linijų piešimas --------------------
 void Taskai ()
```

```
 {int i,x, y, R = 50,cx,cy, k=2;
  int mas[40][3];
  cx = getmaxx()/2;   cy = getmaxy()/2;
  randomize();
  for (i=0; i<10; i++)
  { x = random(R);      y = random (R);
    mas[i*4  ][0] = x +cx;
    mas[i*4  ][1] = y +cy;
    mas[i*4  ][2] = random(15);
    mas[i*4+1][0] = -x +cx;
    mas[i*4+1][1] = y +cy;
    mas[i*4+1][2] = random(15);
    mas[i*4+2][0] = x +cx;
    mas[i*4+2][1] = -y +cy;
    mas[i*4+2][2] = random(15);
    mas[i*4+3][0] = -x +cx;
    mas[i*4+3][1] = -y +cy; }
    mas[i*4+3][2] = random(15);
  for (i=0; i<40; i++)
    putpixel(mas[i][0],mas[i][1],mas[i][2]);
  x =cx;  y = cy;
  for (i=0; i<40; i++)
    {if (i % 2 == 0) k = 2;
     else k = -2;
 Taskas (k,cx,cy,mas[i][0],mas[i][1],mas[i][2]);}
}
```