Lucrarea nr. 1

Elemente de grafica in C++

1. Scopul lucrarii:

Lucrarea de fata isi propune familiarizarea cu lucrul in mod grafic in limbajul C++: initializarea modului grafic, alegerea culorilor, tratarea erorilor, realizarea figurilor grafice de baza.

2. Notiuni teoretice:

Functii de baza pt. modul grafic

1. ***Funcţii pt. iniţialiarea modului grafic:***

FUNCŢIA	SINTAXĂ	DESCRIERE
Initgraph	void initgraph(int graphdriver,int graphmode, char pathtodriver);	Iniţializează sistemul în modul grafic prin încarcarea driver-lui de pe disc.
Detectgraph	void detectgraph(int graphdriver, int graphmode)	Testează hardware-ul şi determină driver-ul şi modul utilizabil.
Closegraph	void closegraph(void);	Termină lucrul cu modul grafic.
Getdrivername	char getdrivername(void);	Returnează numele driver-ului grafic utilizat.
Getgraphmode	int getgraphmode(void);	Returnează codul modului grafic actual.
Getmodename	char getmodename(int mode_number);	Returnează numele modului grafic actual.
Getmaxmode	int getmaxmode(void)	Returnează numărul maxim de moduri ce pot fi folosite de driver-ul actual.
Getmoderange	void getmoderange(int graphdriver, int lomode, int himode);	Determină valorile extreme ale codului modului grafic ce se poate utiliza, corespunzătoare unui driver dat.
Graphdefaults	void graphdefaults(void);	Poziţionează pointerul curent în colţul stânga sus de coordonate (0, 0) şi variabilele sistemului grafic la valorile implicite.
Getmaxx	int getmaxx(void);	Numărul de ordine al coloanei din drepta (rezoluţia X) a driver-ului şi a modului actual
Getmaxy	int getmaxy(void)	Numărul de ordine al liniei situată la bază (rezoluţia Y) a

		driver-ului si al modului actual.
Getx	int getx(void)	Returnează abscisa X a cursorului actual
Gety	int gety(void)	returnează ordonata Y a cursorului actual
Installuserdriver	int installuserdriver(char name, int huge (detect)(void))	Instalarea unui driver propriu
Registerbgidriver	int registerbgidriver(void (driver)(void));	pe baza unei valori ce reprezintă un driver returnează un cod (de tip intreg), ce reprezintă numărul driver-ului
Registerbgifont	int registerbgifont(void (font)(void))	returnează un cod care reprezintă numărul setului de caractere fiind definită astfel:
Restorecrtmode	void restorecrtmode(void)	Ecranul fiind în mod grafic, există posibilitatea restabilirii modului caracter existând înaintea iniţializării modului grafic.
Setgraphbufsize	unsigned setgraphbufsize(unsigned bufsize);	permite utilizatorului să schimbe dimensiunea tamponului grafic utilizat pentru realizarea haşurării cu modelul actual a unei zone închise; tamponul iniţial este de dimensiunea 4K0, care permite haşurarea unui poligon pînă la 655 de vârfuri. Prin modificarea dimensiunii tamponului este posibilă efectuarea haşurării poligoanelor cu mai multe vârfuri.
Setgraphmode	void setgraphmode(int mode)	Revenirea din modul caracter în modul grafic

2. ***Funcţii pt. definire de ferestre şi pagini***

FUNCTIA	SINTAXA	DESCRIERE
Setviewport	void setviewport(int left, int top, int right, int bottom, int clip)	toate comenzile de desenare / scriere operează în regiunea rectangulară definită
Getviewsettings	void getviewsettings (struct viewporttype viewport)	utilizatorul poate obţine informaţii referitoare la fereastra actuală şi felul tăierii
Clearviewport	void clearviewport(void)	şterge fereastra grafică actuală

Cleardevice	void cleardevice(void)	şterge ecranul grafic actual şi poziţionează pointe-ul actual în poziţia (0, 0),
Setactivepage	void setactivepage(int page)	fixează pagina activată pentru echipamentul grafic de ieşire.
Setvisualpage	void setvisualpage(int page)	fixează pagina vizuală

3. ***Funcţii pentru definire puncte***

FUNCTIA	SINTAXA	DESCRIERE
Putpixel	void putpixel(int x, int y, int color)	produce un punct de culoarea dată
Getpixel	unsigned getpixel(int x, int y)	determină culoarea unui punct de coordonate date

4. ***Funcţii pentru definire linii***

FUNCTIA	SINTAXA	DESCRIERE
Line	void line(int x1, int y1, int x2, int y2)	desenează o linie între punctele de coordonate (x1, y1) şi (x2, y2).
Lineto	void lineto(int x, int y)	desenează o linie de la poziţia pointer-ului actual la un punct dat de coordonate (x, y).
Linerel	void linerel(int dx, int dy)	desenează o linie de la poziţia pointer-ului actual până la un punct dat definit de destinaţie DX şi DY
Moveto	void moveto(int x, int y)	mută pointer-ul actual la coordonatele x, y
Moverel	void moverel(int dx, int dy)	mută pointer-ul actual din poziţia curentă la un punct definit de distanţele DX, DY
Setlinestyle	void setlinestyle(int linestyle, unsigned upattern, int thickness)	stabileşte stilul şi grosimea liniei
Getlinesettings	void getlinesettings(struct linesettingstype lineinfo)	returnează stilul şi grosimea liniei utilizate
Setwritemode	void setwritemode(int mode)	stabileşte modul de scriere pe ecran a unei linii

5. ***Funcţii pentru definire arcuri, cercuri şi alte curbe***

FUNCTIA	SINTAXA	DESCRIERE
Circle	void circle(int x, int y, int radius)	desenează un cerc de centru şi rază dată;
Arc	void arc(int x, int y, int stangle, int endangle, int radius)	desenează un arc de cerc
Ellipse	void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius)	desenează o elipsă
Getarccoords	void getarccoords(struct arccoordstype arccoords)	returnează centrul şi coordonatele de început şi de

		sfrărşit ale ultimului arc desenat
Pieslice	void pieslice(int x, int y, int stangle, int endangle, int radius)	desenează şi haşurează un sector de cerc
Sector	void sector(int x, int y, int stangle, int endangle, int xradius, int yradius)	desenează şi haşurează un sector de elipsă
Fillellipse	void fillellipse(int x, int y, int xradius, int yradius)	desenează şi haşurează o elipsă
Getaspectratio	void getaspectratio(int xasp, int yasp)	determină dimensiunile maximale ale ecranului grafic
Setaspectratio	void setaspectratio(int xasp, int yasp)	defineşte raportul dimensional implicit

6. ***Funcţii pentru definire poligoane şi haşurări***

FUNCTIA	SINTAXA	DESCRIERE
Rectangle		desenează un dreptunghi
Bar	void bar(int left, int top, int right, int bottom)	desenează un dreptunghi şi îl haşurează cu modelul şi culoarea satabilite
Bar3d	void bar3d(int left, int top, int right, int bottom,int depth, int topflag)	desenează un paralelipiped dreptunghic şi îl haşurează cu modelul şi culoarea actuală
Drawpoly	void drawpoly(int numpoints, int polypoints);	desenează o linie poligonală
Fillpoly	void fillpoly(int numpoints, int polypoints);	desenează şi haşurează un poligon
Floodfill	void floodfill(int x, int y, int border);	haşurează cu modelul actual o zonă închisă
Setfillstyle	void setfillstyle(int pattern, int color);	fixează modelul de haşurare utilizat şi culoarea de haşurare
Getfillsettings	void getfillsettings (struct fillsettingstype fillinfo)	permite utilizatorului să obţină informaţii despre modelul actual de haşurare şi de culoare
Setfillpattern	void getfillpattern(char pattern);	oferă utilizatorului posibilitatea de a defini un model propriu de haşurare
Getfillpattern	void setfillpattern(char upattern, int color);	permite obţinerea informaţiilor despre ultimul model de haşurare stabilit.

7. ***Funcţii pentru salvarea imaginilor***

FUNCTIA	SINTAXA	DESCRIERE
Imagesize	unsigned imagesize(int left, int top, int right, int bottom)	returnează numărul de octeţi necesari salvării unei imagini.
Getimage	void getimage(int left, int top, int right,	salvează o imagine într-o

	int bottom, void bitmap)	zonă de memorie definită de utilizator
Putimage	void putimage(int left, int top, void bitmap, int top)	suprapune regiunea salvată peste ecran

8. ***Funcţii pentru definirea textelor***

FUNCTIA	SINTAXA	DESCRIERE
Settextstyle	void settextstyle(int font, int direction, int charsize)	defineşte forma caracterelor textului ce se vor utiliza la scrierea textelor, direcţia de scriere precum si dimensiunea caracterelor
Setusercharsize	void setusercharsize(int multx, int divx, int multy, int divy)	utilizatorul poate sa definească înălţimea şi laţimea proprie a caracterelor speciale.
Installuserfont	int installuserfont(char name)	
Settextjustify	void settextjustify(int horiz, int vert)	fixează valorile de aliniere ale textului
Gettextsettings	void gettextsettings(struct textsettingstype texttypeinfo)	retunează numărul setului de caractere, direcţia de scriere dimensiunea caracterelor, valoarea alinieri verticale şi orizontale utilizate actual.
Textheight	int textheight(char textstring)	returneaza înălţimea unui lanţ de caractere în puncte imagine (pixeli).
Textwidth	int textwidth(char textstring)	returnează laţimea unui lanţ de caractere in pixeli.
Outtext	void outtext(char textstring)	trimite un şir de caractere la dispozitivul standard de ieşire, cordonatele punctului de început al textului fiind determinate de poziţia pointerului actual.
Outtextxy	void outtextxy(int x, int y, char textstring)	trimite un şir de caractere la dispozitivul standard de ieşire, coordonatele punctului de început al textului sunt determinate de coordonatele (x,y) ale ecreanului (şi nu de poziţia pointer-ului actual).

9. ***Funcţii pentru definirea de culori şi palete***

FUNCTIA	SINTAXA	DESCRIERE
Getdefaultpalette	struct palettetype getdefaultpalette(void);	încarcă intr-o variabilă (de tip predefinit **PaletteType**) paleta implicită.
Setcolor	void setcolor(int color)	permite stabilirea culori

		scrisului.
Getbkcolor	int getbkcolor(void)	returnează culoarea curentă a background-ului.
Getcolor	int getcolor(void)	returnează culoarea curentă de desenare.
Getmaxcolor	int getmaxcolor(void)	returnează cea mai mare valoare validă a culorii pentru driverul grafic curent.
Getpalette	void getpalette(struct palettetype palette)	dă informaţii despre dimensiunea şi culorile paletei curente.
Getpalettesize	int getpalettesize(void)	Returnează dimensiunea paletei curente de culori .
Setallpalette	void setallpalette(struct palettetype palette)	Schimbă toate paletele de culori.
Setpalette	void setpalette(int colornum, int color);	Setează o singură paletă de culori.
Setrgbpalette	void setrgbpalette(int colornum, int red, int green, int blue)	Defineşte culorile pentru IBM-8514.

10. ***Funcţii pentru tratarea erorilor grafice***

FUNCTIA	SINTAXA	DESCRIERE
Graphresult	int graphresult(void)	retunează un cod de eroare care raportează starea ultimei operaţii grafice executate
Grapherrormsg	char grapherrormsg(int errorcode)	returnează textul mesajului de eroare corespunzător codului de eroare obţinut de **GraphResult**

Paleta de culori:

Culoare	Valoare
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
BLINK	128

Drivere grafice

Constant	Valoare
DETECT	0 (autodetectare)
CGA	1
MCGA	2
EGA	3
EGA64	4
EGAMONO	5
IBM8514	6
HERCMONO	7
ATT400	8
VGA	9
PC3270	10

Moduri grafice

Drivere grafice	Moduri grafice	Valoare	Col×Lin	Paleta
CGA	CGAC0	0	320 x 200	C0
	CGAC1	1	320 x 200	C1
	CGAC2	2	320 x 200	C2
	CGAC3	3	320 x 200	C3
	CGAHI	4	640 x 200	2 color
MCGA	MCGAC0	0	320 x 200	C0
	MCGAC1	1	320 x 200	C1
	MCGAC2	2	320 x 200	C2
	MCGAC3	3	320 x 200	C3
	MCGAMED	4	640 x 200	2 color
	MCGAHI	5	640 x 480	2 color
EGA	EGALO	0	640 x 200	16 color
	EGAHI	1	640 x 350	16 color
EGA64	EGA64LO	0	640 x 200	16 color
	EGA64HI	1	640 x 350	4 color
EGA-MONO	EGAMONOHI	3	640 x 350	2 color
	EGAMONOHI	3	640 x 350	2 color
HERC	HERCMONOHI	0	720 x 348	2 color
ATT400	ATT400C0	0	320 x 200	C0
	ATT400C1	1	320 x 200	C1
	ATT400C2	2	320 x 200	C2
	ATT400C3	3	320 x 200	C3
	ATT400MED	4	640 x 200	2 color
	ATT400HI	5	640 x 400	2 color
VGA	VGALO	0	640 x 200	16 color
	VGAMED	1	640 x 350	16 color
	VGAHI	2	640 x 480	16 color
PC3270	PC3270HI	0	720 x 350	2 color
IBM8514	IBM8514HI	1	1024 x 760	256 color
	IBM8514LO	0	640 x 480	256 color

3. Desfasurarea lucrarii: sa se ruleze urmatoarele programe asupra carora se vor face si alte modificari.

programul nr.1

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

int main(void)
{
   /* autodetectare */
   int gdriver = DETECT, gmode, errorcode;
   int midx, midy;
   int stangle = 45, endangle = 135;
   int radius = 100;

   /* initializeaza variabilele locale si globale */
   initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

   /* rezultatele initializarii */
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Apasa orice tasta pentru a se opri:");
      getch();
      exit(1);
   }

   midx = getmaxx() / 2;
   midy = getmaxy() / 2;
   setcolor(getmaxcolor());

   /* deseneaza arc de cerc */
   arc(midx, midy, stangle, endangle, radius);
   getch();
   closegraph();
   return 0;
}
```

programul nr.2

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

int main(void)
{
```

```
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int midx, midy, i;

  /* initializeaza variabilele locale si globale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }
  midx = getmaxx() / 2;
  midy = getmaxy() / 2;

  for (i=SOLID_FILL; i<USER_FILL; i++)
  {
    /* seteaza felul fill-ului */
    setfillstyle(i, getmaxcolor());

    /* deseneaza dreptunghiul */
    bar(midx-50, midy-50, midx+50,  midy+50);
    getch();
  }
 closegraph();
  return 0;
}
```

programul nr.3

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int midx, midy, i;

  /* initializeaza variabilele locale si globale*/
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
```

```
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;

  for (i=EMPTY_FILL; i<USER_FILL; i++)
  {
    setfillstyle(i, getmaxcolor());

    bar3d(midx-50, midy-50, midx+50,
         midy+50, 10, 1);

    getch();
  }

  closegraph();
  return 0;
}
```

programul nr.4

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int midx, midy;
  int radius = 100;

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }
```

```
  midx = getmaxx() / 2;
  midy = getmaxy() / 2;
  setcolor(getmaxcolor());

  /* deseneaza cercul */
  circle(midx, midy, radius);

    getch();
  closegraph();
  return 0;
}
```

programul nr.5

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

#define CLIP_ON 1

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int ht;

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  setcolor(getmaxcolor());
  ht = textheight("W");

  /* message in default full-screen viewport */
  outtextxy(0, 0, "* <-- (0, 0) in default viewport");

  /* create a smaller viewport */
  setviewport(50, 50, getmaxx()-50, getmaxy()-50, CLIP_ON);

  /* display some messages */
  outtextxy(0, 0, "* <-- (0, 0) in smaller viewport");
```

```
  outtextxy(0, 2*ht, "Press any key to clear viewport:");

  /* wait for a key */
  getch();

  /* clear the viewport */
  clearviewport();

  /* output another message */
  outtextxy(0, 0, "Press any key to quit:");


  getch();
  closegraph();
  return 0;
}
```

programul nr.6

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int maxx, maxy;

  int poly[10];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)

  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
      exit(1);
  }

  maxx = getmaxx();
  maxy = getmaxy();

  poly[0] = 20;        /* 1st vertext */
  poly[1] = maxy / 2;
```

```
  poly[2] = maxx - 20; /* 2nd */
  poly[3] = 20;

  poly[4] = maxx - 50; /* 3rd */
  poly[5] = maxy - 20;

  poly[6] = maxx / 2;  /* 4th */
  poly[7] = maxy / 2;
  poly[8] = poly[0];
  poly[9] = poly[1];

  /* deseneaza poligonul */
  drawpoly(5, poly);

  getch();
  closegraph();
  return 0;
}
```

programul nr.7

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int midx, midy;
  int stangle = 0, endangle = 360;
  int xradius = 100, yradius = 50;

  /* initializeaza variabilele locale si globale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)

  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;
```

```
  setcolor(getmaxcolor());

  /* deseneaza elipsa */
  ellipse(midx, midy, stangle, endangle, xradius, yradius);

  getch();
  closegraph();
  return 0;
}
```

programul nr.8

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int i, maxx, maxy;

  /* matricea poligonului */
  int poly[8];

  /* initializeaza variabilele locale si globale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)

  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  maxx = getmaxx();
  maxy = getmaxy();

  poly[0] = 20;        /* 1st vertex */
  poly[1] = maxy / 2;

  poly[2] = maxx - 20; /* 2nd */
  poly[3] = 20;

  poly[4] = maxx - 50; /* 3rd */
  poly[5] = maxy - 20;
```

```
  poly[6] = maxx / 2;
  poly[7] = maxy / 2;

  for (i=EMPTY_FILL; i<USER_FILL; i++)
  {
    setfillstyle(i, getmaxcolor());

    /* deseneaza un poligon plin */
    fillpoly(4, poly);
    getch();
  }

  closegraph();
  return 0;
}
```

programul nr.9

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int maxx, maxy;

  /* initializeaza variabilele locale si globale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)

  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  maxx = getmaxx();
  maxy = getmaxy();

  /* selecteaza culoarea de desenat */
  setcolor(getmaxcolor());

  /* selecteaza culoarea de umplere */
```

```
    setfillstyle(SOLID_FILL, getmaxcolor());

    /* deseneaza o margine a ecranului */
    rectangle(0, 0, maxx, maxy);

    /* deseneaza cateva cercuri */
    circle(maxx / 3, maxy /2, 50);
    circle(maxx / 2, 20, 100);
    circle(maxx-20, maxy-50, 75);
    circle(20, maxy-20, 25);
    getch();
    floodfill(2, 2, getmaxcolor());

    getch();
    closegraph();
    return 0;
 }
```

programul nr.10

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

int main(void)
{
   /* autodetectare */
   int gdriver = DETECT, gmode, errorcode;
   int bkcolor, midx, midy;
   char bkname[35];

/* initializare variabile globale si locale */
   initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

/* rezultatele initializarii */
   errorcode = graphresult();

   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Apasa orice tasta pentru a se opri:");
      getch();
      exit(1);
   }

   midx = getmaxx() / 2;
   midy = getmaxy() / 2;
   setcolor(getmaxcolor());
```

```
/* centrarea textului pe monitor */
  settextjustify(CENTER_TEXT, CENTER_TEXT);

  bkcolor = getbkcolor();

/* converteste constanta de culoare intr-un sir */
  itoa(bkcolor, bkname, 10);
  strcat(bkname, " is the current background color.");

/* afiseaza un mesaj */
  outtextxy(midx, midy, bkname);

  getch();
  closegraph();
  return 0;
}
```

programul nr.11

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int midx, midy;
  char colstr[80];

  /* initializare variabile globale si locale */
   initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;
  /* converteste informatia de culoare intr-un sir */
  sprintf(colstr, "Acest mod suporta culorile 0..%d", getmaxcolor());

   settextjustify(CENTER_TEXT, CENTER_TEXT);
  outtextxy(midx, midy, colstr);
  getch();
```

```
  closegraph();
  return 0;
}
```

programul nr.12

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

/* numele fonturilor */
char *font[] = { "DEFAULT_FONT",
                 "TRIPLEX_FONT",
                 "SMALL_FONT",
                 "SANS_SERIF_FONT",
                 "GOTHIC_FONT"
               };

/* numele directiei de scriere */
char *dir[] = { "HORIZ_DIR", "VERT_DIR" };

/* pozitionarea orizontala a textului */
char *hjust[] = { "LEFT_TEXT", "CENTER_TEXT", "RIGHT_TEXT" };

/* pozitionarea verticala a scrisului */
char *vjust[] = { "BOTTOM_TEXT", "CENTER_TEXT", "TOP_TEXT" };

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  struct textsettingstype textinfo;
  int midx, midy, ht;
  char fontstr[80], dirstr[80], sizestr[80];
  char hjuststr[80], vjuststr[80];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  midx = getmaxx() / 2;
```

```
  midy = getmaxy() / 2;

  /* informatii despre setarile curente ale textului */
  gettextsettings(&textinfo);

  /* converteste informatiile despre text intr-un sir */
  sprintf(fontstr, "%s is the text style.", font[textinfo.font]);
  sprintf(dirstr, "%s is the text direction.", dir[textinfo.direction]);
  sprintf(sizestr, "%d is the text size.", textinfo.charsize);
  sprintf(hjuststr, "%s is the horizontal justification.", hjust[textinfo.horiz]);
  sprintf(vjuststr, "%s is the vertical justification.", vjust[textinfo.vert]);

  ht = textheight("W");
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  outtextxy(midx, midy, fontstr);
  outtextxy(midx, midy+2*ht, dirstr);
  outtextxy(midx, midy+4*ht, sizestr);
  outtextxy(midx, midy+6*ht, hjuststr);
  outtextxy(midx, midy+8*ht, vjuststr);

  getch();
  closegraph();
  return 0;
}
```

programul nr.13

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int xmax, ymax;

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();

  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }
```

```
  setcolor(getmaxcolor());
  xmax = getmaxx();
  ymax = getmaxy();

  /* deseneaza o line  */
  line(0, 0, xmax, ymax);


  getch();
  closegraph();
  return 0;
}
```

programul nr.14

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

 int main(void)
 {
   /* autodetectare */
   int gdriver = DETECT, gmode, errorcode;
   char msg[80];

   /* initializare variabile globale si locale */
   initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

   /* rezultatele initializarii */
   errorcode = graphresult();
   if (errorcode != grOk)
   {
     printf("Graphics error: %s\n", grapherrormsg(errorcode));
     printf("Apasa orice tasta pentru a se opri:");
     getch();
     exit(1);
   }

   moveto(20, 30);

   /* creare mesaj la coordonatele (20, 30) */
   sprintf(msg, " (%d, %d)", getx(), gety());
   outtextxy(20, 30, msg);

      linerel(100, 100);

   sprintf(msg, " (%d, %d)", getx(), gety());
   outtext(msg);
```

```
    getch();
    closegraph();
    return 0;
 }
```

programul nr.15

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  char msg[80];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }
  moveto(20, 30);

  /* create and output a  message at (20, 30) */
  sprintf(msg, " (%d, %d)", getx(), gety());
  outtextxy(20, 30, msg);

  /* deseneaza o linie la coordonatele (100, 100) */
  lineto(100, 100);

  /* create and output a message at C.P. */
  sprintf(msg, " (%d, %d)", getx(), gety());
  outtext(msg);

  getch();
  closegraph();
  return 0;
}
```

programul nr.16

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

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  char msg[80];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }
  moveto(20, 30);

  putpixel(getx(), gety(), getmaxcolor());

  sprintf(msg, " (%d, %d)", getx(), gety());
  outtextxy(20, 30, msg);

  moverel(100, 100);

  putpixel(getx(), gety(), getmaxcolor());

  sprintf(msg, " (%d, %d)", getx(), gety());
  outtext(msg);

  getch();
  closegraph();
  return 0;
}
```

programul nr.17

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

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
```

```
  int midx, midy;

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;

  /* text la mijlocul ecranului */
  /* Nota: pointer-ul curent nu se schimba */
  outtextxy(midx, midy, "This is a test.");

    getch();
  closegraph();
  return 0;
}
```

programul nr.18

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

#define PIXEL_COUNT 1000
#define DELAY_TIME  100  /* in milisecunde */

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int i, x, y, color, maxx, maxy,
      maxcolor, seed;

/* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

/* rezultatele initializarii */
  errorcode = graphresult();
```

```
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();

    exit(1);
  }

  maxx = getmaxx() + 1;
  maxy = getmaxy() + 1;
  maxcolor = getmaxcolor() + 1;

  while (!kbhit())
  {
    seed = random(32767);
    srand(seed);
    for (i=0; i<PIXEL_COUNT; i++)
    {
      x = random(maxx);
      y = random(maxy);
      color = random(maxcolor);
      putpixel(x, y, color);
    }

    delay(DELAY_TIME);
    srand(seed);
    for (i=0; i<PIXEL_COUNT; i++)
    {
      x = random(maxx);
      y = random(maxy);
      color = random(maxcolor);
      if (color == getpixel(x, y))
        putpixel(x, y, 0);
    }
  }


  getch();
  closegraph();
  return 0;
}
```

programul nr.19

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

int main(void)
{
```

```
  /* select a driver and mode that supports */
  /* multiple background colors.          */
  int gdriver = EGA, gmode = EGAHI, errorcode;
  int bkcol, maxcolor, x, y;
  char msg[80];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  /* indexul de culoare maxima suportata */
  maxcolor = getmaxcolor();

  /* centrarea mesajului text */
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  x = getmaxx() / 2;
  y = getmaxy() / 2;

  /* ciclu in culorile disponibile */
  for (bkcol=0; bkcol<=maxcolor; bkcol++)
  {
    /* sterge ecranul */
    cleardevice();

    /* selecteaza o noua culoare a ecranului */
    setbkcolor(bkcol);

    /* afiseaza un mesaj */
    if (bkcol == WHITE)
      setcolor(EGA_BLUE);
    sprintf(msg, "Background color: %d", bkcol);
    outtextxy(x, y, msg);
    getch();
  }

    closegraph();
  return 0;
}
```

programul nr.20

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

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

int main(void)
{
  /* select a driver and mode that supports */
  /* multiple drawing colors.              */
  int gdriver = EGA, gmode = EGAHI, errorcode;
  int color, maxcolor, x, y;
  char msg[80];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  maxcolor = getmaxcolor();

  settextjustify(CENTER_TEXT, CENTER_TEXT);
  x = getmaxx() / 2;
  y = getmaxy() / 2;

  for (color=1; color<=maxcolor; color++)
  {
    cleardevice();

        setcolor(color);

    sprintf(msg, "Color: %d", color);
    outtextxy(x, y, msg);
    getch();
  }

  closegraph();
  return 0;
}
```

programul nr.21

```
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
```

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

/* numele stilurilor de umplere  */
char *fname[] = { "EMPTY_FILL",
                  "SOLID_FILL",
                  "LINE_FILL",
                  "LTSLASH_FILL",
                  "SLASH_FILL",
                  "BKSLASH_FILL",
                  "LTBKSLASH_FILL",
                  "HATCH_FILL",
                  "XHATCH_FILL",
                  "INTERLEAVE_FILL",
                  "WIDE_DOT_FILL",
                  "CLOSE_DOT_FILL",
                  "USER_FILL"
                };

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;
  int style, midx, midy;
  char stylestr[40];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;

  for (style = EMPTY_FILL; style < USER_FILL; style++)
  {
    /* selecteaza stilul de umplere */
    setfillstyle(style, getmaxcolor());

    /* converteste stilul intr-un sir */
    strcpy(stylestr, fname[style]);
```

```
    /* umple un dreptunghi */
    bar3d(0, 0, midx-10, midy, 0, 0);

    outtextxy(midx, midy, stylestr);

    getch();
    cleardevice();
  }


  getch();
  closegraph();
  return 0;
}
```

programul nr.22

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

/* the names of the line styles supported */
char *lname[] = {
  "SOLID_LINE",
  "DOTTED_LINE",
  "CENTER_LINE",
  "DASHED_LINE",
  "USERBIT_LINE"
  };

int main(void)
{
  /* autodetectare */
  int gdriver = DETECT, gmode, errorcode;

  int style, midx, midy, userpat;
  char stylestr[40];

  /* initializare variabile globale si locale */
  initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

  /* rezultatele initializarii */
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
```

```
  }

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;

  /* a user defined line pattern */
  /* binary: "0000000000000001"  */
  userpat = 1;

  for (style=SOLID_LINE; style<=USERBIT_LINE; style++)
  {
    /* selecteaza stilul liniei */
    setlinestyle(style, userpat, 1);

    /* converteste stilul intr-un sir */
    strcpy(stylestr, lname[style]);

    /* deseneaza o linie */
    line(0, 0, midx-10, midy);

    /* deseneaza un dreptunghi */
    rectangle(0, 0, getmaxx(), getmaxy());

    outtextxy(midx, midy, stylestr);

    getch();
    cleardevice();
  }

  closegraph();
  return 0;
}
```

programul nr.23

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

 #define CLIP_ON 1   /* activates clipping in viewport */

 int main(void)
 {
   /* autodetectare */
   int gdriver = DETECT, gmode, errorcode;

   /* initializare variabile globale si locale */
   initgraph(&gdriver, &gmode, "c:\\bc\\borlandc\\bgi");

   /* rezultatele initializarii */
```

```
  errorcode = graphresult();
  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Apasa orice tasta pentru a se opri:");
    getch();
    exit(1);
  }

  setcolor(getmaxcolor());

  /* mesaj in fereastra implicita */
  outtextxy(0, 0, "* <-- (0, 0) in default viewport");

  /* crearea unei ferestre grafice mai mici */
  setviewport(50, 50, getmaxx()-50, getmaxy()-50, CLIP_ON);

  outtextxy(0, 0, "* <-- (0, 0) in smaller viewport");

  getch();
  closegraph();
  return 0;
}
```