Geometrie Computațională

Laborator 1

Un program simplu de grafică în C este următorul:

```
//#include <graphics.h>
#include "graphics.h"
int main()
{
       initwindow(800, 600, "Exemplu lab 1", 200, 200);
       setbkcolor(3); //setbkcolor(CYAN);
       cleardevice(); //stabileste culoarea ecranului
       outtextxy(80, 90, "Am scris");
       getchar(); //asteapta o tasta
       closegraph();
       return 0;
      }
```

1. Iniţializarea ecranului grafic

Limbajul C are peste 70 de funcţii grafice, de la funcţii de nivel înalt (setviewport(), bar3d(), drawpoly(), etc.) până la funcţii orientate pe bit (getimage(), putimage(), etc.), incluse toate în biblioteca de funcţii grafice graphics.h. Biblioteca grafică conţine numeroase stiluri de linii şi modele de umplere, câteva fonturi cărora li se poate modifica dimensiunea, care pot fi aliniate şi pot fi orientate orizontal sau vertical. De asemenea, pachetul grafic include drivere pentru perifericele grafice (*.BGI) şi fişiere de fonturi (*.CHR). Pentru a putea folosi funcţiile grafice trebuie ca fiecare fişier care foloseşte grafică să conţină directiva:

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

Pentru a iniţializa modul grafic, trebuie mai întâi apelată funcţia **initwindow,** care identifică adaptorul grafic, încarcă şi iniţializează cel mai potrivit driver, trece sistemul în modul grafic şi întoarce controlul rutinelor grafice. Putem cere funcţiei **initgraph** să folosească un anumit driver grafic şi un anumit mod, sau să autodetecteze adaptorul grafic şi să aleagă driverul corespunzător.

Funcţia **initwindow** resetează toate setările grafice la valorile lor implicite (poziţia curentă, paleta, culoarea, viewport-ul etc.), resetează codul de eroare la 0 şi încarcă driverul grafic alocând memorie pentru el. Această funcţie poate fi folosită singură sau împreună cu o altă funcţie numită **detectgraph** care determină parametric adaptorului grafic. Prototipul ei este:

```
void initwindow(int x1,int y1,char* nume,int x2,int y2);
```

unde (x1,y1) reprezintă dimensiunile ferestrei grafice, date în pixeli, *nume* este numele ferestrei, care va apare explicit pe bara albastră superioară de control a poziţiei ferestrei (dacă lipseşte *nume*, valoarea implicită utilizată este “Win BGI”), iar (x2,y2) este poziţia relativă a colţului stânga sus al ferestrei faţă de colţul stânga sus al ecranului.

Din modul grafic se poate ieşi apelând funcţia **closegraph** de prototip:

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

2. Culorile modului graphic

Adaptoarele grafice sunt prevăzute cu o zonă de memorie în care se păstrează date specifice gestiunii ecranului. Acestă zonă de memorie poartă numele de *memorie video*.

În cazul adaptoarelor color, unui pixel îi corespunde o culoare. Culoarea pixelilor se păstrează pe biţi în memoria video. Memoria video necesară pentru a păstra starea ecranului setat în mod grafic, se numeşte *pagină video*. Gestiunea culorilor este dependentă de tipul de adaptor grafic existent la microprocesor.

În cele ce urmează considerăm adaptoarele grafice de tip EGA/VGA. Numărul maxim al culorilor care pot fi afişate cu ajutorul unui adaptor EGA este de 64. Culorile se codifică prin numerele întregi din intervalul [0,63]. Cele 64 de culori nu pot fi definite/afişate simultan pe ecran.

În cazul adaptorului EGA se pot afişa simultan pe ecran cel mult 16 culori. Culorile se codifică prin numerele întregi din intervalul [0,15]. Mulţimea culorilor care pot fi afişate pe ecran simultan se numeşte *paletă*. Culorile din componenţa unei palete pot fi modificate de utilizator prin intermediul funcţiilor standard. La iniţializarea modului grafic se setează o paletă implicită. Paleta se defineşte cu ajutorul unui tablou de 16 elemente pentru adaptorul EGA. Elementele acestui tablou au valori din intervalul [0,63], unde fiecare element din acest tablou reprezintă codul unei culori.

Codurile culorilor din paleta implicită au denumiri simbolice definite în fişierul **graphics.h**. Funcţiile de gestiune a culorilor pot avea ca parametri nu numai codurile culorilor, ci şi indecşi în tabloul care defineşte culorile unei palete. De aceea, indicii din intervalul [0,15] pot fi referiţi prin constante simbolice definite în fişierul **graphics.h**. Aceste denumiri sugerează culoarea din compunerea paletei.

Culoarea fondului de ecran (**background**) este întotdeauna cea corespunzătoare indicelui 0. Culoarea pentru desenare (**foreground**) este cea corespunzătoare indicelui 15.

Culoarea de fond poate fi modificată cu ajutorul funcţiei **setbkcolor**. Aceasta are prototipul:

```
void setbkcolor(int culoare);
```

unde: `culoare` este index în tabloul care defineşte paleta. De exemplu, dacă se utilizează apelul:

```
setbkcolor(BLUE);
```

atunci culoarea de fond a textului ce urmează a fi scris pe ecran devine albastră. Pentru a schimba culoarea întregul ecran de desenare se foloseşte instrucţiunea suplimentară

```
void cleardevice( );
```

la începutul sesiunii de lucru:

```
setbkcolor(BLUE);
cleardevice();
```

Pentru a cunoaşte culoarea de fond curentă se poate apela funcţia **getbkcolor** de prototip:

```
int getbkcolor(void);
```

Ea returnează indexul în tabloul care defineşte paleta pentru culoarea de fond.

Culoarea pentru desenare poate fi modificată folosind funcţia **setcolor** de prototip:

```
void setcolor(int culoare);
```

unde: `culoare` este indexul unei culori în tabloul care defineşte paleta.

În sfârşit, afişarea şirurilor de caractere în modul grafic diferă de afişarea normală din C (cu `printf()`) sau C++ (cu `cout`). Pentru afişarea unui mesaj *c* începând din locaţia punctului se utilizează:

```
void outtextxy(int x,int y, *char c);
```

unde (*x,y*) sunt coordonatele de la care se afişează stringul *c*. Pentru afişarea numerelor se va folosi o conversie a acestora în şiruri de caractere, de exemplu prin funcţia

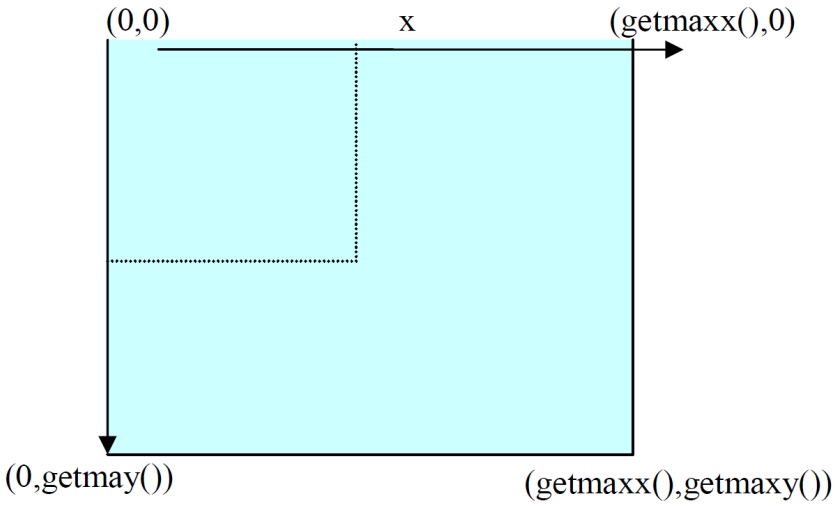
```
*char itoc(int n).
```

În tabelul următor se indică codurile culorilor pentru paleta implicită.

Culorile în C	
denumire simbolică	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
WHITE	15

3. Coordonate-ecran ale pixelilor

În mod grafic, ecranul se compune din *n*m* pixeli, în funcţie de rezoluţia adaptorului grafic. Aceasta înseamnă că pe ecran se pot afişa *m* linii a cate *n* pixeli fiecare. Poziţia unui pixel se defineşte printr-o pereche ordonată de numere întregi: (x,y) numite coordonatele pixelului. Coordonata x defineşte coloana pixelului, iar y defineşte linia acestuia. Pixelul aflat în colţul din stânga sus are coordonatele (0,0). Coloanele se numerotează de la stânga spre dreapta, iar liniile de sus în jos.



Ecranul grafic în C

Biblioteca grafică a programului conţine funcţii care permit utilizatorului să obţină următoarele informaţii relativ la ecran:

- coordonata maximă pe orizontală;
- coordonata maximă pe verticală;
- poziţia curentă (pixel curent).

Prototipurile acestor funcţii sunt:

```
int getmaxx(void);
```

funcţia returnează coordonata maximă pe orizontală (abscisa maximă);

```
int getmaxy(void);
```

funcţia returnează coordonata maximă pe vertical (ordonata maximă);

```
int getx(void);
```

funcţia returnează poziţia orizontală (abscisa) a pixelului curent;

```
int gety(void);
```

funcţia returnează poziţia verticală (ordonata) a pixelului current

4. Funcţiile elementare de desenare

Biblioteca standard a sistemului pune la dispoziţia utilizatorului o serie de funcţii care permit desenarea şi colorarea unor figuri geometric

Un punct colorat (pixel) se afişează cu ajutorul funcţiei **putpixel** de prototip:

```
void putpixel(int x, int y, int culoare);
```

unde: `(x,y)` - defineşte poziţia punctului;
`culoare` - defineşte culoarea punctului şi este un întreg din intervalul [0,15].

Pentru trasarea liniilor se pot folosi trei funcţii: **line, lineto, linerel**.

Funcţia **line** are prototipul:

```
void line (int xstart,int ystart,int xfin,int yfin);
```

Funcţia trasează un segment de dreaptă ale cărui capete sunt punctele de coordonate: `(xstart,ystart)` şi `(xfin,yfin)`.

Funcţia **lineto** are prototipul:

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

Ea trasează un segment de dreaptă care are ca origine poziţia curentă, iar ca şi punct final cel de coordonate `(x,y)`. Punctul final devine poziţia curentă. De notat că funcţia **moveto** permite definirea poziţiei curente.

Funcţia **linerel** are prototipul:

```
void linerel (int x, int y);
```

Dacă notăm cu `xcrt` şi `ycrt` coordonatele poziţiei curente, atunci funcţia **linerel** trasează un segment de dreaptă ale cărui capete sunt punctele de coordonate: `(xcrt,ycrt)` şi `(xcrt+x,ycrt+y)`.

Alte funcţii care permit trasări de figuri geometrice utilizate frecvent sunt:

```
void arc (int xcentru, int ycentru, int unghistart, int unghifin, int raza);
```

care trasează un arc de cerc; unghiurile sunt exprimate în grade sexagesimale.

`void` **`circle`** `(int xcentru, int ycentru, int raza);`

trasează un cerc; `(xcentru,ycentru)` fiind coordonatele centrului arcului de cerc şi respectiv cercului trasat de aceste funcţii; parametrul `raza` defineşte mărimea razei curbelor respective.

`void` **`ellipse`** `(int xcentru, int ycentru, int unghistart, int unghifin, int semiaxamare, int semiaxamica);`

trasează un arc de elipsă cu centrul în punctul de coordinate (xcentru,ycentru) având semiaxa mică definită de parametrul `semiaxamica`, iar semiaxa mare de parametrul `semiaxamare`.

`void` **`rectangle`** `(int st, int sus, int dr, int jos);`

trasează un dreptunghi definit de colţurile sale opuse: `(st,sus)` – colţul din stânga sus şi `(dr,jos)` – colţul din dreapta jos.

`void` **`drawpoly`** `(int nr, int *tabpct);`

trasează o linie poligonală, unde:

`nr` - numărul laturilor;

`tabpct` - este un pointer spre întregi care defines coordonatele liniei poligonale. Acestea sunt păstrate sub forma: `abscisa_i, ordonata_i` unde `i` are valorile `1,2,...nr+1`. Linia poligonală este închisă dacă primul punct coincide cu ultimul (au aceleaşi coordonate). Coordonatele utilizate ca parametri la apelul acestor funcţii sunt relative la fereastra activă. Culoarea de trasare este cea curentă.

```
// Template project.
#include "graphics.h"
#include <math.h>
#include <stdlib.h>
#include <dos.h>
#include <windows.h>
#include <iostream>

using namespace std;

int main()
{
       initwindow(800, 600, "Exemplu lab 1", 200, 200);
       setbkcolor(3); //setbkcolor(CYAN);
       cleardevice(); //stabileste culoarea ecranului
       outtextxy(80, 90, "Am scris");
       setcolor(5);
       line(100, 200, 400, 500);
       circle(300, 350, 50);
       getchar(); //asteapta o tasta
       closegraph();
       return 0;
     }
```

Aplicaţii:

1. Modificați primul program astfel încât să afișați câte un element din fiecare cele studiate.
2. Desenați un cerc centrat în mijlocul ferestrei grafice. Plasați 3 triunghiuri de coordonate arbitrare în interiorul acestuia, schimbând culorile de desenare ale acestora. **Indicație:** folosiți funcția **rand()**.