

**Міністерство освіти і науки України Національний  
університет “Львівська Політехніка”**



**Лабораторна робота №20  
з дисципліни «Програмування частина  
№ 2»**

**Виконав:**  
Студент групи АП-11  
Іщенко Василь

**Прийняв:**  
Чайковський І.Б.

**Тема роботи:** Дослідження графічного режиму роботи мови програмування С.

**Мета роботи:** Дослідження основних принципів відображення графічної інформації на екрані дисплея.

### Теоретичні відомості

Для оформлення діалогу користувача з комп'ютером (програмою) потрібна розвинена система функцій управління роботою екрану. Пакет функцій управління екраном ділиться на дві частини. Перша підтримує текстовий режим (text mode) роботи. У текстовому режимі екран монітора умовно розбивається на окремі ділянки, частіше всього на 25 рядків по 80 символів (знакомісць). У кожне знакомісце може бути виведений один з 256 заздалегідь заданих символів. Друга частина забезпечує роботу екрану в графічному режимі (graphics mode). Він призначений для виведення на екран графіків, діаграм, малюнків тощо. У цьому режимі екран монітора є безліччю точок, кожна з яких може бути одним із декількох кольорів. Кількість точок по горизонталі і вертикалі називається роздільною здатністю монітора в цьому режимі.

#### Ініціалізація графічного режиму.

До складу графічного пакету входять:

заголовний файл graphics.h;

бібліотечний файл graphics.lib;

драйвери графічних пристроїв (\*.bgi);

шрифти (\*.chr).

Кольори задають числами, або англійськими назвами

таблиця 1.

BLACK (0)	<u>чорний</u>	DARKGRAY (8)	<u>темний сірий</u>
BLUE (1)	<u>синій</u>	LIGHTBLUE (9)	<u>яскравий синій</u>
GREEN (2)	<u>зелений</u>	LIGHTGREEN (10)	<u>яскравий зелений</u>
CYAN (3)	<u>блакитний</u>	LIGHTCYAN (11)	<u>яскравий блакитний</u>
RED (4)	<u>червоний</u>	LIGHTRED (12)	<u>червоний</u>
MAGENTA (5)	<u>фіолетовий</u>	LIGHTMAGENTA (13)	<u>фіолетовий</u>
BROWN (6)	<u>коричневий</u>	YELLOW (14)	<u>жовтий</u>
LIGHTGRAY (7)	<u>світлий сірий</u>	WHITE (15)	<u>білий</u>

#### Основні функції для графічних побудов:

setcolor (колір);

setbkcolor (колір) колір тла.;

```

putpixel (z,y,колір);
line(x1,y1,x2,y2);
lineto(x,y);
bar(x1,y1,x2,y2);
rectangle(x1,y1,x2,y2);
circle(x,y,R);
arc(x,y,початк. кут, кінцевий кут, радіус);
closegraph();
outtext(текст);
outtextxy(x,y,текст);
settextstyle(шрифт, напрям, розмір);
getmaxx() – повертає розмір екрану по горизонталі;
getmaxy() – повертає розмір екрану по вертикалі;
getcolor() – повертає значення поточного кольору;
getx(), gety() – повертають координати поточного пікселя.

```

### Приклад 1

```

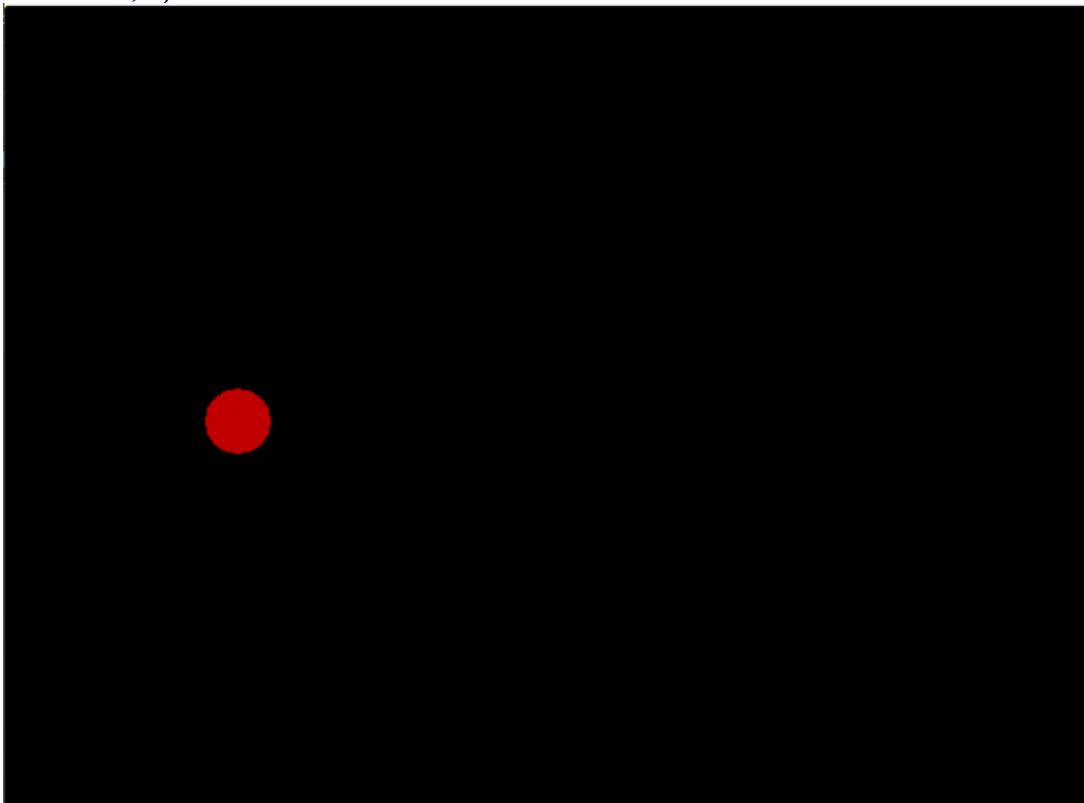
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
int main(void){
    /* автоматичне визначення графічного драйвера */
    int gdriver = DETECT, gmode, errorcode; // gdriver і gmode використовуються для
автоматичного визначення графічного режиму, errorcode для перевірки помилок
    int i, size; // і використовується як лічильник в циклі, size для зберігання розміру
буфера
    void *buf; // указівник для зберігання зображення
    /* ініціалізація графічного режиму */
    initgraph(&gdriver, &gmode, ""); // Ініціалізація графічного режиму з
автоматичним визначенням драйвера і режиму
    /* перевірка результату ініціалізації */
    errorcode = graphresult(); // Отримання коду результату ініціалізації
    /* якщо сталася помилка */
    if (errorcode != grOk){
        printf("Графічна помилка: %s\n", grapherrormsg(errorcode)); // Виведення
повідомлення про помилку
        printf("Натисніть будь-яку клавішу для завершення:"); // Запит на натискання
клавіші для завершення
        getch(); // Очікування натискання клавіші
        exit(1); // Завершення програми з кодом 1 (помилка)
    }
    setcolor(BLACK); // Встановлення кольору для подальших графічних операцій
    setfillstyle(1, RED); // Встановлення стилю заливки (1 - суцільна заливка, RED -
червоний колір)

```

```

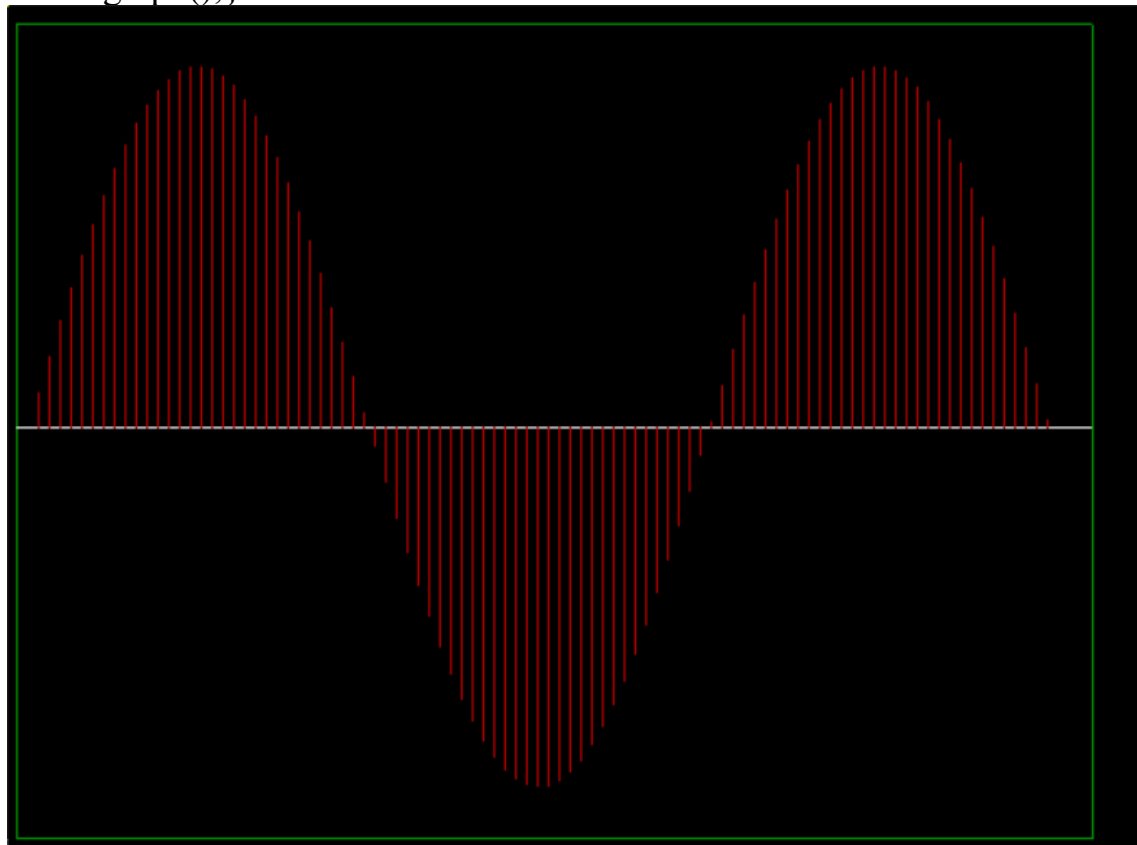
    fillellipse(21, 240, 20, 20); // Малювання заповненого еліпса з центром у точці (21,
240) і радіусами 20
    /* розрахунок розміру буфера, необхідного для збереження зображення */
    size = imagesize(1, 220, 41, 260); // Обчислення розміру пам'яті, необхідної для
збереження області зображення
    buf = malloc(size); // Виділення пам'яті для збереження зображення
    if (buf == NULL) {
        printf("Помилка виділення пам'яті\n"); // Виведення повідомлення про
помилку, якщо пам'ять не була виділена
        closegraph(); // Закриття графічного режиму
        exit(1); // Завершення програми з кодом 1 (помилка)
    }
    getimage(1, 220, 41, 260, buf); // Збереження області екрана розміром 41x260 з
координатами лівого верхнього кута (1, 220) у буфер
    setfillstyle(1, BLACK); // Встановлення стилю заливки (1 - суцільна заливка,
BLACK - чорний колір)
    for(i = 1; i <= 620; i++) // Цикл для переміщення еліпса по екрану
    {
        bar(i - 1, 220, i + 39, 260); // Очищення попереднього положення еліпса.
Задається прямокутник, який перекриває попереднє положення еліпса
        putimage(i, 220, buf, COPY_PUT); // Відображення зображення з буфера у
новій позиції. (i, 220) - нова координата лівого верхнього кута зображення
        delay(10); // Затримка на 10 мс для створення ефекту анімації
    }
    free(buf); // Звільнення виділеної пам'яті для буфера
    closegraph(); // Закриття графічного режиму і повернення екрану у текстовий
режим
    return 0; }

```



## Завдання 2

```
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
main(){
    float a = 5;
    int x = 0;
    int grdrv=DETECT, grmod;
    initgraph(&grdrv,&grmod, "C:\\TC\\BGI");
    setbkcolor(BLACK);
    setcolor(GREEN);
    line(5,10,600,10);
    line(5,460,600,460);
    line(5,460,5,10);
    line(600,460,600,10);
    setcolor(WHITE);
    line(5,getmaxy()/2,600,getmaxy()/2);
    for(float i = 0; i<=3*M_PI;i+=0.1){
        float b = sin(i);
        a = a+(getmaxxx()/91);
        setcolor(RED);
        line(a,getmaxy()/2,a,(getmaxy()/2)-(b*200));
        delay(10);
    }
    getch();
    closegraph();}
```



### Завдання 3

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<windows.h>
main(){
    SetConsoleCP(65001);
    SetConsoleOutputCP(65001);
    char a[50];
    int b=1;
    int c = 0;
    printf("Введіть текст ");
    scanf("%s",&a);
    printf("Введіть множник розміру тексту ");
    scanf("%d",&b);
    printf("Введіть стиль тексту ");
    scanf("%d",&c);
    int grdrv=DETECT, grmod;
    initgraph(&grdrv,&grmod, "C:\\TC\\BGI");
    setbkcolor(BLACK);
    setcolor(RED);
    settextstyle(c, 0, b);
    outtextxy(100,100, a);
    getch();
    closegraph();
}
```



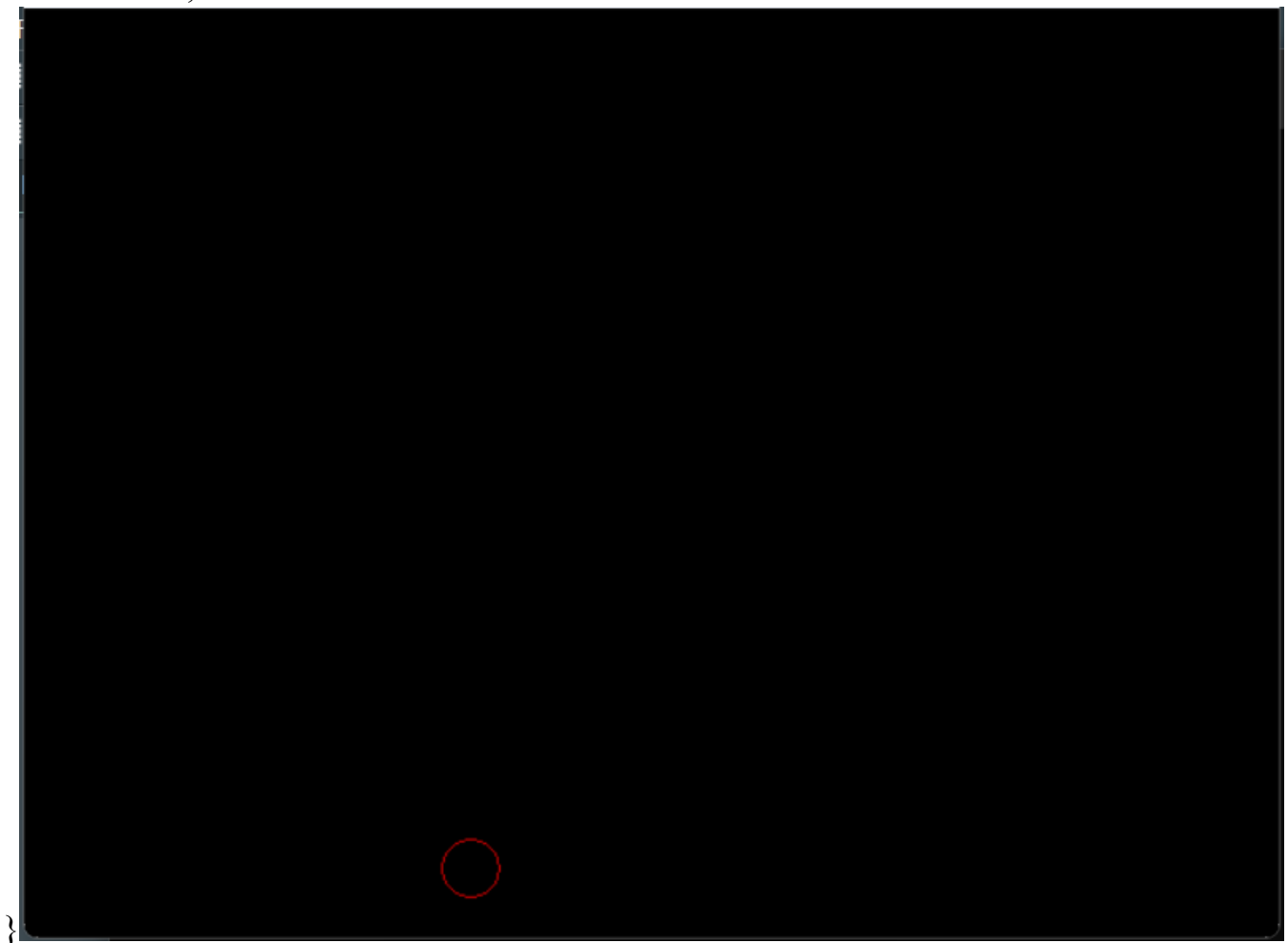
### Приклад 4

```
#include <graphics.h>
#include <conio.h>
int main() {
    int gd = DETECT, gm; // Змінні для графічного режиму та графічного режиму
    initgraph(&gd, &gm, ""); // Ініціалізація графічного режиму
    int x = getmaxx()/2, y = getmaxy()/2; // Початкові координати круга
    int dx = 1, dy = 1, r = 15; // Кроки переміщення по осях x та y
    // Безкінечний цикл, поки не буде натиснута клавіша
```

```

while (!kbhit()) {
    cleardevice(); // Очищення вікна
    int color = RED;
    circle(x, y, r); // Малювання круга з центром у (x, y) та радіусом 15
    x += dx; // Зміна координати x
    y += dy; // Зміна координати y
    delay(2); // Затримка для плавного анімованого ефекту
    // Перевірка зіткнення з межами вікна по осі x
    if (x <= r || x >= getmaxx()-r) {
        dx = -dx; // Зміна напрямку руху по осі x
        color = WHITE;
        setcolor(color);
    }
    // Перевірка зіткнення з межами вікна по осі y
    if (y <= r || y >= getmaxy()-r) {
        dy = -dy; // Зміна напрямку руху по осі y
        color = RED;
        setcolor(color);
    }
}
closegraph(); // Закриття графічного вікна
return 0;

```



### Приклад 5

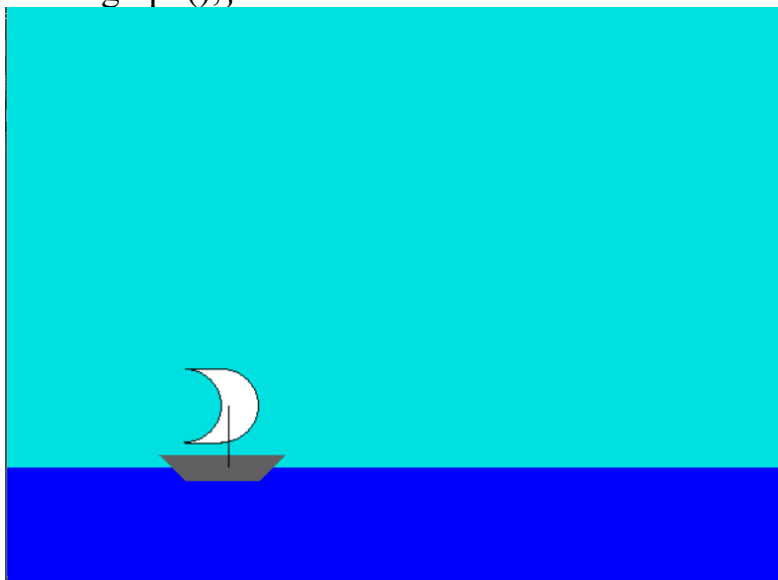
```
#include <graphics.h>
#include <conio.h>
main(){
    int xr=100;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    setcolor(WHITE);
    while (!kbhit()) {
        for (xr = 100; xr >= 0; xr-=2) {
            cleardevice();
            fillellipse(getmaxx()/2, getmaxy()/2, xr, 100);
            delay(20);
        }
        for (xr = 0; xr <= 100; xr+=2) {
            cleardevice();
            fillellipse(getmaxx()/2, getmaxy()/2, xr, 100);
            delay(20);
        }
    }
    closegraph();
}
```





## Приклад 6

```
#include<graphics.h>
#include<conio.h>
#include<math.h>
main(){
    int grdrv=DETECT, grmod,size;
    initgraph(&grdrv,&grmod, "C:\\TC\\BGI");
    setbkcolor(CYAN);
    int x=0,y=getmaxy()*4/5-10;
    void *buf;
    cleardevice();
    setcolor(BLUE);
    setfillstyle(SOLID_FILL, BLUE);
    bar(0,getmaxy()*4/5,getmaxx()+1,getmaxy()+1);
    setcolor(DARKGRAY);
    setfillstyle(SOLID_FILL, DARKGRAY);
    int base[] = {x, y, x + 100, y, x + 80, y+20, x + 20, y+20};
    fillpoly(4, base);
    setcolor(BLACK);
    setfillstyle(SOLID_FILL, WHITE);
    arc(50,getmaxy()*4/5-50,-90,90,30);
    arc(20,getmaxy()*4/5-50,-90,90,30);
    line(20,getmaxy()*4/5-80,50,getmaxy()*4/5-80);
    line(20,getmaxy()*4/5-20,50,getmaxy()*4/5-20);
    line(55,getmaxy()*4/5-50,55,getmaxy()*4/5);
    floodfill(60,getmaxy()*4/5-50 , BLACK);
    buf=malloc(size);
    getimage(0,getmaxy()*4/5-90,110,getmaxy()*4/5+20, buf);
    setfillstyle(1, CYAN);
    for(int i=1;i<=getmaxx();i++){
        bar(i - 1,getmaxy()*4/5-90,110,getmaxy()*4/5);
        putimage(i, getmaxy()*4/5-90, buf, COPY_PUT);
        delay(10);}
    closegraph();}
```



## **відповіді на контрольні запитання**

### **1) Текстовий режим роботи:**

Характеристики: У текстовому режимі програма виводить результати своєї роботи у вигляді тексту у консольному вікні.

Призначення: Текстовий режим використовується для виконання програм, які виводять інформацію у текстовому форматі, такі як звичайні текстові операції, обробка файлів, консольні інтерфейси користувача тощо.

Приклад функціоналу: Виведення тексту у консоль за допомогою функцій `printf()`, зчитування введених користувачем значень за допомогою функцій `scanf()`.

### **Графічний режим роботи:**

Характеристики: У графічному режимі програма виводить результати своєї роботи у вигляді графічних об'єктів на екрані комп'ютера.

Призначення: Графічний режим використовується для створення візуальних інтерфейсів користувача, малювання графіків, анімації, ігор, роботи з фото та відео, інтерактивних візуалізацій та інших графічних застосувань.

Приклад функціоналу: Малювання графічних об'єктів (ліній, кіл, прямокутників) за допомогою функцій з графічної бібліотеки (наприклад, `line()`, `circle()`, `rectangle()`), анімація об'єктів, робота з мишкою та клавіатурою, відображення зображень тощо.

### **2) Керування кольором і вибір палітри в мові C:**

У мові C, для керування кольором і вибору палітри використовуються функції з графічної бібліотеки, наприклад, з `graphics.h`. Зазвичай це функції типу `setcolor()`, `setbkcolor()`, `setpalette()` та інші. Наприклад:

```
setcolor(RED);    // Встановлення кольору контуру
```

```
setbkcolor(WHITE); // Встановлення кольору фону
```

### **3) Основні функції для графічного режиму роботи в мові C:**

Основні функції для графічного режиму роботи в мові C зазвичай включають ініціалізацію графічного режиму, очищення екрану, малювання графічних об'єктів та завершення роботи з графікою. Деякі з цих функцій включають:

`initgraph()`: Ініціалізація графічного режиму.

`closegraph()`: Завершення графічного режиму.

`cleardevice()`: Очищення екрану.

Функції малювання об'єктів, такі як `line()`, `circle()`, `rectangle()` і т. д.

### **4) Принципи роботи з частинами графічного екрану в мові C:**

В мові C для роботи з частинами графічного екрану використовуються функції для малювання та очищення областей екрану. Наприклад, функція `bar()` використовується для малювання прямокутних областей, а функція `cleardevice()` - для очищення екрану. Також можна використовувати функції для роботи з кольорами, які дозволяють встановлювати кольори для контуру, заливки фону та інших елементів.

Приклад:

```
setcolor(BLUE);           // Встановлення кольору контуру
```

```
setfillstyle(SOLID_FILL, BLUE); // Встановлення стилю заливки
```

```
bar(0, y + 20, getmaxx(), getmaxy()); // Малювання прямокутника
```

**Висновок:** Я дослідив основні принципи відображення графічної інформації на екрані дисплея.