Міністерство освіти і науки України
Національний університет “Львівська Політехніка”

Лабораторна робота №20
З дисципліни
“Програмування частина 2”

Виконав:
Студент групи АП-11
Гишка Остап

Прийняв:
Чайковський І.Б.

Львів-2024

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

Основні функції для графічних побудов:

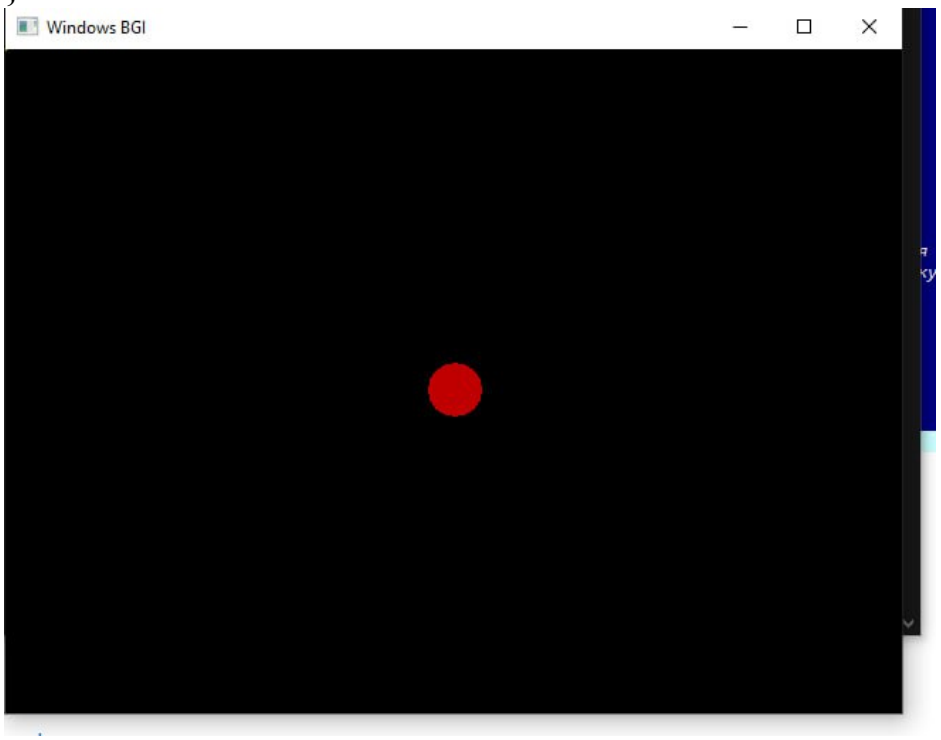
setcolor (колір);
setbkcolor (колір) колір тла.;
putpixel (z,y,колір);
line(x1,y1,x2,y2);
lineto(x,y);
bar(x1,y1,x2,y2);
rectangle(x1,y1,x2,y2);
circle(x,y,R);
arc(x,y,початк. кут, кінцевий кут, радіус);
closegraph();
outtext(текст);
outtextxy(x,y,текст);
settextstyle(шрифт, напрям, розмір);
getmaxx() – 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;
}
```

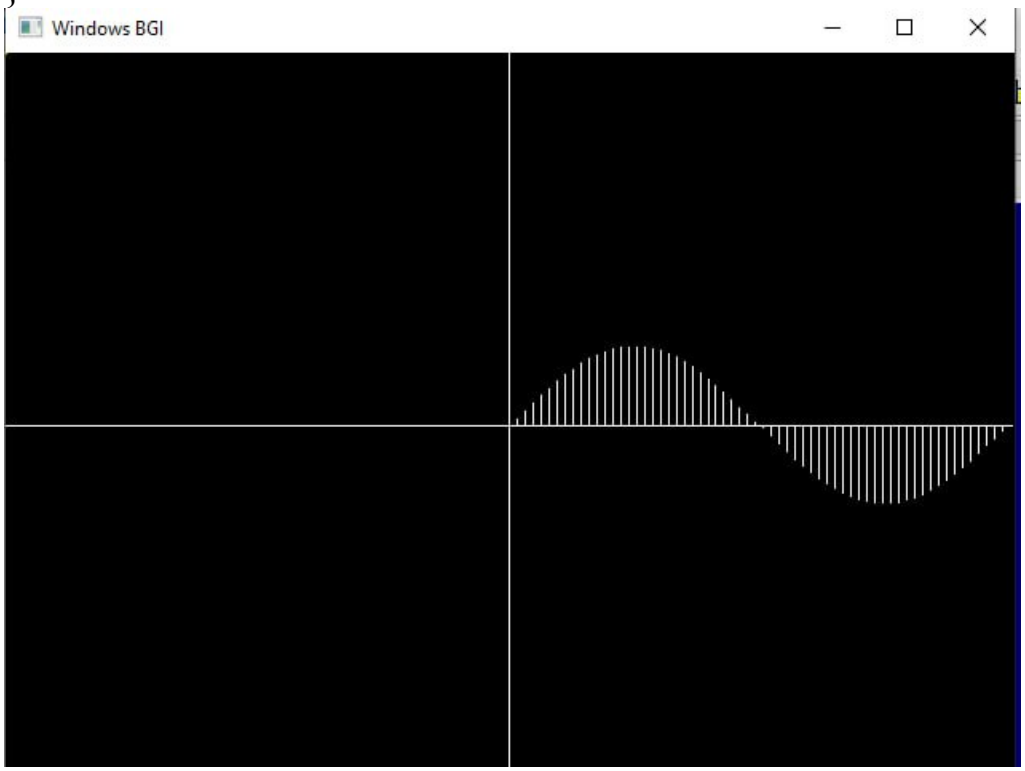


Приклад 2

```
#include <graphics.h>
#include <math.h>
#include <stdio.h>
int main() {
    // Ініціалізація графічного режиму
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    // Встановлення початкової точки та масштабування
    int originX = getmaxx() / 2;  // Початкова точка по осі x (центр екрану)
    int originY = getmaxy() / 2;  // Початкова точка по осі y (центр екрану)
    float scaleX = 50;            // Масштабний коефіцієнт для осі x
    float scaleY = 50;            // Масштабний коефіцієнт для осі y

    // Намалювати осі x та y
    line(0, originY, getmaxx(), originY);  // Ось x
    line(originX, 0, originX, getmaxy());  // Ось y

    // Намалювати графік функції y = sin(x)
    float x;
    for (x = 0; x <= 3 * M_PI; x += 0.1) {
        float y = sin(x);                    // Знаходження значення y
        int screenX = originX + x * scaleX;       // Перетворення координати x на
```

екран

```
        int screenY = originY - y * scaleY;       // Перетворення координати y на екран
        line(screenX, originY, screenX, screenY);  // Намалювати вертикальну лінію
    }

    // Очікування натискання клавіші
    getch();

    // Закриття графічного вікна
    closegraph();
    return 0;
}
```

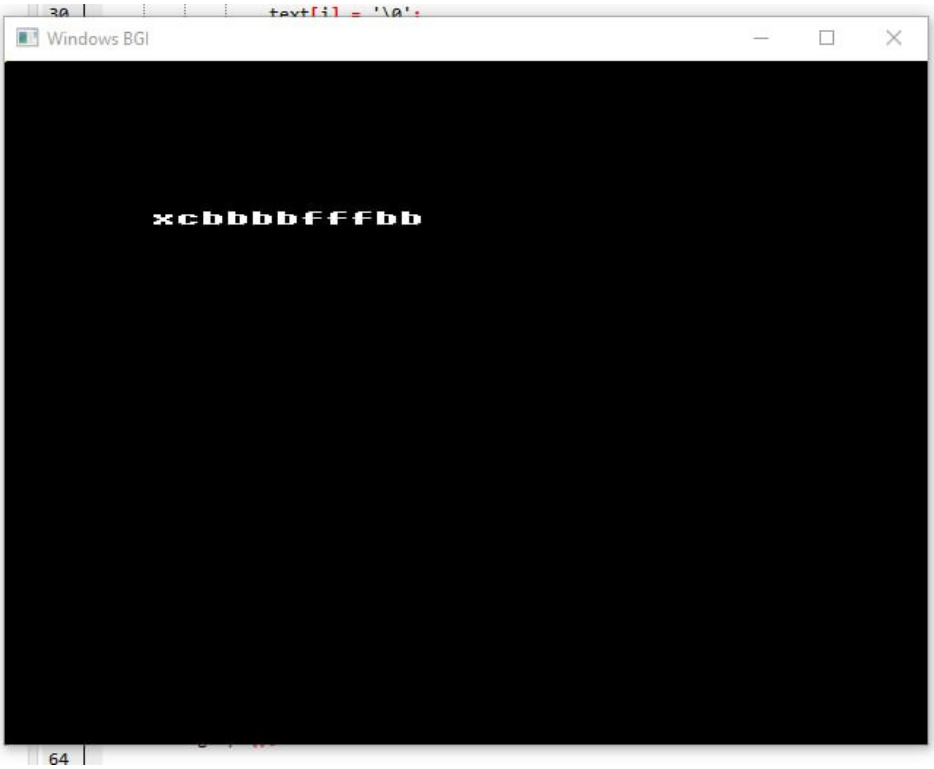


Приклад 3

```
#include <graphics.h>
#include <stdlib.h>
#include <conio.h
int main() {
    int gd = DETECT, gm;  // Змінні для графічного режиму та графічного режиму
    initgraph(&gd, &gm, "");  // Ініціалізація графічного режиму

    int font = DEFAULT_FONT;  // Встановлення шрифту
    int size = 2;             // Встановлення розміру шрифту
    settextstyle(font, HORIZ_DIR, size);  // Встановлення стилю тексту

    int x = 100, y = 100;  // Початкові координати тексту
```

```
char text[100] = "";  // Масив для зберігання введеного тексту
char ch;              // Змінна для зберігання введеного символу
int i = 0;            // Індекс для масиву тексту

// Безкінечний цикл для введення тексту
while (1) {
   ch = getch();  // Отримання введеного символу з консолі

   if (ch == 13)  // Якщо натиснуто Enter
      break;     // Вийти з циклу
   else if (ch == 27)  // Якщо натиснуто Esc
      return 0;        // Завершити програму
   else if (ch == 8) {  // Якщо натиснуто Backspace
      if (i > 0) {     // Якщо є символи для видалення
         i--;        // Зменшити індекс

         // Видалення останнього введеного символу
         text[i] = '\0';

         // Виведення тексту на екран (чорним кольором)
         setcolor(BLACK);
         outtextxy(x, y, text);

         // Додавання пробілу на місце видаленого символу (білий колір)
         text[i] = ' ';
         setcolor(WHITE);
         outtextxy(x, y, text);

         text[i] = '\0';  // Завершення рядка
      }
   } else {
      text[i] = ch;  // Додавання символу до рядка
      i++;           // Збільшення індексу
      text[i] = '\0';  // Завершення рядка

      // Виведення тексту на екран (чорним коліром)
      setcolor(BLACK);
      outtextxy(x, y, text);

      // Виведення останнього введеного символу (білий колір)
      text[i] = ch;
      setcolor(WHITE);
      outtextxy(x, y, text);
```

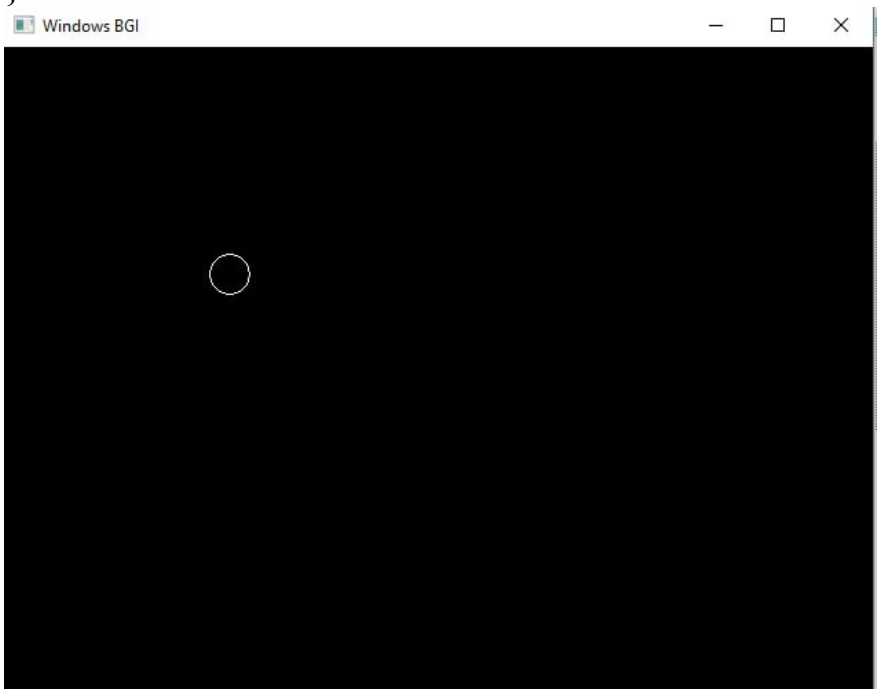
```
        }
    }

    getch();  // Очікування натискання будь-якої клавіші
    closegraph();  // Закриття графічного режиму
        return 0
}
```



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 = 0, y = 0;  // Початкові координати круга
    int dx = 1, dy = 1;  // Кроки переміщення по осях x та y

    // Безкінечний цикл, поки не буде натиснута клавіша
    while (!kbhit()) {
        cleardevice();  // Очищення вікна
        circle(x, y, 15);  // Малювання круга з центром у (x, y) та радіусом 15
        x += dx;  // Зміна координати x
        y += dy;  // Зміна координати y
        delay(20);  // Затримка для плавного анімованого ефекту
```

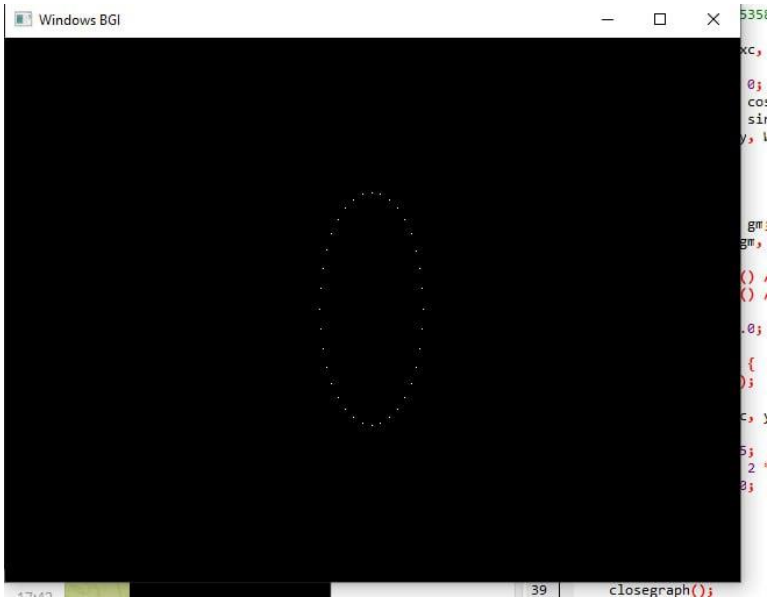
```
        // Перевірка зіткнення з межами вікна по осі x
        if (x <= 0 || x >= getmaxx()) {
            dx = -dx;  // Зміна напрямку руху по осі x
        }

        // Перевірка зіткнення з межами вікна по осі y
        if (y <= 0 || y >= getmaxy()) {
            dy = -dy;  // Зміна напрямку руху по осі y
        }
    }

    closegraph();  // Закриття графічного вікна
        return 0;
}
```



Приклад 5

```
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include <stdio.h>
#define PI 3.14159265358979323846  // Визначення значення числа Пі

void drawSphere(int xc, int yc, int r, double angle) {
    int x, y;
    // Цикл для обходу всіх кутів 0-360 градусів
    for (int theta = 0; theta < 360; theta += 10) {
        // Обчислення координат кожної точки сфери
        x = xc + r * cos(theta * PI / 180) * cos(angle);  // x координата
```

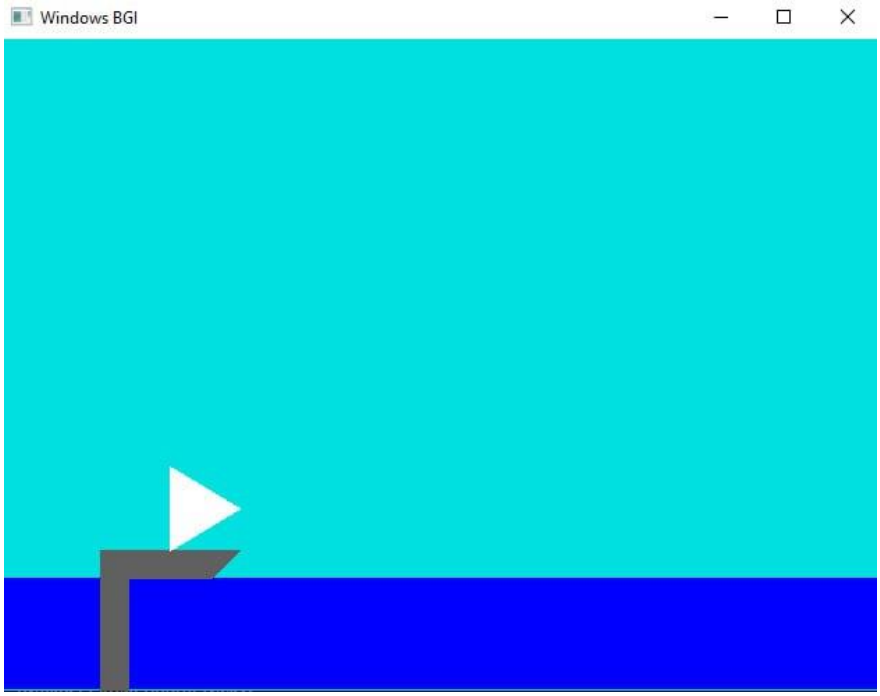
```
            y = yc + r * sin(theta * PI / 180);              // y координата
            putpixel(x, y, WHITE);  // Малювання пікселя кольору білий
        }
    }

    int main() {
        int gd = DETECT, gm;  // Змінні для графічного режиму та графічного режиму
        initgraph(&gd, &gm, "");  // Ініціалізація графічного режиму

        int xc = getmaxx() / 2;  // Координата x центру сфери
        int yc = getmaxy() / 2;  // Координата y центру сфери
        int r = 100;             // Радіус сфери
        double angle = 0.0;      // Кут повороту сфери

        // Безкінечний цикл, поки не буде натиснута клавіша
        while (!kbhit()) {
            cleardevice();  // Очищення вікна

            drawSphere(xc, yc, r, angle);    // Малювання сфери з заданими параметрами та кутом обертання

            angle += 0.05;  // Збільшення кута обертання
            if (angle >= 2 * PI) {  // Якщо кут перевищив 360 градусів
                angle = 0;  // Повернути кут обертання в початкове положення
            }

            delay(50);  // Затримка для плавного анімаційного ефекту
        }
        closegraph();  // Закриття графічного вікна
        return 0;  }
```



Приклад 6

```
#include <graphics.h>
#include <stdio.h>
#include <windows.h>
// Функція для малювання корабля
void drawBoat(int x, int y) {
    // Малювання корпусу корабля
    setcolor(DARKGRAY);  // Встановлення кольору контуру
    setfillstyle(SOLID_FILL, DARKGRAY);  // Встановлення стилю заливки
    int base[] = {x, y, x + 100, y, x + 80, y + 20, x + 20, y + 20, x};  // Масив координат корпусу
    fillpoly(5, base);  // Заповнення багатокутника

    // Малювання вітрила
    setcolor(WHITE);  // Встановлення кольору контуру
    setfillstyle(SOLID_FILL, WHITE);  // Встановлення стилю заливки
    int sail[] = {x + 50, y - 60, x + 100, y - 30, x + 50, y};  // Масив координат вітрила
    fillpoly(3, sail);  // Заповнення багатокутника
}

int main() {
    // Ініціалізація графічного режиму
    int gd = DETECT, gm;  // Змінні для графічного режиму та графічного режиму
    initgraph(&gd, &gm, NULL);  // Ініціалізація графічного режиму

    // Встановлення кольору фону
    setbkcolor(CYAN);  // Встановлення кольору фону
    cleardevice();      // Очищення екрану

    // Початкові координати корабля
    int x = 0, y = getmaxy() - 100;  // Початкові координати x та y

    // Безкінечний цикл для руху корабля
    while (x < getmaxx()) {
        cleardevice(); // Очищення екрану

        // Малювання води
        setcolor(BLUE);  // Встановлення кольору контуру
        setfillstyle(SOLID_FILL, BLUE);  // Встановлення стилю заливки
        bar(0, y + 20, getmaxx(), getmaxy());  // Малювання прямокутника
```

```
// Малювання корабля
    drawBoat(x, y);  // Виклик функції для малювання корабля

    // Збільшення координат для руху корабля
    x += 2;  // Зміна координати x

    // Затримка для плавного руху (змініть цей параметр для зміни швидкості)
    Sleep(50);  // Пауза на 50 мілісекунд

    // Відображення кадру
    delay(50);  // Затримка на 50 мілісекунд
  }

  // Завершення графічного режиму
  getch();       // Очікування натискання клавіші
  closegraph();  // Закриття графічного вікна
  return 0;  }
```



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 С:

У мові С, 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 С:

Основні функції для графічного режиму роботи в мові С 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 С:

В мові С 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
```