Національний університет
водного господарства
та природокористування

Міністерство освіти і науки України
Національний університет водного господарства та природокористування
Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

04-03-151

Методичні вказівки
до виконання лабораторних робіт №27-28
з навчальної дисципліни
"Інформатика та комп'ютерна техніка"
для студентів спеціальностей:
151 "Автоматизація та комп'ютерно-інтегровані технології",
141 "Електроенергетика, електротехніка та електромеханіка"
денної та заочної форм навчання

Національний університет
водного господарства
та природокористування

Рекомендовано науково-методичними комісіями: спеціальності 151 **"**Автоматизація та комп'ютерно-інтегровані технології**"** протокол № 4
від 10 жовтня 2016 р.;
спеціальності 141 **"**Електроенергетика, електротехніка та електромеханіка" протокол №3
від 14 вересня 2016 р.

Рівне – 2016

Національний університет
водного господарства

Методичні вказівки до виконання лабораторних робіт №27-28 з навчальної дисципліни “Інформатика та комп’ютерна техніка” для студентів спеціальностей: 151 “Автоматизація та комп’ютерно-інтегровані технології”, 141 “Електроенергетика, електротехніка та електромеханіка” денної та заочної форм навчання / А.П.Сафоник – Рівне: НУВГП, 2016. – 36 с.

Упорядник: **А.П. Сафоник**, доктор технічних наук, професор кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій НУВГП.

Національний університет
водного господарства
та природокористування

Відповідальний за випуск: В.В. Древецький, д.т.н., професор, завідувач кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Лабораторна робота №27 3
Лабораторна робота №28 29

© Сафоник А.П., 2016
© НУВГП, 2016

Національний університет
водного господарства
та природокористування

Лабораторна робота №27.
Побудова зображень в текстовому та графічному режимах

27.1. Мета роботи

Навчитися користуватися функціями текстового і графічного режиму мови С.

27.2. Теоретичні відомості

Текстовий режим. Всі функції керування екраном у текстовому режимі мають свої прототипи в заголовному файлі CONIO.H. Там же знаходяться деякі константи і макроси, використовувані цими функціями.

Більшість підпрограм зв'язано з вікнами (windows), а не з всім екраном. Вікно - це прямокутна область екрана, яку програма використовує для видачі повідомлень. Вікно може займати повний екран або бути настільки маленьким, що в нього можна вивести тільки декілька символів.

Borland C++ дозволяє встановлювати розмір і місце розташування вікон на екрані. Після того як ви задали вікно, підпрограми, що маніпулюють текстом, використовують не весь екран, а тільки задане вікно. Приміром, функція clrscr( ) очищає активне вікно, а не весь екран. Всі координати виявляються зв'язаними з активним вікном. Для текстового екрана верхній лівий кут вікна має координати (1,1). По змовчуванню активним вікном є весь екран.

Функція, що створює вікно заданого розміру і визначеного місця розташування, має прототип

```
void window (int left, int top, int right, int bottom);
```

Якщо якась координата задана невірно, то ніякої дії не відбувається. Якщо виконання функції було успішним, то всі посилання на поточні координати робляться щодо цього вікна.

При створенні вікна за допомогою функції window( ) координати left, top, right, bottom є абсолютними, тобто розглядаються щодо всього екрана. Тому знову створювані вікна можуть перекриватися один одним.

Однієї з найбільш використовуваних функцій є функція очищення активного вікна

```
void clrscr(void);
```

Активне вікно очищається, а курсор встановлюється в лівому верхньому кутку вікна.

Наступна функція - це функція позиціювання курсору. Її прототип -

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

де x и y - координати щодо активного вікна. Якщо якась координата виходить за межі області, то ніякої дії не відбувається, однак помилка

Національний університет
водного господарства
та природокористування

також не фіксується. Функції

```
int wherex( );
int wherey( );
```

повертають координати x і y поточного положення курсору.

Інша корисна функція - c1reo1().Вона очищає рядок із поточної позиції курсору до правої межі вікна. Особливо зручно з її допомогою звільняти місце для введення. Прототип має вид

```
void clreol(void);
```

Наступні дві функції знищують рядок, у якому знаходиться сміття, і вставляють порожній рядок. Функція delline( ) із прототипом

```
void delline(void);
```

знищує рядок, у якому знаходиться курсор, а всі рядки, що знаходяться під ним, піднімаються нагору. Функція

```
void insline (void);
```

вставляє порожній рядок, а той рядок, у якому знаходився курсор, і рядки, що були нижче, опускаються на одну позицію.

Функції введення/виводу, такі, як printf( ), не зв'язані з віконним інтерфейсом, тому в Borland C++ додатково включені функції введення/виводу, орієнтовані на роботу з використанням вікон. Такими функціями є:

cprintf( ) - форматований вивід в активне вікно,
cputs( ) - вивід рядка в активне вікно,
putch( ) - вивід символу в активне вікно,
getche( ) - ввід символу з активного вікна,
cgets( ) - ввід рядка з активного вікна,
cscanf( ) - форматований ввід.

Ці функції працюють майже так само, що й відповідні функції printf( ), gets( ), puts( ), за тим винятком, що зв'язані з активним вікном. Крім того ці функції не є перенаправляючими. Функція cputs( ) не переводить курсор на новий рядок, а функція cprintf( ) не розбиває символ '\n' на '\n\r'.

Ми припускаємо, що встановлений поточний відеорежим комп'ютера, котрий майже завжди наступний: текстовий режим 25 рядків x 80 символів у рядку. Існує декілька текстових режимів. Текстовим режимом керує функція

```
textmode( );
```

із прототипом

```
void textmode(int mode);
```

де аргумент mode може бути одного з наступних розмірів:

Макрос	Розмір	Режим
BW40	0	40 символів x 25 рядків, чорно- білий
C40	1	40 символів x 25 рядків, кольоровий

Національний університет
водного господарства
та природокористування

BW80	2	40 символів х 25 рядків, колірний
C80	3	40 символів х 25 рядків, кольоровий
MONO	7	80 символів х 25 рядків, монохромний
LASTMOD	-1	Попередній режим, що був до заміни
C4350	64	EGA 80 х 43 VGA 80 х 50, кольоровий

Сучасна техніка дозволяє виводити текст в кольорі. Можна задати як колір тексту, так і колір фону. Однак використовувати колір можуть тільки спеціальні, зв'язані з віконним інтерфейсом функції. Такі функції, як printf( ), для цього не призначені.

Функція textcolor() змінює колір тексту. Її прототип має вигляд

void textcolor(int color);

Аргумент color може приймати значення від 0 до 15. Кожне значення відповідає своєму кольору. У файлі CONIO.H визначені макроси, що відповідають цим кольорам. Вони приведені нижче.

Макрос	Код
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGREY	7
DARKGREY	8
L1GHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
L1GHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15
BLINK	128

Національний університет
водного господарства
та природокористування

Національний університет
водного господарства
та природокористування

Крім того, функція textcolor( ) дозволяє робити текст миготливим (blink):

Зміна кольору тексту відбувається тільки після виклику функції textcolor( ). Текст, що вже знаходиться на екрані, свого кольору не змінює.

Для того, щоб зробити текст миготливим, потрібно застосувати побітну операцію "OR" до кольору і розміру BLINK. Наприклад, щоб вивести текст червоними миготливими символами, використовуємо виклик функції

textcolor(RED||BLINK);

Функція textbackground( ) встановлює колір фону під текстом, що виводиться. Прототип цієї функції -

void textbackground(int color);

де розмір color може приймати значення від нуля до шести, тобто доступні тільки перші 7 кольорів із приведеної вище таблиці.

Функція clrscr( ) не тільки очищає активне вікно, але і заповнює кольором, заданим функцією background( ).

Приведемо приклад використання деяких з описаних вище функцій:

```
#include <conio.h>
// Приклад. Використання функцій роботи з текстовим екраном
main(void)
{ register int ctext, cback;
textmode(C80);
for (ctext=BLUE; ctext<=WHITE; ctext++);
for (cback=BLACK; cback<= LIGHTGRAY; cback++);
{ textcolor(ctext);
textbackground(cback);
cprintf("ТЕСТ ");
}
cprintf(\n);
}
textcolor (WHITE||BLINK);
textbackground(BLACK);
cprintf("КІНЕЦЬ ТЕКСТУ");
textmode(LASTMODE);
return(0);
}
```

Організація відеопам'яті, тобто ділянки пам'яті, що відображається на екрані у випадку роботи з текстовим екраном, дуже проста. Кожному символу відповідають 2 байти. В однім із них зберігається код символу, в іншому - атрибут символу. Байт атрибута має наступний вигляд:

Національний університет
водного господарства
та природокористування

7	6	5	4	3	2	1	0
м	ф	ф	ф	с	с	с	с

де СССС - 4-бітовий двійковий код кольору символу; ФФФ - 3-бітовий двійковий код кольору фону; м - ознака миготіння (1 - миготіння включене, 0 - виключено).

Оскільки, для кольору фону виділено 3 біта, то кольорів може бути тільки 8, а для кольору символу, що задається чотирма бітами, - 16 кольорів.

Більш того, як ми знаємо з фізики, будь-який колір формується змішанням червоного (R-red), зеленого (G-green) і синього (B-blue) кольорів. У байті атрибута ми можемо зазначити, який біт за який колір відповідає:

7	6	5	4	3	2	1	0
b	r	g	b	c	R	g	b

При цьому 3-й біт відповідає за яскравість кольору символу. Якщо ми будемо розглядати 4 останніх біта, то 0001 відповідає кольору BLUE (1), у той час як 1001 відповідає кольору LIGHTBLUE (9). Відповідно 0000 - чорний колір (BLACK), 1111 - білий колір (WHITE).

Функція, що задає атрибут, має вигляд

```
void textattr(int newattr);
```

Якщо ми хочемо задати атрибут, що відповідає миготливому червоному символу на чорному фоні, це можна зробити наступними способами:

```
textattr((BLACK<<4) + RED + BLINK);
```

або

```
textattr(0x84);
```

Двійковий запис шістнадцяткового числа 0x84 має вигляд 10000100. Порівняєте його з бітами атрибута.

Функція highvideo( ) встановлює підвищену яскравість символу, lowvideo( ) - стандартну яскравість символу, normvideo( ) відновлює стандартні атрибути (білі символи на чорному фоні без миготіння).

Функція

```
void setcursor type(int cursortype);
```

встановлює розмір курсору, значення параметра можуть бути наступними:

NOCURSOR	0	курсор незначний;
SOLIDCURSOR	1	курсор займає все знакомісце;
NORMALCURSOR	2	курсор має нормальний вигляд.

Змінна _wscroll керує прокручуванням вікна. Якщо привласнити _wscroll значення, рівне нулю прокручування у вікні буде заборонене, якщо ж привласнити значення, рівне одиниці, то прокручування буде дозволене.

Національний університет
водного господарства
та природокористування

У бібліотеці передбачені функції, що дозволяють зберігати, відновлювати і переміщати вміст частини екрана. Цими функціями є відповідно

int gettext(int left, int top, int right, int bottom, void * destin);

int puttext(int left, int top, int right, int bottom, void * source);

int movetext(int left, int top, int right, int bottom, int destleft, int desttop);

Область екрана задається координатами лівого верхнього і правого нижнього кутів, область пам'яті, у котру (із якої) переноситься частина відеопам’яті, задається вказівниками destin і source. У функції movetext( ) два останні аргументи вказують на координати верхнього лівого кута нового розташування області екрана.

Оскільки кожен символ у відеопам’яті займає 2 байти, то при збереженні ділянки екрана потрібно виділити буфер відповідного розміру:

```
#include <conio.h>
/* Приклад. Використання gettext( ), puttext( ) */
main(void)
{
 char buffer[288],i;
 char str[ ]="This is a string ! ";
 textbackground(BLACK);
 clrscr( );
 window(5,5,20,10);
 textattr((GREEN<<4)+RED);
 clrscr( );
 cputs("\n Hello, World !\n\n\r");
 textattr((GREEN<<4)+RED+BLINK);
 cputs("Press any key... ");
 getch( );
 window(1, 1,80,25);
 gettext(4, 4, 21, 11, buffer);
 textbackground(BLUE);
 textcolor(WHITE);
 for(i=1;i<23;i++)
  { gotoxy(1, i);
   cputs(str); cputs(str); cputs(str); cputs(str);
  }
 getch( );
 puttext(24, 5, 41,12, buffer);
 getch( );
 for(i=1;i<5;i++)
   movetext(22,4,32,8,10*i, 17);
```

Національний університет
водного господарства
та природокористування

Національний університет
водного господарства
та природокористування

```
 getch( );
 normvideo( );
 getc( );
 return 0;
}
```

Змінна directvideo керує виводом на консоль: безпосередньо в оперативну пам'ять дисплея (directvideo=l), або через виклик BIOS (directvideo=0). По змовчуванню значення directvideo=l. Вивід через BIOS працює повільніше, але можна бути впевненим, що програма буде працювати на будь-якому IBM-сумісному комп'ютері.

Графічний режим. Розглянемо, як відбувається робота в графічному режимі.

У сучасних ПК, в основному, використовуються растрові дисплеї, у яких найменшим елементом зображення є маленька світна пляма - pixel (від англ. picture element); з таких окремих крапок складається зображення.

Роздільна здатність дисплея - це кількість пікселів по горизонталі і вертикалі (стандартний дозвіл 640*480 крапок). Зображення, що виводиться на екран дисплея, у закодованому виді зберігається в спеціальній області пам'яті центрального процесора - так званій відеопам'яті. Дані з цієї області періодично зчитуються, перетворюються у відеосигнали і відображаються на екрані. Перетворення кодів зображення у відеосигнали здійснює спеціальна електронна схема - відеоадаптер. Найбільш розповсюджені типи адаптерів: VGA, SVGA.

Як кодується зображення у відеопам'яті? Кожному пікселю екрана ставиться у відповідність фіксована кількість бітів - атрибут пікселя. Звичайно атрибут пікселя складається з 1, 2, 4, 8 біт (у залежності від використовуваного графічного режиму). Якщо атрибуту приділяється 1 біт, то графіка буде двоколірною, наприклад чорно-білою (конкретні кольори залежать від типу монітора). Якщо кожен піксель представляється N бітами, то є можливість представлення відтінків. Звичайно, в кольорових моніторах ПК IBM використовується поділ кольору на RGB-компоненти (red-green-blue). Крім того, є можливість збільшення яскравості трьох компонентів, установивши режим I - підвищеної інтенсивності. Таким чином, колір кодується чотирма бітами: IRGB, тому базова палітра складає 2^4=16 кольорів.

У Turbo C і C++ є графічна бібліотека, що містить велику кількість функцій, що дозволяють здійснити доступ до окремих елементів відеопам'яті, керувати кольором, створювати зображення різної форми, виводити графічні та текстові повідомлення, керувати курсором. Настроювання цих функцій на роботу з конкретним відеоадаптером

досягається за рахунок підключення потрібного графічного драйвера. Драйвер - це спеціальна програма для керування тим чи іншим пристроєм ПК.

Графічні драйвери практично для усіх видів адаптерів розроблені фірмою Borland International. Вони знаходяться в окремих файлах з розширеннями BGI (Borland Graphics Interface). Для підключення графічного драйвера служить спеціальна функція initgraph( ).

Багато графічних функцій використовують поняття вказівника поточної позиції (це графічний аналог курсору в текстовому режимі, але він є невидимий).

Вказівник ідентифікує обраний піксель і характеризується парою цілих чисел: номером у рядку і номером рядка на екрані (тобто горизонтальною і вертикальною координатами). Нумерація ведеться зліва направо і зверху вниз (з нуля).

Наприклад, для режиму EGA з розширенням 640*480 координати кутових точок екрана і його центра будуть такими (рис.27.1).

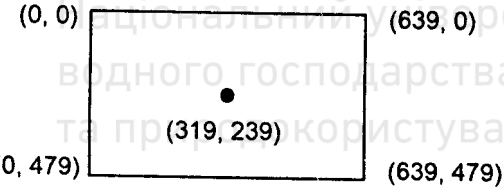


Рис.27.1. Координати характерних точок екрана.

Ініціалізація графічної системи. Обробка помилок. Переключення режимів. Вибір драйвера і графічного режиму виконує функція

detectgraph (&gd, &gm);

Проводить тестування апаратури відеоадаптера, автоматичний вибір придатного драйвера і графічного режиму. Цілочисельні змінні, вказівники на які передаються функції в якості аргументів, повертають номер драйвера і номер режиму.

Завантаження драйвера, ініціалізацію графічної системи виконує функція

initgraph (&gd, &gm, "шлях до BGI-файлів");

Значеннями змінних gd і gm повинні бути номер потрібного драйвера і номер бажаного графічного режиму.

Якщо файл із зазначеним драйвером знайдений, то йому динамічно виділяється необхідна пам'ять і виконується його завантаження.

Національний університет
водного господарства
та природокористування

Якщо пам'яті недостатньо, функція завершиться аварійно з кодом "-5". Після нормального розміщення драйвера в пам'яті, функція initgraph( ) організує внутрішній графічний буфер, динамічно запитуючи блок ОП розміром 4К (за замовчуванням). Потім відбувається ініціалізація графічних змінних (колір символів, колір фону, поточного вікна та ін.) і адаптер дисплея переводиться в графічний режим. При цьому екран очищується і вказівник поточної позиції встановлюється в лівому верхньому куті.

Якщо BGI-файли знаходяться в поточній директорії, то в якості третього параметра функції initgraph ( ) можна задати порожній рядок

initgraph (&gd, &gm, " ");

Обробка помилкових ситуацій. У Turbo C і C++ передбачені засоби обробки помилок, що виникають при роботі з графікою. Функція

graphresult ( );

повертає код завершення останньої використовуваної графічної функції. У випадку успіху повертається "0".

Для різних типів помилок передбачені свої коди завершення.

Константа	Код помилки	Повідомлення
grOK	0	Нема помилки
grNoInitGraph	-1	графіка не ініційована
…	…	…
grNoLoadMem	-5	Недостатньо пам'яті для
…	…	…
grFontNotFound	-8	Файл шрифту не знайдений
…	…	…

Є можливість роздрукувати повідомлення, що відповідає значенню коду помилки. Для цього використовується функція

grapherrormsg (код помилки);

Вона повертає вказівник на рядок, що містить опис коду завершення, переданого в якості аргументу.

Звичайна початкова послідовність дій при роботі з графікою виглядає таким чином:

```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
void main ( )
{
int gd, gm, error;
detectgraph (&gd, &gm);
initgraph (&gd, &gm, " ");
error = graphresult ( );
```

```
if (error != grOk )
 {
  puts (" помилка графіки");
  puts (grapherrormsg (error)) ;
  exit  (1) ;
 }

      // Тіло програми

closegraph ( );
exit (0) ;
}
```

Переключення режимів. Для тимчасового переходу в текстовий режим дисплейного адаптера в графічній бібліотеці передбачена функція

restorecrtmode ();

Вона відновлює той текстовий режим, що був перед звертанням до функції initgraph( ) текстовий вміст буфера (а, значить, і екрана) не відновлюється, оскільки його знищує функція initgraph(). Повернутися в графічний режим можна за допомогою функції

setgraphmode (gm);

Аргументом її є цілочисельний номер режиму, допустимий для даного драйвера. Можна відновити попередній графічний режим, якщо його номер був завчасно визначений за допомогою функції

getgraphmode ( );

і збережений у програмі.

Одержати максимальне значення номера графічного режиму, допустиме для поточного драйвера, можна за допомогою функції

getmaxmode ();

По закінченні роботи з графічною системою необхідно звільнити пам'ять, виділену динамічно під потреби графіки, очистити буфер відеоадаптера, відновити попередній текстовий режим. Усі ці дії виконує функція

closegraph ();

Установка кольорів, шрифтів, стилів ліній і стилів зафарбування. Для задання кольору у програмі можна скористатися константами перераховуваного типу або відповідними їм цілими значеннями.

Стандартна колірна палітра така

BLACK	0	чорний
BLUE	1	синій

Національний університет водного господарства та природокористування

GREEN	2	зелений
GRAY	3	сірий
RED	4	червоний
MAGENTA	5	малиновий
BROWN	6	коричневий
LIGHTGRAY	7	ясно-сірий
DARKGRAY	8	темно-сірий
LIGHTBLUE	9	ясно-синій
LIGHTGREEN	10	ясно-зелений
LIGHTCYAN	11	світло-бірюзовий
LIGHTRED	12	ясно-червоний
LIGHTMAGENTA	13	світло-малиновий
YELLOW	14	жовтий
WHITE	15	білий

Кольором фону можна керувати за допомогою функції

setbkcolor(колір);

Поточне значення кольору, що малює, (кольору виведених символів і ліній) установлюється функцією

setcolor (колір);

У графічному режимі при виводі текстового повідомлення є можливість вибору одного з декількох шрифтів, розміру виведених символів і напрямку тексту. Ці параметри задаються за допомогою функції

settextstyle (шрифт, напрямок, розмір);

Припустимі значення для параметра шрифт

0 DEFAULT_FONT (стандартний)
1 TRIPLEX_FONT (типу триплекс; у файлі TRIP.CHR)
2 SMALL_FONT (зменшений; у файлі LITT.CHR)
3 SANS_SERIF_FONT (прямий; у файлі SANS.CHR)
4 GOTHIC_FONT (готичний; у файлі GOTH.CHR)

Допустимі значення для параметра напрямок

HORIZ_DIR	зліва направо
VERT_DIR	знизу вгору

Аргумент, керуючий розміром шрифту, може змінюватися від 1 до 10. Для стандартного шрифту ця величина показує, у скільки разів треба збільшити кожен символ (цей шрифт визначений на матриці 8*8. Якщо аргумент розмір дорівнює 4, то символи будуть збільшені до матриці 32*32 пікселя). Для інших шрифтів цей параметр задасть не лінійну, а експонентну шкалу масштабування. Базовий варіант символу відповідає

розміру, рівному 4. Тому, якщо розмір дорівнює 7, то символи збільшаться в 2 рази; якщо 8 - то в 3 рази; якщо 9 – то в 4 рази.

Використання вертикального розташування рядка робить символи нижчими і ширшими, ніж у горизонтальному рядку. Це зв'язано з ефектом неквадратності пікселів на деяких типах дисплеїв.

Символи тексту завжди виводяться суцільними тонкими лініями.

Для установки характеру і товщини ліній геометричних об'єктів використовується функція

setlinestyle (вид, зразок, товщина);

Допустимі значення для параметра товщина

1 NORM_WIDTH (лінія в один піксель)

3 THICK_WIDTH (лінія в три пікселі)

Коди для параметра вид (тільки для кусочно-лінійних графічних примітивів)

0 SOLID_LINE (суцільна)

1 DOTTED_LINE (із крапок)

2 CENTER_LINE (із крапок і тире)

3 DASHED_LINE (пунктирна)

3 USERBIT_LINE (обумовлена користувачем)

Параметр зразок задається тільки, коли вид дорівнює 4 (в інших випадках він ігнорується, тому його можна робити рівним 0). З його допомогою можна задати будь-який періодично повторюваний малюнок лінії з періодом до 16 пікселів. Якщо в лінії потрібний піксель, що світиться, у шаблоні задається біт, рівний 1; якщо немає -0.

Наприклад, шаблон для пунктирної лінії може бути таким: 0x3333, що відповідає послідовності бітів 0011001100110011.

У графічному режимі є можливість зафарбувати виділену на екрані замкнуту область визначеним способом.

Для установки стилю зафарбування використовується функція

setfillstyle (тип зафарбування, колір);

Допустимі значення параметра тип зафарбування

0	EMPTY_FILL	0. штрихування кольором фону заповнення
1	SOLID_FILL	1. зазначеним кольором штриховка
2	LINE_F1LL	2. горизонтальними лініями
3	LTSLASH_FILL	3. штрихування похилими лініями /////
4	SLASH_ FILL	4. штрихування потовщеними лініями ////

5	BKSLASH_FILL	5. штрихування потовщеними лініями \\\\\\
6	LTBKSLASH_FILL	6. штрихування похилими лініями \\\\\
7	HATCH_FILL	7. прямокутне горизонтальне штрихування
8	XHATCH_FILL	8. косе штрихування
9	INTERLEAVE_FILL	9. косе штрихування, що перекривається
10	WIDE DOT_FILL	10. заповнення рідко розташованими точками
11	CLOSE_DOT_FILL	11. заповнення частими точками

Графічні примітиви. Основне призначення графічних примітивів - забезпечити програміста зручним набором програмних засобів для малювання різних геометричних об'єктів. Розглянемо спочатку функції, призначені для малювання об'єктів контурного типу.

Спосіб взаємодії виведених прямих ліній установлює функція

setwritemode (режим);

Можливі коди параметра режим

0 COPY_PUT

1 XOR_PUT

Будь-яке інше значення автоматично береться по модулю 2. За допомогою цієї функції вказується спосіб, яким код кольору, що малює, установлений за допомогою функції setcolor( ), буде взаємодіяти з атрибутами пікселів, що вже знаходяться на місці об'єкта, що малюється.

Якщо встановлений режим 0, то лінія, що вичерчується, "затирає" те, що було на екрані. Коли ж заданий режим 1, то для комбінування лінії з існуючим на екрані зображенням використовується логічна операція ВИКЛЮЧАЮЧЕ АБО. Чудова властивість цієї операції в тому, що вивід двічі на одне і теж місце лінії приводить до її стирання і відновлення вихідного зображення на екрані (це використовується при програмуванні рухомих об'єктів).

Функція setwritemode( ) працює без "побічних эффектів" тільки з кусково-лінійними зображеннями, у випадку кривих 2-го порядку бувають дивні наслідки впливу даної функції.

Креслення лінії можна здійснити з допомогоюу кожної з трьох функцій:

1) line(x1,y1,x2,y2);

де x1, y1 - координати початку відрізка прямої;

x2, y2 - координати кінця відрізка прямої.

При цьому положення покажчика поточної позиції не змінюється.

2) linerel (dx, dy);

де dx, dy - приріст координат поточної точки. Лінія малюється з поточної крапки в крапку з новими координатами. При цьому покажчик зміщається

Національний університет
водного господарства
та природокористування

зі старої точку в нову.

3) lineto (x, y);

де x, y - нові координати кінця відрізка. Лінія малюється з поточної точки в точку з новими координатами. При цьому покажчик поточної позиції зміщається з старої точки в нову.

Лінії малюються поточним стилем, установленим функцією setlinestyle( ), і поточним кольором, установленим функцією setcolor( ). Використовується система координат біжучого вікна і встановлений режим відсікання.

Приклад.

Два способи креслення ромба.

```
спосіб 1
setwritemode (XOR_PUT); setcolor (GREEN);
moveto (100, 10); lineto (50, 90); lineto (100, 170);
setcolor (RED); lineto (150, 90); lineto (100, 10);

спосіб 2
setcolor (WHITE); setcolor (GREEN);
line (100, 10, 50. 90); line (50, 90, 100, 170);
setcolor (RED);
line (100,170, 150, 90); line (l50, 90, 100, 10);
```

Національний університет
водного господарства
та природокористування

Контур прямокутника можна накреслити за допомогою функції.

rectangle (x1, y1, x2, y2);

де x1, y1 - координати лівого верхнього кута;
x2, y2 - координати правого нижнього кута.

Якщо спробувати намалювати квадрат, то на багатьох моніторах це не вийде, тому що піксель має форму прямокутника, витягнутого по вертикалі. Необхідно зробити коректування кількості пікселів по горизонтальній і вертикальній сторонам квадрата. Істинні пропорції пікселів, необхідні для такого коректування, можна взнати за допомогою функцій

getaspectratio (&xasp, &yasp);

Відношення xasp/yasp і є відношення горизонтального і вертикального розмірів пікселя. Тому, якщо горизонтальний рядок квадрата має розмір G пікселів, то довжина вертикального рядка повинна бути рівна

(int) (G * (float) xasp/yasp);

Накреслити ламану лінію дозволяє функція

Національний університет
водного господарства
та природокористування

drawpoly (кількість вершин, вказівник на масив цілих);

Кожна пара чисел масиву інтерпретується як пара координат чергової вершини ламаної.

Для побудови кривих використовуються функції:

Креслення окружності

circle (x, y, радіус);

де x, y - координати центра;

радіус - радіус окружності в пікселях по горизонталі. "Ступінь неквадратності" пікселів автоматично враховується.

Креслення дуги окружності

arc (x, y, поч_кут, кін_кут, радіус);

Кути виражаються в градусах і відраховуються проти годинникової стрілки. Нульовий кут відповідає горизонтальному напрямку вектора зліва направо. Значення кутів перетворюються до еквівалентних значень з інтервалу [0..360]. Таким чином, arc (x, y, -45, 45, r) і arc (x, y, 675, -315, r) задають ту саму дугу в чверть окружності. Креслення дуги еліпса

ellipse (x, y, поч_кут, кін_кут, rx, ry);

де rx, ry - довжини півосей еліпса в пікселях. Осі еліпса завжди паралельні осям координат.

Перейдемо тепер до розгляду двовимірних графічних примітивів, що будують деякі геометричні фігури і зафарбовують їхні внутрішні області.

Зафарбування прямокутника виконує функція

bar (x1, y1, x2, y2);

де x1, y1 - координати лівого верхнього кута;

x2, y2 - координати правого нижнього кута. Сторони прямокутника паралельні осям координат; контур його не вимальовується. Стиль зафарбування встановлює функцією setfillstyle( ).

Паралелепіпед із зафарбованою передньою гранню можна намалювати, використовуючи функцію

bar3d (x1, y1, x2, y2, глибина, дах);

Паралелепіпед обрамляється зовнішнім контуром. Якщо параметр глибина буде рівним 0, то вийде обрамлений прямокутник. Якщо параметр дах дорівнює 1 (TOP_ON), то малюється верхня грань; якщо він дорівнює 0 (TOP_OFF), то верхня грань не викреслюється (це корисно, якщо треба нарисувати один на одному кілька паралелепіпедів).

Приклад.

Паралелепіпеди, що стоять один на одному

```
setkcolor (WHITE);
```

Національний університет
водного господарства
та природокористування

```
setcolor (GREEN);
bar3d (230, 50,250, 150, 15, 1);
bar3d (220, 150, 260, 180, 15, 1);
bar3d (300, 150, 340, 180, 15, 0);
bar3d (300, 50, 340, 150, 15, 1);
```

Побудувати і зафарбувати багатокутник дозволяє функція

fillpoly (n, покажчик на масив цілих);

де n - кількість вершин.

Координати кожної вершини задаються двома величинами цілого типу. Ця функція завжди з'єднує першу точку списку вершин з останньою, замикає контур і зафарбовує його. Шаблон і колір зафарбовування можуть бути задані функціями setfillstyle( ) і setfillpattern( ).

Зафарбований еліпс із контуром можна одержати, застосувавши функцію

fillellipse (x, y, rx, ry);

де x, y координати центра,

rx, ry - довжини півосей еліпса в пікселях.

Осі еліпса паралельні осям координат. Зафарбовування здійснюється поточним кольором.

Зафарбований круговий сектор з контуром малює функція

pieslice (x, y, поч_кут, кін_кут, радіус);

Після приведення кутів до діапазону ([0..360] сектор малюється від меншого значення кута до більшого, тому неможливо зобразити сектор, що перетинає позитивний напрямок осі OX. Контур (дуга і два радіуси) малюється після зафарбування сектора, причому тип і товщина ліній беруться з установок функції setlinestile( ). Якщо потрібний сектор без контуру, це можна зробити так:

```
setcolor (BLACK);  setbkcolor (BLUE) ;
setwritemode (XOR_PUT); setflllstile (WIDE_DOT_FILL,RED);
pieslice (200, 100, 45, 90, 50);
```

Приклад.

Кругові сектори з написами в них

```
setbkcolor (BLUE); setcolor (RED) ;
setfillstyle (1, 3);
x=getmaxx ( )/2; y=getmaxy ( )/2;
pieslice (x, y, 270, 360, 100);
setfillstyle (1, 2);
```

Національний університет
водного господарства
та природокористування

```
pieslice (x, y, 0, 270, 100);
settextstyle (1, 0, 2);
moveto (x-20, y-40); outtext ("75%");
moveto (x+20, y+20); outtext ("25%");
```

Зафарбований еліптичний сектор з контуром одержимо, застосувавши функцію

sector (x, y, поч_кут, кін_кут, rx, ry);

Функція діє аналогічно функцій pieslice ( ).

Зафарбування довільної замкнутої області виконується функцією

floodfill (x, y, межа);

де x, y – координати точки всередині області.

Контур, що обмежує область, повинен бути замкнутим. Колір контура повинен співпадати з параметром межа. Зафарбовування (колір і тип) встановлюююбся функцією setfillstyle ( ).

Приклад.

Сім концентричних червоно-синіх кілець.

Національний університет
водного господарства
та природокористування

```
setlinestyle(0, 0, 1);
setbkcolor (GREEN);
for(i=7; i>=1; i--)
 if ( i%2 )
     {
      setcolor(BLUE);
      circle(x, y, 5*i);
      setfillstyle(SOLID_FILL, BLUE);
      floodfill(x, y, BLUE);
     }
  else
    {
      setcolor(RED);
      circle(x, y, 5*i);
      setfillstyle(SOLID_FILL, RED);
      floodfill(x, y, RED);
    }
```

Поради. Приступаючи до написання програм цього розділу, варто враховувати, що:

- у графічному режимі екран являє собою сукупність точок, кожна з який може бути пофарбована в один; з 16-ти кольорів;

Національний університет
водного господарства
та природокористування

- координаты точок зростають зліва направо і зверху вниз. Ліва верхня точка має координати (0,0), права нижня (639,479);
- для того щоб програма могла виводити на екран графічні примітиви (лінії, окружності, прямокутники), необхідно ініціалізувати графічний режим.

Шаблон графічної програми виглядає таким чином:

```
// шаблон графічної програми
#include <graphics.h>
#include <conio.h>
void main(void)
{
 int gdriver = DETECT; // драйвер.
 int gmode;                  // режим
 int errorcode;              // код помилки
 initgraph(&gdriver, &gmode, C:\\BCPP\\BGI\\);
 errorcode = graphresult();
 if (errorcode != grOk)     //помилка ініціалізації
                            // графічного режиму
 {
  printf("Помилка: %d\n", errorcode);
  puts("Для завершення програми натисніть <Enter>") ;
  getch();
  return;
 }
    // далі інструкції програми
  closegraph;             // завершення графічного режиму
}
```

Національний університет
водного господарства
та природокористування

27.3. Опис лабораторного обладнання

1. Персональний комп'ютер.
2. Програмне забезпечення: Borland C++ 3.1.

27.4. Програма роботи

1.Запустити середовище Borland C++.

2.Скласти алгоритми програм до завдань, вказаних викладачем (додатки 1, 2, 3), згідно свого варіанту.

3.Написати програми до завдань, вказаних викладачем (додатки 1, 2, 3, 4) згідно свого варіанту.

Вимоги до програм

Національний університет
водного господарства
та природокористування

- початкові дані ввести оператором введення;
- на екран вивести графічне зображення згідно свого варіанту.

Вимоги до звіту

Звіт повинен включати в себе:

- Титульний лист із зазначенням номеру варіанту
- Мету роботи
- Програму роботи
- Завдання
- Тексти програм
- Результати виконання програм, скопійовані з монітора комп‘ютера

27.5. Контрольні запитання

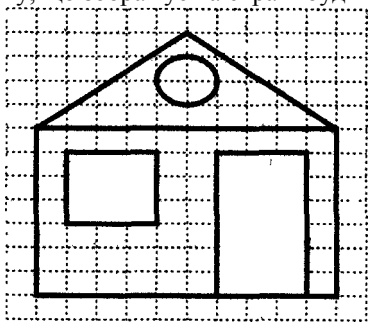
1. Назвати спільні риси та відмінності текстового і графічного відеорежимів.
2. Назвати основні функції заголовного файла graphics.h.
3. Назвати загальні функції для керування графічним режимом.
4. Назвати функції для керування кольором зображення і фону.
5. Назвати функції для роботи з окремими пікселями.
6. Назвати функції для побудови відрізків ліній .
7. Назвати функції для побудови прямокутників і многокутників.
8. Назвати функції для побудови кіл та їх елементів.
9. Назвати функції для побудови еліпсів та їх елементів.
10. Назвати функції для виводу тексту в графічному режимі.
11. Представити фрагмент програми для ініціалізації графічного режиму.

Національний університет
водного господарства
та природокористування

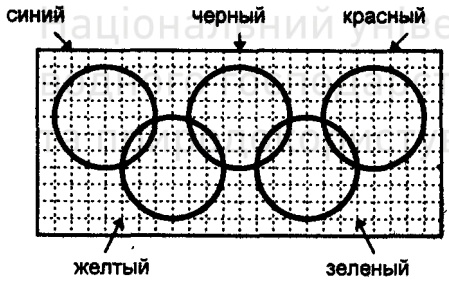
Додаток 1

Варіанти:

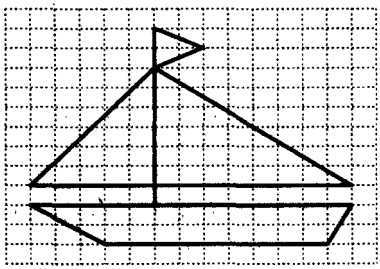
1. Написати програму, що зображує на екрані будиночок.



2. Написати програму, що виводить на екран прапор Олімпійських ігор. Зображення прапора приведене нижче (одній клітці відповідає п'ять пікселів).

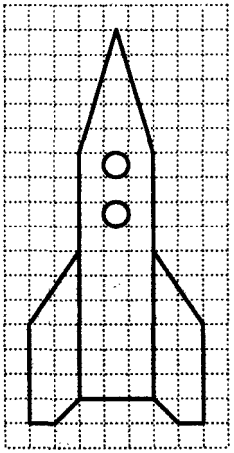


3. Написати програму, що креслить на екрані кораблик.

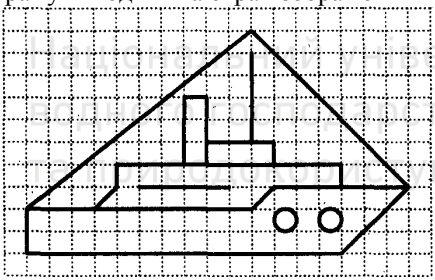


Національний університет
водного господарства
та природокористування

4. Написати програму, що зображає на екрані ракету.



5. Написати програму виводить на екран зображення кораблика.

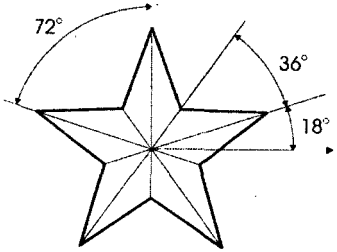


6. Написати програму, що будує на екрані візерунок з 100 окружностей випадкового діаметру і кольору.

7. Написати програму, що будує на екрані візерунок з 50 прямокутників випадкового розміру і кольору.

8. Написати програму, що будує на екрані візерунок — ламану лінію з 100 ланок з випадковими координатами, випадкового кольору.

9. Написати програму, що виводить на екран контур п'ятикутної зірки.



10. Написати програму, що виводить на екран п'ятикутну зірку червоного кольору з білою окантовкою.

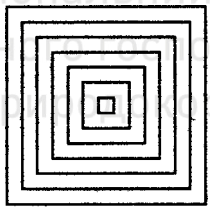
11. Написати програму, що будує на екрані шестикутник.

12. Написати програму, що малює на екрані Державний прапор України.

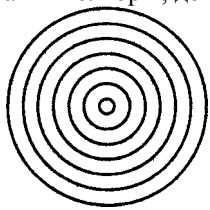
13. Написати програму, що малює на екрані зображення веселого чоловічка жовтого кольору.

14. Написати програму, що малює на екрані зображення невеселого чоловічка .

15. Написати програму, що виводить на екран зображений нижче візерунок.

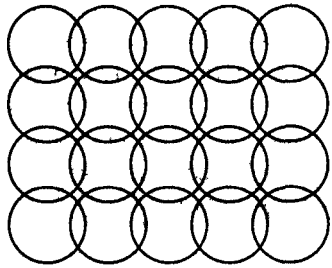


16. Написати програму, що виводить на екран зображений нижче візерунок. Окружності повинні бути різного кольору: від синього до білого (дивись таблицю кодування кольорів, довідник функція setcolor()).

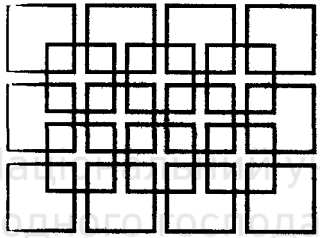


Національний університет
водного господарства
та природокористування

17. Написати програму, що виводить на екран зображений нижче візерунок.

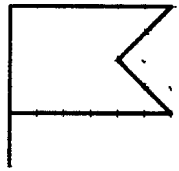


18. Написати програму, що виводить на екран зображений нижче візерунок

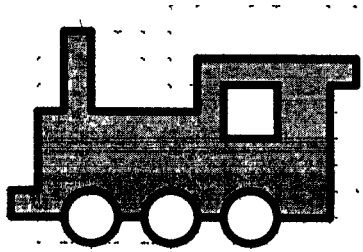


19. Написати програму, що виводить на екран зображення шахівниці.

20. Написати програму, що малює на екрані прапорець червоного кольору.

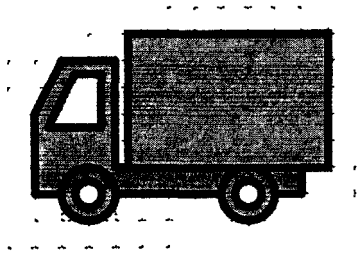


21. Написати програму, що малює на екрані паровоз. Використовуйте метод базової крапки.



22. Написати програму, що малює на екрані автомобіль. Інструкції, що забезпечують креслення колеса автомобіля, оформіть як функцію.

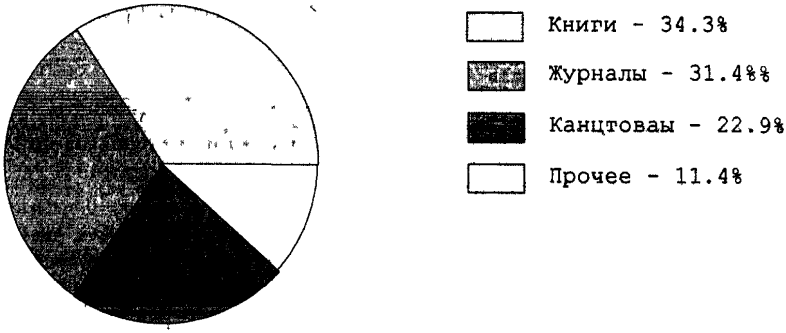
Національний університет
водного господарства
та природокористування



23. Написати програму, що виводить на екран пронумеровану координатну сітку.

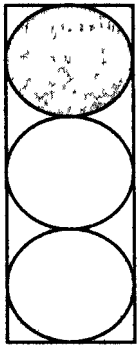
25. Написати функцію, що малює на екрані кораблик. Як параметри функція повинна одержувати координати базової крапки і колір, яким варто малювати.

26. Написати програму, що виводить кругову діаграму, що відображає товарообіг (у відсотках) книгарні. Вихідні дані (обсяг продажів у гривнях по категоріях: книги, журнали, листівки і канцтовари) вводяться під час роботи програми. Приклад діаграми приведений нижче.



27. Написати програму, що виводить на екран зображення світлофора.

Національний університет
водного господарства
та природокористування



Додаток 2

Скласти програму згідно варіанту для побудови геометричного зображення просторової фігури.

Варіанти:

1. трикутної зрізаної піраміди;
2. правильної шестикутної піраміди;
3. обеліску;
4. октаедра;
5. келиха;
6. глобуса;
7. пружини;
8. Сатурна з кільцями;
9. Сонячної системи;
10. болта з гайкою;
11. прямої п'ятикутної призми;
12. правильної шестикутної призми;
13. еліпсоїда;
14. гіперболічного параболоїда;
15. однопорожнинного гіперболоїда;
16. канцелярської кнопки;
17. зрізаного конуса;
18. монети;
19. сходів;
20. балки;
21. хреста;
22. тора;
23. кубаРубіка;
24. мушлі;
25. капелюха;
26. дзвону;
27. купола;
28. гантелі;
29. штанги;
30. швелера;
27. двотавра;
28. ока;
33. діаманта;
34. піщаного годинника;
35. ковша;
36. стінного блоку;
37. молота;
38. шахову фігуру- короля;
39. шахову фігуру- туру;
40. шахову фігуру- ферзя.

Додаток 3

Національний університет
водного господарства
та природокористування

Скласти програму згідно варіанту для побудови простого зображення.

Варіанти:

1. веселки (райдуги);
2. атома водню;
3. хід променів в лінзі;
4. лещат;
5. циферблату;
6. кілець Ньютона;
7. телефонного апарату;
8. акустичної стереосистеми;
9. штангенциркуля;
10. калькулятора;
11. дискети;
12. клавіатури піаніно;
13. шахівниці;
14. блискавки;
15. аудіокасети;
16. космічної станції;
17. герба України;
18. мікрометра;
19. олімпійського прапора;
20. мікроскопа;
21. герба Рівного;
22. фотоапарату;
23. готувальні;
24. кульмана;
25. плотера;
26. відеокамери;
27. плану студмістечка НУВГП;
28. медичної емблеми;
29. принтера;
30. комп'ютера;
27. саксофона;
32 яхти;
33. гірського пейзажу;
34. замка з ключем;
35. лабіринту;
36. мішені;
37. будівельного крана;
38. тюльпана;
39. пальми;
40. місячного кратера.

Національний університет
водного господарства
та природокористування

Національний університет
водного господарства
та природокористування

Лабораторна робота №28.
Побудова графіків функцій

28.1. Мета роботи

Навчитися працювати з графічними вікнами, вивчити алгоритми побудови графіків функцій.

28.2. Теоретичні відомості

Робота з вікнами і координатами

Очищення екрана виконує функція

cleardevice ( );

Усі встановлені раніше графічними процедурами параметри скидаються і набувають значення за замовчуванням.

Максимальні значення координат точок по горизонталі і вертикалі, допустимі в даному графічному режимі, можна одержати, використовуючи функції

getmaxx ( );

(повертає максимальну координату по горизонталі),

getmaxy ( );

(повертає максимальну координату по вертикалі).

Відкриття вікна на графічному екрані. Усередині основного масиву точок екрана завжди можна виділити деякий підмасив - прямокутне вікно з власною системою координат. Це робить функція

setviewport (x1, y1, x2, y2, clip);

де x1, y1 - координати лівого верхнього кута вікна;

x2, y2 - координати правого нижнього кута вікна;

clip - відсікання.

Якщо параметр clip дорівнює 1, то ті елементи зображення, що не вміщаються у вікні, будуть відсічені; якщо ж він дорівнює 0, то границі вікна проігноруються. При успішному виконанні цієї функції вказівник поточної графічної позиції переміститься в початок координат вікна.

Очищення графічного вікна виконує функція

clearviewport ( );

Поточні координати вказівника позиції в системі координат вікна можна одержати за допомогою функцій

getx( );

(повертає горизонтальну координату),

gety( );

(повертає вертикальну координату).

Зміну вказівника позиції виконують функції

moveto (x, y);

Національний університет
водного господарства
та природокористування

moverel (dx, dy);

де x, y - нові координати в системі координат вікна;

dx, dy - приріст відносно старих координат у вікні.

Для відновлення параметрів, прийнятих по замовчуванню, служить функція

graphdefaults ( );

Відновлюються вікно, вказівник позиції, колір, стилі ліній, шрифти - такими, якими їх робить initgraph( ).

Запис пікселя у відеопам'ять здійснює функція

putpixel (x, y, колір);

де x, y - координати пікселя в системі координат вікна. Наприклад, у результаті виконання фрагмента

```
for(i=0; i<160; i++)
{
  putpixel (i, 10, GREEN);
  putpixel (160+i, 10, RED);
}
```

буде виведена зелено-червона пряма.

Національний університет
водного господарства
та природокористування

Вивід тексту у вікно

Стандартні функції виводу printf( ), puts( ) успішно працюють у графічному режимі. Однак, вони обмежені видом і розміром символів шрифту, а також можливістю розміщення символів тільки в тих позиціях екрана, що допускаються в текстовому режимі. Спеціальні ж, графічні функції виводу тексту дозволяють працювати з ними, як з повноправним елементом графіки. Стиль виведеного тексту задасться за допомогою функції settextstyle(int шрифт, int напрям, int розмір). Є всього дві функції виводу графічного тексту у вікно:

outtext (sp);

виводиться рядок, починаючи з поточної графічної позиції, sp - це вказівник на рядок; що виводиться;

outtextxy (x, y, sp);

рядок виводиться, починаючи з позиції (x, y).

Приклад. Демонстрація роботи в графічному вікні

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

Національний університет
водного господарства
та природокористування

```
{
 int gdriver=DETECT,gmode,errorcode;
          //автоматичний вибір   режиму найбільшого розширення
 initgraph(&gdriver,&gmode,"c:\\bcpp\\bgi");
                    //ініціалізація відеоадаптера
 errorcode=graphresult( );
 if(errorcode !=grOk)
  {
   printf("Помилка графічного режиму типу: %s \n" ,grapherrormsg
(errorcode));
  printf("Натисніть будь-яку клавішу для виходу");
  getch( );
  exit(1);
  }
 //Суцільна діагональ червоного кольору товщиною в 3 пікселя
 //на зеленому екрані
 setbkcolor (GREEN);
 setcolor (RED);
 setlinestyle (SOLID_LINE, 0, 3);
 line (0, 0, getmaxx ( ), getmaxy ( ) );
 // Після натискання клавіші очищаємо екран
 getch ( ); cleardevice ( );
 // Задаємо вікно виводу і очищаємо його.
 setviewport (100, 100, getmaxx( )-50, getmaxy ( )-50, 1);
 clearviewport ( );
 rectangle (100, 100, getmaxx( )-50, getmaxy( )-50);
 settextstyle (TRIPLEX_FONT, HORIZ_DIR, 3);
 setbkcolor (BLUE);
 setcolor (WHITE);
 // Виводимо текст в лівому верхньому кутку вікна
 outtextxy (1, 1, "А ось і нове вікно");
 getch ( );
 closegraph ( );
 return (0);
}
```

Побудова графіків функцій в декартовій системі координат

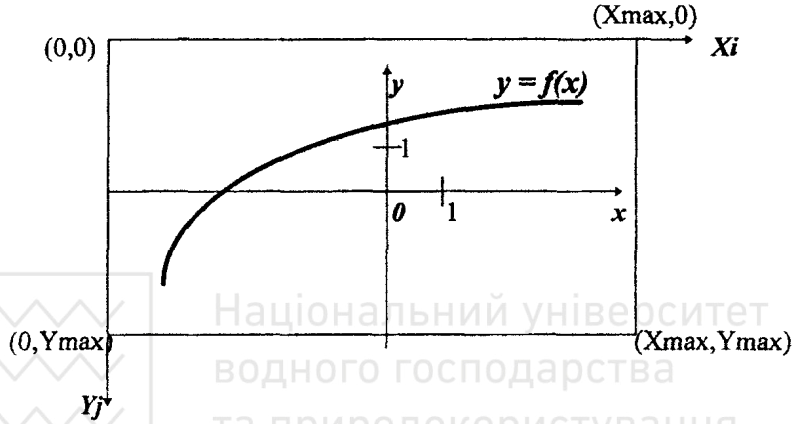
Сучасні комп'ютери виконують мільйони арифметичних операцій за секунду, що дозволяє швидко обчислювати з дуже великою точністю значення і будувати графіки складних математичних функцій.

Національний університет
водного господарства
та природокористування

Особливістю побудови заданих у вигляді y = f(x) функцій є необхідність врахування відмінності між системами екранних і математичних координат.

Екранні координати точок задаються цілими числами Xi і Yj, які відповідають номерам стовпчиків і рядків утвореної пікселями матриці.

Математичні координати x і y, які використовуються при обчисленні y = f(x) повинні бути дійсними числами , причому напрями вертикальних осей Yj і y протилежні.



Національний університет
водного господарства
та природокористування

Алгоритм побудови графіка функції в прямокутній системі координат

1) Визначаються ділянки на екрані (вікна) для побудови графіка, пояснюючих підписів, діалогових повідомлень та ін. Побудову декількох графіків можна виконати в різних вікнах, спільно в одному вікні або по черзі змінювати.

2) Для універсальності програми функція y=f(x) оформляється у вигляді окремої підпрограми, яку можна потім легко змінювати.

3) В робочому вікні для побудови графіка визначається положення центра координат Xc, Yc, масштаби по осях тX, тY.

4) З допомогою циклу для кожного значення екранної координати Xi обчислюється відповідне значення математичної координати x.

$$x=(Xi-Xc)/тX;$$

5) Обчислюється y =f(x).

6) Перехід від математичної координати y до екранної

$$Yj=Yc-(y*тY);$$

7) Побудова на екрані невеликого кола з центром (Xi,Yj) або зв'язування точки (Xi,Yj) відрізком з попередньою точкою графіка.

Національний університет
водного господарства
та природокористування

Приклад.

Побудова графіка функції f(x)=50sin(x)/(1+|x|)

```
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#include<stdio.h>
#define CLIP_ON 4

double f(double x);
void main( )
{
 int mX=3,mY=30;    //масштаби по осях
 int Xmax,Ymax;     //розміри екрана
 int X1,Y1,X2,Y2;     //координати вікна для графіка
 int Xm,Ym;          //розміри вікна
 double Xc,Yc;       //координати центра графіка у вікні
 double Xi,Yi;       //відповідають екранним координатам
 double x,y;         //відповідають математичним координатам
 int gdriver=DETECT,gmode,errorcode;
                  //автоматичний вибір   режиму найбільшого
розширення
 initgraph(&gdriver,&gmode,"c:\\bcpp\\bgi");
                  //ініціалізація відеоадаптера
 errorcode=graphresult( );
 if(errorcode !=grOk)
  {
  printf("Помилка графічного режиму типу: %s \n" ,grapherrormsg
(errorcode));
  printf("Натисніть будь-яку клавішу для виходу");
  getch( );
  exit(1);
  }
 Xmax=getmaxx( ); Ymax=getmaxy( );    //уточнення розмірів екрана
 rectangle(0,0,Xmax,Ymax);             // рамка навколо екрана
 setviewport(5,5,Xmax-5,45,5);         //вікно 1 для пояснюючого
//підпису
 rectangle(0,0,Xmax-10,40);            // рамка навколо вікна 1
 outtextxy(100,20,"Graf f(x)=50sin(x)/(1+|x|)"); //пояснюючий підпис
//для графіка
  X1=5;Y1=50;
```

Національний університет
водного господарства
та природокористування

```
 X2=Xmax-5;Y2=Ymax-5;       //обчислення координат для вікна 2
 Xm=X2-X1;Ym=Y2-Y1;                  //розміри вікна 2
 setviewport(X1,Y1,X2,Y2,CLIP_ON);  //вікно 2 для графіка
 rectangle(0,0,Xm,Ym);                //рамка навколо вікна 2
 Xc=Xm/2;Yc=Ym/2;                    //центр графіка у вікні 2
 line(0,Yc,Xm,Yc);                   // вісь абсцис
 line(Xc,0,Xc,Ym);                   // вісь ординат
 do
  for(Xi=0;Xi<=Xm;Xi++)
  {
      x=(Xi-Xc)/mX;
      y=f(x);
      Yi=Yc-y*mY;
      if(Xi==0) moveto(Xi,Yi);
          else lineto(Xi,Yi);
  }
 while(getch( )==0);
 closegraph( );
}

double f(double x)
 {
 return(50*sin(x)/(1+abs(x)));
 }
```

Національний університет
водного господарства
та природокористування

Функції перетворення форматів даних

Для виведення числових даних у графічному режимі змінні необхідно перетворювати у символьний формат, оскільки функці] outtext та outtextxy виводять лише дані символьного типу.

Функція	Перетворення
itoa	int → string
atoi	string → int
ecvt	double → string
fcvt	double → string
gcvt	double → string
atof	string → double

Національний університет
водного господарства
та природокористування

```
#include <stdlib.h>
char *itoa(value,string,radix);
int value;      //Число, яке перетворюється
char *string;   //Результат
int radix;      //Основа value
```

Опис

Функція itoa перетворює символи числа value на символьний рядок, який закінчується NULL-символом, і запам‘ятовує результат у string. Аргумент radix визначає основу системи числення для value; його значення може лежати у межах від 2 до 36. Якщо radix=10 і value – від‘ємне число, то першим символом у рядку результата буде знак мінус.

Значення, що повертається

Функція itoa повертає вказівник на рядок. У випадку помилки повертаємого значення немає. Див. також ltoa, ultoa (Tech doc/Bcpp).

Примітка! Розмір пам‘яті, яка відводиться для string, повинен бути більше, ніж потрібно для повертаємого рядка. Функція може повертати рядок довжиною до 17 байтів.

Національний університет
водного господарства
та природокористування

Приклад:

```
#include <stdlib.h>
int radix=8;
char buffer[20];
char *p;
p=itoa(-3445,buffer,radix); /*p="1712113"*/
```

28.3. Опис лабораторного обладнання

3. Персональний комп’ютер.
4. Програмне забезпечення: Borland C++ 3.1.

28.4. Програма роботи

1. Ознайомитись з особливостями роботи в графічних вікнах і алгоритмом побудови графіка функції в декартовій системі координат.
2. Написати та відлагодити програму для виконання завдання (додаток 28.1) згідно варіанту.
3. Зберегти програму на магнітних носіях.

Вимоги до програм

- початкові дані ввести оператором введення;
- на екран вивести графічне зображення згідно свого варіанту.

Національний університет
водного господарства
та природокористування

Вимоги до звіту

Звіт повинен включати в себе:

- Титульний лист із зазначенням номеру варіанту
- Мету роботи
- Програму роботи
- Завдання
- Тексти програм
- Результати виконання програм, скопійовані з монітора комп‘ютера

28.5. Контрольні питання

1. Які відмінності між математичними і екранними координатами ?
2. Як виконується перехід між математичними і екранними координатами для визначення положення точок графіка на екрані ?
3. Як можна розмістити на екрані графіки декількох функцій?
4. Які особливості побудови графіків в декартовій системі координат ?
5. Як побудувати зображення системи координат і координатну сітку?
6. Якими функціями моделюються затухаючі коливання ?
7. Якими способами досягається точність і достовірність побудованого графіка функції?

Національний університет
водного господарства
та природокористування

Додаток 28.1

Побудувати графік функції y=f(x) згідно варіанту

1. $x\cos^2 x$	11. $5\cos^2 x$	21.10 ln(1+|x|)	31. $12e^{1/\sin 2x}$
2. $x\sin^2 x$	12. $10\sin^3 x$	22. sin 2x+cos3x/sin 2x	32. |x|/10
3. $x^2\cos^2 x$	13. 5cos x+sin 5x	23.12 $e^{-x/20}$-sin x;	33. $x^3/100$-|x|
4. $x^2\sin^2 x$	14. 10/x+x	24. 5 $e^{-x/10}\cos 2x$	34. $16/(1+x^2)$
5. |x|$\cos^2 x$	15. sin x+x^3/300-x	25. 10 $e^{1/(2+\sin x)}$	35. 15/(1+|x|)
6. |x|$\sin^2 x$	16. tg x+tg 3x	26. $x(x^2-25)(x^2-90)/1000$	36. 5 $e^{\sin x}$
7. |x|sin x	17. sin 2x/cos 3x	27. 12 $e^{-x/20}$sin x	37. 20 $e^{\sin x}$
8. |x|cos x	18. sin 2x+cox 3x	28. x^2/10- x+1	38. 20cos x/(1+cos |x|)
9. x^cos x	19. 10/x+x	29. 5sin x+sin 5x+sin 15x	39. |x/6|+x^2
10. x^sin x	20. sin x+4x	30. 12 $e^{-x/20}$sin x	40. |x|.