ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра обчислювальної техніки та систем управління

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни
«КОМП'ЮТЕРНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

Частина 2

Харків – 2013

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 26 лютого 2013 р., протокол № 8.

Методичні вказівки призначені для студентів факультету УПП і відповідають робочій програмі дисципліни «Комп'ютерна техніка та програмування» для всіх форм і строків навчання.

Укладачі:

доц. С.Є. Бантюков
старш. викл. С.О. Бантюкова

Рецензент

проф. Р.В. Вовк

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни
«КОМП'ЮТЕРНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

Частина 2

Відповідальний за випуск Бантюков С.Є.

Редактор Буранова Н.В.

Підписано до друку 25.03.13 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 0,75. Тираж 100. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ
ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

Кафедра обчислювальної техніки і систем управління

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ

Методичні вказівки
до лабораторних робіт з дисципліни
«Комп'ютерна техніка та програмування»
для студентів факультету УПП

Частина 2

Харків 2013

Методичні вказівки з дисципліни «Комп’ютерна техніка та програмування» розглянуто та рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 26 лютого 2013 р., протокол № 8.

Методичні вказівки призначені для студентів факультету УПП і відповідають робочій програмі дисципліни «Комп’ютерна техніка та програмування» для всіх форм і строків навчання.

Укладачі:

доц. С.Є. Бантюков
старш. викл. С.О. Бантюкова

Рецензент

проф. Р.В. Вовк

РОБОТА 1

ВИКОРИСТАННЯ СТРУКТУР В ПРОГРАМАХ МОВОЮ СІ

1 Мета роботи – набуття навичок у визначенні і використанні структур даних.

2 Завдання та порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні запитання.

2.3 Розробити програми відповідно до одержаного завдання.

2.4 Ввести розроблені програми у ПЕОМ, відладити їх, виконати і отримати результати.

3 Контрольні запитання

3.1 Що розуміється під структурою в мові програмування Сі?

3.2 Загальна форма визначення структури.

3.3 Напишіть приклад визначення структури.

3.4 Як оголошуються змінні типу структури?

3.5 Як здійснюється звернення до окремих елементів структури?

3.6 Як оголошуються масиви структур?

3.7 Як здійснюється доступ до конкретної структури в масиві структур?

4 Зміст звіту

4.1 Номер роботи, її назва, визначення мети.

4.2 Стислий зміст основних теоретичних положень і відповіді на контрольні запитання.

4.3 Результати виконання завдання: схеми алгоритмів, їх стислий опис, тексти програм, результати роботи програм.

4.4 Висновки по роботі.

5 Навчальний матеріал

5.1 Поняття структури

В мові програмування Сі **структура** є сукупністю змінних різних типів, на яку посилаються за допомогою одного імені. Структура забезпечує зручні засоби сумісного зберігання зв'язаної інформації. Визначення структури утворює маску, яку можна використовувати для створення змінних структури. Змінні, що складають структуру, називаються **елементами структури**.

Загальна форма визначення структури має такий вигляд:

```
struct <ім'я_типу_структури> {
                    <ім'я_змінної1>;
                    <ім'я_змінної2>;
                    <ім'я_змінної3>;
                          . . .
                    < ім'я_змінноїN>;
                                  } [<змінні_структури>];
```

Звичайно всі елементи в структурі логічно зв'язані один з одним. Наприклад, у структурі можна представити інформацію про ім'я і адресу. Наведений нижче фрагмент коду оголошує маску структури, що визначає поля імені та адреси. Ключове слово *struct* повідомляє компілятору, що визначається маска структури.

```
struct addr  {
      char name [ 30 ];
      char street [ 40 ];
      char    city [ 20 ];
      char    state [ 3 ];
      int  zip;
                  };
```

На даному етапі в цьому коді ніякі змінні в дійсності не оголошені. Він тільки визначає вигляд даних. Для оголошення **фактичної змінної** цієї структури необхідно записати

```
struct addr addr_info;
```

Цей рядок оголосить змінну структури типу *addr*, що називається *addr_info*. Допускається оголошувати одну чи більше змінних.

5.2 Звернення до елементів структури

До окремих елементів структури звертаються за допомогою **оператора . (крапка)**. Загальний формат має такий вигляд:

<ім'я_структури>. <ім'я_елемента>

Наприклад, наведений нижче код буде присвоювати поштовий індекс 12345 полю *zip* структурної змінної *addr_info*, що була оголошена раніше.

```
addr_info. zip = 12345;
```

Таким чином, для виведення на екран поштового індексу можна записати

```
printf ( "%d ", addr_info. zip );
```

У такому ж вигляді можна використати масив символів *addr_info. name* у виклику функції *scanf( )*, як це показано нижче:

```
scanf ( "%s ", addr_info. name );
```

Це звернення до функції *scanf()* буде передавати символьний покажчик на початок елемента *name.*

5.3 Масиви структур

Загальним використанням структур є їх застосування в масивах структур. Для оголошення **масиву структур** необхідно спочатку визначити структуру, а після цього оголосити змінну масиву цього типу. Наприклад, для оголошення 100-елементного масиву структур *addr*, що був визначений раніше, можна написати

```
struct  addr  addr_info [ 100 ];
```

Цей рядок створює 100 наборів змінних, що організуються так, як це визначено в структурі *addr*.

Для доступу до конкретної структури індексується ім'я структури. Наприклад, щоб надрукувати поштовий індекс із структури 3, можна записати

printf ("%d ", addr_info [2]. zip);

Подібно до всіх змінних типу масиву, масиви структур починають індексування своїх елементів з нуля.

5.4 Приклад використання структур

Розробити програму, що використовує масив структур для зберігання і виведення інформації про адресатів: ім’я (15 символів), вулиця (20 символів), місто (10 символів).

```
// Програма, що використовує масив структур даних
#include <stdio. h>
#include <conio. h>
#define size 100
main ( )
{
int i, kol;
struct addr {
        char name [ 16 ];
        char street [ 21 ];
        char city [ 11 ];
                } addr_info [ size ];
clrscr ( );
printf ( "Уведіть кількість адресатів ( до 100 ) : " );
scanf ( "%d ", &kol );
printf ( " \n " );
for ( i=0; i<kol; i++ )  {
        printf ( " Уведіть ім'я %d - го адресата ( 15 симв. ) : ",
i+1 );
        scanf ( "%s ", addr_info [ i ]. name );
    printf ( " Уведіть вулицю %d - го адресата ( 20 симв. ) : ",
i+1);
```

```
        scanf ( "%s ", addr_info [ i ]. street );
      printf ( " Уведіть місто %d - го адресата ( 10 симв. ) : ",
i+1 );
        scanf ( "%s ", addr_info [ i ]. city );
      printf ( " \n " );
                    }
printf ( "┌──┬──────────┬──────────┬──────────┐          \n
" );
printf ( "│ №│      Ім'я│          Вулиця│          Місто      \n " );
printf ( "├──┼──────────┼──────────┼──────────┤          \n
" );
for (i=0; i<kol; i++ )  {
      printf ("│ %3d│ %15.15s│ %20.20s│ %10.10s    \n ", i+1, \
     addr_info [ i ]. name, addr_info [ i ]. street, addr_info [ i ].
city );
                          }
printf ( "└──┴──────────┴──────────┴──────────┘          \n
" );
}
```

6 Варіанти індивідуальних завдань

Розробити алгоритм і програму, що використовує масив структур для зберігання і виведення інформації у вигляді відомості про:

1-й рівень

1 Результати легкоатлетичного кросу: прізвище (20 позицій); рік народження (4 позиції); факультет (3 позиції); результат (3 позиції).

2 Вартість будівельних матеріалів: назва (15 позицій); кількість (5 позицій); вартість (5 позицій).

3 Видачу авансу робітникам цеху № N за M місяць R року: прізвище (20 позицій); табельний номер (6 позицій); сума (4 позиції). Значення N, M, R вводяться в діалозі.

4 Облік пропускання занять студентами групи G за M місяць R року: прізвище (20 позицій); дата пропускання (8 позицій); кількість годин (3 позиції). Значення G, M, R вводяться в діалозі.

5 План-графік заміни рейок на дільниці в M місяці R року: перегін (15 позицій); тип рейок (3 позиції); довжина заміни (4 позиції); дата початку робіт (8 позицій); дата кінця робіт (8 позицій). Значення M, R вводяться в діалозі.

2-й рівень

6 Результати поточної успішності студентів: прізвище (20 позицій); контрольна точка 1 (1 позиція), контрольна точка 2 (1 позиція), контрольна точка 3 (1 позиція). Зробити перевірку на правильність введення оцінки.

7 Результати складання екзамену з дисципліни N студентами G групи D дати: прізвище (20 позицій); номер залікової книжки (10 позицій); екзаменаційна оцінка (1 позиція). Значення N, G, D вводяться в діалозі. Зробити перевірку на правильність введення оцінки.

8 Розрахунок перевезення вантажів по V відділенню, Z залізниці, на M місяць, R року: назва маршруту (30 позицій); вага вантажу, т (4 позиції); дальність перевезення, км. (4 позиції); обсяг, ткм (6 позицій). Врахувати, що «обсяг» = «вага вантажу» * * «дальність перевезення». Значення V, Z, M, R вводяться в діалозі.

9 Вартість матеріалів для ремонту: назва (15 позицій); кількість (3 позиції); ціна за од. (4 позиції); сума (6 позицій). Врахувати, що «сума» = «кількість» * «ціна за од.». Визначити загальну вартість всіх матеріалів.

10 Результати складання сесії студентами G групи: прізвище (20 позицій); оцінка з математики (1 позиція); оцінка з фізики (1 позиція); оцінка з обчислювальної техніки (1 позиція).

Визначити середню оцінку з кожної дисципліни і дисципліну, що була складена найкраще. Значення G вводиться в діалозі.

3-й рівень

11 Список студентів групи: прізвище (15 позицій); ім'я (10 позицій); по батькові (15 позицій); рік народження (4 позиції). Вивести на екран інформацію про студентів за заданим ключем – роком народження.

12 Студентів G групи K курсу F факультету: прізвище (15 позицій); ім'я (10 позицій); по батькові (15 позицій); стать (1 позиція); рік народження (4 позиції). Зробити перевірку на правильність запровадження статі. Визначити, скільки відсотків у групі складають чоловіки і жінки. Значення G, K, F запроваджуються в діалозі.

13 Розрахунок навчального навантаження студентів K курсу F факультету у S семестрі NR навчального року, тривалість семестру N тижнів: назва дисципліни (20 позицій); лекції, год (3 позиції); практичні заняття, год (3 позиції); лабораторні роботи, год (3 позиції); всього з дисципліни, год (3 позиції). Врахувати, що «всього з дисципліни» знаходиться як сума «лекції» + «практичні заняття» + «лабораторні роботи». Знайти тижневе завантаження студентів. Значення K, F, S, NR, N вводяться в діалозі.

14 Результати легкоатлетичного кросу: прізвище студента (20 позицій); рік народження (4 позиції); група (10 позицій); результат (3 позиції). Визначити студентів з найкращим і найгіршим результатами.

15 Список студентів групи: прізвище (15 позицій); ім'я (10 позицій); по батькові (15 позицій); рік народження (4 позиції). Упорядкувати список студентів за зростанням року народження.

РОБОТА 2

ВИКОРИСТАННЯ ГРАФІКИ В ПРОГРАМАХ МОВОЮ СІ

1 Мета роботи – набуття практичних навичок розробки програм з використанням функцій графічної бібліотеки.

2 Завдання і порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні запитання.

2.3 Розробити програми відповідно до одержаного завдання.

2.4 Ввести розроблені програми у ПЕОМ, відладити їх, виконати і отримати результати.

3 Контрольні запитання

3.1 В якій бібліотеці описуються графічні функції?

3.2 Яка функція використовується для ініціалізації графічної системи?

3.3 Яким чином можливе автовизначення графічного драйвера?

3.4 Які функції використовуються для визначення координат?

3.5 Які функції використовуються для установлення основного кольору і кольору фону?

3.6 За допомогою яких функцій можливо організувати вивід графічного тексту?

3.7 Чи можливе використання функції *setwritemode()* для роботи з функціями *circle(), ellipse()*?

3.8 Чим відрізняються функції *line()* і *lineto()*?

3.9 Яким чином відключається графічна система?

4 Зміст звіту

4.1 Номер роботи, її назва, визначення мети.

4.2 Стислий зміст основних теоретичних положень і відповіді на контрольні запитання.

4.3 Результати виконання завдання: схеми алгоритмів, їх стислий опис, тексти програм, результати роботи програм.

4.4 Висновки з роботи.

5 Навчальний матеріал

5.1 Загальні відомості

В мові програмування Сі є графічна бібліотека, що містить велику кількість функцій, що дозволяють здійснювати різноманітні графічні операції. Настройка цих функцій на роботу з конкретним відеоадаптером досягається за рахунок підключення необхідного графічного драйвера. **Адаптер** – це електронна схема, що управляє зовнішнім приладом ПЕОМ. **Драйвер** – це спеціальна програма для управління тим чи іншим приладом комп'ютера.

Графічні драйвери містяться в окремих файлах з розширеннями BGI (Borland Graphics Interface).

При необхідності використання графічних функцій у вхідному модулі програми треба помістити директиву *#include<graphics.h>*, що підставляє на своє місце вміст файла заголовка з прототипом функцій.

5.2 Ініціалізація графічної системи. Перемикання режимів

Виконання кожної програми починається в текстовому режимі. Для переведення системи у графічний режим необхідне виконання функції *initgraph ( )*:

initgraph(&<графічний_драйвер>, &<графічний_режим>,
"<шлях_до_bgi_файлів>");

Графічні драйвери використовуються такі:

CGA	EGA64
EGA	IBM8514
EGAMONO	ATT400
HERCMONO	PC3270
VGA	DETECT (Автовизначення
MCGA	драйвера)

Деякі графічні режими:

EGAHI	640 x 350	16 кольорів
EGAMONOHI	640 x 350	2 кольори
HERCMONOHI	720 x 348	2 кольори
VGALO	640 x 200	16 кольорів
VGAMED	640 x 350	16 кольорів
VGAHI	640 x 480	16 кольорів
PC3270HI	720 x 350	2 кольори
IBM8514LO	640 x 480	256 кольорів
IBM8514HI	1024 x 768	256 кольорів

Якщо BGI-драйвери містяться в поточній директорії, то замість параметра *"<шлях_до_bgi_файлів>"* можна задати пустий рядок:

initgraph (&gdriver, &gmode, "");

У противному разі шлях до драйвера записується в такому форматі:

".. \\bgi\\drivers "

Існує функція *graphresult()*, яка аналізує причини неправильного виконання графічної операції:

graphresult ( );

Якщо виконання графічної функції завершилося успішно, функція повертає значення 0. У противному разі значення, що повертається, ідентифікує причину невдачі.

Існує можливість надрукувати повідомлення, що відповідає значенню коду помилки. Для цього використовується функція

grapherrormsg (<код_помилки>);

Наприклад,

error = graphresult( );

printf ("Помилка графіки : %s ", grapherrormsg (error));

Перемикання режимів. Для тимчасового переходу у текстовий режим у графічній бібліотеці існує функція

restorecrtmode ( );

що відновлює той текстовий режим, що був перед зверненням до функції *initgraph()*. Повернутися у графічний режим можна за допомогою функції

setgraphmode (<графічний_режим>);

де *графичний_режим* – цілочисельний номер режиму, що допущений для даного драйвера. Попередній графічний режим можна відновити, якщо його номер був своєчасно визначений за допомогою функції

getgraphmode( );

і збережений у програмі.

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

getmaxmode ( );

Після закінчення роботи з графічною системою її необхідно відключити. Це виконує функція

closegraph ( );

5.3 Робота з вікнами і координатами

Після встановлення графічного режиму весь екран за умовчанням стає графічним вікном. Всередині основного графічного вікна (екрана) можна виділити прямокутне вікно зі своєю системою координат. Виконується це за допомогою функції

setviewport (<x1>, <y1>, <x2>, <y2>, <відсікання>);

де *x1, y1* – координати лівого верхнього кута вікна;
x2, y2 – координати правого нижнього кута вікна.

Якщо параметр <*відсікання*> дорівнює 1, то ті елементи зображення, що не вміщаються у вікні, будуть відсічені, а якщо 0, то межі вікна проігноруються.

Очищення графічного вікна виконує функція

clearviewport ( );

Очищення екрана виконується за допомогою функції

cleardevice ( );

Всі встановлені раніше графічними процедурами параметри скидаються і набувають значення за умовчанням.

Для відновлення параметрів, прийнятих за умовчанням, є функція

graphdefaults ( );

Відновлюються параметри, встановлені функцією *initgraph ( ).*

Координати лівого верхнього кута графічного екрана (0, 0), а координати правого нижнього кута визначаються функціями

getmaxx ( ); і *getmaxy ( );* .

Ці функції повертають максимальну координату по горизонталі і по вертикалі відповідно.

Координати графічного курсора завжди відраховуються щодо лівого верхнього кута вікна. Графічний курсор є невидимим, але його поточні горизонтальні і вертикальні координати можуть бути визначені за допомогою функцій

getx ( ); і *gety ( );* .

Ці координати подаються даними типу *int* і можуть бути як додатними, так і від'ємними.

Переустановлення покажчика позиції виконує функція

moveto (<x>, <y>);

де *x, y* – нові координати в системі координат вікна.

5.4 Установлення кольорів, стилів ліній і зафарбовування, шрифтів

Існує можливість встановлювати окремо основний колір, що малює, і колір фону. Поточне значення основного кольору (колір символів і ліній, що виводяться) встановлюється функцією

setcolor (<колір>);

Кольором фону можна управляти за допомогою функції

setbkcolor (<колір>);

де колір – один з перелічених кольорів чи код.

КОД	КОЛІР	ЗНАЧЕННЯ
0	BLACK	чорний
1	BLUE	синій
2	GREEN	зелений
3	CYAN	бірюзовий
4	RED	червоний
5	MAGENTA	малиновий
6	BROWN	коричневий
7	LIGHTGRAY	світло-сірий
8	DARKGRAY	темно-сірий
9	LIGHTBLUE	світло-синій
10	LIGHTGREEN	світло-зелений
11	LIGHTCYAN	світло-бірюзовий
12	LIGHTRED	світло-червоний
13	LIGHTMAGENTA	світло-малиновий
14	YELLOW	жовтий

15	WHITE	білий

Максимальне значення кольору повертає функція

getmaxcolor ( );

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

setlinestyle (<стиль>, <зразок>, <товщина>);

Коди для параметра *стиль* (тільки для кусково-лінійних графічних примітивів)

Код	**Стиль**
0	SOLID_LINE (суцільна)
1	DOTTED_LINE (з крапок)
2	CENTER_LINE (з крапок і тире)
3	DASHED_LINE (пунктирна)
4	USERBIT_LINE (що визначається користувачем)

Параметр *зразок* задається тільки тоді, коли *стиль* дорівнює 4 (в інших випадках він ігнорується, тому його можна робити рівним 0).

Допущені значення для параметра *товщина*:

1 NORM_WIDTH (лінія в один піксель)
3 THICK_WIDTH (лінія в три пікселі)

Існує можливість зафарбувати виділену на екрані замкнуту область певним засобом. Для установлення стилю і кольору зафарбовування використовується функція

setfillstyle (<стиль>, <колір>);

Допущені значення параметра *стиль*:

0	EMPTY_FILL	Заповнення пустотою
1	SOLID_FILL	Заповнення суцільним кольором
2	LINE_FILL	Заповнення горизонтальною лінією ---
3	LTSLASH_FILL	Заповнення ///
4	SLASH_FILL	Заповнення стовщеними лініями ///
5	BKSLASH_FILL	Заповнення стовщеними лініями \\\
6	LTBKSLASH_FILL	Заповнення \\\
7	HATCH_FILL	Горизонтальні грати
8	XHATCH_FILL	Скісні грати
9	INTERLEAVE_FILL	Коса лінія з прошарками
10	WIDE_DOT_FILL	Рідко розташовані крапки
11	CLOSE_DOT_FILL	Щільно розташовані крапки

При виводі текстового повідомлення існує можливість вибору одного з декількох шрифтів, розміру виводимих символів і напрямку тексту. Ці параметри задаються за допомогою функції

settextstyle (<шрифт>, <напрямок>, <розмір_символів>);

Допущені значення параметра *шрифт*:

0	DEFAULT_FONT (стандартний)
1	TRIPLEX_FONT (типу триплекс)
2	SMALL_FONT (зменшений)
3	SANS_SERIF_FONT (прямий)
4	GOTHIC_FONT (готичний)

Допущені значення параметра *напрямок* :

0	HORIZ_DIR (зліва_направо)
1	VERT_DIR (знизу_вгору)

Існує можливість вирівняти текст за допомогою функції

settextjustify (<горизонтально>, <вертикально>);

Допущені значення параметрів *горизонтально* і *вертикально:*

Параметр	**Назва**	**Значення**	**Вирівнювання**
горизонтально	LEFT_TEXT	(0)	ліворуч <
	CENTER_TEXT	(1)	> в центрі <
	RIGHT_TEXT	(2)	> праворуч
вертикально	BOTTOM_TEXT	(0)	знизу вгору
	CENTER_TEXT	(1)	в центрі
	TOP_TEXT	(2)	зверху до низу

Виведення тексту у вікно можливе за допомогою стандартних функцій *printf ( )* і *puts ( )*. Але вони обмежені виглядом 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 ним, як з повноправним елементом графіки.

Функцій виводу графічного тексту – дві:

outtext (<строка>);

виводить рядок символів, починаючи з поточної графічної позиції;

outtextxy (<x>, <y>, <строка>);

виводить рядок символів, починаючи із позиції *(x, y)*.

5.5 Графічні примітиви

Графічні примітиви призначені для малювання різноманітних геометричних об'єктів.

Засіб взаємодії прямих ліній, що виводяться, задає функція

setwritemode (<режим>);

Допущені значення параметра *режим:*

0	COPY_PUT
1	XOR_PUT

За допомогою цієї функції встановлюється засіб, яким код кольору, що малює, встановлений функцією *setcolor()*, буде взаємодіяти з атрибутами пікселів, що вже знаходяться на місці об'єкта, що малюється. Якщо установлено режим 0, то лінія, що малюється, затирає те, що було на екрані. Якщо задано режим 1, то лінія, що малюється, комбінується з тим, що було на екрані. Для цього використовується операція XOR. Цікава властивість цієї операції в тому, що вивід двічі на одне і те саме місце лінії призводить до її затирання і відновлення вхідного зображення на екрані. Ця функція застосовується для роботи з кусково-лінійними зображеннями.

Для виведення пікселя в специфікованій точці екрана використовується функція

putpixel (<x>, <y>, <колір>);

де *x, y* – координати пікселя в системі координат вікна.

Малює лінію між двома специфікованими точками функція

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.

За допомогою функції

lineto (<x>, <y>);

малюється лінія з поточної точки у точку з новими координатами. При цьому покажчик поточної позиції зміщується зі старої точки у нову.

Лінії малюються поточним стилем, встановленим функцією *setlinestyle( )*, і поточним кольором, встановленим функцією *setcolor( ).*

Малює прямокутник функція

rectangle (<x1>, <y1>, <x2>, <y2>);

де *x1, y1* – координати лівого верхнього кута;

x2, y2 – координати правого нижнього кута.

Зафарбування прямокутника встановленим стилем виконує функція

bar (<x1>, <y1>, <x2>, <y2>);

де *x1, y1* – координати лівого верхнього кута;

x2, y2 – координати правого нижнього кута.

Паралелепіпед із зафарбованою передньою гранню можна намалювати за допомогою функції

bar3d (<x1>, <y1>, <x2>, <y2>, <глибина>, <дах>);

Паралелепіпед обрамляється зовнішнім контуром. Якщо параметр *глибина* буде дорівнювати 0, то буде отримано обрамований прямокутник. Якщо параметр *дах* дорівнює 1 (TOP_ON), то буде намальована верхня грань; якщо він дорівнює 0 (TOP_OFF), то верхня грань не малюється. Наприклад, паралелепіпеди, що розташовані один на одному:

```
setbkcolor ( WHITE );
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 );
```

Накреслити ламану лінію дозволяє функція

drawpoly (<кількість_вершин>, <покажчик_на_масив_цілих>);

Кожна пара чисел інтерпретується як пара координат чергової вершини ламаної. Наприклад,

```
int  x [ 10 ] = { 20, 200, 500, 20, 300, 150, 300, 200 };
x [ 8 ] = x [ 0 ];          // замикання контуру
x [ 9 ] = x [ 1 ];
drawpoly ( 5, x );
```

Побудувати і зафарбувати многокутник дозволяє функція

fillpoly (<кількість_вершин>, <покажчик_на_масив_цілих>);

Ця функція завжди замикає контур, з'єднуючи першу точку списку вершин з останньою.

Для побудови кривих використовуються певні функції.

Для малювання кола з центром у точці *(x,y)* із заданим радіусом у пікселях:

circle (<x>, <y>, <радіус>);

Для малювання дуги кола:

arc (<x>, <y>, <початковий_кут>, <кінцевий_кут>, <радіус>);

Кут виражається в градусах і відраховується проти годинникової стрілки.

Для малювання дуги еліпса:

ellipse(<x>, <y>, <початковий_кут>, <кінцевий_кут>, <x_радіус>, <y_радіус>);

де *x, y* – центральна точка;

x_радіус, y_радіус – довжини півосей еліпса в пікселях.

Зафарбований еліпс з контуром можна одержати, використовуючи функцію

fillellipse (<x>, <y>, <x_радіус>, <y_радіус>);

Зафарбований круговий сектор з контуром малює функція

pieslice(<x>, <y>, <початковий_кут>, <кінцевий_кут>, <радіус>);

Зафарбований еліптичний сектор з контуром малює функція

sector (<x>, <y>, <початковий_кут>, <кінцевий_кут>, <x_радіус>, <y_радіус>);

Заповнення кольором обмеженої області виконується функцією

floodfill (<x>, <y>, <колір_межі>);

де *x, y* – координати точки, з якої починається заповнення замкнутої області в усіх напрямках до *кольору_межі;*

Встановлення кольору заповнення слід здійснювати раніше за допомогою функції *setfillstyle ( )*, наприклад,

```
setcolor ( 4 );
rectangle ( 10, 10, maxx-20, maxy-20 );
setfillstyle ( SOLID_FILL, 2 );
floodfill ( 12, 12, 4 );
```

Приклад програми, що реалізує деякі наведені функції бібліотеки *graphics.h :*

```
#include <graphics.h>
#include <stdlib.h>    // бібліотека стандартних функцій
#include <string.h>   // бібліотека обробки рядків
#include <stdio.h>   // бібліотека стандартного вводу-виводу
#include <conio.h>
main ( )
{
// Назва стилів
char *fname [ ] = { "EMPTY_FILL ", "SOLID_FILL ",
"LINE_FILL",
            "LTSLASH_FILL ", "SLASH_FILL ", "BKSLASH_FILL",
```

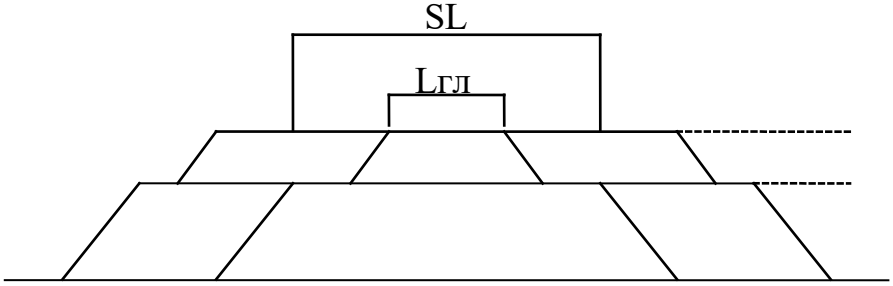
```
                "LTBKSLASH_FILL",
"HATCH_FILL","XHATCH_FILL",
                "INTERLEAVE_FILL", "WIDE_DOT_FILL",
                "CLOSE_DOT_FILL", "USER_FILL"
                        };
 // Автовизначення графічного драйвера, оголошення змінних
int gdriver = DETECT, gmode, errorcode;
int style, midx, midy;
char stylestr [ 40 ];
// Ініціалізація графічної системи
initgraph ( &gdriver, &gmode, " " );

// Аналіз результатів ініціалізації
errorcode = graphresult ( );
if ( errorcode != grOk )              // випадок помилки
    {
  printf ( "Графічна помилка : %s \n ", grapherrormsg
( errorcode ));
    printf ( "Натисніть будь-яку клавішу для зупинки... " );
    getch ( );
    exit ( 1 );                       // Завершення з кодом помилки
    }
midx = getmaxx ( ) / 2;
midy = getmaxy ( ) / 2;
for ( style = EMPTY_FILL; style < USER_FILL; style++ )
     {
      // Вибір стилю заповнення і кольору
      setfillstyle ( style, style );

     // Перетворення стилю у рядок символів
      strcpy ( stylestr, fname [ style ] );

     // Виконання смуги
      bar3d ( 0, 0, midx-10, midy, 0, 0 );

     // Виведення повідомлення про стиль
     outtextxy ( midx, midy, stylestr );
```

```
    // Очікування натиску клавіші
     getch ( );
     cleardevice ( );
      }
// Відключення режиму графіки
outtextxy ( 10, 450, "Натисніть будь-яку клавішу ... " );
getch ( );
closegraph ( );
return 0;
}
```

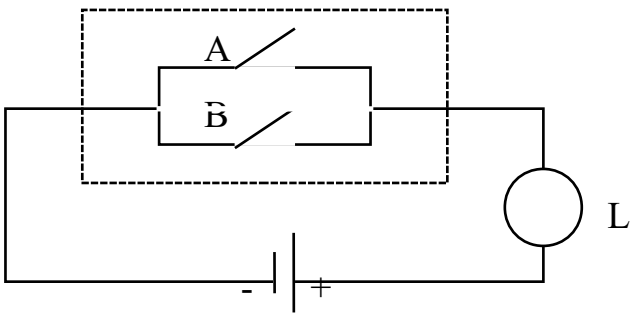
6 Варіанти індивідуальних завдань

1-й рівень

1 Розробити програму для побудови в графічному режимі на екрані дисплея креслення поперечного розрізу земельного полотна. Забезпечити виведення елементів креслення в різних кольорах.



2 Розробити програму для побудови у графічному режимі на екрані дисплея зображення принципу роботи елемента АБО з використанням перемикачів. Забезпечити виведення елементів креслення в різних кольорах.



3 Розробити програму для побудови в графічному режимі на екрані дисплея графіка функції

$$y = 30*\sin^3(0,1*x)$$

на інтервалі $x \in [-150; 150]$ з кроком 0,1.

4 Розробити програму для побудови в графічному режимі на екрані дисплея епіциклоїди за заданим параметричним уявленням (параметричне уявлення кривої на площині з координатами x, y – це дві функції x=x(t) і y=y(t), що визначені на одній числовій множині):

$$x=(a+b)*\cos(t) - a*\cos((a+b)*t/a),$$
$$y=(a+b)*\sin(t) - a*\sin((a+b)*t/a)$$

для $a>0$, $b>0$, $t \in [0, 2\pi]$. Розглянути випадки, коли b/a=1, b/a=2, b/a=3, b/a=4, b/a=5. Величини a, b задаються в діалоговому режимі.

5 Розробити програму для побудови в графічному режимі на екрані дисплея завитку Паскаля за заданим параметричним уявленням:

$$x=a*\cos^2(t) + b*\cos(t);$$
$$y=a*\cos(t)*\sin(t) + b*\sin(t),$$

для $a>0$, $b>0$, $t \in [0, 2\pi]$. Розглянути випадки, коли b>a, a>b. Величини a, b задаються в діалоговому режимі.

2-й рівень

6 Зобразити на екрані прямокутник і забезпечити можливість його пересування за допомогою клавіш управління курсором. При пересуванні прямокутника по екрану за ним повинен залишатися слід.

7 Зобразити на екрані секундомір зі стрілкою. Забезпечити виведення елементів у різних кольорах.

8 Зобразити на екрані прямокутник і забезпечити можливість його пересування за допомогою клавіш управління

курсором і передбачити можливість зміни кольору прямокутника в процесі його руху після натискування клавіш 1-9.

9 Зобразити на екрані прямокутник і забезпечити можливість його збільшення чи зменшення при натисканні на клавіші + або - .

10 Зобразити на екрані прямокутник, забезпечити можливість його пересування за допомогою клавіш управління курсором і передбачити можливість зміни швидкості пересування після натискування клавіш 1-9.

3-й рівень

11 Задана інформація про об'єкт вивчення у вигляді одновимірного масиву даних на п'ять елементів. Забезпечити можливість графічного відображення заданої інформації на екрані дисплея у вигляді кругової секторної діаграми.

12 Задана інформація про об'єкт вивчення у вигляді одновимірного масиву даних на N елементів. Забезпечити можливість графічного відображення заданої інформації на екрані дисплея у вигляді гістограми (стовпчикової діаграми).

13 Зобразити на екрані світлофор. Забезпечити можливість установлення кольорів (червоний, жовтий, зелений) на будь-якій лампі при натискуванні відповідних клавіш.

14 Зобразити на екрані семисегментний індикатор для відображення десяткових чисел. Передбачити можливість установлення на ньому чисел від 0 до 9 після натискування відповідної клавіші.

15 Зобразити на екрані секундомір на двох семисегментних індикаторах, що відображає числа від 00 до 99. Секундомір запускається і зупиняється натисненням на будь-яку клавішу.

РОБОТА 3

ВИКОРИСТАННЯ ПОКАЖЧИКІВ У ПРОГРАМАХ МОВОЮ СІ

1 Мета роботи – вивчення правил використання покажчиків; набуття практичних навичок розробки програм з використанням покажчиків.

2 Завдання та порядок виконання

2.1 Вивчити теоретичний матеріал.

2.2 Підготувати відповіді на контрольні запитання.

2.3 Розробити програми відповідно до одержаного завдання.

2.4 Ввести розроблені програми у ПЕОМ, відладити їх, виконати і отримати результати.

3 Контрольні запитання.

3.1 Що називається покажчиком?

3.2 Що визначає базовий тип покажчика?

3.3 Які оператори використовують з покажчиками? Які дії вони виконують?

3.4 Які арифметичні операції допускається застосовувати до покажчиків? Наведіть приклади.

3.5 На що вказує ім'я масиву без індексу?

3.6 Як здійснюється ініціалізація покажчиків?

4 Зміст звіту

4.1 Номер роботи, її назва, визначення мети.

4.2 Стислий зміст основних теоретичних положень і відповіді на контрольні запитання.

4.3 Результати виконання завдання: схеми алгоритмів, їх стислий опис, тексти програм, результати роботи програм.

4.4 Висновки з роботи.

5 Навчальний матеріал

5.1 Покажчик-змінна

Покажчик – це змінна, що містить адресу байта пам'яті. В більшості випадків це адреса розміщення в пам'яті іншої змінної. Коли одна змінна містить адресу іншої змінної, кажуть, що перша змінна показує на другу (посередня адресація).

Якщо змінна повинна містити покажчик, то її необхідно оголосити відповідним чином. Загальний формат оголошення покажчика-змінної має вигляд

*<тип> *<ім'я_змінної>;*

де *тип* – будь-який з допустимих у мові Сі базових типів;
ім'я_змінної – ім'я покажчика-змінної.

Базовий тип покажчика визначає, на який тип змінної він може показувати. Наприклад, у наступних операторах оголошуються покажчики на символьну і цілі змінні:

```
char  *p;
int  *temp, *p21;
```

5.2 Оператори з покажчиками

З покажчиками застосовують два оператори: * і *&*. Оператор *&* повертає адресу комірці пам'яті свого операнда. Наприклад,

addr_uk = &uk;

поміщає у змінну *addr_uk* адресу байта пам'яті змінної *uk*. Ця адреса указує на місце розташування змінної в пам'яті комп'ютера. Треба мати на увазі, що адреса змінної не має нічого спільного із значенням змінної. Наприклад, якщо змінна *uk* розміщується за адресою *4000*, то після виконання зазначеного вище оператора змінна *addr_uk* буде мати значення *4000.*

Оператор * повертає значення змінної, що розташована за даною адресою. Наприклад, якщо *addr_uk* містить адресу байта пам'яті змінної *uk*, тоді

*val = *addr_uk;*

розмістить значення змінної *uk* у змінну *val*.

Оператори & і * мають більш високий пріоритет, ніж усі інші арифметичні оператори, за винятком унарного мінуса, з яким за пріоритетом вони рівні.

Приклад програми, в якій використовуються два наведених оператори присвоєння для виводу на екран числа 100:

```
#include <stdio.h>
main ( )
{
int  *addr_uk,  uk=100,  val;
addr_uk = &uk;              // присвоєння адреси uk
val = *addr_uk;             //  присвоєння значення, що знаходиться
за
                            //  цією адресою
printf ( "%d ", val );    //  виведення на екран числа 100
}
```

5.3 Вирази з покажчиками

Як і в разі з будь-якою змінною, покажчик можна записувавати в правій частині оператора присвоювання для присвоювання значення покажчику іншої змінної, як показано в такому фрагменті:

```
int x, *p1, *p2;
p1 = &x;
p2 = p1;
printf ( "%p ", p2 );        // друк адреси  x
```

Специфікатор формату *%p* у функції *printf()* визначає, що буде виведено значення покажчика в шістнадцятковому коді.

Над покажчиками можна виконувати тільки дві арифметичні операції: + і - . Припустимо, що *p1* – покажчик на ціле значення з поточною величиною *4000*. Після виконання виразу *p1++*; вміст *p1* буде *4002*, оскільки при кожному прирості покажчик показує на наступне ціле. Те ж саме

справедливо і для від'ємного приросту. Наприклад, при виконанні *p1--*; *p1* буде мати значення *3998.*

До покажчиків можна додавати і віднімати від них цілі величини. Так, у виразі

p1 = p1 + 9;

p1 буде показувати на дев'ятий елемент базового типу *p1* по відношенню до поточного.

5.4 Покажчики і масиви

В мові Сі ім'я масиву без індексів трактується як адреса початкового елемента, тобто **ім'я масиву є покажчиком на масив**. Розглянемо фрагмент:

```
char  str [ 80 ], *p1;
p1 = str;
```

В даному фрагменті покажчику *p1* буде присвоєна адреса першого елемента масиву *str*. Таким чином, звернення до п’ятого елемента масиву можна записати як *str [4]* або як **(p1+4)*.

5. 5 Покажчики і рядки

Оскільки ім'я масиву без індексу є покажчиком на перший елемент цього масиву, то при обробці рядків з неіндексованими іменами рядкових масивів їм будуть передаватися не самі рядки, а покажчики на них. Для ілюстрації цього розглянемо програму, яка порівнює два рядки :

```
#include <stdio.h>
#include <stdlib.h>      //  для функції  exit ( )
main ( )
{
char  *s1,  *s2;

printf ( "Введіть рядки : \n " );
scanf ( "%s ", s1 );
scanf ( "%s ", s2 );
while ( *s1 )
```

```
    if ( *s1-*s2 )  {
        printf ( "\n Нерівні !" );
        exit ( 1 );   //  переривання виконання програми
                    }
    else            {
        s1++;
        s2++;
                    }
printf ( "\n Рівні !" );
}
```

Всі **рядки в мові Сі закінчуються нуль-символом**, який має значення «хибно». Таким чином, умова в операторі

```
while ( *s1 )
```

буде істинною, доки покажчик не досягне кінця рядка (рядок закінчується символом нуля '\0').

При використанні рядкової константи в будь-якому вислові комп'ютер сприймає цю константу як покажчик на перший символ у рядку. Наприклад, у програмі на екран буде виведено фразу *"Ця програма працює "* :

```
#include <stdio.h>
main ( )
{
char *s;

s= "Ця програма працює ";
printf ( s );
}
```

5.6 Масив покажчиків

В мові Сі існує можливість організовувати масиви покажчиків. Так, щоб оголосити масив цілих покажчиків розміром 10, необхідно записати

```
int  *x [ 10 ];
```

Щоб присвоїти адресу цілої змінної з ім'ям *var* третьому елементу масиву покажчиків, необхідно записати

x [2] = &var;

Для одержання значення змінної *var* необхідно записати

**x [2].*

Найчастіше масив покажчиків використовується для утримання покажчиків, що указують на різноманітні повідомлення. Наприклад, нижче наведена програма, яка буде виводити повідомлення про помилку при завданні відповідного коду.

```
#include <stdio.h>
main ( )
{
// оголошення масиву покажчиків
char  *err [ ] = {
                "помилка відкриття файла \n",
                "помилка читання \n",
                "помилка запису \n ",
                "відмова системи \n "
                   };
int  num;
printf ( "Введіть код помилки ( 0-3 ) : " );
scanf ( "%d ", &num );
printf ( "%s ", err [ num ] );
}
```

5.7 Ініціалізація покажчиків

Перед використанням покажчика йому необхідно присвоїти певне значення, інакше використання неініціалізованого покажчика може призвести до аварійного завершення програми.

Можлива така ініціалізація:

*char *p= "Ініціалізація покажчика ";*

Покажчику *p* присвоюється адреса першого символа рядка *"Ініціалі-зація покажчика "*. В програмі покажчик *p* можна використати як будь-який рядок. Наприклад, нижченаведена програма друкує заданий рядок у прямому і зворотному напрямках.

```
#include <stdio.h>
#include <string.h>  // для використання функції strlen ( )
main ( )
{
int  i;
char  *p= "Ініціалізація покажчика ";

 // друкування рядка у прямому і зворотному напрямках
printf ( p );
for ( i=strlen ( p ); i>=0; i-- )  printf ( "%c ", *( p+i ) );
}
```

В цьому прикладі використовується бібліотечна функція визначення довжини рядка *strlen ( )*.

6 Варіанти індивідуальних завдань

За допомогою покажчиків вирішити поставлене завдання.

1-й рівень

1 Задано масив цілих чисел *a [5] ={1, 2, 3, 4, 5}*. Вивести на екран адреси елементів масиву і значення за цими адресами.

2 Масив *a* містить 10 елементів a_i , $i = 0, 9$. Визначити середнє арифметичне значення позитивних елементів масиву.

3 Дано масив *a* з елементами $a_{i\ j}$, $i = 0, 5; j=0, 10$. Визначити різницю між добутком всіх елементів і сумою від'ємних елементів масиву.

4 Дано масив *a* з елементами $a_{i\ j}$, $i = 0, 10; j=0, 15$. Визначити різницю між добутком і сумою елементів $a_{i\ j} > 10$ у парних рядках масиву.

5 Розробити функцію, що визначає довжину запровадженого рядка.

2-й рівень

6 Задано рядок символів s= "Це програма". Надрукувати цей рядок у зворотному напрямку, тобто "амаргорп еЦ".

7 Ввести рядок із великих літер і перевести його у рядок із малих літер, використовуючи бібліотечну функцію tolower(). Здійснити зворотне перетворення, тобто рядок із малих літер перевести у рядок з великих літер, використовуючи бібліотечну функцію toupper().

8 Ввести рядок і визначити, скільки разів у ньому зустрівся зазначений символ.

9 Задано цілочисельний масив з *n* елементів. Необхідно стиснути цей масив, вилучивши з нього нульові елементи. Якщо нульових елементів немає, то повідомити, що стиснути масив неможливо.

10 Надрукувати введений рядок, усунувши з нього символи, що не є літерами і цифрами.

3-й рівень

11 Дано натуральне число *n*. Обчислити суму чисел цього числа.

12 Дано натуральне число *n*. Обчислити суму *k* правих чисел цього числа.

13 Дано натуральне число *n*. Обчислити суму чисел цього числа на непарних позиціях (нумерація позицій починається зліва).

14 Ввести рядок. На основі введеного рядка сформувати новий рядок, що складається тільки з цифр, що входять у введений рядок.

15 У введеному тексті підрахувати кількість символів у слові максимальної довжини (слова розділяються пропусками).