Лабораторна робота № 16
Тема: «Управління виведенням даних на екран»

Мета роботи: отримання навичок роботи з функціями виводу даних на екран в текстовому режимі.

Завдання для підготовки до роботи:

1. Ознайомитися зі стандартними функціями для роботи в текстовому режимі.
2. Розробити алгоритм і скласти програму для вирішення завдання з урахуванням виведення даних на екран в текстовому режимі.
3. Для створення файлів вихідних даних використовувати будь-який текстовий редактор або допоміжні програми.
4. Підібрати набори тестових даних.

Зміст звіту:

1. Тема та мета роботи.
2. Завдання за варіантом.
3. Блок-схема алгоритму роботи програми.
4. Код програми.
5. Контрольний приклад виконання програми: набори тестових даних з обґрунтуванням їх вибору, скриншоти з результатами роботи програми.
6. Висновки про виконану роботу. Опис і аналіз помилок, виявлених при налаштуванні програми.

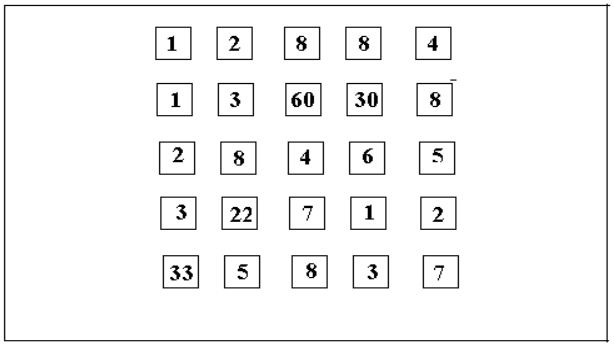
Завдання 16.1.

Дано двовимірний масив 5x5. Організувати введення елементів, виведення результату у вигляді вікон. Змінити після паузи колір вікон, в яких зберігаються задані згідно з умовою елементи. *Роботу виконуємо виключно із застосуванням бібліотеки curses.h.*

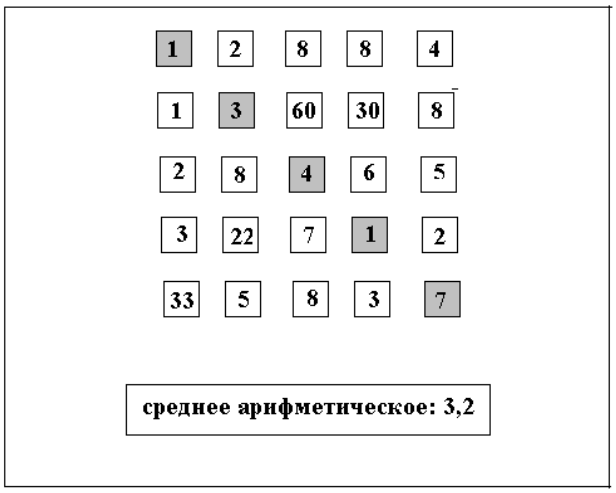
№	Завдання
1	Знайти суму непарних елементів. Замінити знайденим числом всі елементи першого, третього і п’ятого стовпців.
2	Знайти суму непарних елементів. Замінити знайденим числом всі елементи, які знаходяться вище побічної діагоналі.
3	Знайти добуток всіх елементів першого стовпця. Замінити знайденим числом всі елементи першого рядка і п’ятого стовпця.
4	Знайти суму елементів, які знаходяться на непарних позиціях (сума індексів $(i + j)$ для X_{ij} – парне число). Замінити знайденим числом елементи другого і четвертого рядка.
5	Знайти добуток елементів, які знаходяться вище головної діагоналі. Замінити знайденим числом всі непарні елементи.
6	Знайти суму всіх парних елементів, які знаходяться вище побічної діагоналі. Замінити знайденим числом всі елементи четвертого рядка.
7	Знайти суму елементів i-рядка і j-стовпця (i, j – попередньо вводити з клавіатури). Замінити знайденим числом всі елементи матриці, які менші за цю суму.
8	Знайти середнє арифметичне позитивних елементів, які знаходяться нижче головної діагоналі. Замінити знайденим числом всі негативні елементи, розташовані вище головної діагоналі.
9	Знайти суму елементів, які знаходяться на непарних позиціях (сума індексів

	$(i+j)$ для X_{ij} – парне число). Замінити знайденим числом всі негативні елементи.
10	Знайти середнє арифметичне елементів на головній діагоналі і середнє арифметичне побічної. Поміняти місцями елементи на діагоналях.
11	Знайти добуток елементів *i*-рядка і *j*-стовпця (*i*, *j* – попередньо вводити з клавіатури). Замінити знайденим числом всі елементи нижче побічної діагоналі.
12	Знайти середнє арифметичне від'ємних елементів, які знаходяться нижче головної діагоналі. Замінити знайденим числом всі нульові елементи.
13	Знайти добуток елементів, які знаходяться вище побічної діагоналі. Замінити знайденим числом центральний елемент матриці (на перетині головної і побічної діагоналей).
14	Знайти суму елементів, які знаходяться на непарних позиціях (сума індексів $(i+j)$ для X_{ij} – парне число). Замінити знайденим числом всі елементи першого і п'ятого стовпця.
15	Знайти добуток всіх елементів, які знаходяться на головній діагоналі. Замінити знайденим числом всі елементи п'ятого рядка і другого стовпця.
16	Знайти середнє арифметичне позитивних елементів, які знаходяться вище головної діагоналі. Замінити знайденим числом всі негативні елементи.
17	Знайти суму елементів, які знаходяться вище головної діагоналі. Замінити знайденим числом всі парні елементи.
18	Знайти суму елементів, які знаходяться на парних позиціях (сума індексів $(i+j)$ для X_{ij} – непарне число). Замінити знайденим числом всі нульові елементи.
19	Знайти середнє арифметичне елементів на головній діагоналі і середнє арифметичне на побічній. Замінити елементи на діагоналях на одиничні.
20	Знайти добуток всіх елементів, які знаходяться на побічній діагоналі. Замінити знайденим числом всі елементи четвертого стовпця.
21	Знайти середнє арифметичне додатних елементів, які знаходяться вище третього рядка. Замінити знайденим числом перший і другий стовпці.
22	Знайти середнє арифметичне елементів першого і останнього стовпців. Замінити елементи на діагоналях на число –1.
23	Знайти середнє арифметичне парних елементів. Замінити знайденим числом всі елементи останнього стовпця.
24	Знайти добуток елементів, які знаходяться на парних позиціях (сума індексів $(i+j)$ для X_{ij} – непарне число). Замінити знайденим числом всі елементи на головній діагоналі.
25	Знайти середнє арифметичне непарних елементів. Замінити знайденим числом всі елементи побічної діагоналі.
26	Знайти суму всіх парних елементів, які знаходяться нижче головної діагоналі. Замінити знайденим числом всі елементи третього стовпця.
27	Знайти добуток елементів, які знаходяться на непарних позиціях (сума індексів $(i+j)$ для X_{ij} – парне число). Замінити знайденим числом всі елементи п'ятого рядка.
28	Знайти добуток елементів *i*-рядка і *j*-стовпця (*i*, *j* – попередньо вводити з клавіатури). Замінити знайденим числом всі елементи головної і побічної діагоналей.
29	Знайти середнє арифметичне додатних елементів. Замінити знайденим числом всі елементи другого стовпця.
30	Знайти суму від'ємних елементів. Замінити знайденим числом всі елементи побічної діагоналі.

Наприклад: Знайти середнє арифметичне елементів, що розташовані на головній діагоналі двовимірного масиву.



Після паузи:



Теоретичні відомості:

Підключення бібліотеки pdcurses до Code::Blocks

Якщо ви працюєте під Windows, то можна використовувати бібліотеку "**Public Domain Curses**" ("pdcurses"), що доступна за адресою: http://sourceforge.net/projects/pdcurses/files/latest/download.

Перед використанням бібліотеки її потрібно скомпілювати.

Розпаковуємо архів, наприклад, в папку, куди встановили Code::Blocks. Але краще в папку, наприклад, ...\CodeBlocks\pdcurses\, так буде зручніше.

Для компіляції можна скористатися компілятором, який поставляється разом з Code::Blocks.

Заходимо в командний рядок від імені адміністратора (натискаємо Пуск->Всі програми->Службові –Windows->Командний рядок натиснути правою клавішею –> Запуск від адміністратора) і набираємо (рис. 16.1):

path = ...\MinGW\bin

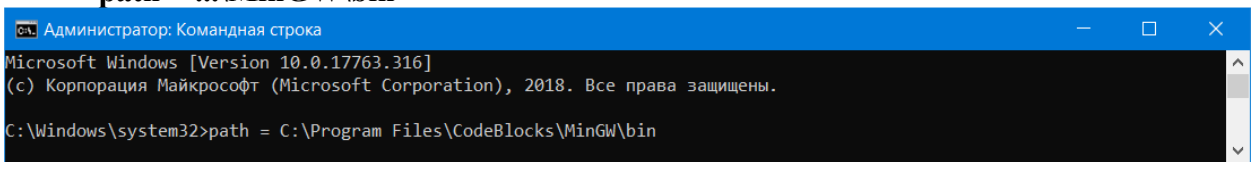


Рисунок 16.1 – Шлях до компілятора

Ця команда додасть в path шлях до папки bin вашого компілятора. Тепер переходимо в папку, в яку розпакували архів з вихідними кодами бібліотеки і переходимо в папку wincon. У цій папці знаходиться makefile, який створений для спрощення збірки, його ім'я Makefile. Його потрібно передати програмі mingw32-make, щоб вона

скомпілювала бібліотеку. Наступна команда виконує зазначену дію (рис. 16.2):

mingw32-make -f Makefile

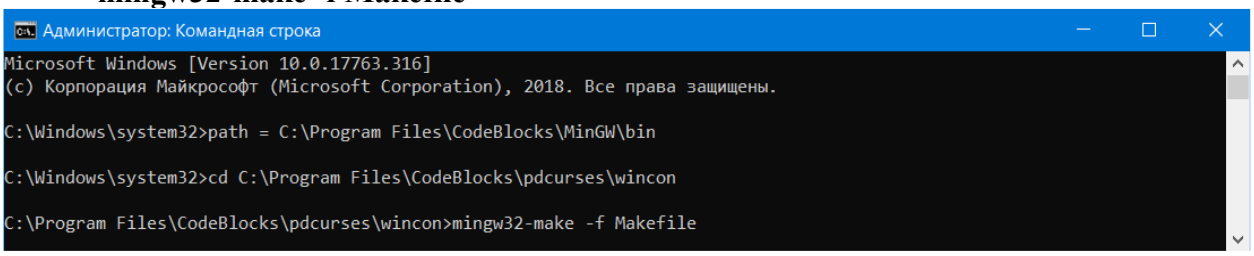


Рисунок 16.2 – Команда компіляції бібліотеки

Якщо з’явилися помилки, додайте до команди make INFOEX=N, тобто команда буде мати вигляд

mingw32-make -f Makefile INFOEX=N

https://sourceforge.net/projects/pdcurses/files/pdcurses/

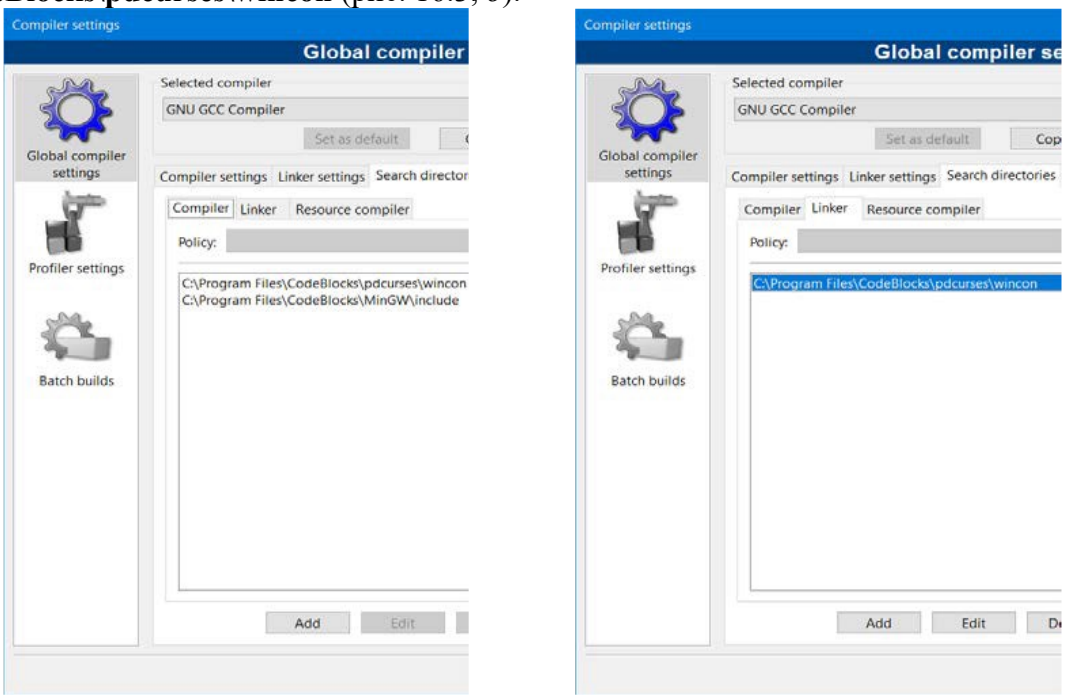
Після закінчення компіляції в папці wincon з'явиться файл **pdcurses.a**, це статична бібліотека, призначена для компоновщика.

Наступним кроком необхідно перенести цей файл в папку зі статичними бібліотеками компілятора. Для CodeBlocks це папка **...\CodeBlocks\MinGW\lib**. Також в папці, в яку розпаковано PDcurses, є заголовочний файл **curses.h**. Його потрібно перенести в папку з заголовочними файлами компілятора. Для CodeBlocks це папка **...\CodeBlocks\MinGW\include**. Тепер PDcurses готова до використання.

Для підключення цієї бібліотеки до CodeBlocks, необхідно в середовищі розробки додати параметри "**Compiler Search Directories**" і "**Linker Search Directories**", де вказати шлях до розпакованих файлів:

Меню **Settings->Compiler->Search Directories->Compiler->Add** вказати шлях **..\CodeBlocks\pdcurses\wincon** (рис. 16.3, *а*);
Меню **Settings->Compiler->Search Directories->Compiler->Add** вказати шлях **..\CodeBlocks\MinGW\include** (рис. 16.3, *а*);

Меню **Settings->Compiler->Search Directories->Linker->Add** вказати шлях **..\CodeBlocks\pdcurses\wincon** (рис. 16.3, *б*).



а) б)

Рисунок 16.3 – Підключення бібліотеки до CodeBlocks

Меню **Settings->Compiler->Linker settings->Add** вказати шлях **..\CodeBlocks\MinGW\lib\pdcurses.a** (рис. 16.4)

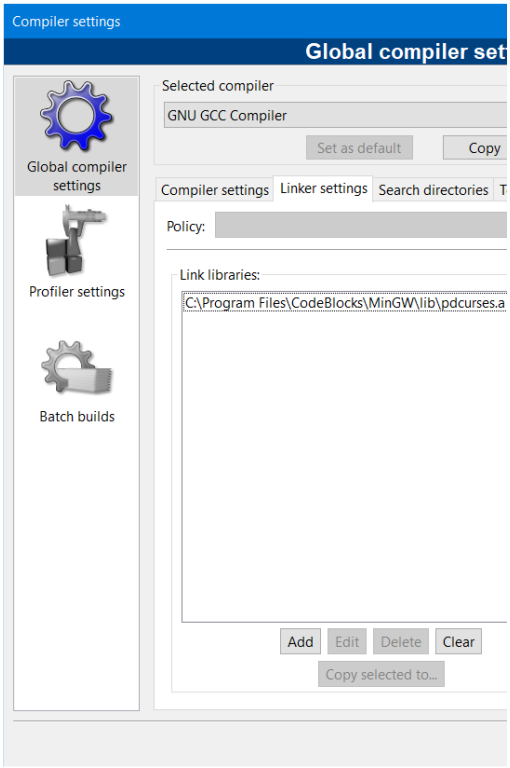


Рисунок 16.4 – Підключення бібліотеки

Для підключення бібліотеки до самого проекту необхідно натиснути правою кнопкою миші на ім'я вашого проекту і обрати пункт "**Properties**..." (рис. 16.5).

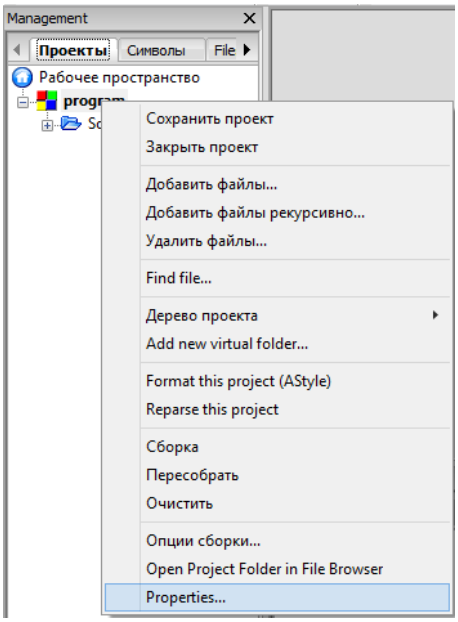
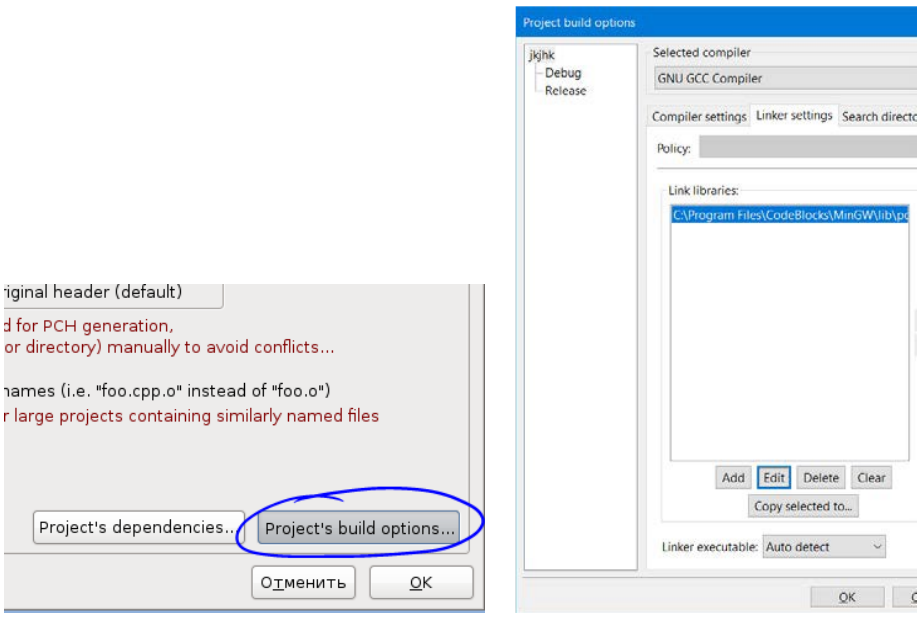


Рисунок 16.5 – Підключення бібліотеки до проекту

У вікні, що з'явилося, натиснути кнопку "**Project's build options...**" в правому нижньому кутку (рис. 16.6, *а*).

У вікні, що з'явилося **Linker settings->Add** вказати шлях **..\CodeBlocks\MinGW\lib\pdcurses.a** (рис. 16.6, *б*)



а) б)

Рисунок 16.6 – Підключення бібліотеки до проекту

Підключення бібліотеки pdcurses до Visual Studio

Качаємо бібліотеку за адресою:
https://sourceforge.net/projects/pdcurses/files/pdcurses/3.4/pdc34dll.zip/download.

Підключення бібліотеку pdcurses до проекту.

Створюємо проект **Win32 Console Application** -> в папку з проектом розархівовуємо вміст архіву **pdc34dll.zip** (не в першу папку, а всередині, де розташовані інші заголовочні файли з розширенням "**.h**") -> повертаємося в проект -> Оглядач рішень (Solution Explorer) -> правою кнопкою миші на імені проекту -> Властивості (Properties) -> Налаштування компоновщика (Linker) -> Введення (Input) -> додаткові залежності (Additional Dependencies -> <Edit ...> -> В саме верхнє поле вводимо «**pdcurses.lib**». Скрізь тиснемо «**OK**».

Пишемо код. Компілюємо.

У коді програми бібліотека підключається як

#include <curses.h>

Робота с бібліотекою ncurses

Кожна програма, яка використовує ncurses, повинна мати наступну структуру:

```
#include <ncurses.h>
...
initscr(); // ініціалізація роботи ncurses

робота з ncurses

endwin(); // завершення роботи з ncurses
```

Виведення

Виведення символу

int addch(const chtype ch)

виводить символ **ch** в поточну позицію курсора і переміщує курсор на один символ вправо або на початок наступного рядка, якщо курсор знаходиться біля правої межі екрану.

(аналогічна функції **putchar** з **<stdio.h>**) chtype тип символів з якими працює ncurses (він включає в себе код символу, колір і додаткові атрибути).

Вставка символу

int insch(chtype ch)

вставляє символ **ch** зліва від курсору і всі символи, що розташовані після курсору, переміщаються на одну позицію вправо.

Виведення рядка з перетворенням за форматом

int printw(const char *fmt,...)

приклад:

```
...
i=1;
printw("Значення i=%d",i);
...
```

виведе **Значення i=1**

(аналогічна функції printf)

*Виведення рядка типу chtype**

int addchnstr(const chtype *chstr, int n)

виводить перші *n* символів або весь рядок символів *chstr*, якщо *n* = -1 на позицію, де розташований курсор.

*Виведення рядка типу char**

int addstr(const char *str)

виводить рядок **str** на позицію, де розташований курсор.

Вставка рядка

int insnstr(const char* str, int n)

вставляє перші *n* символів або весь рядок символів *str*, якщо *n* = -1 на позицію, де розташований курсор. Положення курсору не змінюється, те що стояло праворуч від курсору зміщується вправо.

Вставка порожнього рядка

int insertln()

вставляє порожній рядок. Рядки, що розташовані нижче, починаючи з поточного зсуваються вниз на один рядок.

Режими виведення

Для символів типу **chtype** можна встановлювати такі атрибути, як миготіння або колір символу і фону. Для додавання символу атрибуту миготіння потрібно включити прапорець **A_BLINK**:

chtype ch = 'w' | A_BLINK;

Тепер при виведенні цього символу він буде блимати, якщо звичайно це дозволяє зробити термінал. (**A_DIM** – знижена яскравість, **A_BOLD** – підвищена яскравість, **A_NORMAL** – нормальне відображення, **A_UNDERLINE** – підкреслений, **A_REVERSE** – інверсний).

З включенням кольору трохи складніше. Перед використанням кольорів потрібно проініціалізувати палітру. Палітра – це структура, на якій певній цифрі відповідає певний колір. У нашому випадку одній цифрі відповідають відразу два кольори символів і фону.

```
...
chtype ch;
...
if (!has_colors())
{
  endwin();
  printf("Кольори не підтримуються");
  exit(1);
}
start_color();

// 1 колір у палітрі - червоні символи на чорному фоні
init_pair(1, COLOR_RED, COLOR_BLACK);

// 2 колір у палітрі - зелені символи на жовтому фоні
init_pair(2, COLOR_GREEN, COLOR_YELLOW);
...
ch = 'w' | COLOR_PAIR(1); // символ з кольором 1 з палітри
```

Функція **has_colors** дозволяє дізнатися чи можна використовувати кольори. Функція **start_color ()** повинна викликатися до завдання палітри. Функція **init_pair ()** потрібна щоб задати якій цифрі який колір буде відповідати від **1** до **COLOR_PAIRS-1** (0 зарезервований для стандартного відображення). Для використання кольору в символі потрібно включити прапорець **COLOR_PAIR (номер з палітри)**.

Список кольорів:
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE

Наступні функції дозволяють встановити атрибути виведення за замовчуванням:

Включення атрибутів
int attron(int attrs)
включає атрибути attrs. Наприклад, **attron (COLOR_PAIR (1));** встановлює колір 1 з палітри.

Відключення атрибутів
int attroff(int attrs)
відключає атрибути attrs. Наприклад, **attroff (A_BLINK);** відключає миготіння.

Встановлення атрибутів
int attrset(int attrs)
замінює поточні атрибути атрибутами attrs. Наприклад, **attrset (A_NORMAL);** замінює поточні атрибути на **A_NORMAL**.

Встановлення атрибутів очищення
void bkgdset(chtype ch)
Встановлює атрибути, з якими очищається екран такими функціями як **clear ()**. Наприклад, **bkgdset (COLOR_PAIR (1));** очищення буде здійснюватися кольором 1 з палітри.

Введення

Введення символу
int getch()
повертає введений символ (аналогічна функції getchar). Якщо включено режим обробки командних клавіш, можна дізнатися про натискання функціональних клавіш і клавіш управління курсором. Константи, що відповідні кодам цих клавіш, можна знайти в файлі ncurses.h. Нижче представлено найважливіші з них.

Константа	Клавіша
KEY_DOWN, KEY_UP, KEY_LEFT, KEY_RIGHT	клавіши стрілок
KEY_F(n)	функціональні клавіши 0..63
KEY_BACKSPACE	Backspace
KEY_DC	Delete
KEY_IC	Insert
KEY_HOME	Home
KEY_END	End
KEY_NPAGE	Page Down
KEY_PPAGE	Page Up

Введення рядка за форматом
int scanw(char *fmt,...)
Аналогічна функції scanf. Наприклад, для введення користувачем числа в змінну і виклик функції:
scanw("%d", &i);

Режими введення
Автоматичне відображення при введенні
int noecho() и int echo()
Функція **noecho** відключає автоматичне відображення при введенні. Це потрібно, якщо програміст сам хоче вирішувати виводити йому отриманий символ чи ні. Функція **echo** скасовує дію функції **noecho**.

Встановлення часу очікування
int halfdelay(int tenths)
За замовчуванням такі функції як **getch** чекають введення до тих пір, поки користувач не натисне кнопку. Функція **halfdelay** встановлює режим, в якому введення очікується **tenths** десятих долей секунди, потім в **getch** повертається ERR, якщо користувач не натискав клавіші. Скасувати цей режим можна викликавши функцію **nocbreak** ().

Обробка командних клавіш
int keypad(WINDOW *win, bool bf)
За замовчуванням обробку таких клавіш, як клавіші управління курсором і функціональні клавіші, бере на себе термінал. Щоб встановити режим обробки командних клавіш, потрібно викликати функцію **keypad** з **TRUE** в якості другого параметра. Перший

параметр вказує для якого вікна потрібно встановити цей режим. Якщо вікно не використовується можна вказати **stdscr**. Для відключення потрібно передати **FALSE** в другому параметрі.

Управління курсором

!!! Увага: Всі координати в ncurses відраховуються від верхнього лівого кута, починаючи з 0. Таким чином, верхній лівий кут має координати (0,0).

Переміщення курсору
int move(int y, int x)
встановлює курсор в позицію **x**, **y**.

Отримання поточних координат курсору
void getyx(WINDOW *win, int y, int x)
в змінні **x, y** записуються поточні координати вікна **win**. Якщо вікно не використовується, в якості першого параметра можна вказати **stdscr**.
Отримати розміри екрану можна викликавши функції **getmaxx (stdscr)** і **getmaxy (stdscr)** вони повертають максимальне значення **x** і **y** відповідно для даного екрану.

Інші корисні функції

Очистка екрана
int clear()
заповнює весь екран пробілами

Очищення від курсору до кінця рядка
int clrtoeol()
замінює пробілами інтервал від курсору до кінця рядка.

Очищення від курсору до кінця екрану
int clrtobot()
замінює пробілами інтервал від курсору до кінця екрану.

Вставка/видалення рядків
int insdelln(int n)
для позитивного *n* вставляє *n* порожніх рядків, для негативного – видаляє *n* рядків.

Видалення символу
int delch()
видаляє символ на якому стоїть курсор (символи розташовані праворуч від курсору зсуваються вліво).

Видалення рядку
int deleteln()
видаляє рядок, на якому стоїть курсор (рядки розташовані нижче, зсуваються вгору).

Вікна

Вікно – прямокутна ділянка екрану, з якою можна працювати як з цілим екраном (очищати, виводити текст і т.д.). Тобто здійснювати виведення тільки в певну прямокутну область екрану.

Після ініціалізації створюється вікно **stdscr** з максимально можливими для даного терміналу розмірами. Якщо вікна не будуть використовуватися у всіх функціях, що вимагають вікно, можна вказувати **stdscr**.

Всередині одного вікна також можна створити вікна, які будуть називатися підвікнами.

Створення вікна
WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x)
створює вікно в якому **nlines** рядків і **ncols** стовпців (якщо **nlines** або **ncols** рівні 0, то їм буде присвоєно **begin_y** або **begin_x**) з координатами лівого верхнього кута (**begin_x**, **begin_y**). При виклику з нульовими аргументами, функція створить вікно розміром з екран.

Створення підвікна
WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x)
створює підвікно, в якому **nlines** рядків і **ncols** стовпців (якщо **nlines** або **ncols** рівні 0, то їм буде присвоєно **begin_y** або **begin_x**) з координатами лівого верхнього кута (**begin_x**, **begin_y**) щодо всього екрану. **origwin** – батьківське вікно.

Створення підвікна 2
WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x)
робить те ж саме, що і **subwin**, за винятком того, що **begin_x** і **begin_y** координати щодо вікна батька **origwin**.

Видалення вікна/підвікна
int delwin(WINDOW *win)
видаляє вікно/підвікно **win**.

Переміщення вікна
int mvwin(WINDOW *win, int y, int x)
переміщує вікно **win** в нову позицію **(x, y)** лівого верхнього кута.

Переміщення підвікна
int mvderwin(WINDOW *win, int par_y, int par_x)
переміщує підвікно **win** в нову позицію **(x, y)** лівого верхнього кута щодо батьківського вікна.
Вікно грає роль обмежувача виведення на екран. При введенні-виведенні в вікна, що перекриваються, дані в одному вікні можуть бути затерті даними, виведеними в інше вікно, в місці перекриття вікон.

Приклад створення вікна:

```
#include <ncurses.h>
int main() {
  WINDOW* window;
  initscr();
  cbreak();
  refresh();
  window = newwin(10, 40, 0, 0);
  box(window, 0, 0);
  mvwprintw(window, 0, 1, "press any key to exit!");
  wrefresh(window);
  getch();
  endwin();
  return 0;
}
```

Додаткова інформація про роботу з бібліотекою (n)curses.
http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/

Приклад:

```
#include <stdio.h>
#include <stdlib.h>
#include <curses.h>
#include <unistd.h>
#include <Windows.h>

int main() {
   int N;
   WINDOW*  windowInput, windowOutput;
   initscr(); // Старт режиму curses
   if (!has_colors()) { // ініціалізація кольору
     endwin();
     printf("Colors are not supported!");
     return 1;
   }
   curs_set(0);// curs_set(FALSE); curs_set(TRUE);
   start_color();

   // визначаємо наші пари кольорів "текст - фон"
   init_pair( 1, COLOR_BLACK, COLOR_GREEN );
   init_pair( 2, COLOR_BLACK, COLOR_MAGENTA);
   init_pair( 3, COLOR_BLUE, COLOR_YELLOW);
   init_pair( 4, COLOR_WHITE, COLOR_BLACK  );
   init_pair( 5, COLOR_BLACK, COLOR_WHITE  );
   move(2, 27); // переміщення курсора.
   attron(COLOR_PAIR(1));
   printw("Good");
   attron(COLOR_PAIR(2));
   printw("Morning!");
   attron(COLOR_PAIR(3));
   printw("Have a nice day!");
   attron(COLOR_PAIR(4));
   int x=0, y=3;
   for (x = 20; x < 60; x++)
   {
      mvaddch(y,x,'-'); // переміщує поточну позицію і додає символ
      refresh(); // оновлення екрану з максимально можливою швидкістю.
      Sleep(10); // призупиняє виконання програми на час
   }
    cbreak();
   refresh();
   windowInput = newwin(15, 70, 6, 5); // створює вікно в якому nlines рядків і ncols стовпців з координатами лівого верхнього кута
  wbkgd(windowInput, COLOR_PAIR(5));  // зафарбовує вікно
  box(windowInput, 0, 0); //
  mvwprintw(windowInput,0,1,"First window");
  wrefresh(windowInput);
```

```
  windowOutput = newwin(10, 16, 8, 8); // створює вікно в якому nlines рядків і ncols стовпців з координатами лівого верхнього кута
  box(windowOutput, 0, 0);
  mvwprintw(windowOutput, 0, 1, "window");

wrefresh(windowOutput);
  move(8 + 2, 8 + 1);
    attron(COLOR_PAIR(4));
    printw("Input N: ");
    scanw("%d", &N);

    // початок необов'язкового коду
box(windowInput, 0, 0);
  mvwprintw(windowInput,0,1,"First window");
  wrefresh(windowInput);
init_pair( 14, COLOR_WHITE, COLOR_BLACK  );
    box( windowOutput, 0, 0);
    mvwprintw(windowOutput, 0, 1, "window");
    wrefresh(windowOutput);
    move(8 + 2, 8 + 1);
    attron(COLOR_PAIR(14));
    printw("Input N: %d", N);
    // кінець необов'язкового коду
    move(8 + 4, 8 + 1);
    printw("N = %d", N);

   getch();
    endwin(); // Кінець режиму curses
    return 0;
}
```

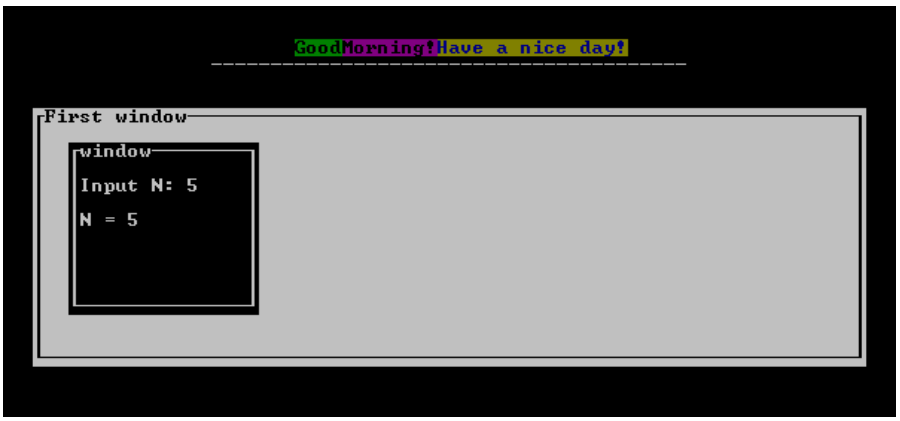


Рисунок 16.7 – Результат роботи програми

Контрольні запитання

1. Які текстові режими підтримують відеоадаптери VGA/SVGA. Як змінити текстовий режим на інший?
2. Що називається текстовим вікном?
3. Як відкрити текстове вікно?
4. За допомогою яких функцій можна керувати позицією і формою текстового курсору?
5. На які групи поділяються клавіші клавіатури ПК?