四時讀書樂 (冬)

木落水盡千崖枯，迥然吾亦見真吾。
坐對韋編燈動壁，高歌夜半雪壓廬。
地爐茶鼎烹活火，四壁圖書中有我。
讀書之樂何處尋？數點梅花天地心。

~ 元・翁森

NCURSES Library

- Functions for Screen Handling

rand()

```
#include <iostream>
using std::cout;
using std::endl;

int main()
{
    for (int i=0; i<9; i++)
    cout << rand() % 6 + 1 << endl;
    return 0;
}
```

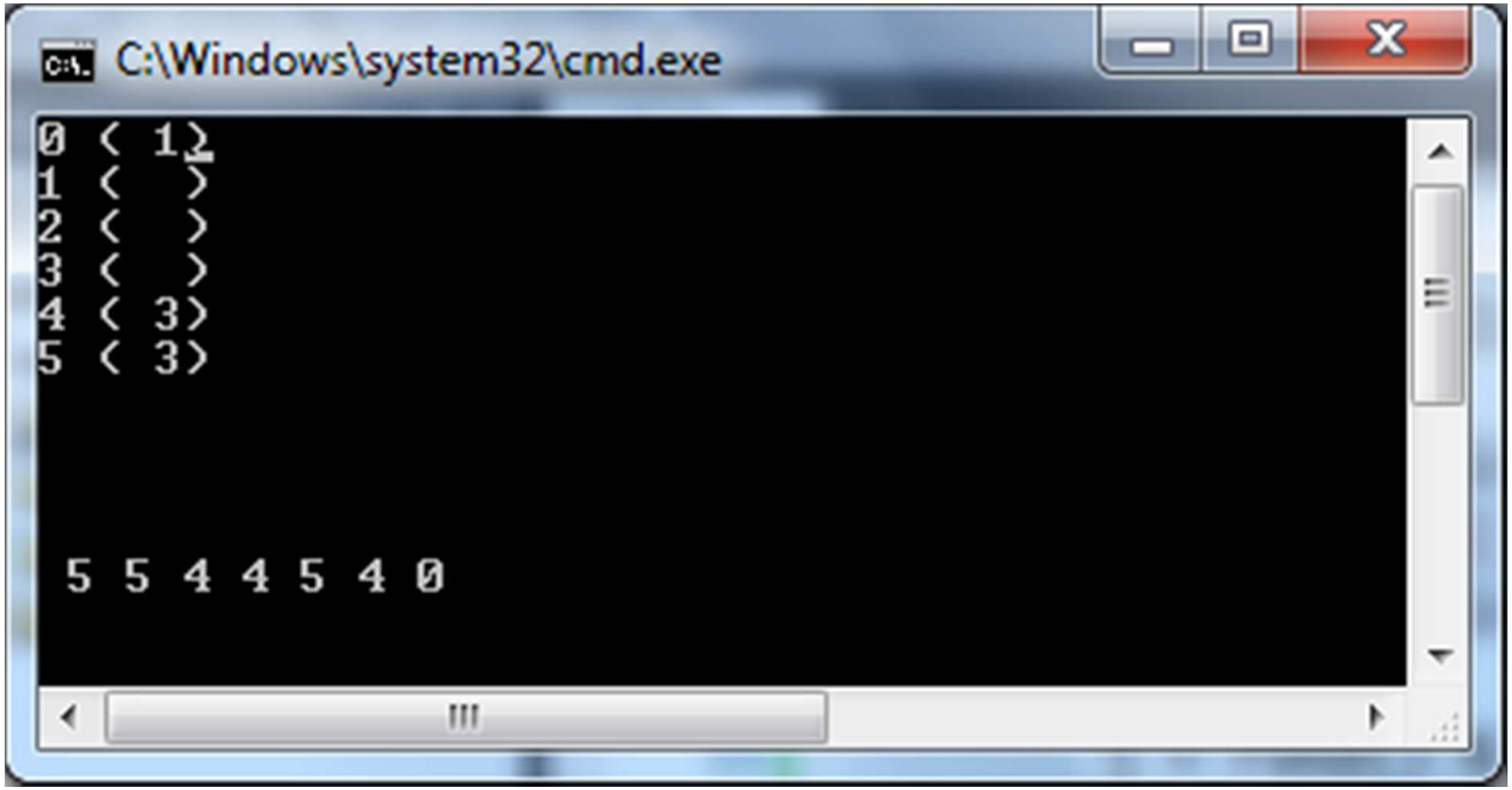
Sleep( )

```
#include <iostream>
#include <Windows.h>
#include <Winbase.h>

using std::cout;
using std::endl;

int main()
{
    for (int i=0; i<9; i++)
    {
       cout << rand() % 6 + 1 << endl;
       Sleep(2000); // 2000 ms = 2s
    }
    return 0;
}
```

Screen Handling



```
0 ( 1)
1 (  )
2 (  )
3 (  )
4 ( 3)
5 ( 3)




 5 5 4 4 5 4 0
```

pdcurses_1.cpp

```
#include <iostream>
#include <Windows.h>
#include <Winbase.h>
#include <curses.h>

int main()
{
        int i, j;
        int a[6] = { 0 };
        initscr();
        for (i=0; i<6; i++)
                printw("%d (  )\n", i);
        for (i=0; i<9; i++)
        {
                j = rand() % 6;
                move(10, i*2);        printw("%2d", j);
                move(j, 3);           printw("%2d", ++a[j]);
                refresh();
                Sleep(2000);    // 2000 ms = 2s
        }
        endwin();
        return 0;
}
```

Definition of **curses** on Wikipedia

- **curses** is
  a terminal control library for Unix-like systems, enabling the construction of text user interface (TUI) applications.
- The name is a pun on the term "cursor optimization". It is a library of functions that manage an application's display on character-cell terminals (e.g., VT100)

Basic Functions

- move(y,x)
  - Move cursor to (y,x) in screen
- addch(ch)
  - Add a character to screen
- addstr(str)
  - Add a string to screen by calling addch()
- printw(fmt, arg1, arg2, …)
  - Formatted print to screen by calling addstr()
- refresh()
  - Update screen

Initialize and Terminate Curses

- initscr()
  - Initialize curses
- endwin()
  - End curses.
  - This function should be called when your program is finished.
  - It will release the space allocated to screen handling in your program.

- Remember to `#include <curses.h>` at the beginning of your program.

Use Curses Library in Visual C++ 2010

- Download Public Domain Curses Library
- Uncompress it and save the 4 files under a local directory, say *L:\PDCurses*.
- In Visual C++ 2010 Express,
  - Project – Property (or `Alt-F7`)
    - Under **Configuration Properties**
      - **Debugging – Environment**
        - PATH=*L:\PDCurses*
      - **C/C++** - **Additional Include Directories**
        - *L:\PDCurses*
      - Expand **Linker**, choose **Input**. In **Additional Dependencies**, insert "*L:\PDCurses\pdcurses.lib;*"

Exercise

- Modify pdcurses_1.cpp so that in addition to the counting number, in the same row you also show a bar (e.g. "*******") to show the same number of stars.

Homework: 8 Queens

- Imagine there are 8 queens attending a race. On your computer screen each queen is represented by a character 'Q'. On each row there is one queen.
  - If you think a single character is monotonous, you may modify this program to handle 8 horses ~/-\^
- Use a random number generator to determine which queen will move forward.
  - You may use the Sleep() function to slow down the program.
- Suppose the length of each lane is 50. The first queen who arrives at the destination wins the race.

四時讀書樂 (春)

山光照檻水繞廊，舞雩歸詠春風香。
好鳥枝頭亦朋友，落花水面皆文章。
蹉跎莫遣韶光老，人生惟有讀書好。
讀書之樂樂何如，綠滿窗前草不除。

~ 元・翁森

四時讀書樂 (夏)

新竹壓檐桑四圍，小齋幽敞明朱暉。
晝長吟罷蟬鳴樹，夜深燼落螢入帷。
北窗高臥羲皇侶，只因素稔讀書趣。
讀書之樂樂無窮，瑤琴一曲來薰風。

~ 元・翁森

四時讀書樂 (秋)

昨夜庭前葉有聲，籬豆花開蟋蟀鳴。
不覺商意滿林薄，蕭然萬籟涵虛清。
近床賴有短檠在，趁此讀書功更倍。
讀書之樂樂陶陶，起弄明月霜天高。

~ 元・翁森

四時讀書樂 (冬)

木落水盡千崖枯，迴然吾亦見真吾。
坐對韋編燈動壁，高歌夜半雪壓廬。
地爐茶鼎烹活火，四壁圖書中有我。
讀書之樂何處尋？數點梅花天地心。

～ 元・翁森

Getting Characters from the Terminal

- getch()
  - Get a character from the terminal
- getstr(str)
  - Get a string from the terminal
- scanw(fmt, arg1, arg2, …)
  - Formatted input from the terminal like scanf().

pdcurses_3.cpp

```
#include <curses.h>

int main()
{
   char text[10];
   int i, j, c;
   initscr();
   getstr(text);              // input the string "1,2"
   addstr(text); addch('\n');

   scanw("%d,%d", &i, &j);    // input the string "1,2" again
   printw("%d\t%d\n", i, j);

   c = getch();
   endwin();
   return 0;
}
```

noecho()

```
#include <curses.h>

int main()
{
   int c;
   initscr();
   // noecho();
   do {
        c = getch();
        printw(" %d\n", c);
   } while (c != '0');

   endwin();
   return 0;
}
```

pdcurses_4.cpp

```
// pdcurses_4.cpp
#include <curses.h>

int main()
{
   int y=10, x=10;
   char c;
   initscr();
   noecho();
   do {
        move(y, x); addch('Q');
        c = getch();
        move(y, x); addch(' ');
        switch (c)
        {
        case 'h':
                x--;
                break;
        case 'l':
                x++;
                break;
        case 'j':
                y++;
                break;
        case 'k':
                y--;
                break;
        }
   } while (c != 'q');

   endwin();
   return 0;
}
```

`curs_set()`

- `curs_set()` alters the appearance of the text cursor.
- `int curs_set(int visibility);`
  - A value of 0 for visibility makes the cursor disappear;
  - a value of 1 makes the cursor appear "normal" (usually an underline)
  - 2 makes the cursor "highly visible" (usually a block).

pdcurses_4a.cpp

```
// pdcurses_4.cpp
#include <curses.h>

int main()
{
   int y=10, x=10;
   char c;
   initscr();
   noecho();
   curs_set(0);  // no cursor
   do {
        move(y, x); addch('Q');
        c = getch();
        move(y, x); addch(' ');
        switch (c)
        {
        case 'h':
                x--;
                break;
        case 'l':
                x++;
                break;
        case 'j':
                y++;
                break;
        case 'k':
                y--;
                break;
        }
   } while (c != 'q');

   endwin();
   return 0;
}
```

Function Keys

- Call `keypad()` to enable the handling of Function keys and arrow keys.
  - `int keypad(WINDOW *win, bool bf);`
  - `keypad(stdscr, TRUE);`
- `getch()` returns an integer corresponding to the key pressed.
  - If it is a normal character, the integer value will be equivalent to the ASCII code of the character.
  - Otherwise it returns a number which can be matched with the constants defined in `curses.h`.
    - For example if the user presses F1, the integer returned is 265.

Function Keys (cont.)

- With keypad() enabled, you can check the returned value of getch() with the constants defined in curses.h
  - KEY_UP, KEY_DOWN, KEY_LEFT, KEY_RIGHT
  - KEY_HOME, KEY_END,
  - KEY_F(n)

Key Definitions

- `#define KEY_IC        0x14b  /* insert char or enter ins mode (Insert) */`
- `#define KEY_DC        0x14a  /* delete character (Delete) */`
- `#define KEY_HOME      0x106  /* home key */`
- `#define KEY_END       0x166  /* end key */`
- `#define KEY_PPAGE     0x153  /* previous page (PageUp) */`
- `#define KEY_NPAGE     0x152  /* next page (PageDown) */`
- `#define PADENTER      0x1cb  /* enter on keypad */`

You may check curses.h to see more definitions.

Colors

- To start using color, you should first call the function `start_color()`.
  - To find out whether a terminal has color capabilities or not, you can use `has_colors()` function, which returns FALSE if the terminal does not support color.
- Colors are always used in pairs.
  - A color-pair consists of a foreground color and a background color.
  - Initializes a color-pair with the routine `init_pair()`. After it has been initialized, `COLOR_PAIR(n)` is used to represent the color attribute.

pdcurses_2.cpp

```
#include <curses.h>

int main()
{
        initscr();
        start_color();

        init_pair( 1, COLOR_WHITE, COLOR_RED );
        attron( COLOR_PAIR(1) );
        printw("Background red");
        attroff( COLOR_PAIR(1) );

        refresh();
        getch();
        endwin();
        return 0;
}
```

Pre-defined Colors on Unix

- COLOR_BLACK = 0
- COLOR_RED = 1
- COLOR_GREEN = 2
- COLOR_YELLOW = 3
- COLOR_BLUE = 4
- COLOR_MAGENTA = 5
- COLOR_CYAN = 6
- COLOR_WHITE = 7

Exercise: Arrow Keys

- Modify pdcurses_4a.cpp so that users can use arrow keys and HJKL to control the movement of ‘Q’.
- Moreover, try to allow users to use both uppercase ‘H’ and lowercase ‘h’ to do the same movement.
- Users can also change the color of ‘Q’ by pressing ‘0’…’7’.