Лабораторна робота № 1

(4 години)

Тема: Консольні Win32-програми для Windows.

Мета: Ознайомитися із принципами і засобами створення консольних програм для Windows. Навчитися використовувати функції Win32 API для роботи з консоллю.

Короткі теоретичні відомості

Створення консолі

Microsoft Windows створює нову консоль, коли вона запускає процес символьного режиму. Наприклад, Windows створює нову консоль, коли вона запускає командний процесор cmd.exe. Коли командний процесор запускає новий процес консолі, користувач може визначити чи система створює нову консоль для нового процесу чи вона успадковує консоль командного процесора.

Якщо запустити консольний додаток із командного рядка, то додаток використовує "чужу" консоль. Для створення своєї консолі потрібно від'єднатися від чужої консолі і створити свою. Для цього використовуються наступні функції:

`BOOL FreeConsole(VOID)` - від'єднує процес від консолі.

`BOOL AllocConsole(VOID)` - виділяє нову консоль для процесу.

Для роботи із консоллю потрібно одержати дескриптор консолі:

```
HANDLE GetStdHandle(DWORD nStdHandle);
```

де *nStdHandle* визначає пристрій, для якого одержують дескриптор:

`STD_INPUT_HANDLE`	(-10)	Стандартний пристрій вводу
`STD_OUTPUT_HANDLE`	(-11)	Стандартний пристрій виводу
`STD_ERROR_HANDLE`	(-12)	Стандартний пристрій помилок

Наприклад:
```
hStdIn = GetStdHandle(STD_INPUT_HANDLE);
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
```

Після цього усі операції із консоллю здійснюються через її ***дескриптор***. Якщо функція завершується фатально, то вона повертає значення `INVALID_HANDLE_VALUE`.

Настройка консолі

Для настройки параметрів консолі використовують такі функції:

Функція API	Коментарі
DWORD **GetConsoleTitle(** LPTSTR **lpConsoleTitle**, DWORD **nSize);**	*Одержати заголовок вікна консолі* адреса буфера розмір буфера
BOOL **SetConsoleTitle**(LPCTSTR **lpConsoleTitle);**	*Встановити заголовок вікна консолі* адреса нового заголовка
BOOL **SetConsoleMode**(HANDLE **hConsoleHandle,** DWORD **dwMode);**	*Встановити режим консолі* дескриптор вхідного або екранного буфера вхідний або вихідний режим
BOOL **SetConsoleCursorPosition**(HANDLE **hConsoleOutput,** COORD **dwCursorPosition);**	*Встановити нові координати курсору* дескриптор екранного буфера нові координати курсору
BOOL **SetConsoleTextAttribute**(HANDLE **hConsoleOutput,** WORD **wAttributes);**	*Встановити атрибути тексту консолі* дескриптор екранного буфера кольори тексту і фону

Закрити консоль можна за допомогою функції **FreeConsole**. Якщо інші процеси підключені до консолі, вона зберігається. Коли останній процес від'єднується від консолі, вона закривається.

Функції для роботи з консоллю

Функції Win32 API допускають два рівні доступу до консолі: високорівневий і низькорівневий.

Високорівневі вхідні функції фільтрують вміст вхідного буфера і повертають лише потік символів з клавіатури, відкидаючи службову інформацію та інші події.

Високорівневі функції виведення записують потік символів у буфер екрану, які відображаються в поточному розташуванні курсора.

Високорівневі функції також підтримують перепризначення стандартних дескрипторів і керують режимами консолі.

Повний перелік функцій для роботи з консоллю можна знайти в довідковій системі Microsoft Win32 Programmer's Reference у розділі Consoles and Character-Mode Support → Console Functions.

Високорівневі функції

Програма може використовувати файлові функції вводу/виводу **ReadFile** і **WriteFile**, ф також спеціалізовані консольні функції **ReadConsole** і **WriteConsole**, для високорівневого I/O, який надає непрямий доступ до вхідного і екранного буферів.

```
BOOL ReadConsole(
  HANDLE hConsoleInput,              // дескриптор вхідного буфера консолі
  LPVOID lpBuffer,                   // адреса буфера прийому даних
  DWORD nNumberOfCharsToRead,        // кількість символів для читання
  LPDWORD lpNumberOfCharsRead,       // адреса числа прочитаних символів
  LPVOID lpReserved );               // зарезервовано, завжди NULL

BOOL WriteConsole(
  HANDLE hConsoleOutput,             // дескриптор екранного буфера консолі
  CONST VOID *lpBuffer,              // адреса буфера даних для виводу
  DWORD nNumberOfCharsToWrite,       // кількість символів для запису
  LPDWORD lpNumberOfCharsWritten,    // вказівник числа прочитаних символів
  LPVOID lpReserved);                // зарезервовано, завжди NULL
```

Функції **ReadConsole**, **WriteConsole** оперують з текстовими рядками. При підготовці текстових рядків для цих функцій корисною може бути функція Win API **wsprintf**.

Ця функція призначена для форматованого виведення послідовності символів і значень у текстовий буфер. Дія функції схожа на функцію **printf**, але вона виводить не на екран, а в текстовий буфер. Функція автоматично додає кінцевий нуль, хоча він не входить у число виведених символів, яке повертає функція по завершенні своєї роботи.

```
int wsprintf(
  LPTSTR lpOut,        // pointer to buffer for output
  LPCTSTR lpFmt,       // pointer to format-control string
  ...                  // optional arguments
 );
```

Параметри

lpOut вказівник на буфер для прийому символів.

lpFmt вказівник на рядок форматування із кінцевим нулем.

Після цього йдуть значення, тип і кількість яких відповідає рядку форматування.

Функція повертає кількість виведених у буфер символів (за винятком кінцевого нуля).

Низькорівневі функції

Низькорівневі функції вводу забезпечують прямий доступ до вхідного буфера і дають можливість додаткам отримувати детальні дані про події клавіатури і миші, а також про події, що визначають взаємодію користувача із вікном консолі.

Низькорівневі функції виводу дають можливість додатку читати або записувати певну послідовність комірок символів у буфер екрану.

Високорівневі і низькорівневі методи I/O не взаємовиключні, і програма може використовувати будь-яку комбінацію цих функцій. Проте, зазвичай прикладні програми використовують або один підхід або інший.

Низькорівневі функції вводу

Win32 API надає п'ять низькорівневих функцій для доступу до вхідного буфера консолі:

ReadConsoleInput	Читає і видаляє записи із вхідного буфера. Функція не повертається, поки є хоча б один запис, доступний для читання. Прочитані записи переміщаються в буфер викликаючого процесу. Непрочитані записи залишаються у вхідному буфері для наступної операції читання даних. Функція повідомляє загальне число записів, які були прочитані.
PeekConsoleInput	Читає вхідні записи без видалення із вхідного буфера. Якщо жоден запис недоступний, функція повертається негайно. Функція повідомляє загальне число записів, які були прочитані.
GetNumberOfConsole-InputEvents	Визначає число непрочитаних вхідних записів у вхідному буфері.
WriteConsoleInput	Розміщує записи у вхідному буфері позаду наявних записів. Вхідний буфер зростає динамічно, якщо необхідно утримувати додаткові вхідні записи.
FlushConsoleInputBuffer	Відкидає всі непрочитані події у вхідному буфері (очищає буфер).

Низькорівневі функції виводу

Низькорівневі функції виводу консолі надають прямий доступ до символьних комірок буфера екрану. Одні функції читають або записують до послідовних комірок екранного буфера, починаючи із будь-якої позиції. Інші функції читають або записують до прямокутних блоків комірок.

Наступні функції читають або записують до вказаної послідовності символьних комірок в буфері екрану, починаючи із вказаної.

ReadConsoleOutputCharacter	Копіює рядок символів Unicode або ANSI із буфера екрану.
WriteConsoleOutputCharacter	Записує рядок символів Unicode або ANSI до буфера екрану.
ReadConsoleOutputAttribute	Копіює атрибути кольору рядка тексту і фону з буфера екрану.
WriteConsoleOutputAttribute	Записує атрибути кольору рядка тексту і фону до буферу екрану.
FillConsoleOutputCharacter	Записує єдиний Unicode або символ ANSI до вказаної кількості послідовних комірок в буфері екрану.
FillConsoleOutputAttribute	Записує атрибути кольору рядка тексту і фону до вказаної кількості послідовних комірок в буфері екрану.

Низькорівневі функції вимагають писати більше коду і вибирати серед більшого діапазону функцій, проте він також надає програмі більшої гнучкості.

Для роботи із вхідним буфером консолі за допомогою низькорівневих функцій слід ознайомитися із структурою запису у вхідному буфері. Кожний запис у вхідному буфері складається із типу події (WORD EventType) і запису про подію. Тип події приймає наступні значення: KEY_EVENT, MOUSE_EVENT, WINDOW_BUFFER_SIZE_EVENT, MENU_EVENT, FOCUS_EVENT (подія клавіатури, миші, зміни розміру буфера, меню, фокусу).

Розглянемо для прикладу структуру записів про подію від клавіатури і миші. Вони мають такі прототипи:

Подія клавіатури	**Подія миші**

typedef struct _KEY_EVENT_RECORD {
  BOOL bKeyDown;
  WORD wRepeatCount;
  WORD wVirtualKeyCode;
  WORD wVirtualScanCode;
  union {
  WCHAR UnicodeChar;
  CHAR AsciiChar;
  } uChar;
  DWORD dwControlKeyState;
} KEY_EVENT_RECORD;
	typedef struct _MOUSE_EVENT_RECORD {
	  COORD dwMousePosition;
	  DWORD dwButtonState;
	  DWORD dwControlKeyState;
	  DWORD dwEventFlags;
	} MOUSE_EVENT_RECORD;

Повний запис у вхідному буфері для подій клавіатури і миші має такий вигляд:

EventType	**Event**						
MOUSE EVENT	MOUSE_EVENT_RECORD						
MOUSE_-EVENT	dwMouse-Position		dwButtonState		dwControlKeyState		dwEventFlags
	wX	wY					
0	4	6	8		12		16
KEY EVENT	KEY_EVENT_RECORD						
KEY_EVENT	bKeyDown		wRepeat Count	wVirtual-KeyCode	wVirtual-ScanCode	uChar	dwControl-KeyState
0	4		8	10	12	14	16

Записи у вхідному буфері консолі містять поля, які відображають стан кнопок миші (dwButtonState), стан керуючих клавіш клавіатури (dwControlKeyState) та тип події миші (dwEventFlags). Ці поля можуть приймати наступні значення.

dwButtonState:	**dwControlKeyState:**
FROM_LEFT_1ST_BUTTON_PRESSED RIGHTMOST_BUTTON_PRESSED FROM_LEFT_2ND_BUTTON_PRESSED FROM_LEFT_3RD_BUTTON_PRESSED FROM_LEFT_4TH_BUTTON_PRESSED	RIGHT_ALT_PRESSED LEFT_ALT_PRESSED RIGHT_CTRL_PRESSED LEFT_CTRL_PRESSED SHIFT_PRESSED NUMLOCK_ON SCROLLLOCK_ON CAPSLOCK_ON ENHANCED_KEY
dwEventFlags: DOUBLE_CLICK MOUSE_MOVED MOUSE_WHEELED	

Розглянемо приклади низькорівневих функцій:

Функція читання запису із вхідного буфера консолі:

```
BOOL ReadConsoleInput(
    HANDLE hConsoleInput,           // дескриптор вхідного буфера консолі
    PINPUT_RECORD lpBuffer,         // адреса буфера для читання даних
    DWORD nLength,                  // кількість записів, які слід прочитати
    LPDWORD lpNumberOfEventsRead    // адреса числа прочитаних записів
   );
```

Функція запису в екранний буфер консолі:

```
BOOL WriteConsoleOutputCharacter(
    HANDLE hConsoleOutput,          // дескриптор екранного буфера консолі
    LPCTSTR lpCharacter,            // вказівник на буфер, із якого виводять
    DWORD nLength,                  // кількість комірок для запису
    COORD dwWriteCoord,             // координати першої комірки
    LPDWORD lpNumberOfCharsWritten // вказівник на число записаних комірок
   );
```

Функція установки координат курсору:

```
BOOL SetConsoleCursorPosition(
    HANDLE hConsoleOutput,          // дескриптор екранного буфера консолі
    COORD dwCursorPosition          // нові координати курсору
 );
```

Завдання для виконання роботи

1. Написати програму для одержання відомостей про систему, використавши функції Win32 API типу: **GetSystemDirectory()**, **GetWindowsDirectory()**, **GetComputerName()**, **GetUserName()**, **GetVersionEx()**, **GetKeyboardType()**. Для виведення результатів використати функції **wsprintf()**, **WriteConsole()**.
2. Написати програму для одержання відомостей про систему, використавши функції Win32 API: **GetSystemInfo()**, **GetSysColor(), GetSystemMetrics()**. Передбачити введення команд з клавіатури.
3. Написати програму, яка виводить в задану позицію консолі інформацію про координати миші, стан керуючих клавіш (Alt, Ctrl, Shift), скан-код та ASCII-код натиснутої клавіші. Для виведення результатів використати функції **SetConsoleCursorPosition()**, **wsprintf()**, **WriteConsole()**,

Контрольні питання

1. Що таке консольна Windows-програма?
2. Які стандартні пристрої введення-виведення використовує консоль?
3. Що таке вхідний буфер та екранний буфер консолі?
4. Що таке високорівневий та низькорівнеаий доступ до буферів консолі?
5. Які високорівневі консольні функції Win32 API ви знаєте?
6. Які можливості надають низькорівневі консольні функції?
7. Як визначити стан кнопок миші?
8. Як визначити стан керуючих клавіш клавіатури?
9. Як визначити код символа, введеного з клавіатури?

Додаток 1

Скелет консольної програми

```
#include <windows.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
  HANDLE hIn, hOut;
  DWORD Read, Written;
  DWORD Numb = 1;
  char Name[] = "Using of Console functions\n\n";

        FreeConsole();
        AllocConsole();
        hIn = GetStdHandle(STD_INPUT_HANDLE);
        hOut = GetStdHandle(STD_OUTPUT_HANDLE);

        WriteConsole(
                hOut,                   // handle to a console screen buffer
                Name,                   // pointer to buffer to write from
                strlen(Name),           // number of characters to write
                &Written,               // pointer to number of characters written
                NULL                    // reserved
        );
  . . .
        ReadConsole(
                hIn,                    // handle of a console input buffer
                Buf,                    // address of buffer to receive data
                Numb,                   // number of characters to read
                &Read,                  / address of number of characters read
                NULL                    // reserved
        );

  return 0;
}
```

Додаток 2

Приклад використання низькорівневих функцій

```
HANDLE hIn, hOut;
  DWORD Read, Written;
  DWORD ToRead = 1;
  INPUT_RECORD InRec;
  COORD Coor;

  Coor.X = 0;
  Coor.Y = 5;

      /* Читаємо запис вхідного буферу, якщо  KEY_EVENT – виводимо його поля */

        while ((int)InRec.Event.KeyEvent.uChar.AsciiChar != 27)
         {
                ReadConsoleInput(hIn, &InRec, ToRead, &Read);
                SetConsoleCursorPosition(hOut, Coor);
                if (InRec.EventType == KEY_EVENT)
                 {                                                  // стираємо рядок і виводимо поля
                        FillConsoleOutputCharacter(hOut,' ',80,Coor,&Written);
                        printf("%u %u %u %u %#x - %c %#x",
                                InRec.Event.KeyEvent.bKeyDown,
                                InRec.Event.KeyEvent.wRepeatCount,
                                InRec.Event.KeyEvent.wVirtualKeyCode,
                                InRec.Event.KeyEvent.wVirtualScanCode,
                                InRec.Event.KeyEvent.uChar.UnicodeChar,
                                InRec.Event.KeyEvent.uChar.UnicodeChar,
                                InRec.Event.KeyEvent.dwControlKeyState);
                 };
         };
```