

## Лабораторна робота № 2

(4 години)

**Тема:** Створення віконних програм у Windows.

**Мета:** Ознайомитися із структурою та етапами створення віконних програм для Windows

### Короткі теоретичні відомості

Кожна Windows-програма має два суттєвих компоненти: функцію `WinMain()`, яка ініціалізує вікно, і функцію `WindowProc()`, яка обслуговує повідомлення Windows. Ця структура становить основу всіх програм Windows.

Функція `WinMain()` викликається Windows при запуску кожної програми і здійснює необхідну ініціалізацію і налаштування вікна програми. Вона також містить цикл повідомлень для одержання повідомлень із черги повідомлень програми.

Функція `WindowProc()`, яка інколи має ім'я `WndProc()` або інше, викликається операційною системою кожного разу, коли повідомлення має передаватися вікну вашої програми. Зазвичай кожне вікно програми має свою функцію `WindowProc()`.

У кожній віконній програмі можна виділити чотири суттєвих компоненти:

1. Реєстрація класу вікна.
2. Створення вікна.
3. Цикл повідомлень.
4. Процедура вікна (віконна функція).

Перших три компоненти складають головну функцію `WinMain()` а останній є віконною процедурою `WindowProc()`.

Розглянемо усі компоненти віконної програми.

### Функція WinMain

Функція `WinMain` це точка входу до програми Windows. Вона ініціалізує програму, відображує вікно програми на екрані і запроваджує основний цикл повідомлень.

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                  LPSTR lpCmdLine, int nCmdShow );
```

*hInstance* дескриптор примірника. Це 32-розрядне число, що ідентифікує примірник нашої програми в середовищі ОС. Цей номер надає Windows, коли програма запускається на виконання.

*hPrevInstance* дескриптор попереднього примірника, завжди NULL. Це спадок від 16-розрядної Windows.

*lpCmdLine* параметри командного рядка (програми Windows все ще можуть бути запуснені з командного рядка).

*nCmdShow* визначає, як вікно відображуватиметься. (мінімізоване, максимізоване або приховане).

Функція `WinMain` завершується, коли отримує повідомлення `WM_QUIT`.

### Реєстрація класу вікна

Перед створенням вікна, слід зареєструвати у Windows його клас. Для цього заповнюють структуру `WNDCLASSEX` і викликають функцію реєстрації класу вікна `RegisterClassEx()`.

```
typedef struct _WNDCLASSEX {
    UINT    cbSize;           // розмір структури
    UINT    style;           // стиль класу
    WNDPROC lpfnWndProc;     // вказівник на віконну функцію
    int     cbClsExtra;      // додаткові байти класу
    int     cbWndExtra;      // додаткові байти примірника вікна
```

```

HANDLE hInstance;           // дескриптор примірника
HICON hIcon;                // дескриптор іконки
HCURSOR hCursor;           // дескриптор курсору
HBRUSH hbrBackground;      // дескриптор пензля фону
LPCTSTR lpszMenuName;       // вказівник на ім'я меню
LPCTSTR lpszClassName;     // вказівник на ім'я класу
HICON hIconSm;              // мала іконка вікна
} WNDCLASSEX;

```

Створюємо змінну WNDCLASSEX wc і заповнюємо структуру WNDCLASSEX:

```

wc.cbSize      = sizeof(WNDCLASSEX);
wc.style       = CS_HREDRAW | CS_VREDRAW;
wc.lpfnWndProc = WindowProc;
wc.cbClsExtra  = 0;
wc.cbWndExtra  = 0;
wc.hInstance   = hInstance;
wc.hIcon       = LoadIcon(0, IDI_APPLICATION);
wc.hCursor     = LoadCursor(0, IDC_ARROW);
wc.hbrBackground = GetStockObject(GRAY_BRUSH);
wc.lpszMenuName = 0;
wc.lpszClassName = "MyWindClass";
wc.hIconSm     = 0;

```

*Примітка.* Для вибору кольору фону вікна використовують кілька способів.

```

wc.hbrBackground = (HBRUSH) (COLOR_WINDOW+1);
wc.hbrBackground = (HBRUSH) GetStockObject(GRAY_BRUSH);
wc.hbrBackground = CreateSolidBrush(RGB(255,255,220));

```

Є також макрос **GetStockBrush (i)**, описаний в заголовковому файлі windowsx.h.

Функція *реєстрації класу* вікна:

```

if(!RegisterClassEx(&wc))
{
    MessageBox(NULL, "Window Registration Failed!", "Error!",
        MB_ICONEXCLAMATION | MB_OK);
    return 0;
};

```

## Створення вікна

Для створення вікна використовують функцію **CreateWindow()**, яка повертає дескриптор створеного вікна. Функція використовує дані із структури **WNDCLASSEX** через вказівник на ім'я зареєстрованого класу lpClassName.

```

HWND CreateWindow(
    LPCTSTR lpClassName,           // вказівник на ім'я зареєстрованого класу
    LPCTSTR lpWindowName,         // вказівник на ім'я вікна
    DWORD dwStyle,                // стиль вікна
    int x,                        // горизонтальна та
    int y,                        // вертикальна позиція вікна
    int nWidth,                   // ширина вікна
    int nHeight,                  // висота вікна
    HWND hWndParent,              // дескриптор батьківського вікна (або власника)
    HMENU hMenu,                  // дескриптор меню або ідентифікатор дочірнього вікна
    HANDLE hInstance,             // дескриптор примірника програми
    LPVOID lpParam                 // вказівник на дату створення вікна
);

```

Наприклад:

```
hWnd = CreateWindow(szClassName, "My First Window", WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, NULL, NULL, hInstance, NULL);
```

Для відображення вікна на екрані комп'ютера слід викликати функцію ShowWindow:

```
BOOL ShowWindow(hWnd, nCmdShow);
```

Перший параметр функції ідентифікує вікно і є дескриптором, поверненим функцією CreateWindow(), а другий – значення nCmdShow, яке передавалося WinMain() і вказує, як вікно з'являється на екрані. Приклади nCmdShow: SW\_HIDE, SW\_RESTORE, SW\_SHOWNORMAL. Можна просто передати значення, одержане від WinMain.

Після цього слід викликати функцію, яка повинна перемалювати (оновити) клієнтську область вікна:

```
BOOL UpdateWindow(HWND hWnd);
```

### *Цикл повідомлень*

В кінці головної функції WinMain() створюється цикл повідомлень, призначення якого – діставати повідомлення із черги повідомлень і направляти їх у віконну процедуру для обробки.

Цей цикл зазвичай має вигляд:

```
while (GetMessage (&msg, NULL, 0, 0)) // дістати повідомлення
{
    TranslateMessage (&msg); // транслювати повідомлення
    DispatchMessage (&msg); // відправити повідомлення
}
```

Цикл виконується весь час роботи програми до одержання повідомлення WM\_QUIT, після чого цикл припиняється і програма завершує свою роботу.

### **Віконна функція**

Весь код, який визначає поведінку програми, включається у віконну функцію (процедуру обробки повідомлень). Це функція **WindowProc()**, на яку посилається **WinMain()** через структуру **WNDCLASSEX**. Windows звертається до цієї функції кожного разу, коли надходить повідомлення для основного вікна програми.

Функція **WindowProc()**, аналізує потік повідомлень із черги, відбирає повідомлення, на які вона повинна відреагувати, і викликає відповідні функції (ділянки коду), відповідальні за обробку тих чи інших подій (обробники подій).

Заголовок віконної функції:

```
LRESULT CALLBACK WindowProc(
    HWND hWnd, // дескриптор вікна
    UINT uMsg, // ідентифікатор повідомлення
    WPARAM wParam, // перший параметр повідомлення
    LPARAM lParam // другий параметр повідомлення
);
```

Структура віконної функції:

```
LRESULT CALLBACK WindowProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch(msg)
    {
        case WM_CLOSE:
            DestroyWindow(hWnd);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
    }
}
```

```

        break;
    default:
        return DefWindowProc (hwnd, msg, wParam, lParam);
    }
    return 0;
}

```

### **Обробка подій у віконній функції**

Для обробки подій у віконній функції `WindowProc` використовують оператор **switch**, що являє собою список блоків **case <мітка>**, кожна із яких містить оператори для обробки окремих повідомлень.

#### **Макрос `HANDLE_MSG`**

Замість конструкції **case <мітка>** часто використовують макрос **HANDLE\_MSG**, описаний у заголовковому файлі **windows.h**: `HANDLE_MSG (hwnd, message, fn)`.

```

HANDLE_MSG (hwnd, WM_PAINT, OnPaint);           // case WM_PAINT: OnPaint ();

```

Цей макрос дає можливість спростити функцію `WindowProc` і винести усі обробники подій в окремі функції. Оператор **switch** набуває вигляду:

```

switch (msg) {
    HANDLE_MSG (hwnd, WM_PAINT, OnPaint);
    HANDLE_MSG (hwnd, WM_DESTROY, OnDestroy);
    default:
        return (DefWindowProc (hwnd, msg, wParam, lParam));
}

```

Крім цього додається кілька функцій, які є обробниками подій. Наприклад:

```

void OnPaint (HWND hwnd) {
    char szText [] = "Рядок для виводу";
    PAINTSTRUCT ps;                               // структура для малювання
    HDC hdc = BeginPaint (hwnd, &ps);             // одержання контексту пристрою
    TextOut (hdc, 5, 30, szText, strlen (szText)); // виведення рядка тексту
    EndPaint (hwnd, &ps);                          // звільнення контексту пристрою
}

```

У цьому обробнику використовується функція `TextOut`, призначена для виведення тексту в робочій області вікна.

```

TextOut (hdc, x, y, psText, iLength);

```

Тут `hdc` – дескриптор контексту, який одержується функцією **BeginPaint** і звільняється функцією **EndPaint**.

### **Повідомлення від миші**

Повідомлення від миші (наприклад, `WM_MOUSEMOVE`, `WM_LBUTTONDOWN` та ін.) мають параметри `wParam` і `lParam`, які містять таку інформацію:

```

fwKeys = wParam;           // прапорці клавіш
xPos = LOWORD (lParam);    // горизонтальна позиція курсору
yPos = HIWORD (lParam);    // вертикальна позиція курсору

```

Для розшифровки координат миші використовують змінну типу `POINT`:

```

POINT pt;

```












Обробник повідомлення миші може мати вигляд:

```
case WM_LBUTTONDOWN:
{
    pt.x = LOWORD(lParam);
    pt.y = HIWORD(lParam);
    wsprintf(str, "Co-ordinates are: X=%i and Y=%i", pt.x, pt.y);
    InvalidateRect(hWnd, NULL, TRUE);
}
```

Цей обробник розшифровує координати миші і заносить їх у рядковий масив str.

Функція InvalidateRect оголошує робочу область вікна (або її частину) недійсною (неправильною), що призводить до її перемальовування.

### Завдання для виконання роботи

1. Створити віконну програму для Windows.
2. Замінити іконку програми на одну із наступних: , , , .
3. Замінити курсор вікна на , , ,  або інший.
4. Змінити колір фону робочої області на , ,  або інший (кольори фону COLOR\_APPWORKSPACE, COLOR\_BTNshadow, COLOR\_BTNHIGHLIGHT та ін.).
5. Змінити спосіб відображення вікна на екрані (nCmdShow = SW\_SHOWNORMAL, SW\_MINIMIZE, SW\_MAXIMIZE, SW\_SHOWNA, SW\_HIDE та ін.).
6. Доповнити віконну функцію програми новими обробниками повідомлень.
  - a. обробити натискання лівої кнопки миші і вивести в робочій області повідомлення про координати миші.
  - b. обробити натискання правої кнопки миші і вивести діалогове вікно із текстовим повідомленням про цю подію.
  - c. оформити обробники повідомлень за допомогою макросу HANDLE\_MSG.

### Контрольні питання

1. Яка структура Windows-програми із графічним інтерфейсом?
2. Яку назву має головна функція віконної програми і які функції вона виконує?
3. Що означає термін «реєстрація класу вікна»?
4. Які дії необхідно виконати у програмі для того, щоб вікно відобразилося на екрані монітора?
5. Що таке цикл повідомлень і як він організується у програмі?
6. Яким чином повідомлення про події в системі потрапляють у програму і як вони обробляються?
7. Яку назву має функція вікна і для чого вона призначена?
8. Де у віконній програмі розташовуються обробники подій?
9. Яким чином можна налаштувати параметри вікна програми (задати розміри, вибрати іконку, встановити курсор)?

### Завдання для самостійної роботи

1. Ознайомитися із системними повідомленнями, які надходять від миші та клавіатури.