Лабораторна робота № 3
(4 години)

Тема: Використання ресурсів у програмах для Windows.

Мета: Ознайомитися із засобами створення та використання ресурсів у програмах для Windows

Короткі теоретичні відомості

Ресурси це заздалегідь визначені дані у програмах Windows, які знаходяться у виконуваному файлі у двійковому вигляді і можуть використовуватися при роботі програми. Ці дані не завантажуються в пам'ять при запуску програми. Windows забезпечує функції, що явно завантажують ресурси програми в пам'ять таким чином, що вони можуть використовуватися програмою. Прикладами таких функцій є LoadIcon і LoadCursor, які використовуються при ініціалізації структури класу вікна програми.

До ресурсів програм Windows відносять такі типи ресурсів:

• значки	Icons
• курсори	Cursors
• бітові матриці	Bitmaps
• меню	Menus
• прискорювачі клавіатури	Keyboard accelerators
• діалогові вікна	Dialog boxes
• рядки символів	Character strings
• ресурси користувача	Custom resources

Кожний ресурс має свій опис – сценарій (script), який зберігається у файлі ресурсів із розширенням **.rc**. Наприклад:

```
MY_ICON          ICON      DISCARDABLE    "my_icon.ico"
MY_CURSOR        CURSOR    DISCARDABLE    "my_arrow.cur"
```

Файли ресурсів створюються за допомогою спеціальних редакторів ресурсів, або текстових редакторів.

Крім файлу ресурсів до проекту зазвичай включається окремий заголовковий файл **resource.h**. Цей файл містить визначення символічних констант ресурсів, що використовуються як у головній програмі, так і у файлі ресурсів. Наприклад:

```
#define MY_ICON          101
#define MY_CURSOR        104
```

Сучасні інтегровані середовища розробки програм автоматично формують файли ресурсів і заголовковий файл.

Компіляція файлу ресурсів здійснюється компілятором ресурсів, в результаті чого він перетворюється на об'єктний модуль із розширенням **.res**.

Піктограми

Більшість програм Microsoft Windows включають персональний значок (іконку або icon). Цей значок Windows відображує у лівому верхньому кутку заголовка вікна програми, в меню Пуск, на панелі задач внизу екрану, у папці або на ярлику робочого столу. У віконній програмі Windows цей значок використовується при реєстрації класу вікна:

```
wc.hIcon = LoadIcon (NULL, IDI_APPLICATION)
```

Є два варіанти:

1) програма використовує один із стандартних значків Windows

IDI_APPLICATION	значок програми
IDI_ASTERISK	інформаційний значок
IDI_EXCLAMATION	значок попередження
IDI_HAND	значок помилки
IDI_QUESTION	значок запитання

2) програма використовує свій власний значок

Власний значок будується за допомогою редактора ресурсів і зберігається у файлі із розширенням **.ico**. Наприклад у Microsoft Visual Studio виконується команда **Insert → Resource → Icon**. Найкраще мати два значки, розміром 32×32 та 16×16, кожний із яких створюється окремо.

Значок має ідентифікатор, наприклад, IDI_ICON1, MY_ICON та ін.

Для включення значка у вікно програми використовують функції LoadIcon() та LoadImage(). Функція LoadIcon() завантажує значок із файлу програми, а функція LoadImage() – із окремого файлу з розширенням **.ICO**.

```
HICON LoadIcon(

  HINSTANCE hInstance,          // дескриптор примірника програми
  LPCTSTR lpIconName            // рядок з іменем значка або ідентифікатор ресурсу
);
```

При використанні стандартних значків Windows параметр hInstance задають NULL. Другий параметр *lpIconName* вказує на рядок, який містить ім'я ресурсу-значка. Приклади використання функцій:

```
wc.hIcon = LoadIcon (NULL, IDI_APPLICATION) ;
wc.hIcon = LoadIcon (hInstance, MAKEINTRESOURCE (IDI_ICON)) ;
wc.hIcon = LoadIcon (hInstance, MAKEINTRESOURCE (101)) ;
wc.hIcon = LoadIcon (hInstance, "MyIcon") ;

wc.hIcon = LoadImage(NULL, "alt_icon.ico", IMAGE_ICON, 32, 32,
                     LR_LOADFROMFILE);
wc.hIconSm = LoadImage(NULL, " alt_icon.ico", IMAGE_ICON, 16, 16,
                     LR_LOADFROMFILE);
```

Курсори

Програма може використовувати власні курсори, хоча курсори, які поставляє Windows, є цілком адекватними для більшості задач. Використання замовних курсорів миші у програмі подібне до використання власних значків. Замовні курсори загалом монохромні розміром 32×32 пікселів. Курсори створюються за допомогою редактора ресурсів: **Insert → Resource → Cursor**. Після створення курсору слід визначити його "гарячу точку" (HotSpot) – точку, яка визначає точну позицію курсору.

Необхідний курсор можна встановити при реєстрації класу вікна за допомогою функції LoadCursor():

```
HCURSOR LoadCursor(

  HINSTANCE hInstance,          // дескриптор примірника програми
  LPCTSTR lpCursorName          // рядок імені або ідентифікатор ресурсу курсору
);
```

Під час роботи програми курсори можна змінювати за допомогою функції **SetCursor()**.

```
HCURSOR SetCursor(

  HCURSOR hCursor                       // дескриптор курсору
);
```

Для використання цієї функції потрібно мати дескриптори курсорів, які можна одержати за допомогою функцій **CreateCursor()** або **LoadCursor()**.

Меню

Більшість програм Windows мають "головне меню" (main menu) або "Меню верхнього рівня". Елементи головного меню зазвичай викликають "випадаючі" (спливаючі або pop-up) меню.

Меню складаються із пунктів. Є два типи пунктів: пункти-команди і пункти-підменю (Submenu). Можна також визначати численні вкладення підменю: тобто, пункт спливаючого меню може викликати інше спливаюче меню.

Якщо вибраний пункт-команда, то Windows посилає повідомлення WM_COMMAND, а якщо вибраний пункт-підменю, то виводиться наступне спливаюче меню.

Пункти меню можуть бути ***дозволені*** (enabled), ***заборонені*** (disabled) або ***недоступні*** (grayed). Пункти меню можуть бути позначеними значком ✓ (checked). В цьому випадку вони використовуються як перемикачі (check box).

Щоб створити опис чи шаблон меню у Developer Studio, вибирають пункт меню ***Insert → Resource → Menu***. До макета меню, який з'являється додають пункти і визначають їх властивості. У файл ресурсів при цьому заносяться рядки типу:

```
IDR_MAINMENU MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",                       IDM_FILE_OPEN
        MENUITEM "&Save",                       IDM_FILE_SAVE
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       IDM_APP_EXIT
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&Help...",                    IDM_APP_HELP
        MENUITEM "&About MenuDemo...",          IDM_APP_ABOUT
    END
END
```

Є кілька способів підключити меню до своєї програми.

.

1. Підключення меню до програми при реєстрації класу вікна:

```
wc.lpszMenuName = szMenuName;
```

2. Завантаження меню функцією *LoadMenu*. Функція *LoadMenu* повертає дескриптор меню. Наприклад:

```
hMenu = LoadMenu (hInstance, TEXT ("MyMenu")) ;
```

```
hMenu = LoadMenu (hInstance, MAKEINTRESOURCE (ID_MENU)) ;
```

Одержаний дескриптор використовується функцією *CreateWindow*, як її дев'ятий параметр:

```
hwnd = CreateWindow (TEXT ("MyClass"), TEXT ("Window Caption"),
                     WS_OVERLAPPEDWINDOW,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, hMenu, hInstance, NULL) ;
```

У цьому випадку меню, визначене дескриптором, перевантажує меню, задане класом вікна.

При виборі пунктів меню Windows посилає віконній функції повідомлення WM_COMMAND. Це повідомлення вказує, що користувач вибрав дозволений пункт із

меню вікна програми. Молодше слово параметра *wParam* передає ідентифікатор вибраного пункту меню.

Аналізуючи LOWORD (*wParam*) у WndProc(), можна визначити ідентифікатор пункту меню (ID):

```
case WM_COMMAND:
   switch(LOWORD(wParam))
   {
     case IDM_FILE_OPEN:        // код для File Open
     case IDM_FILE_SAVE:        // код для File Save

     case ID_FILE_EXIT: PostMessage(hwnd, WM_CLOSE, 0, 0);
     break;

     case IDM_APP_HELP:         // код для Help

     case IDM_APP_ABOUT: MessageBox (...);
     break;
   }
break;
```

Меню можна змінювати під час роботи програми за допомогою функцій:

AppendMenu	додає пункт в кінець меню
InsertMenu, InsertMenuItem	вставляє новий пункт меню
DeleteMenu	видаляє пункт меню (також знищує підменю)
RemoveMenu	видаляє пункт меню

Є ще цілий ряд функцій для зміни пунктів меню: SetMenuItemInfo, CheckMenuItem, CheckMenuRadioItem, EnableMenuItem, ModifyMenu.

Діалогові вікна

Діалогові вікна – це спеціальні вікна, призначені для діалогу з користувачем. Деякі операції по створенню вікна система робить автоматично (створення та ініціалізація елементів керування, керування передачею фокусу).

Діалогові вікна бувають або "модальні", або "немодальні." Коли програма виводить модальне діалогове вікно, користувач зобов'язаний прийняти певне рішення і закрити діалогове вікно передбаченим чином (зазвичай натискаючи одну із кнопок OK, Cancel, Retry та ін.). Немодальні вікна можна покинути, не закриваючи їх.

Перший крок по створенню діалогового вікна – це створення сценарію діалогу. Деталі залежать від компілятора або IDE. У Developer Studio, для прикладу, вибирають пункт меню *Вставка* → *Ресурс* → *Діалог* (**Insert → Resource → Dialog**) і створюють діалог візуально, додаючи до заготовки необхідні компоненти. Файл ресурсу створюється автоматично. Для прикладу, файл ресурсу може містити такий текст сценарію діалогу:

```
IDD_ABOUT DIALOG DISCARDABLE 0, 0, 239, 66
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "My About Box"
FONT 8, "MS Sans Serif"
BEGIN
     DEFPUSHBUTTON "&OK",IDOK,174,18,50,14
     PUSHBUTTON "&Cancel",IDCANCEL,174,35,50,14
     GROUPBOX "About this program...",IDC_STATIC,7,7,225,52
     CTEXT "An example program showing how to use Dialog
Boxes\r\n\r\nby theForger",IDC_STATIC,16,18,144,33
END
```

Коли програмі потрібно а при виборі відповідного пункту меню одержує повідомлення WM_COMMAND із молодшим словом *wParam* IDM_APP_ABOUT, вона повинна викликати функцію *DialogBox* для відображення діалогового вікна:

```
int DialogBox(
  HINSTANCE hInstance,      // дескриптор примірника програми
  LPCTSTR lpTemplate,       // ідентифікатор шаблону діалогового вікна
  HWND hWndParent,          // дескриптор батьківського вікна
  DLGPROC lpDialogFunc      // вказівник процедури діалогового вікна
 );
```

Функція повертає числове значення, яке формується після натискання однієї із кнопок діалогового вікна. Найчастіше це ідентифікатор натиснутої кнопки. Наприклад:

```
case IDM_APP_ABOUT:
{
   int ret = DialogBox(GetModuleHandle(NULL),
                     MAKEINTRESOURCE(IDD_ABOUT), hwnd, AboutDlgProc);
   if(ret == IDOK){
        MessageBox(hwnd, "Dialog exited with IDOK.", "Notice",
                MB_OK | MB_ICONINFORMATION);
   }
   else if(ret == IDCANCEL){
        MessageBox(hwnd, "Dialog exited with IDCANCEL.", "Notice",
                MB_OK | MB_ICONINFORMATION);
   }
   else if(ret == -1){
        MessageBox(hwnd, "Dialog failed!", "Error",
                MB_OK | MB_ICONINFORMATION);
   }
}
break;
```

Діалогове вікно повинно мати свою віконну процедуру. Вона приймає такі ж параметри як і звичайна віконна функція, але має певні відмінності. Наприклад:

```
BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message,
                            WPARAM wParam, LPARAM lParam)
{
     switch (message)
     {
     case WM_INITDIALOG :
          return TRUE ;

     case WM_COMMAND :
          switch (LOWORD (wParam))
          {
          case IDOK :
          case IDCANCEL :
               EndDialog (hDlg, 0) ;
               return TRUE ;
          }
          break ;
     }
     return FALSE ;
}
```

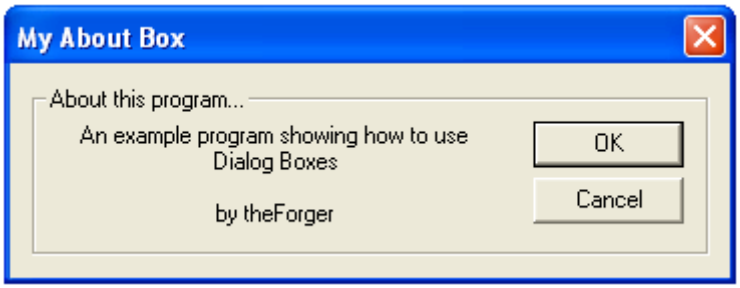
Процедура діалогового вікна має тип BOOL: якщо повідомлення оброблене – повертається TRUE (nonzero), якщо ні – FALSE (0).

Для ініціалізації параметрів процедура діалогового вікна оброблює повідомлення WM_INITDIALOG.

Основним повідомленням, яке потрібно обробити є WM_COMMAND. Це повідомлення посилає кнопка батьківському вікну при її натисканні мишею або клавішею Space-bar. Ідентифікатор кнопки (наприклад, IDOK) знаходиться у молодшому слові *wParam*. Для цього повідомлення процедура діалогового вікна викликає функцію *EndDialog*, яка вказує Windows знищити діалогове вікно.

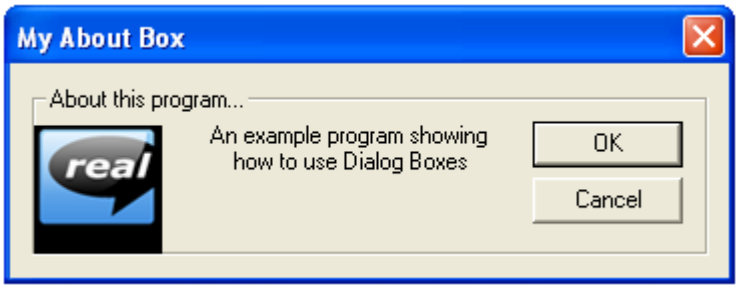
Завдання для виконання

1. Створити власний значок програми і оформити його як ресурс.
2. Створити власний курсор програми і оформити його як ресурс.
3. Створити власне меню на 2-3 пункти із спливаючими меню на 3-4 пункти. Оформити меню як ресурс. Відредагувати файл ресурсу в текстовому редакторі, додавши новий пункт меню (на 2-3 підпункти).
4. Написати обробники кількох пунктів меню, які виводять повідомлення про обраний пункт.
5. Створити власне діалогове вікно типу:



У процедурі діалогового вікна обробити події від натискання кнопок OK і Cancel. У віконній процедурі програми вивести результат натискання кнопок в діалоговому вікні.

6. Додати в діалогове вікно картинку. Для цього спочатку в проект додають ресурс типу Bitmap. Потім в діалогове вікно додають елемент Picture із панелі інструментів. В кінці настроюють властивості (Properties) елемента Picture. Задають ідентифікатор, тип картинки Bitmap і вибирають ідентифікатор ресурсу IDB_Bitmap1.



7. Використати функцію ShellAbout для виведення стандартного діалогового вікна Windows "О программе".
8. Створити пункт меню, який виводить діалогове вікно MessageBox із кількома кнопками (Yes, No і Cancel). Проаналізувати які значення повертає функція MessageBox і вивести повідомлення про те, яка кнопка була натиснута.

Контрольні питання

1. Що таке ресурси і яке значення вони мають для Windows-програм?
2. Яким чином можна створити ресурси і які файли можуть їх містити?
3. Які типи ресурсів ви знаєте і які дані вони містять?
4. Як можна використати у програмах власні іконки або курсори? Як можна змінити курсор під час роботи програми?
5. Як створюються меню програми? Які типи меню ви знаєте?
6. Що відбувається при виборі пункту меню під час роботи програми?
7. Як написати обробник меню?
8. Як створюються діалогові вікна? Як викликати діалогове вікно?
9. Як обробити натискання кнопок діалогового вікна?
10. Як дізнатися про те, яка кнопка діалогового вікна була натиснута?