#include "framework.h" // Включаем заголовочный файл framework.h, предположительно содержащий общие библиотеки и определения.

#include "WinAPI.h" // Включаем заголовочный файл WinAPI.h, где, вероятно, определены функции и структуры для работы с WinAPI.

#define MAX\_LOADSTRING 100 // Макрос, устанавливающий максимальную длину строки (часто используется для буферов строк).

// Глобальные переменные:

**HINSTANCE** hInst; // Текущий экземпляр

**WCHAR** szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

**WCHAR** szWindowClass[MAX\_LOADSTRING]; // Имя класса главного окна

// Отправить объявления функций, включенных в этот модуль кода:

**ATOM** MyRegisterClass(**HINSTANCE** hInstance); // Прототип функции для регистрации класса окна. Принимает дескриптор экземпляра приложения (hInstance).

**BOOL** InitInstance(**HINSTANCE**, **int**); // Прототип функции для инициализации экземпляра и создания главного окна. Принимает дескриптор экземпляра и флаг отображения окна.

**LRESULT** **CALLBACK** WndProc(**HWND**, **UINT**, **WPARAM**, **LPARAM**); // Прототип функции обработки сообщений главного окна. Принимает дескриптор окна и параметры сообщения.

**INT\_PTR** **CALLBACK** About(**HWND**, **UINT**, **WPARAM**, **LPARAM**); // Прототип функции обработки сообщений для окна "О программе". Принимает дескриптор окна и параметры сообщения.

// Точка входа приложения

**int** APIENTRY wWinMain(\_In\_ **HINSTANCE** hInstance,

\_In\_opt\_ **HINSTANCE** hPrevInstance,

\_In\_ **LPWSTR** lpCmdLine,

\_In\_ **int** nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance); // Подавление предупреждения об неиспользуемом параметре (не используется в этой функции).

UNREFERENCED\_PARAMETER(lpCmdLine); // Подавление предупреждения об неиспользуемом параметре (не используется в этой функции).

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING); // Загрузка строки заголовка приложения из ресурсов

LoadStringW(hInstance, IDC\_WINAPI, szWindowClass, MAX\_LOADSTRING); // Загрузка имени класса окна из ресурсов

MyRegisterClass(hInstance); // Регистрация класса окна

// Выполнить инициализацию приложения:

**if** (!InitInstance(hInstance, nCmdShow)) // Создание и отображение главного окна

{

**return** FALSE; // В случае ошибки инициализации возвращаем FALSE.

}

**HACCEL** hAccelTable = **LoadAccelerators**(hInstance, **MAKEINTRESOURCE**(IDC\_WINAPI)); // Загрузка таблицы акселераторов для обработки хоткеев

**MSG** msg;

// Цикл основного сообщения:

**while** (**GetMessage**(&msg, **nullptr**, 0, 0))

{

**if** (!**TranslateAccelerator**(msg.hwnd, hAccelTable, &msg)) // Обработка акселераторов

{

**TranslateMessage**(&msg); // Преобразование сообщений клавиш в символы

**DispatchMessage**(&msg); // Передача сообщения в функцию окна для обработки

}

}

**return** (**int**)msg.wParam; // Возвращаем код выхода из приложения после завершения цикла сообщений

}

// Функция регистрации класса окна

**ATOM** MyRegisterClass(**HINSTANCE** hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = **sizeof**(**WNDCLASSEX**); // Размер структуры WNDCLASSEX

wcex.style = CS\_HREDRAW | CS\_VREDRAW; // Стиль класса окна: перерисовка при изменении горизонтального или вертикального размера

wcex.lpfnWndProc = WndProc; // Указатель на функцию обработки сообщений окна

wcex.cbClsExtra = 0; // Дополнительные данные для класса окна

wcex.cbWndExtra = 0; // Дополнительные данные для экземпляра окна

wcex.hInstance = hInstance; // Дескриптор экземпляра приложения

wcex.hIcon = **LoadIcon**(hInstance, **MAKEINTRESOURCE**(IDI\_WINAPI)); // Загрузка иконки приложения

wcex.hCursor = **LoadCursor**(**nullptr**, IDC\_ARROW); // Загрузка стандартного курсора

wcex.hbrBackground = (**HBRUSH**)(COLOR\_WINDOW + 1); // Кисть для фона окна (стандартный цвет окна + 1)

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_WINAPI); // Имя ресурса для меню

wcex.lpszClassName = szWindowClass; // Имя класса окна

wcex.hIconSm = **LoadIcon**(wcex.hInstance, **MAKEINTRESOURCE**(IDI\_SMALL)); // Загрузка маленькой иконки приложения

**return** RegisterClassExW(&wcex); // Регистрация класса окна и возврат дескриптора класса

}

// Функция инициализации экземпляра и создания главного окна

**BOOL** InitInstance(**HINSTANCE** hInstance, **int** nCmdShow)

{

hInst = hInstance; // Сохранить текущий экземпляр в глобальной переменной

// Создание главного окна

**HWND** hWnd = CreateWindowW(szWindowClass, szTitle, **WS\_OVERLAPPEDWINDOW**,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, **nullptr**, **nullptr**, hInstance, **nullptr**);

**if** (!hWnd)

{

**return** FALSE; // В случае ошибки при создании окна возвращаем FALSE

}

**ShowWindow**(hWnd, nCmdShow); // Отображение окна с заданным режимом отображения

**UpdateWindow**(hWnd); // Обновление окна

**return** TRUE; // Успешная инициализация

}

// Функция обработки сообщений главного окна

**LRESULT** **CALLBACK** WndProc(**HWND** hWnd, **UINT** message, **WPARAM** wParam, **LPARAM** lParam)

{

**switch** (message)

{

**case WM\_COMMAND**:

{

**int** wmId = **LOWORD**(wParam); // Получаем идентификатор команды из параметра wParam

// Разбор выбора в меню:

**switch** (wmId)

{

**case IDM\_ABOUT**:

// Отображение диалогового окна "О программе"

**DialogBox**(hInst, **MAKEINTRESOURCE**(IDD\_ABOUTBOX), hWnd, About);

**break**;

**case IDM\_EXIT**:

// Закрытие главного окна

**DestroyWindow**(hWnd);

**break**;

**default**:

// Если идентификатор команды не соответствует вышеописанным, передаем управление стандартной процедуре окна

**return** **DefWindowProc**(hWnd, message, wParam, lParam);

}

}

**break**;

**case WM\_PAINT**:

{

**PAINTSTRUCT** ps; // Структура для хранения информации о рисовании

**HDC** hdc = **BeginPaint**(hWnd, &ps); // Начало процесса рисования

**EndPaint**(hWnd, &ps); // Завершение процесса рисования

}

**break**;

**case WM\_DESTROY**:

// Постановка в очередь сообщения о завершении приложения

**PostQuitMessage**(0);

**break**;

**default**:

// Если сообщение не было обработано выше, передаем управление стандартной процедуре окна

**return** **DefWindowProc**(hWnd, message, wParam, lParam);

}

**return** 0;

}

// Обработчик сообщений для окна "О программе".

**INT\_PTR** **CALLBACK** About(**HWND** hDlg, **UINT** message, **WPARAM** wParam, **LPARAM** lParam)

{

UNREFERENCED\_PARAMETER(lParam);

**switch** (message)

{

**case WM\_INITDIALOG**:

// Обработка сообщения инициализации диалогового окна.

// Возвращаем TRUE, чтобы сообщить, что мы обработали это сообщение.

**return** (**INT\_PTR**)TRUE;

**case WM\_COMMAND**:

// Обработка сообщений от элементов управления в диалоговом окне.

// В данном случае, если нажата кнопка OK или Cancel, закрываем диалоговое окно.

**if** (**LOWORD**(wParam) == IDOK || **LOWORD**(wParam) == IDCANCEL)

{

**EndDialog**(hDlg, **LOWORD**(wParam)); // Завершение диалогового окна с кодом, соответствующим кнопке OK или Cancel.

**return** (**INT\_PTR**)TRUE; // Возвращаем TRUE, чтобы сообщить, что мы обработали это сообщение.

}

**break**;

}

**return** (**INT\_PTR**)FALSE; // Возвращаем FALSE, чтобы сообщить, что мы не обработали данное сообщение.

}

