БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий

и робототехники

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

О Т Ч Ё Т

о лабораторной работе № 2

Дисциплина

«Системное программирование»

Тема

«Простейшее приложение на базе WIN32/64 API»

Выполнил: студент гр. 10702217 Храмков Д. С.

Проверил: Разорёнов Н. А.

Минск 2019

***Лабораторная работа № 2***

**ДИАЛОГОВЫЕ ОКНА**

**Цель работы:** изучить вопросы проектирования и создания модальных и немодальных диалоговых панелей на базе Win32/64 API.

**Изучаемые вопросы**

1. Назначение и классификация диалоговых окон.
2. Создание и отображение диалогового окна на экране.
3. Шаблон, ресурсы диалогового окна.
4. Задание размеров диалогового окна.
5. Управляющие элементы диалогового окна.
6. Обработка сообщений от элементов управления диалогового окна.
7. Сообщения работы с модальными и немодальными диалоговыми окнами.
8. Обмен информации и проверка корректности вводимых данных.
9. Элементы управления:
   1. Поле для ввода. Кнопка. Список. Комбинированный список. Слайдер
   2. Сообщения к элементам управления SendMessage, SendDlgItemMessage.
   3. Обработка сообщений от элементов диалогового окна.
   4. Корректность вводимых данных.
10. Окна сообщений.
11. Стандартные диалоги.

***Постановка задачи***

Создать приложение (рис. 2.1), использующее модальный, системный модальный, немодальный и стандартный диалоги и элементы управления типа кнопка, меню, список, движок, поле для ввода.

1. **Архитектура WIN32/64 программы.**

Архитектура приложения представляет из себя следующую схему:

* Функция запуска цикла обработки сообщений wWinMain();
* Функция регистрации окон приложения MyRegisterClass();
* Функция создания дескрипторов объектов InitInstance();
* Функция обработки поступающих сообщений WndProc().

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow);

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow);

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

1. **Цикл обработки сообщений (GetMessage).**

Цикл обработки сообщений состоит из управляющей структуры while, которая при каждом проходе цикла получает очередное сообщение из очереди, посредством функции GetMessage(), после чего сравнивает нажатия клавиши с таблицей клавиш-ускорителей при помощи TranslateAccelerator(), и, если она находит соответствие, переводит нажатия клавиш в сообщения WM\_COMMAND (или WM\_SYSCOMMAND) и отправляет сообщение оконной процедуре. Затем переводит все сообщения от виртуальных клавиш в символьное сообщение с помощью функции TranslateMessage(), и после этого отсылает полученное сообщение на обработку оконной процедуре, используя функцию DispatchMessage().

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

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

1. **Оконная функция, назначение, параметры, обработка сообщений.**

Оконная функция вызывается операционной системой Windows, когда программе нужно послать сообщение. В качестве параметров оконная функция получает сообщение, посылаемое Windows.

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

Принимаемые параметры: hWnd – идентификатор окна, message – код текущего сообщения, wParam и lParam – дополнительная информация о сообщении.

Данная функция обрабатывает сообщения в соответствии с её описанием.

1. **Структура оконного класса, поля, их назначение, инициализация.**

Структура оконного класса представлена в приложении следующим образом:

typedef struct tagWNDCLASS {

UINT style; //описывает стиль окна

WNDPROC lpfnWndProc; //задаёт функцию, обрабатывающую процессы окна

int cbClsExtra; //устанавливает размер дополнительных байтов для класса

int cbWndExtra; //устанавливает размер дополнительных байтов для окна

HINSTANCE hInstance; //дескриптор экземпляра приложения

HICON hIcon; //задаёт иконку для окна

HCURSOR hCursor; //задаёт вид курсора в данном окне

HBRUSH hbrBackground; //устанавливает цвет фона

LPCSTR lpszMenuName; //задаёт имя меню

LPCSTR lpszClassName; //задаёт имя класса

} WNDCLASS;

1. **Классы окон, регистрация оконного класса.**

Регистрация класса окна осуществляется функцией ATOM MyRegisterClass(HINSTANCE hInstance), в котором при помощи структуры WNDCLASSEXW задаются параметры окна (размер, стиль, связанная функция, …).

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB1));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB1);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

1. **Создание экземпляра оконного класса.**

Для создания окна в Win32 API нужно выполнить команду CreateWindow. Если функция завершена успешно, то возвращаемое значение является дескриптором нового окна.

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

20, 20, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

1. **Отображение окна на экране.**

Чтобы отобразить окно на экране используется функция ShowWindow(). Для отрисовки обновлений в окне выполняется команда UpdateWindow, которая вызывает событие WM\_PAIN.

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

1. **Управление положением и размерами окна.**

Положение и размер окна можно задать при создании. Также, для изменения параметров окна во время выполнения программы используется SetWindowPos.

window2 = CreateWindow(L"UserWindow2", L"Window2", dwWindowStyle,

0, dy / 3, dx / 2, dy / 3, hWnd, nullptr, hInst, nullptr);

SetWindowPos(window2, 0, 0, dy / 3, dx / 2, dy / 3, SWP\_SHOWWINDOW);

1. **Обновление оконной области на экране.**

Обновление окна происходит в WM\_PAINT.

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

EndPaint(hWnd, &ps);

}

1. **Установки фона.**

Фон окна задаётся при регистрации окна класса.

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

1. **Курсор приложения. Типы курсоров, их прикрепление к приложению.**

Курсор задаётся при регистрации класса окна функцией LoadCursor.

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);