МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

В.Г. Рудалев,

Д.А. Щеглаков

ИНТЕРФЕЙС API WIN32 НА ПРИМЕРАХ

Методическое пособие для вузов

Издательско-полиграфический центр
Воронежского государственного университета
2010

Содержание

Введение

Windows API (Application Programming Interfaces) представляет собой набор базовых функций, структур, сообщений, макросов и интерфейсов программирования приложений для операционных систем семейств Windows и Windows NT.

Все 32-разрядные Windows-платформы обладают общим набором API, получившим название Win32 API.

Работа через Windows API — это наиболее «близкий к системе» способ взаимодействия с ней прикладных программ.

По типу предоставляемых возможностей выделяются несколько групп наборов Windows API. Среди них:

– Window Management (управление окнами). Средства для создания пользовательского интерфейса и управления им.
– Window Controls (элементы управления). Предназначены для создания общих элементов пользовательского интерфейса. Их применение делает единообразным интерфейсы графической оболочки и остальных приложений, а также сокращает время разработки.
– Shell Features (функции оболочки). Предназначены для доступа к системным устройствам и ресурсам – дисководам, принтерам, файлам, сетевым ресурсам и т.д.

ђddђуеҮ ЬӖ dеdЁђҮзе d ѕаоѕоҹ ЭddђҮdүә

1.2. Сообщения и оконная функция

Взаимодействие оконного приложения с пользователем осуществляется посредством механизма обработки *сообщений*. Этот механизм определяет реакцию приложения на то или иное действие пользователя или изменение среды в операционной системе. В процессе работы приложение получает большое количество сообщений от операционной системы. Их обработка происходит в одной конкретной процедуре (а точнее функции), называемой *оконной*. Такая схема функционирования во многом напоминает реакцию на события компонентов VCL. Отличие заключается в том, что у компонентов VCL для обработки каждого события назначается своя процедура-обработчик, в то время как оконная процедура предназначена для обработки всех сообщений.

Для работы с сообщениями в Delphi предусмотрены записи `TMsg` и `TMessage`

```
TMSG = record
    hwnd: HWND; //дескриптор окна, принимающего сообщение
    message: UINT; //идентификатор сообщения
    wParam: WPARAM; //параметры, содержимое которых
    lParam: LPARAM; //меняется в зависимости от типа сообщения
    time: DWORD; //время отправки сообщения
    pt: TPoint; //позиция курсора на экране
  end;
```

`TMsg` – «родная» структура для функций Windows API. В Delphi для хранения сообщений предусмотрена структура `TMessage`, содержащая часть значений из соответствующей структуры `TMsg`. В структуре `TMessage` меньше записей, чем в `TMsg`. Это связано с тем, что обработку остальных параметров сообщения Delphi берет на себя.

```
TMessage = record
    Msg: Cardinal; //идентификатор сообщения
    case Integer of
      0: (
        WParam: Longint;//параметры, содержимое которых
        LParam: Longint;//меняется в зависимости от типа
                        //сообщения
        Result: Longint);// результат выполнения
      1: (
        WParamLo: Word;// 16 - битные части соответствующих
        WParamHi: Word;// параметров wParam и lParam ( млад-
        LParamLo: Word;// шее и старшее слово)
        LParamHi: Word;
        ResultLo: Word;
        ResultHi: Word);
     End;
End;
```

Как следует из объявлений, значения параметров `WParam` и `LParam` зависят от вида (типа) сообщения. Тип сообщения соответствует событию, о котором информируется приложение (окно). Чтобы определить тип, используется параметр `message`. Через этот параметр передается константа-идентификатор, по значению которой оконная функция определяет тип сообщения (см. табл. 2).

Таблица 2

Константа	Значение	Тип сообщения
WM_ACTIVATE	$0006	Активация или деактивация окна
WM_CHAR	$0102	Клавиша клавиатуры нажата и отпущена
WM_CLOSE	$0010	Закрытие окна
WM_KEYDOWN	$0100	Нажата клавиша клавиатуры
WM_KEYUP	$0101	Отпущена клавиша клавиатуры
WM_LBUTTONDOWN	$0201	Нажата левая клавиша мыши
WM_MOUSEMOVE	$0200	Перемещение указателя мыши
WM_PAINT	$000F	Перерисовка клиентской области окна
WM_COMMAND	$0111	Выбор элемента меню либо от дочернего оконного элемента должно быть получено сообщение
WM_SETFONT	$0030	Изменение шрифта
WM_CLOSE	$0010	Закрытие окна
WM_DESTROY*	$0002	Уничтожение окна

В таблице приводятся лишь некоторые константы-идентификаторы сообщений. Объявления всех таких констант находятся в модуле messages.pas.

1.3. Структура Windows-приложения

Рассмотрим пример создания простейшего однооконного приложения с использованием Win32 API.

В среде Delphi создадим новый проект (File→New→Application). Затем удалим из него все модули и формы: View→Project Manager, воспользовавшись пунктом Remove (рис. 1).

Открыв файл проекта (Project→View Source), отредактируем текст программы.

* После передачи приложению сообщения WM_DESTROY, его оконной функцией вызывается функция DestroyWindow, которая, в свою очередь, передает окнам приложения сообщение WM_CLOSE.

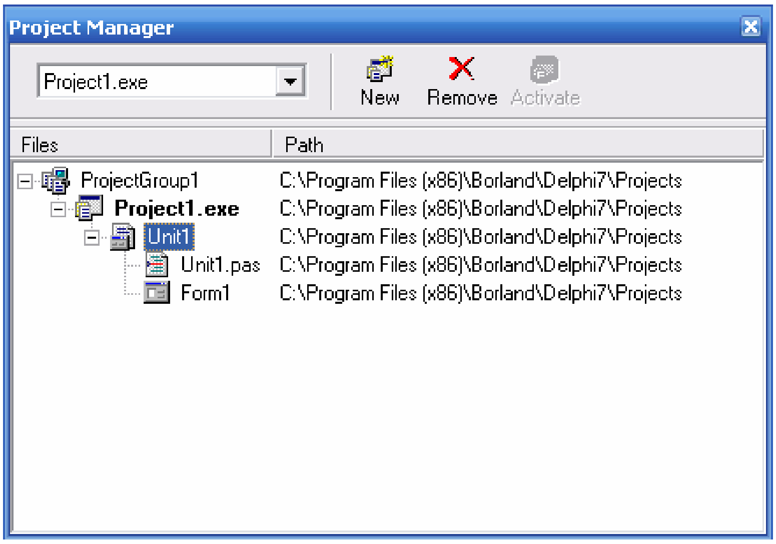

Рис. 1

```
program Project1;

uses
  Windows,
  Messages,
  SysUtils;
var
  hWndMain:HWND;// дескриптор окна приложения
  WC:TWndClass;//структура для хранения параметров класса окна
  msg:TMsg;//  сообщение
  Instance:HWnd;// дескриптор приложения

// оконная процедура:
Function WindowProc(Window: HWND; AMessage:Cardinal;
wParam, lParam: Longint): Longint; stdcall;
begin
case AMessage of
  WM_DESTROY:
  begin
    PostQuitMessage( 0 );
    Result := 0;
    Exit;
  end;
    else
      Result := DefWindowProc(Window ,AMessage,
                wParam, lParam);
   end;
end;

// Блок инициализации программы
begin
  instance:=GetModuleHandle(nil);
  with WC do
  begin
```


ŋqdqyɘY bƎ dɘqƎbYɜɘ d ɛɔɑbɔɣ ℈qdqYqɣè