

Утверждаю

Ректор университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Н. Федонин

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020г.

**ПРОГРАММИРОВАНИЕ WINDOWS-ПРИЛОЖЕНИЙ**

Работа с диалоговыми окнами

**Методические указания**

**к выполнению лабораторной работы №6**

**для студентов очной формы обучения**

**по направлению подготовки**

**02.03.03 «Математическое обеспечение и администрирование информационных систем», профиль «Технология программирования»**

**Брянск 2020**

**УДК 004.01**

Программирование Windows-приложений. Работа с диалоговыми окнами [Электронный ресурс]: методические указания к выполнению лабораторной работы №6 для студентов очной формы обучения по направлению подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем», профиль «Технология программирования» – Брянск: БГТУ, 2020. – 18 с.

Разработал:

Е.А. Белов,

канд. техн. наук, доц.

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол № 1 от 31.01.2020г.)

**Методические издания публикуются в авторской редакции**

**ЦЕЛЬ РАБОТЫ**

Изучить работу с диалоговыми окнами в приложениях WinAPI 32. В результате выполнения работы студент должен знать типы диалоговых, функции создания диалоговых окон, события, правила их обработки.

Продолжительность работы - 2 часа.

**2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Диалоговое окно – это окно, содержащее множество элементов управления (они являются его дочерними окнами), через которые проходит обмен данными между пользователем и приложением.

Для расположения элементов управления на поверхности диалоговой панели используется:

* включение в файл ресурсов приложения текстового описания шаблона диалоговой панели.
* использование специального редактора диалогов. Этот редактор позволяет нарисовать диалоговую панель и сохранить ее текстовое описание в файле ресурсов приложения.
* создание шаблона диалоговой панели в памяти во время работы приложения.

Создание диалогового окна:

* Создать шаблон диалога.
* Определить функцию диалога.
* Вызвать одну из функций создания диалога.

Типы диалоговых окон:

**Модальные окна** – диалоговые окна, при выводе на кран которых работа приложения приостанавливается, т.е. функции главного окна приложения и всех дочерних окон перестают получать сообщения от мыши и клавиатуры – они попадают в окно диалоговой панели.

Модальная диалоговая панель позволяет пользователю переключиться на работу с другими приложениями.

Создание: при помощи функции DialogBox и отображает это окно как модальное диалоговое окно.

int iCode=DialogBox(hInst,MAKEINTRESOURCE(IDR\_DIALOG1),hWnd,DlgProc);

**Немодальные окна** – диалоговые окна, при выводе на экран которых работа приложения не приостанавливается, делая при этом недоступным владельца диалогового окна.

Немодальные диалоговые окна остаются над поверхностью своих окон владельцев, даже если это окно-владелец получает фокус.

Создание: с помощью функции CreateDialog, сразу возвращает управление, а ее возвращаемым значением является дескриптор окна диалога. Немодальные окна не возвращают значения своему владельцу.

HWND hDlgModaless=NULL; // глобальная переменная, дескриптор окна диалога

. . .

hDlgModaless=CreateDialog(hInst,MAKEINTRESOURCE(IDR\_DIALOG2),hWnd, DlgModalessP roc);

Немодальные диалоговые окна и их владельцы могут обмениваться через вызовы SendMessage. Для этого необходимо изменить цикл обработки сообщений:

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

{

if(hDlgModaless==NULL||!IsDialogMessage(hDlgModaless,&msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

Немодальный диалог разрушается при вызове функции DestroyWindow. Приложение отвечает за разрушение всех немодальных диалоговых окон перед своим завершением.

case WM\_CLOSE:

{ DestroyWindow(hDlg);

hDlgModaless=NULL;

}; return TRUE;

Работа с элементами управления диалога.

Большинство дочерних окон элементов управления посылают своему родительскому окну сообщения WM\_COMMAND. Родительское окно может изменять состояние своих дочерних окон элементов управления, посылая дочерним окнам управления сообщения.

Для посылки сообщений от родительского окна дочернему окну управления с дескриптором hWndControl используется оператор следующего вида:

SendMessage(hWndControl, сообщение, параметр\_wParam, параметр\_lParam);

В Windows существует функция, которая позволяет получить дескриптор окна элемента управления диалоговой панели hDlg по идентификатору id элемента управления.

SendMessage(GetDlgItem(hDlg,id),сообщение, параметр\_wParam, параметр\_lParam);

Для работы в процедурах диалога с элементами управления диалоговых окон в программном интерфейсе Windows существует ряд функций.

Функция SendDlgItemMessage. Следующие вызовы эквивалентны:

SendDlgItemMessage(hDlg, id, iMsg, wParam, lParam);

и

SendMessage(GetDlgItem(hDlg, id),iMsg, wParam, lParam);

Функции, облегчающие работу с элементами управления в процедуре диалога:

* Для заполнения списка LISTBOX именами файлов, каталогов и дисковых устройств, предназначена функция DlgDirList
* Для списка COMBOBOX определена аналогичная функция DlgDirListComboBox
* Функция DlgDirSelect предназначена для получения из списка LISTBOX строки, выбранной пользователем
* Аналогичная функция предусмотрена для списка COMBOBOX – DlgDirSelectComboBox.

**3. ЗАДАНИЕ**

На основе приложения осуществляющего построение графического изображения, при получении сообщения WM\_PAINT:

1. Добавить в приложение возможность работы с диалоговой панелью, появляющейся после выбора пункта меню "Характеристики" осуществлять ввод значений характеристик изображения и интервала между сообщениями от таймера при помощи элементов управления.
2. Для выбора цвета изображения использовать Стандартную диалоговую панель выбора цвета (ChooseColor), ограничить возможность выбора цвета пользователем (заранее заданный диапазон цветов), заменить шаблон диалоговой панели на пользовательский.
3. **\*** По желанию: использовать хотя бы один из «**общих элементов управления**» (comctl32.dll).

Список сообщений, обязательных для обработки функцией главного окна: WM\_PAINT, WM\_DESTROY, WM\_CREATE, WM\_LBUTTONDOWN, WM\_RBUTTONDOWN, WM\_KEYDOWN, WM\_TIMER, WM\_COMMAND. Остальные сообщения передать на обработку стандартной оконной функции.

**4. ПОРЯДОК ВЫПОЛНЕНИЯ**

Для создания шаблона диалоговой панели в файле ресурсов следует выполнить следующий действия: выбрать меню Insert -> выбрать пункт Resourse -> отметить тип ресурса Dialog -> выбрать кнопку New -> отредактировать шаблон диалога (по умолчанию он имеет идентификатор (IDD\_DIALOG1) -> сохранить изменения.  
  
Созданный и отредактированный шаблон меню должен иметь элементы управления, позволяющие изменять (редактировать) такие обязательные характеристики изображения как размер, местоположение и цвет (для каждой компоненты цвета предусмотреть отдельный элемент управления), а также характеристики вида изображения и интервала между сообщениями от таймера при помощи элементов управления.

В исходном файле приложения изменить тип памяти для структуры с характеристиками изображения и переменной для хранения интервала dTimer - сделать их глобальными переменными.

Создать диалоговую процедуру DlgProc, в ней (для работы с элементами управления использовать функции SendMessage, SendDlgItemMessage или другие функции для работы с элементами управления): при обработке сообщения WM\_INITDIALOG в соответствующие поля панели диалога поместить значения характеристик изображения (из структуры характеристик) и интервала dTimer между сообщениями WM\_TIMER; при обработке сообщения WM\_COMMAND от кнопки IDOK - присвоить значения из полей панели диалога в соответствующие поля структуры характеристик и переменную dTimer.

В оконной функции главного окна при обработке сообщения WM\_COMMAND от пункта меню ID\_SETVALUE удалить выдачу сообщения функцией MessageBox. Вместо этого при помощи функции DialogBox вызвать диалоговую панель, шаблон которой описан в виде ресурса IDD\_DIALOG1, а диалоговой функцией является функция DlgProc.

Проверить возвращаемое функцией DialogBox значение. Если оно равно IDOK, то сделать следующие действия:  
 объявить всю рабочую область как недействительную (функция InvalidateRect);  
 послать окну сообщение о перерисовке (функция UpdateWindow);  
 уничтожить таймер с номером nTimer;  
 создать новый таймер с номером nTimer и интервалом dTimer.

Реализовать выбор цвета рисунка пользователем при помощи стандартного диалога Цвет(**Color**). Подключить функцию **ChooseColor,** которая создает стандартное диалоговое окно Цвет(**Color**).

Синтаксис:

BOOL ChooseColor

(

LPCHOOSECOLOR lpcc *// указывает на структуру с данными инициализации*

);

Параметры **lpcc**  
Указатель на структуру **CHOOSECOLOR**, которая содержит информацию, используемую, чтобы инициализировать диалоговое окно. Когда функция **ChooseColor** возвращает значение, эта структура содержит информацию о цвете, выбранном пользователем.

Возвращаемые значения. Если пользователь щелкает мышью по кнопке **OK** диалогового окна, возвращаемое значение не нуль. Элемент **rgbResult** структуры **CHOOSECOLOR** содержит значения **RGB** цвета, выбранного пользователем.  
Если пользователь отменяет или закрывает Диалоговое окно Цвет (**Color**), или происходит ошибка, возвращаемое значение нулевое. Чтобы получить дополнительные данные об ошибке, вызовите функцию **CommDlgExtendedError**, которая может возвратить одно из следующих значений:

CDERR\_FINDRESFAILURE CDERR\_MEMLOCKFAILURE

CDERR\_INITIALIZATION CDERR\_NOHINSTANCE

CDERR\_LOCKRESFAILURE CDERR\_NOHOOK

CDERR\_LOADRESFAILURE CDERR\_NOTEMPLATE

CDERR\_LOADSTRFAILURE CDERR\_STRUCTSIZE

CDERR\_MEMALLOCFAILURE

Замечания. Диалоговое окно Цвет (**Color**) не поддерживает палитры. Выбор цветов, предлагаемый диалоговым окном, ограничен цветами системы и размытыми версиями этих цветов.  
Вы можете предоставить фильтр - процедуру **CCHookProc** для диалогового окна Цвет (**Color**). Фильтр - процедура может обрабатывать сообщения, посылаемые в диалоговое окно. Чтобы подключить фильтр - процедуру, установите флажок **CC\_ENABLEHOOK** в элементе **Flags** структуры **CHOOSECOLOR** и установите адрес ее в элементе **lpfnHook**.

CHOOSECOLOR cc; // структура

static COLORREF acrCustClr[16]; // массив цветов

HWND hwnd; // окно - владелец

HBRUSH hbrush; // дескриптор кисти

static DWORD rgbCurrent; // изначальный цвет

// Заполняем CHOOSECOLOR

ZeroMemory(&cc, sizeof(cc));

cc.lStructSize = sizeof(cc);

cc.hwndOwner = hwnd;

cc.lpCustColors = (LPDWORD) acrCustClr;

cc.rgbResult = rgbCurrent;

cc.Flags = CC\_FULLOPEN | CC\_RGBINIT;

if (ChooseColor(&cc)==TRUE) {

hbrush = CreateSolidBrush(cc.rgbResult);

rgbCurrent = cc.rgbResult;

}

Для ограничения выбора пользователем диапазон цветов используйте функцию-фильтр.

Фильтр - процедура **CCHookProc** - определяемая программой или определенная библиотекой процедура повторного вызова, которая используется стандартным диалоговым окном Цвет (**Color**). Фильтр - процедура принимает сообщения или сообщения, предназначенные для заданной по умолчанию процедуры диалогового окна.

Синтаксис:

UINT APIENTRY CCHookProc

(

HWND hdlg, *// дескриптор окна блока диалога*

UINT uiMsg, *// код сообщения*

WPARAM wParam, *// параметр сообщения*

LPARAM lParam *// параметр сообщения*

);

Параметры  
**hdlg -** дескриптор окна блока диалога Цвет (**Color**), для которого предназначено сообщение. **uiMsg**  
Идентифицирует получаемое сообщение.  
**wParam** - определяет дополнительную информацию о сообщении. Точное значение зависит от значения параметра **uiMsg**.  
**lParam -** определяет дополнительную информацию о сообщении. Точное значение зависит от значения параметра **uiMsg**.  
Если параметр **uiMsg** указывает на сообщение **WM\_INITDIALOG**, **lParam** - указатель на структуру **CHOOSECOLOR**, содержащую значения, установленные, когда было создано диалоговое окно.

Возвращаемые значения. Если фильтр - процедура возвращает нуль, заданная по умолчанию процедура диалогового окна обрабатывает сообщение.  
Если фильтр - процедура возвращает значение отличное от нуля, заданная по умолчанию процедура диалогового окна игнорирует сообщение.

Когда вы используете функцию **ChooseColor**, чтобы создать диалоговое окно Цвет (**Color**), вы можете предоставить фильтр - процедуру **CCHookProc**, чтобы обрабатывать сообщения или предупреждения, предназначенные для процедуры диалогового окна. Чтобы разрешать действие фильтр - процедуры, используйте структуру **CHOOSECOLOR**, которую вы передали в функцию создающую диалоговое окно. Определите адрес фильтр - процедуры в элементе **lpfnHook**, и установите флажок **CC\_ENABLEHOOK** в элементе **Flags**.

Заданная по умолчанию процедура диалогового окна обрабатывает сообщение **WM\_INITDIALOG** перед передачей его в фильтр - процедуру. Все другие сообщения сначала принимает фильтр - процедура. Затем, возвращаемое значение фильтр - процедуры определяет, обрабатывает ли заданная по умолчанию процедура диалогового окна сообщение или игнорирует его.  
Если фильтр - процедура обрабатывает сообщение **WM\_CTLCOLORDLG**, она должна возвратить допустимый дескриптор кисти для закрашивания фона диалогового окна. Вообще, если фильтр - процедура обрабатывает какое-либо сообщение **WM\_CTLCOLOR\***, она должна возвратить допустимый дескриптор кисти для закрашивания фона определяемого элемента управления.

Не вызывайте функцию **EndDialog** из фильтр - процедуры. Вместо этого, фильтр - процедура может вызывать функцию **PostMessage**, чтобы послать сообщение **WM\_COMMAND** со значением **IDABORT** в процедуру диалогового окна. Регистрация **IDABORT** закрывает диалоговое окно и вынуждает функцию блока диалога возвратить значение ЛОЖЬ (**FALSE**). Если Вам нужно знать, почему фильтр - процедура закрыла диалоговое окно, вы должны предоставить свой собственный механизм связи между фильтр - процедурой и вашей прикладной программой.  
Вы можете делить на подклассы стандартные элементы управления общего диалогового окна. Однако, и процедура стандартного диалогового окна может тоже делить на подклассы элементы управления. По этой причине, вы должны делить на подклассы элементы управления тогда, когда ваша фильтр - процедура обрабатывает сообщение **WM\_INITDIALOG**. Это гарантирует, что ваша процедура подкласса получит определенные элементом управления сообщения раньше процедуры подкласса, установленной процедурой диалогового окна.

**CCHookProc** - метка - заместитель для определяемого программой или определенного библиотекой имени функции. Тип **LPCCHOOKPROC** - указатель на фильтр - процедуру **CCHookProc**.

Структура **CHOOSECOLOR** содержит информацию, которую использует функция **ChooseColor**, чтобы инициализировать стандартное диалоговое окно Цвет (**Color**). После того, как пользователь закрывает диалоговое окно, система возвращает информацию о выборе пользователя в этой структуре.

Синтаксис:

typedef struct

{

DWORD lStructSize;

HWND hwndOwner;

HWND hInstance;

COLORREF rgbResult;

COLORREF\* lpCustColors;

DWORD Flags;

LPARAM lCustData;

LPCCHOOKPROC lpfnHook;

LPCTSTR lpTemplateName;

} CHOOSECOLOR;

Элементы:  
**lStructSize -** определяет длину структуры, в байтах.  
**hwndOwner -** идентифицирует окно, которое владеет диалоговым окном. Этот элемент может быть любой правильный дескриптор окна или это может быть значение ПУСТО (**NULL**), если блок диалога не имеет владельца.  
**hInstance** **-** если флажок **CC\_ENABLETEMPLATEHANDLE** установлен в элементе **Flags**, элемент **hInstance** - дескриптор объекта памяти, содержащего шаблон диалогового окна. Если флажок **CC\_ENABLETEMPLATE** установлен, **hInstance** идентифицирует модуль, который содержит шаблон диалогового окна, именованный при помощи элемента **lpTemplateName**. Если ни **CC\_ENABLETEMPLATEHANDLE**, ни **CC\_ENABLETEMPLATE** не установлен, этот элемент игнорируется.  
**rgbResult -** если флажок **CC\_RGBINIT** установлен, элемент **rgbResult** определяет выбранный первоначально цвет, когда создавалось диалоговое окно. Если среди доступных цветов определяемого значения цвета нет, система выбирает самый близкий доступный чистый тон. Если **rgbResult** - нуль или **CC\_RGBINIT** не установлен, первоначально выбранный цвет черный. Если пользователь щелкает мышью по кнопке **OK**, элемент **rgbResult** устанавливает цвет выбранный пользователем.  
**lpCustColors -** указатель на массив из 16 значений **COLORREF**, который содержит для пользовательских палитр цветов значения красного, зеленого, синего (**RGB**) в диалоговом окне. Если пользователь изменяет эти цвета, система модифицирует массив новыми значениями RGBChooseColor, вы должны распределить статическую память для массива.  
Flags - установка битов флажков, которые вы можете использовать, чтобы инициализировать стандартное диалоговое окно Цвет (Color). Когда блок диалога возвращает значение, он устанавливает эти флажки, чтобы показать ввод данных пользователем. Этот элемент может быть комбинацией следующих флажков:

* CC\_ENABLEHOOK - Разрешает действие фильтр - процедуры, которая определена в элементе lpfnHook этой структуры. Этот флажок используется только для того, чтобы инициализировать диалоговое окно.
* CC\_ENABLETEMPLATE - Указывает, что элементы hInstance и lpTemplateName определяют шаблон диалогового окна, чтобы использовать его вместо заданного по умолчанию шаблона. Этот флажок используется только для того, чтобы инициализировать диалоговое окно.
* CC\_ENABLETEMPLATEHANDLE - Указывает, что элемент hInstance идентифицирует блок данных, который содержит предварительно загружаемый шаблон диалогового окна. Система игнорирует элемент lpTemplateName, если этот флажок установлен. Этот флажок используется только для инициализации диалогового окна.
* CC\_FULLOPEN - Заставляет диалоговое окно показывать дополнительные элементы управления, которые позволяют пользователю создавать пользовательские цвета. Если этот флажок не установлен, пользователь должен щелкнуть мышью по кнопке Определить дополнительный цвет (Define Custom Color), чтобы показать элементы управления дополнительным цветом.
* CC\_PREVENTFULLOPEN - Запрещает работу кнопки Определить дополнительные цвета (Define Custom Colors).
* CC\_RGBINIT - Заставляет диалоговое окно использовать цвет, определяемый в элементе rgbResult как исходный выбранный цвет.
* CC\_SHOWHELP - Заставляет диалоговое окно отображать кнопку Справка (Help). Элемент hwndOwner должен определить окно, которое примет зарегистрированные сообщения HELPMSGSTRING, переданное диалоговым окном, когда пользователь щелкает мышью по кнопке Справка (Help).

lCustData - устанавливает определяемые программой данные, которые система передает в фильтр - процедуру, идентифицированную элементом lpfnHook. Когда система передает сообщение WM\_INITDIALOG фильтр - процедуре, параметр lParam сообщения является указателем на структуру CHOOSECOLOR, определяемую, когда диалоговое окно было создано. Фильтр - процедура может использовать этот указатель, чтобы получить значение lCustData.  
lpfnHook - указатель на фильтр - процедуру CCHookProc, которая может обрабатывать сообщения, предназначенные для диалогового окна. Этот элемент игнорируется, если в элементе Flags не установлен флажок CC\_ENABLEHOOK.  
lpTemplateName - указатель на строку с нуль-терминатором в конце, которая именует ресурс шаблона диалогового окна в модуле, идентифицированном элементом hInstance. Этот шаблон заменяет стандартный шаблон диалогового окна. Для перечисления ресурсов диалогового окна, значение lpTemplateName может быть возвращено макрокомандой MAKEINTRESOURCE. Этот элемент игнорируется, если в элементе Flags не установлен флажок CC\_ENABLETEMPLATE.

Стандартные диалоговые окна имеют заданные по умолчанию шаблоны, которые определяют число, тип и позицию стандартных элементов управления в диалоговом окне. Вы можете определять настраиваемый шаблон, чтобы дать пользователям доступ к дополнительным элементам управления, которые явятся уникальными для вашей прикладной программы.

Для всех общих диалоговых окон за исключением диалоговых окон в стиле Проводника (**Explorer-style**) Открыть (**Open** ) и Сохранить как (**Save As**), вы изменяете заданный по умолчанию шаблон, чтобы создать пользовательский шаблон, который заменит заданный по умолчанию. Пользовательский шаблон определяет тип и позицию стандартных элементов управления также как и любых дополнительных элементов управления.

Когда вы создаете пользовательский шаблон диалогового окна, путем изменения шаблона заданного по умолчанию, удостоверитесь, что идентификаторы для любых добавленных элементов управления уникальны и не находятся в противоречии с идентификаторами стандартных средств управления. Следующая таблица перечисляет имя заданного по умолчанию файла шаблона и файла для включения для каждого из стандартных типов диалогового окна.

Тип диалогового окна Файл шаблона Включаемый файл

Цвет (Color) COLOR.DLG COLORDLG.H

Найти (Find) FINDTEXT.DLG DLGS.H

Шрифт (Font) FONT.DLG DLGS.H

Открыть (Open) (многократный выбор) FILEOPEN.DLG DLGS.H

Открыть (Open) (однократный выбор) FILEOPEN.DLG DLGS.H

Параметры страницы (Page Setup) PRNSETUP.DLG DLGS.H

Печатать (Print) PRNSETUP.DLG DLGS.H

Параметры печати (Print Setup) (устаревший) PRNSETUP.DLG DLGS.H

Заменить (Replace) FINDTEXT.DLG DLGS.H

Чтобы включить пользовательский шаблон, вы должны установить флажок в элементе **Flags** соответствующей структуры для диалогового окна. Если шаблон - ресурс в прикладной программе или в динамически компонуемой библиотеке, установите флажок **ENABLETEMPLATE** в элементе **Flags**, а элементы структуры hInstance и lpTemplateName используйте, чтобы идентифицировать имя ресурса и модуль. Если шаблон уже в памяти, установите флажок **ENABLETEMPLATEHANDLE** в элементе **Flags** и используйте элемент **hInstance**, чтобы идентифицировать объект памяти, который содержит шаблон.

В большинстве случаев, вы должны также разрешить работу фильтра (hook) для диалогового окна, чтобы поддерживать и обрабатывать вводимую информацию для дополнительных элементов управления в вашем пользовательском шаблоне.

**5. КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Что такое файл ресурсов? Какими способами его можно создавать?
2. На каком этапе создания приложения ресурсы записываются в загрузочный модуль приложения?
3. Какие данные приложение может хранить в виде ресурсов?
4. Какова последовательность работы приложения с таблицей текстовых строк, содержащейся в ресурсах приложения?
5. Как в ресурсы приложения включить изображение пиктограммы?
6. Какие существуют способы загрузки и использования пиктограммы, содержащейся в ресурсах приложения?
7. Какой оператор используется для включения в ресурсы приложения изображения курсора мыши?
8. Какие существуют способы загрузки и использования изображения курсора мыши, содержащегося в ресурсах приложения?
9. Как включить изображение типа bitmap в ресурсы приложения?
10. Как загрузить и использовать изображение типа bitmap?
11. Какие существуют типы меню?
12. Какими способами можно создать меню?
13. Что такое шаблон меню и где он хранится? Можно ли при помощи шаблона меню создавать несколько уровневые меню?
14. Как в шаблоне меню определяются такие элементы меню как пункты и подменю?
15. Для чего необходимы идентификаторы пунктов меню?
16. Как указать, что все окна класса по умолчанию должны иметь некоторое меню?
17. Каким способом окно может установить для себя меню, отличное от меню класса окон?
18. Можно ли (и как) меню создавать динамически, в процессе работы приложения?
19. Какие сообщения окну приходят от меню? Чем сообщение WM\_COMMAND отличается от сообщения WM\_SYSCOMMAND?
20. Чем сообщение WM\_COMMAND, приходящее от пункта меню, отличается от этого же сообщения, но пришедшего от элемента управления?
21. Можно ли модифицировать системное меню окна? Как это сделать?
22. В чем состоит особенность обработки сообщений WM\_SYSCOMMAND?
23. Как создать использовать независимое плавающее меню? Какие сообщения приходят при выборе его пунктов?
24. Что такое акселераторы? Где они определяются?
25. Как загрузить и использовать таблицу акселераторов? Какие изменения следует внести в цикл обработки сообщений?