Урок 7. Работа с диалоговыми панелями, стандартные диалоги

Типы диалогов, шаблон панели, процедура диалога

Диалоговое окно обычно содержит множество элементов управления (они являются его дочерними окнами), через которые проходит обмен данными между пользователем и приложением. Существует несколько функций Win32, которые помогают в конструировании, отображении и управлении содержимым диалоговых окон.

Для расположения элементов управления на поверхности диалоговой панели используется три способа.

Первый способ предполагает включение в файл ресурсов приложения текстового описания шаблона диалоговой панели. Это описание можно создать при помощи любого текстового редактора.

Второй способ размещения органов управления предполагает использование специального ***редактора диалогов***. Этот редактор позволяет нарисовать диалоговую панель и сохранить ее текстовое описание в файле ресурсов приложения. Такой подход в создании приложений носит зачатки визуального программирования, когда внешний вид и поведение приложения определяется с помощью графических средств проектирования без традиционного программирования на каком-либо алгоритмическом языке.

Третий способ предполагает создание шаблона диалоговой панели в памяти во время работы приложения. Этот способ используется редко, обычно только в тех случаях, когда внешний вид диалоговой панели нельзя определить на этапе создания приложения.

Для создания диалоговой панели не требуется вызывать функцию CreateWindow, так как в программном интерфейсе Windows определены функции, специально предназначенные для создания диалоговых панелей.

Перед вызовом функций, специально предназначенных для создания диалоговых панелей, необходимо создать ***шаблон*** диалоговой панели. Это можно сделать, воспользовавшись одним из описанных выше способов. Помимо шаблона, перед созданием диалоговой панели следует определить ***специальную функцию диалога***, в которую будут поступать сообщения от элементов управления.

Итак, для создания диалоговой панели необходимо предпринять следующие действия:

1. Создать шаблон диалога.
2. Определить функцию диалога.
3. Вызвать одну из функций создания диалога.

Типы диалоговых панелей

Диалоговые окна бывают двух типов: немодальные и модальные.

При выводе на экран **модальной** диалоговой панели работа приложения ***приостанавливается***.

Функции главного окна приложения и всех дочерних окон перестают получать сообщения от мыши и клавиатуры. Все эти сообщения попадают в окно диалоговой панели. Когда работа с диалоговой панелью будет завершена, главное окно приложения и его дочерние окна будут ***разблокированы***. Поэтому диалоговая панель не должна создаваться как дочернее окно - в этом случае она будет заблокирована наряду с остальными дочерними окнами и приложение “зависнет”.

Модальная диалоговая панель, тем не менее, позволяет пользователю переключиться на работу с другими приложениями. Если требуется запретить возможность такого переключения, необходимо использовать системные модальные диалоговые панели.

Модальное диалоговое окно создается и активизируется при помощи функции DialogBox. Эта функция создает диалоговое окно, используя ресурс шаблона диалоговой панели, и отображает это окно как модальное диалоговое окно. Приложение, вызывающее функцию DialogBox, передает адрес функции обратного вызова; DialogBox не возвращает управления (не возвращает значения), пока диалоговое окно не закроется посредством вызова функции EndDialog (возможно, в ответ на такое событие пользовательского интерфейса, как щелчок на кнопке “OK”).

Хотя можно создать модальное окно и без владельца, обычно это не рекомендуется. При создании такого диалогового окна необходимо помнить о следующем: поскольку основное окно приложения не является недоступным для пользователя, требуется предпринять действия для обеспечения обработки посланных или отправленных сообщений для продолжения его работы. Windows не разрушает и не скрывает диалоговых окон без владельцев, даже если другие окна приложения разрушены.

В отличие от модального диалогового окна, открытие **немодальной** диалоговой панели не останавливает выполнение приложения, делая при этом недоступным владельца диалогового окна. Однако немодальные диалоговые окна остаются над поверхностью своих окон-владельцев, даже если это окно-владелец получает фокус. Немодальные окна представляют эффективный способ продолжительного отображения важной для пользователя информации.

Немодальные диалоговые окна обычно создаются с помощью функции CreateDialog. Немодальные окна не возвращают значения своему владельцу. Однако немодальные диалоговые окна и их владельцы могут обмениваться через вызовы SendMessage. Процедура диалогового окна для немодального диалога не должна вызывать функцию EndDialog. Немодальный диалог обычно разрушается при вызове функции DestroyWindow. Эта функция может быть вызвана в ответ на событие пользовательского интерфейса из процедуры диалогового окна. Приложение отвечает за разрушение всех немодальных диалоговых окон перед своим завершением.

Замечание. В отличие от использования модальных диалоговых панелей, при использовании немодального диалогового окна приложение отвечает за получение и отправление сообщений для этого немодального окна. Большинство приложений делает это в своем главном цикле сообщений; однако, для обеспечения реакции диалогового окна на события клавиатуры и для того, чтобы позволить пользователю перемещаться между элементами управления с использованием быстрых клавиш, приложение должно вызывать функцию IsDialogMessage.

Шаблон диалоговой панели

Для определения типов и внешнего вида элементов управления в окне диалога большинство приложений использует ресурс шаблона диалогового окна, созданный как часть файла ресурсов приложения.

Оператор диалогового окна, содержащийся в файле ресурсов приложения, определяет его размещение и внешний вид, а также список всех его элементов управления. Оператор диалогового окна состоит из нескольких строк:

```
ИмяИлиИдентификатор DIALOG x, y, ширина, высота
// необязательные операторы, определяющие свойства диалога
BEGIN
        // операторы, определяющие элементы управления
END
```

Первая строка содержит имя диалога (но может быть и число), ключевое слово DIALOG (DILOGEX) и четыре числовых параметра, определяющих положение верхнего левого угла диалогового окна и его размер.

Информация о размере и расположении в операторе диалогового окна измеряется в диалоговых единицах. Диалоговые единицы происходят от размера шрифта, определенного для диалогового окна. Основные единицы диалогового окна представляют среднюю высоту и ширину символа выбранного шрифта. Четыре горизонтальных диалоговых единицы равны одной горизонтальной основной диалоговой единице; восемь вертикальных диалоговых единиц равны одной вертикальной основной диалоговой единице.

После строки, содержащей ключевое слово DIALOG, в операторе диалогового окна могут следовать несколько необязательных операторов.

Необязательный оператор CAPTION указывает заголовок диалогового окна. По умолчанию диалоговое окно не имеет заголовка.

Необязательный оператор STYLE определяет стиль диалогового окна. Значения стилей обычно определяются в файле заголовков windows.h (их идентификаторы начинаются с префиксов WS_ и DS_); несколько значений можно объединять с помощью операции логического ИЛИ (|).

Необязательный оператор CLASS используется для указания определенного класса окна для диалога. Этот оператор следует использовать достаточно аккуратно, так как он переопределяет поведение диалогового окна. Перед созданием диалоговых панелей с собственным классом этот класс должен быть зарегистрирован. При этом в структуре WNDCLASS, используемой для регистрации, поле cbWndExtra должно иметь значение DLGWINDOWEXTRA.

Необязательный оператор FONT определяет используемый в диалоговом окне шрифт. Оператор FONT позволяет задать шрифт, с использованием которого Windows будет писать текст в диалоговой панели: FONT 10, “MS Serif”. Первый параметр указывает размер шрифта в пунктах, второй - название шрифта. По умолчанию используется системный шрифт.

Необязательный оператор MENU указывает ресурс меню, определяющий меню этого диалогового окна. При отсутствии этого оператора диалоговое окно будет создано без строки меню.

Последовательность операторов, определяющих элементы управления диалогового окна, заключена между ключевыми словами BEGIN-END. Существует два типа таких операторов: оператор определения конкретного элемента управления и оператор CONTROL.

Каждый оператор определения того или иного элемент управления диалогового окна содержит параметры типа элемента управления, текста, идентификатора элемента управления (текстового или числового), его расположения, стиля и расширенного стиля:

```
оператор_элемента_управления   "text", id, x, y, width, height [, style]
```

Поле ввода определяется оператором EDITTEXT.

Оператор элемента управления COMBOBOX определяет раскрывающийся список (комбинированный список).

Оператор LISTBOX используется для определения окна списка.
SCROLLBAR – применяется для полосы прокрутки.

Статические элементы управления определяются операторами LTEXT, CTEXT, RTEXT или ICON. Первые три определяют статические элементы с соответственно левым, центральным и правым выравниванием. Последний оператор определяет статический элемент управления со стилем SS_ICON.

Кнопка определяется одним из следующих ключевых слов: AUTO3STATE, AUTOCHECKBOX, AUTORADIOBUTTON, CHECKBOX, DEFPUSHBUTTON, GROUPBOX, PUSHBUTTON, RADIOBUTTON, STATE3, USERBUTTON.

Существует и другой способ определения элементов управления - при помощи обобщающего оператора CONTROL.

Элементы управления можно определять и с оператора CONTROL. Синтаксис этого оператора несколько отличается от синтаксиса утверждений для других элементов управления:

```
CONTROL        "text", id, classname, style, x, y, width, height
```

Параметр classname определяет класс окна для данного элемента управления. Это может быть один из классов элементов управления Windows (например, "edit", "combobox", "listbox", "scrollbar", "static", "button" и др.). Таким образом, оператор CONTROL можно использовать в качестве альтернативного синтаксиса для утверждений других элементов управления.

Поле стиля style в инструкциях определения элементов управления дает возможность включить в инструкции различные идентификаторы стиля окна, такие, как наличие рамки у окна, отметка о кнопки-переключателя и т.д.

Идентификаторы id в операторах, определяющих элементы управления, представляют собой числа, с помощью которых элементы управления – дочерние окна – идентифицируют себя при посылке сообщений (обычно это сообщения WM_COMMAND) своему родительскому окну, которым является окно диалога. Статические дочерние окна сообщений не посылают, поэтому для них можно устанавливать значения id равными –1.

Замечание. При поступлении сообщений от элементов управления они поступают в оконную процедуру окна диалога DefDlgProc, которая находится внутри Windows. Эта стандартная оконная процедура окна диалога, сделав стандартные действия, посылает затем полученные сообщения диалоговой процедуре, которая включена в приложение.

Понятие процедуры диалога

Диалоговая процедура приложения или процедура диалога приложения обрабатывает сообщения, получаемые окном диалога. Она очень сильно напоминает оконную процедуру, но настоящей оконной процедурой не является.
Как уже отмечалось,

Настоящая оконная процедура окна диалога находится в Windows. Эта оконная процедура вызывает диалоговую процедуру приложения, передавая ей многие сообщения, которые получает сама.

Параметры диалоговой функции те же, что и параметры обычной оконной процедуры; как и оконная процедура, процедура диалога должна быть определена как функция CALLBACK.

Модальная диалоговая панель

Диалоговая процедура модальной диалоговой панели

Простейшая диалоговая процедура DlgProc для модального диалога имеет следующий вид:

```
BOOL CALLBACK DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
        {
                switch(message)
                {
                        case WM_INITDIALOG:
                        {       // различные инициализации
                                . . .
                // приложение само может установить фокус ввода на нужный
                // элемент управления (SetFocus), тогда возвращают FALSE.
                // Для предоставления Windows возможности установки фокуса
                //ввода следует возвратить TRUE.
                                return TRUE;
                        };
                        case WM_COMMAND:
                        {       switch(LOWORD(wParam))
                                {
                                case IDCANCEL:
                // нажаты или ESC, или Ctrl+Break, или пункт
                // меню IDCANCEL, или кнопка IDCANCEL
                        EndDialog(hDlg, IDCANCEL);
                        return TRUE;
                                case IDOK:
        // нажата ENTER и при этом ни одна из кнопок не находилась в фокусе
                // ввода и ни одна из кнопок не имеет стиль BS_DEFPUSHBUTTON,
                // или
                // нажата кнопка IDOK или пункт меню IDOK
                        EndDialog(hDlg, IDOK);
                        return TRUE;
                        // обработка сообщений от других элементов управления
                                        . . .
                                }
                        }; break;
                        case WM_CLOSE:
                        // закрытие окна при помощи системного меню
                        {// моделирование отказа от работы с панелью
                        PostMessage(hDlg, WM_COMMAND, IDCANCEL, 0);
                        }; return TRUE;
                }
                return FALSE;
        }
```

Замечание. Следует обратить внимание на отличия между диалоговой процедурой и оконной процедурой:

Возвращаемое значение: оконная процедура возвращает значение LRESULT, а диалоговая процедура – значение типа BOOL (определяемого как int).

Обработка некоторых сообщений: процедура диалога не обрабатывает сообщения WM_PAINT и WM_DESTROY. Процедура диалога не получит сообщения WM_CREATE; вместо этого она выполняет инициализацию при обработке специального сообщения WM_INITDIALOG.

Обработка сообщений по умолчанию: если оконная процедура не обрабатывает какое-то сообщение, она вызывает DefWindowProc; процедура диалога, если она не обрабатывает сообщение, возвращает FALSE (0), а если обрабатывает, то TRUE (не 0).

Сообщение WM_INITDIALOG является первым сообщением, которое получает процедура диалога (причем оно посылается только процедуре диалога). Если процедура диалога возвращает TRUE, то Windows помещает фокус ввода на первое дочернее окно элемента управления, которое имеет стиль WS_TABSTOP. С другой стороны, при обработке сообщения WM_INITDIALOG процедура диалога может использовать функцию SetFocus для того, чтобы самостоятельно установить фокус на одно из дочерних окон управления, и тогда она должна возвратить значение FALSE.

Сообщение WM_COMMAND с идентификатором IDCANCEL будет появляться тогда, когда пользователь закроет диалоговую панель, нажав клавиши <Esc> или <Ctrl+Break> или выберет кнопку или пункт меню с идентификатором IDCANCEL. Обычно в диалоговую процедуру добавляют моделирование этой же ситуации, если пользователь закроет диалоговую панель с помощью системного меню (обработка сообщения WM_CLOSE).

Сообщение WM_COMMAND с идентификатором IDOK поступает в том случае, если пользователь выбрал кнопку или пункт меню с идентификатором IDOK или нажал клавишу <Enter> в тот момент, когда выполняются одновременно два условия:
ни одна из кнопок, расположенных в диалоговой панели, не имеет фокус ввода;
ни одна из кнопок не имеет стиль BS_DEFPUSHBUTTON.

Если же в диалоговом окне при выполнении первого условия есть кнопка со стилем BS_DEFPUSHBUTTON, то при нажатии на <Enter> вместе с сообщением WM_COMMAND передается идентификатор этой кнопки, выбираемой по умолчанию.

Для того чтобы модальное диалоговое окно прекратило свою работу, диалоговая процедура должна вызвать функцию EndDialog, которая и сообщает Windows о необходимости закрытия окна диалога.

Замечание. Сообщения модального диалога не проходят через очередь сообщений приложения, поэтому не стоит беспокоиться о влиянии быстрых клавиш на работу окна диалога.

Создание окна модальной диалоговой панели

Для того чтобы отобразить модальную диалоговую панель, внешний вид которой определяется шаблоном с идентификатором IDR_DIALOG1, а поведение диалоговой процедурой DlgProc, необходимо воспользоваться функцией DialogBox:

```
int iCode = DialogBox(hInst, MAKEINTRESOURCE(IDR_DIALOG1), hWnd, DlgProc);
```

Для вызова этой функции также необходим дескриптор копии приложения, передаваемый в приложение через параметры функции WinMain (этот дескриптор можно запомнить глобальной переменной и затем использовать по мере необходимости); а также hWnd – дескриптор родительского окна диалоговой панели.

Функция DialogBox, которая вызывается для вывода на экран окна диалога, не возвращает управление в оконную процедуру родительского окна до тех пор, пока окно диалога не будет закрыто диалоговой процедурой при помощи функции EndDialog.

Возвращаемым значением функции DialogBox является второй параметр функции EndDialog. Это значение может быть использовано для анализа того, что послужило причиной выхода пользователя из диалога.

Даже при выведенном окне диалога, оконная процедура родительского окна может продолжать получать сообщения, даже и из диалоговой процедуры. Для посылки сообщения из диалоговой процедуры в оконную функцию родительского окна вызов функции SendMessage (или PostMessage в случае постановки сообщения в очередь) в DlgProc должен начинаться следующим образом:

```
SendMessage(GetParent(hDlg),. . .);
```

Работа с элементами управления диалога

Большинство дочерних окон элементов управления посылают своему родительскому окну сообщения WM_COMMAND (исключение составляют полосы прокрутки). Более того, родительское окно может изменять состояние своих дочерних окон элементов управления (например, устанавливать и снимать метки с флажков), посылая дочерним окнам управления сообщения.

Аналогичным образом можно получать сообщения от элементов управления и изменять их состояние и в процедуре диалога. Кроме этого, для работы с окнами управления в окнах диалога Windows обеспечивает разработчиков приложений еще несколькими возможностями.

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

```
SendMessage(hWndControl, сообщение, параметр_wParam, параметр_lParam);
```

Сложность в том, что процедуре диалога неизвестны дескрипторы дочерних окон элементов управления. Известен лишь идентификатор элемента управления, от которого приходит сообщение.

В Windows существует функция, которая позволяет получить дескриптор окна элемента управления диалоговой панели hDlg по идентификатору id элемента управления. С использованием этой функции предыдущий оператор приобретает следующий вид:

```
SendMessage(GetDlgItem(hDlg, id), сообщение, параметр_wParam, параметр_lParam);
```

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

Первым усовершенствованием является специальная функция SendDlgItemMessage. Следующие вызовы эквивалентны:

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

и

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

Существует также функция, которая снимает контрольные метки со всех радио-переключателей диалоговой панели hDlg с идентификаторами от idFirst до idLast, за исключением радио-переключателя с идентификатором idCheck, который, наоборот, включается:

```
CheckRadioButton(hDlg, idFirst, idLast, idCheck);
```

Похожая функция имеется и для работы с флажками. Если в окне диалога hDlg создается элемент управления CHECKBOX с идентификатором idCheckbox, то снять или установить контрольную метку (iCheck – 0/1) можно с помощью следующей функции:

```
CheckDlgButton(hDlg, idCheckbox, iCheck);
```

Чтобы получить состояние флажка в окне диалога в любой момент времени, можно использовать следующую функцию:

```
int iCheck = IsDlgButtonCkecked(hDlg, idCheckbox);
```

Замечание. Перечислим кратко другие 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.

Функция SetDlgItemText позволяет изменить заголовок элемента управления или записать текст в текстовый редактор.

Функция SetDlgItemInt позволяет записать в заголовок окна элемента управления или текстовый редактор текстовую строку, полученную после преобразования целого числа в формат строки символов.

Для получения строки, связанной с элементом управления, можно использовать функцию GetDlgItemText.

Предусмотрена также функция GetDlgItemInt, получающая из органа управления текстовую строку и выполняющая преобразование этой строки в целое число.

Табуляция и группы

Для того чтобы пользователь, например, имел возможность переходить при помощи клавиши <Tab> от одного дочернего окна элемента управления к другому в обычном недиалоговом родительском окне, приложение должно использовать технику введения новой оконной процедуры.

В окне диалога необходимость применения этой техники отпадает: Windows обеспечивает всю логику, необходимую для перемещения фокуса ввода с одного элемента управления на другое. Однако для этого необходимо включить в шаблон окна диалога при описании элементов управления стили WS_TABSTOP и WS_GROUP: для всех дочерних окон элементов управления, к которым необходим доступ с помощью клавиши <Tab>, задается стиль WS_TABSTOP.

Вторая возможность работы с клавиатурой, которую Windows предоставляет в окне диалога, включает в себя использование клавиш управления курсором. Эта возможность особенно важна для групп радио-переключателей.

Итак, пусть необходимо при помощи клавиши <Tab> передавать фокус отмеченному в данный момент времени радио-переключателю некоторой группы. Для передачи же фокуса от одного переключателя к другому внутри этой группы пусть необходимо использовать клавиши управления курсором.

Для использования клавиши <Tab> и клавиш управления курсором в группе элементов необходимо первому элементу из группы установить стиль WS_GROUP|WS_TABSTOP, а остальным элементам группы ни один из этих стилей не задавать.

Элементу управления, оператор определения которого стоит за определением последнего элемента группы, следует установить хотя бы стиль WS_GROUP (начало новой группы).

Windows будет использовать клавиши управления курсором для передачи фокуса ввода с первого элемента управления, имеющего стиль WS_GROUP (начало группы), на следующие элементы управления группы (но до элемента управления, имеющего стиль WS_GROUP - начало новой группы). При достижении последнего элемента группы, Windows будет циклически переходить снова не первый и т.д.

Замечание. По умолчанию дочерние окна управления LTEXT, CTEXT, RTEXT и ICON (все это статические поля) включают стиль WS_GROUP, который помечает конец предыдущей группы и начало следующей. Для дочерних окон элементов управления других типов часто необходимо добавлять стиль WS_GROUP.

Хотя, как правило, программисты позволяют менеджеру окна диалога брать передачу фокуса ввода внутри группы и при использовании табуляции на себя, в Windows есть две функции, которые дают возможность определить следующую или предыдущую позиции табуляции или окна группы:

```
HWND hWndControl = GetNextDlgTabItem(hDlg, hWndControlCurrent, bPrevious);
```

и

```
HWND hWndControl = GetNextDlgGroupItem(hDlg, hWndControlCurrent, bPrevious);
```

Если параметр bPrevious равен TRUE, то функции возвращают предыдущую позицию табуляции или окна группы, если FALSE – то следующую.

Немодальная диалоговая панель

Создание окна немодальной диалоговой панели

Модальные окна диалога (исключая системные) позволяют пользователю переключаться между окнами диалога и другими программами. Однако пользователь не может перейти в другое окно своей программы, не закрыв модальное окно диалога.

Немодальное окно диалога позволяет пользователю переключаться между окном диалога и окном, в котором оно было создано, а также между окном диалога и остальными программами. Таким образом, немодальное окно диалога больше напоминает обычные всплывающие окна, которые могут создаваться программой.

Немодальные окна диалога предпочтительнее модальных в том случае, когда пользователь хотел бы на некоторое время оставить окно диалога на экране, продолжая работу с основным окном.

В отличие от модальных диалогов, которые создаются функцией DialogBox, возвращающей управление только после закрытия окна диалога, немодальные диалоги создаются в оконной процедуре окна hWnd с помощью функции CreateDialog. Параметры этой функции те же, что и параметры функции DialogBox:

```
HWND hDlgModaless = NULL; // глобальная переменная, дескриптор окна диалога
        . . .
hDlgModaless = CreateDialog(hInst, MAKEINTRESOURCE(IDR_DIALOG2), hWnd,
DlgModalessProc);
```

Отличие состоит в том, что функция CreateDialog сразу возвращает управление, а ее возвращаемым значением является дескриптор окна диалога. Как правило, этот дескриптор хранится в глобальной переменной.

Диалоговая процедура немодальной диалоговой панели

Простейшая диалоговая процедура DlgModalessProc для немодального диалога должна иметь следующий вид:

```
BOOL CALLBACK DlgModalessProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
        {
        switch(message)
                {
                case WM_INITDIALOG:
                        {
                        ... // различные инициализации
                        return TRUE;
                        };
                case WM_COMMAND:
                        {
                        switch(LOWORD(wParam)
                                {
                                case IDCANCEL: // если есть кнопка IDCANCEL
                                CloseWindow(hDlg);
                                break;
                        // обработка сообщений от других элементов управления
                                        ...
                                }
                        }; break;
                case WM_CLOSE: // закрытие через системное меню
```

```
                {       DestroyWindow( hDlg ); // не EndDialog !!!
                        hDlgModaless = NULL;
                }; return TRUE;
        }
        return FALSE;
}
```

Особенности работы с немодальными окнами диалога

Работа с немодальными окнами диалога похожа на работу с модальными, но имеет несколько важных отличий.

Во-первых, немодальные окна диалога обычно содержат строку заголовка и значок системного меню. Строка заголовка и системное меню дают пользователю возможность перемещать немодальное окно диалога по экрану как с помощью мыши, так и клавиатуры. Инструкция STYLE в шаблоне окна диалога для немодального окна в этом случае должна выглядеть примерно так:

```
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_VISIBLE
```

Во-вторых, следует обратить внимание, что в инструкцию STYLE включен стиль WS_VISIBLE. Если этого не сделать, то после вызова функции CreateDialog необходимо вызвать функцию ShowWindow, так как иначе немодальное окно не появится на экране:

```
hDlgModaless = CreateDialog(hInstance, "MyDialog", hWnd, DlgModalessProc);
ShowWindow(hDlgModaless, SW_SHOW);
```

В-третьих, для закрытия немодального диалога следует в диалоговой процедуре использовать функцию DestroyWindow. После вызова функции DestroyWindow глобальная переменная hDlgModaless должна быть установлена в NULL:

```
case WM_CLOSE:
{       DestroyWindow( hDlg );
        hDlgModaless = NULL;
}; return TRUE;
```

В отличие от сообщений для модальных окон диалога и окон сообщений, сообщения для немодальных окон диалога проходят через очередь сообщений программы. Поэтому цикл их обработки должен быть изменен таким образом, чтобы эти сообщения передавались в оконную процедуру окна диалога.

Рассмотрим действия, необходимые для обеспечения передачи сообщений немодальным окнам диалогов.

Во-первых, необходимо объявить глобальную переменную для хранения дескриптора созданного окна немодального диалога, например, hDlgModaless:

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

Во-вторых, в необходимом месте кода приложения следует при помощи функции CreateDialog создать немодальный диалог. При этом необходимо запомнить в глобальной переменной дескриптор окна диалога, который является возвращаемым значением этой функции, например:

```
hDlgModaless = CreateDialog(hInst, MAKEINTRESOURCE(IDR_DIALOG2), hWnd,
DlgModalessProc);
```

Затем необходимо изменить цикл обработки сообщений, чтобы он выглядел так:

```
while(GetMessage(&msg, NULL, 0, 0))
        {
        if(hDlgModaless == NULL || !IsDialogMessage(hDlgModaless, &msg))
                {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
                }
        }
```

Если сообщение предназначено для немодального диалога, то функция IsDialogMessage отправляет его оконной процедуре окна диалога и возвращает TRUE (не 0); в противном случае она возвращает FALSE (0). Функции TranslateMessage и DispatchMessage будут вызываться только в том случае, если hDlgModaless равен NULL или если сообщение не для окна диалога.

Если для окна приложения используются быстрые клавиши, то цикл обработки сообщений должен стать таким:

```
while(GetMessage(&msg, NULL, 0, 0))
        {
        if(hDlgModaless == NULL || !IsDialogMessage(hDlgModaless, &msg))
                {
                if(!TranslateAccelerator(hWnd, hAccel, &msg))
                        {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                        }
                }
        }
```

Замечание. Переменная hDlgModaless может использоваться и в других частях программы для проверки наличия созданного немодального окна диалога.

Контрольные вопросы:

1. Является ли диалоговая панель окном Windows? Чем они отличаются от перекрывающихся окон?
2. Какие существуют способы расположения элементов управления на диалоговой панели?
3. Какие типы диалоговых панелей существуют? Чем они отличаются друг от друга?
4. Что определяет ресурс шаблона диалога?
5. Что такое диалоговая процедура? Является ли она оконной функцией окна диалога?
6. Получает ли диалоговая процедура сообщения WM_PAINT, WM_CREATE, WM_DESTROY? Как происходит обработка этих сообщений?
7. Когда в диалоговую процедуру поступает сообщение WM_COMMAND с идентификатором IDCANCEL?
8. Когда в диалоговую процедуру поступает сообщение WM_COMMAND с идентификатором IDOK?
9. Вызовом какой функции создается модальная диалоговая панель?
10. Возвращает ли функция создания модального диалога управление сразу же после вывода диалоговой панели?

11. Что должна сделать диалоговая процедура для того, чтобы модальная диалоговая панель прекратила свою работу?
12. Проходят ли сообщения для модальных диалоговых окон через очередь сообщений приложения?
13. Чем отличается поведение немодального и модального диалоговых окон?
14. Вызовом какой функции создается немодальная диалоговая панель?
15. Возвращает ли функция создания немодального диалога управление сразу же после вывода диалоговой панели?
16. Что необходимо сделать для того, чтобы немодальная диалоговая панель прекратила свою работу?
17. Можно ли прекратить работу немодального диалога функцией EndDialog?
18. Проходят ли сообщения для немодальных диалоговых окон через очередь сообщений приложения?
19. Какие изменения следует внести в цикл обработки сообщений для того, что немодальное диалоговое окно могло получать сообщения, предназначенные для него?