Диалоги. Знакомство с элементами управления - MFC

Диалоги и элементы управления - очень обширная тема. Например, для полного освещения темы элементов управления потребовалась бы целая книга. Поэтому в данных методических указаниях мы рассмотрим сначала основы, а затем только наиболее широко используемые элементы управления. Это послужит Вам хорошим фундаментом на пути к углубленному изучению MFC.

Диалог (диалоговое окно) представляет собой специальный вид окна, которые предназначены для взаимодействия с пользователем. Обычно они используются для изменения настроек приложения и ввода информации. Диалоги используются очень часто. Например, практически все окна настроек приложения Microsoft Word являются диалогами.

Взаимодействие между диалогом и пользователем Взаимодействие между диалогом и пользователем осуществляется с помощью элементов управления. Это особый тип окон для ввода или вывода. Элемент управления принадлежит окну-владельцу, в данном случае - диалогу. Все версии Windows поддерживают некоторый набор стандартных элементов управления, к которым относятся кнопки, контрольные переключатели, селекторные кнопки, списки, поля ввода, комбинированные списки, полосы прокрутки и статические элементы. Вам должны быть знакомы все эти элементы. Рассмотрим кратко каждый из них:

* Обыкновенная кнопка (push button) - это кнопка, которую пользователь "нажимает" мышью или клавишей Enter, переместив предварительно на нее фокус ввода.
* Контрольный переключатель (check box) может быть либо выбранным, либо нет. Если в диалоге есть несколько контрольных переключателей, то могут быть выбраны одновременно несколько из них.
* Радиокнопка (radio button) - это почти то же самое, что и контрольный переключатель. Только при наличии нескольких кнопок в группе может быть выбрана только одна.
* Список (list box) содержит набор строк, из которого можно выбрать одну или несколько. Широко используется при отображении имен файлов.
* Поле ввода (edit box) - это элемент, позволяющий ввести строку текста.
* Комбинированный список (combo box) представляет собой список со строкой ввода.
* Статический элемент (static control) предназначен для вывода текста или графики, но не предназначен для ввода.

Элементы управления способны как генерировать сообщения в ответ на действия пользователя, так и получать их от приложения. В последнем случае сообщения являются, фактически, командами, на которые элемент управления должен отреагировать.

Классы MFC для элементов управления

В MFC содержатся классы для всех стандартных элементов управления. Эти классы описывают сами элементы, а также содержат функции для работы с ними. Их называют классами управления. Они порождаются от класса CWnd. Таким образом, все они обладают характеристиками окна. Ниже приведены основные классы управления:

|  |  |
| --- | --- |
| Класс | Элемент управления |
| CButton | Кнопки, селекторные кнопки и контрольные переключатели |
| CEdit | Поля ввода |
| CListBox | Списки |
| CComboBox | Комбинированные списки |
| CScrollBar | Полосы прокрутки |
| Cstatic | Статические элементы |

В MFC допускается также и непосредственное обращение к элементам управления, но на практике это происходит очень редко. Намного удобнее пользоваться соответствующими классами. Наиболее часто элементы управления используются с диалоговыми окнами, хотя можно создавать и отдельные элементы, расположенные в главном окне.

Модальные и немодальные диалоги

Есть два типа диалогов: модальные и немодальные. Наиболее распространены первые. В случае модальных диалогов, при активизации диалога основное окно приложения становится пассивным и перестает реагировать на действия пользователя до тех пор, пока он не закроет диалог. В случае немодальных диалогов, диалог существует независимо от других окон, и основное окно также может быть активизировано. Программировать немодальные диалоги немного труднее. Мы рассмотрим оба типа диалогов.

Диалоги как ресурсы

Как уже было упомянуто выше, диалоги не создаются программно. Вместо этого, при необходимости из ресурсов загружаются описания диалогов, и Windows по этому описанию формирует окно и размещает на нем все элементы управления.

Диалоги редактируются визуально из ресурсного редактора. Диалог вместе со всеми элементами управления представляет собой один ресурс со своим идентификатором. Кроме того, каждый элемент управления имеет свой идентификатор, который может быть только числовым. Обычно идентификаторы имеют префикс в соответствии с названием данного элемента управления, хотя при желании можно использовать любые идентификаторы.

Класс CDialog

В MFC все диалоги являются экземплярами либо класса CDialog, либо порожденных от него классов. Лишь самые простые диалоги используют непосредственно класс CDialog. В общем же случае, необходимо определять собственный класс. Класс CDialog имеет конструкторы со следующими прототипами:

CDialog::CDialog(LPCSTR ResourceName, CWnd \*Owner = 0);

CDialog::CDialog(UINT ResourceID, CWnd \*Owner = 0);

CDialog::CDialog();

Параметр ResourceName или ResourceID определяет идентификатор диалога в ресурсах, строковый или числовой. Параметр Owner - это указатель на окно-собственник, если равен 0, то собственником будет главное окно приложения. Последняя форма конструктора предназначена для создания немодальных диалогов, о которых будет сказано в дальнейшем.

Обработка сообщений от диалогов

Все диалоги являются разновидностью окон. Поэтому для них используется такой же механизм сообщений, как и для главного окна. Для каждого диалога организуется собственная очередь сообщений, так же точно, как и для главного окна.

Каждый раз, когда элемент управления диалога активизируется, диалогу посылается сообщение WM\_COMMAND. С этим сообщением передается идентификатор элемента управления. Для обработки сообщений в карту сообщений диалога нужно поместить макрос ON\_COMMAND(). Многие элементы управления генерируют также идентификационный код, который позволяет определить, какое действие было произведено с элементом управления. Во многих случаях по этому коду выбирается тот или иной обработчик. Как будет показано далее, в MFC имеется механизм, подобный макросу ON\_COMMAND(), с помощью которого можно связывать идентификационные коды с обработчиками.

Вызов диалога

После того, как объект класса диалога создан, необходимо вызвать член-функцию DoModal(). Результатом будет модальное отображение диалога. Прототип функции следующий:

virtual int CDialog::DoModal();

Функция возвращает код завершения, генерируемый диалогом при закрытии, или -1, если окно не может быть отображено. Если при отображении диалога произошла ошибка, возвращается IDABORT. Функция не завершается, пока диалог не будет закрыт.

Закрытие диалога

По умолчанию диалог закрывается при получении сообщения с идентификатором либо IDOK, либо IDCANCEL. Они предопределены и обычно связаны с кнопками подтверждения и отмены. Класс CDialog содержит встроенные обработчики для этих двух случаев, OnOK() и OnCancel(). Их не нужно включать в очередь сообщений диалога. Но их можно переопределить, что дает возможность программисту управлять закрытием диалога. Для программного закрытия диалога нужно вызвать член-функцию с прототипом:

void CDialog::EndDialog(int RetCode);

Параметр определят значение, которое вернет функция DoModal(). Обычно возвращаются значения IDOK или IDCANCEL, другие значения используются редко.

Пример программы с диалоговым окном

Ниже приведены исходные тексты примера программы с диалогом. Диалог выбирается с помощью меню. Он имеет три элемента управления - кнопки "Red", "Green" и "Cancel". Также переопределена функция OnCancel(), что позволяет при попытке закрытия диалога вывести окно с подтверждением. Диалог закрывается программно, с возвратом кода завершения 7, который затем возвращается функцией DoModal() и отображается в окне. Это сделано лишь для демонстрации, реально же обычно возвращается IDCANCEL. Диалог создавался визуально в среде Visual C++. Здесь приводится также файл с идентификаторами, сгенерированный средой. В дальнейшем этот файл приводиться не будет, потому что, вообще говоря, он не предназначен для чтения человеком. Так как все идентификаторы в примерах имеют осмысленные имена, то загрузив проект в среду IDE, Вы сможете сразу узнать, какие идентификаторы каким элементам управления соответствуют.

**resource.h**

//{{NO\_DEPENDENCIES}}

// Microsoft Developer Studio generated include file.

// Used by Resources.rc

#define IDC\_RED 1007

#define IDC\_GREEN 1008

#define IDM\_DIALOG 40001

#define IDM\_HELP 40002

#define IDM\_EXIT 40003

// Next default values for new objects

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 1

#define \_APS\_3D\_CONTROLS 1

#define \_APS\_NEXT\_RESOURCE\_VALUE 106

#define \_APS\_NEXT\_COMMAND\_VALUE 40010

#define \_APS\_NEXT\_CONTROL\_VALUE 1008

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif

**Dialogs.hpp**

#include "stdafx.h"

#include "resource.h"

// Класс главного окна

class CMainWin: public CFrameWnd {

public:

// Конструктор немного изменен.

// Title -- заголовок окна, HowShow -- код показа окна

CMainWin(CString Title, int HowShow);

afx\_msg void OnCommand\_Dialog();

afx\_msg void OnCommand\_Exit();

afx\_msg void OnCommand\_Help();

afx\_msg void OnPaint();

// Установка строки, отображаемой в верхней

// части окна

virtual void SetInfoStr(CString str);

private:

CString infoStr;

// Строка, отображаемая в верхней части окна

DECLARE\_MESSAGE\_MAP()

};

// Класс приложения

class CApp: public CWinApp {

public:

BOOL InitInstance();

};

// Класс диалогового окна

class CSampleDialog: public CDialog {

public:

// DialogName -- идентификатор диалога в ресурсах,

// Owner -- окно-владелец (если NULL, то главное окно)

CSampleDialog(char \*DialogName, CWnd \*Owner = NULL);

afx\_msg void OnCommand\_Red();

afx\_msg void OnCommand\_Green();

afx\_msg void OnCancel();

private:

DECLARE\_MESSAGE\_MAP()

};

**Dialogs.cpp**

#include "stdafx.h"

#include "Dialogs.hpp"

#include "resource.h"

CApp App;

//-------------------------------------------------------

CMainWin::CMainWin(CString Title, int HowShow)

:infoStr("")

{

RECT rect;

rect.top = 10; rect.left = 50;

rect.bottom = 460, rect.right = 630;

this->Create(0, Title, WS\_OVERLAPPEDWINDOW, rect, 0, "DIALOGMENU");

this->ShowWindow(HowShow);

this->UpdateWindow();

this->LoadAccelTable("DIALOGMENU");

}

afx\_msg void CMainWin::OnCommand\_Dialog()

{

CSampleDialog sampleDialog("SAMPLEDIALOG", this);

int result = sampleDialog.DoModal();

char s[20];

sprintf(s, "%i", result);

this->SetInfoStr(infoStr + ". " +

"Функция CDialog::DoModal() возвратила " + CString(s));

}

afx\_msg void CMainWin::OnCommand\_Exit()

{

this->MessageBox("Завершение приложения.",

"Dialogs", MB\_ICONEXCLAMATION);

this->SendMessage(WM\_CLOSE);

}

afx\_msg void CMainWin::OnCommand\_Help()

{

this->MessageBox("Программа, использующая диалог.\n"

"Написана с помощью MFC 4.0.",

"Dialogs", MB\_ICONINFORMATION);

}

afx\_msg void CMainWin::OnPaint()

{

CPaintDC paintDC(this);

paintDC.TextOut(10, 10, infoStr);

}

void CMainWin::SetInfoStr(CString str)

{

infoStr = str;

this->InvalidateRect(0);

}

// Карта откликов на сообщения главного окна

BEGIN\_MESSAGE\_MAP(CMainWin, CFrameWnd)

ON\_COMMAND(IDM\_DIALOG, OnCommand\_Dialog)

ON\_COMMAND(IDM\_HELP, OnCommand\_Help)

ON\_COMMAND(IDM\_EXIT, OnCommand\_Exit)

ON\_WM\_PAINT()

END\_MESSAGE\_MAP()

//---------------------------------------------------

BOOL CApp::InitInstance()

{

m\_pMainWnd = new CMainWin("Dialogs - приложение с диалогом",

SW\_RESTORE);

return TRUE;

}

//---------------------------------------------------

CSampleDialog::CSampleDialog(char \*DialogName, CWnd \*Owner)

:CDialog(DialogName, Owner)

{

}

afx\_msg void CSampleDialog::OnCommand\_Red()

{

// Член-функция CWnd::GetOwner() возвращает указатель

// на окно-собственник данного окна.

// Так как в данном случае это главное окно,

// мы проводим преобразование к CMainWin

// и вызываем метод SetInfoStr()

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Red'");

}

afx\_msg void CSampleDialog::OnCommand\_Green()

{

// См. комментарий в CSampleDialog::OnCommand\_Red()

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Green'");

}

afx\_msg void CSampleDialog::OnCancel()

{

// См. комментарий в CSampleDialog::OnCommand\_Red()

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Cancel'");

// Закрываем диалог, только если пользователь

// подтвердил намерение.

// Значение 7 будет возвращено DoModal().

int response = this->MessageBox("Вы уверены ?",

"Cancel", MB\_YESNO);

if(response == IDYES)

this->EndDialog(7);

}

// Карта откликов на сообщения диалога

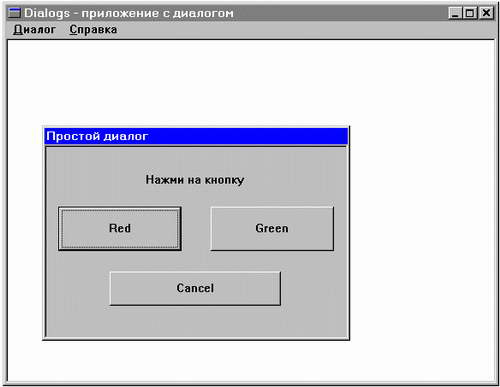
BEGIN\_MESSAGE\_MAP(CSampleDialog, CDialog)

ON\_COMMAND(IDC\_RED, OnCommand\_Red)

ON\_COMMAND(IDC\_GREEN, OnCommand\_Green)

END\_MESSAGE\_MAP()

//------------------------------------------------

  
Рис. 7. Пример приложения с диалогом

Инициализация диалога

Часто различные переменные и элементы управления, связанные с диалогом, должны быть инициализированы до того, как диалог будет отображен. Чтобы позволить диалогу выполнить подобные действия, Windows автоматически посылает ему сообщение WM\_INITDIALOG в момент создания. При получении такого сообщения MFC автоматически вызывает член-функцию OnInitDialog(), которая является стандартным обработчиком, определенным в классе CDialog. Эта функция должна переопределяться в программе, если необходимо выполнение различных инициализационных действий. Прототип функции такой:

virtual BOOL CDialog::OnInitDialog();

Эта функция вызывается до того, как диалог будет отображен. Она должна возвращать TRUE, чтобы Windows могла передать фокус ввода (т. е. сделать активным) на первый элемент управления в окне.

Первым действием в переопределенной функции должен быть вызов функции CDialog::OnInitDialog().

В дальнейшем мы будем использовать эту функцию в примерах.

Списки

Список является одним из наиболее распространенных элементов управления. В MFC работа со списком осуществляется через класс CListBox.

Основы работы со списками

Списки являются элементами управления, требующими двустороннего взаимодействия между ними и программой. То есть список может как посылать, так и принимать сообщения. В частности, сообщения посылаются списку при его инициализации. Сюда входит передача набора строк, которые будут отображены в окне списка (по умолчанию список создается пустым). Когда список инициализирован, он посылает сообщения о действиях, произведенных с ним пользователем.

Прием идентификационных кодов списка

Список может генерировать сообщения различных типов. Например, сообщения посылаются при двойном щелчке на элементе списка, при потере списком фокуса ввода и при выборе другого элемента из списка. Каждое такое событие описывается идентификационным кодом. Этот код является частью сообщения WM\_COMMAND. Некоторые другие элементы также используют идентификационные коды.

Единственный код, которым мы воспользуемся, называется LBN\_DBLCLK. Он посылается, когда пользователь выполняет двойной щелчок на элементе списка. (При определении списка в ресурсах должна быть установлена опция Notify, чтобы он мог генерировать это сообщение.) Когда выбор произведен, необходимо запросить список, чтобы узнать о том, какой элемент выбран.

Для обработки сообщения LBN\_DBLCLK необходимо поместить его обработчик в карту сообщений. Но это будет не макрос ON\_COMMAND(). Вместо этого в данном случае используются специальные макрокоманды. Для нашего сообщения это будет ON\_LBN\_DBLCLK(). Она имеет такой вид:

ON\_LBN\_DBLCLK(ИдентификаторСписка,

ИмяОбработчика)

Многие сообщения обрабатываются подобным образом. Названия всех макросов для таких сообщений начинаются с префикса ON\_LBN\_.

Передача сообщений списку

В традиционных Windows-программах сообщения посылаются элементам управления с помощью API-функций, например SendDlgItemMessage(). Но в программах на MFC для этих целей применяются соответствующие функции-члены. Они автоматически посылают нужное сообщение элементу управления. В этом заключается преимущество MFC по сравнению с традиционным методом программирования.

Списку может быть послано несколько разных сообщений. Для каждого класс CListBox содержит отдельную член-функцию. Например, рассмотрим следующие функции:

int CListBox::AddString(LPCSTR StringToAdd);

int CListBox::GetCurSel() const;

int CListBox::GetText(int Index, LPCSTR StringVariable);

Функция AddString() вставляет указанную строку в список. По умолчанию она вставляется в конец списка. Начало списка имеет индекс 0. Функция GetCurSel() возвращает индекс текущего выделенного элемента. Если ни один элемент не выбран, возвращает LB\_ERR. Функция GetText() получает строку, связанную с указанным индексом. Строка копируется в символьный массив по адресу StringVariable.

Получение указателя на список

Функции CListBox работают с объектами CListBox. Значит, необходимо получить указатель на объект списка, что делается с помощью функции GetDlgItem(), являющейся членом класса CWnd:

СWnd \*CWnd::GetDlgItem(int ItemIdentifier) const;

Эта функция возвращает указатель на объект, чей идентификатор передан как параметр. Если такой объект не существует, то возвращается 0. Необходимо понимать, что указатель, возвращаемый функцией, является временным. Он действителен только в пределах текущего обработчика.

Значение, возвращенное функцией, должно быть приведено к типу указателя на конкретный класс управления. Например, в нашем случае это тип CListBox\*.

Инициализация списка

Посколькlу по умолчанию список создается пустым, он должен инициализироваться каждый раз, когда отображается диалог. Для этого необходимо переопределить функцию OnInitDialog(), в которой в список добавлялись бы строки. Если при добавлении элементов в список их число превысит то, которое помещается в окне списка, то в этом окне автоматически появится вертикальная полоса прокрутки.

Пример программы

Мы расширим предыдущий пример программы и добавим в диалог список. Выбор элемена списка осуществляется либо с помощью двойного щелчка на элементе, либо при нажатии на кнопку "Выбери фрукт". Обратите внимание, что в обоих случаях вызывается один и тот же обработчик, который сначала получает адрес объекта списка, затем получает номер выделенного пункта, а по номеру получает выделенную строку. Строка печатается в окне.

**Listbox.hpp**

#include "stdafx.h"

#include "resource.h"

// Класс основного окна

class CMainWin: public CFrameWnd {

public:

CMainWin(CString Title, int HowShow);

afx\_msg void OnCommand\_Dialog();

afx\_msg void OnCommand\_Exit();

afx\_msg void OnCommand\_Help();

afx\_msg void OnPaint();

virtual void SetInfoStr(CString str);

private:

CString infoStr;

DECLARE\_MESSAGE\_MAP()

};

// Класс приложения

class CApp: public CWinApp {

public:

BOOL InitInstance();

};

// Класс диалога

class CSampleDialog: public CDialog {

public:

CSampleDialog(char \*DialogName, CWnd \*Owner = NULL);

BOOL OnInitDialog();

afx\_msg void OnCommand\_Red();

afx\_msg void OnCommand\_Green();

afx\_msg void OnSelectFruit();

afx\_msg void OnCancel();

private:

DECLARE\_MESSAGE\_MAP()

};

**Listbox.cpp**

#include "stdafx.h"

#include <stdlib.h>

#include "Listbox.hpp"

#include "Resource.h"

//------------------------------------------------

// Реализация CMainWin

CMainWin::CMainWin(CString Title, int HowShow)

:infoStr("")

{

RECT rect;

rect.top = 10; rect.left = 50;

rect.bottom = 460, rect.right = 630;

this->Create(0, Title, WS\_OVERLAPPEDWINDOW, rect, 0,

"MENU");

this->ShowWindow(HowShow);

this->UpdateWindow();

this->LoadAccelTable("DIALOGMENU");

}

afx\_msg void CMainWin::OnCommand\_Dialog()

{

CSampleDialog sampleDialog("SAMPLEDIALOG");

sampleDialog.DoModal();

this->SetInfoStr("Диалог закрыт.");

}

afx\_msg void CMainWin::OnCommand\_Exit()

{

this->SendMessage(WM\_CLOSE);

}

afx\_msg void CMainWin::OnCommand\_Help()

{

this->MessageBox("Демонстрация элемента управления Listbox",

"Listbox", MB\_ICONINFORMATION);

}

afx\_msg void CMainWin::OnPaint()

{

CPaintDC paintDC(this);

paintDC.TextOut(10, 10, infoStr, infoStr.GetLength());

}

void CMainWin::SetInfoStr(CString str)

{

infoStr = str;

this->InvalidateRect(0);

}

BEGIN\_MESSAGE\_MAP(CMainWin, CFrameWnd)

ON\_COMMAND(IDM\_DIALOG, OnCommand\_Dialog)

ON\_COMMAND(IDM\_HELP, OnCommand\_Help)

ON\_COMMAND(IDM\_EXIT, OnCommand\_Exit)

ON\_WM\_PAINT()

END\_MESSAGE\_MAP()

//----------------------------------------------

BOOL CApp::InitInstance()

{

m\_pMainWnd = new CMainWin("Listbox", SW\_RESTORE);

return TRUE;

}

//----------------------------------------------

CSampleDialog::CSampleDialog(char \*DialogName, CWnd \*Owner)

:CDialog(DialogName, Owner)

{

}

afx\_msg void CSampleDialog::OnCommand\_Red()

{

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Red'");

}

afx\_msg void CSampleDialog::OnCommand\_Green()

{

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Green'");

}

// Инициализация диалога

BOOL CSampleDialog::OnInitDialog()

{

CDialog::OnInitDialog();

// Вызвать метод базового класса

// Получить объект списка Listbox

CListBox \*listBoxPtr = (CListBox \*)

(this->GetDlgItem(IDC\_LB1));

// Добавить строки в список

listBoxPtr->AddString("Яблоко");

listBoxPtr->AddString("Слива");

listBoxPtr->AddString("Груша");

listBoxPtr->AddString("Апельсин");

listBoxPtr->AddString("Мандарин");

listBoxPtr->AddString("Абрикос");

listBoxPtr->AddString("Персик");

listBoxPtr->AddString("Ананас");

return TRUE;

}

// Обработчик сообщения выбора элемента

// (двойной щелчок на списке или нажатие

// специальной кнопки)

afx\_msg void CSampleDialog::OnSelectFruit()

{

CListBox \*listBoxPtr = (CListBox \*) (this->GetDlgItem(IDC\_LB1));

// Получить номер текущего выделенного элемента списка

int index = listBoxPtr->GetCurSel();

// Если результат LB\_ERR, то ни один элемент еще не выделен

if(index == LB\_ERR) {

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Ни один фрукт не выделен");

}

else {

char s[100];

// Получить текст выделенного элемента

listBoxPtr->GetText(index, s);

CString infoStr;

infoStr.Format("Выбран фрукт %s с индексом %i", s,

index);

// Вывести сообщение

((CMainWin \*) (this->GetOwner()))

->SetInfoStr(infoStr);

}

}

afx\_msg void CSampleDialog::OnCancel()

{

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Cancel'");

this->EndDialog(TRUE);

}

BEGIN\_MESSAGE\_MAP(CSampleDialog, CDialog)

ON\_COMMAND(IDC\_RED, OnCommand\_Red)

ON\_COMMAND(IDC\_GREEN, OnCommand\_Green)

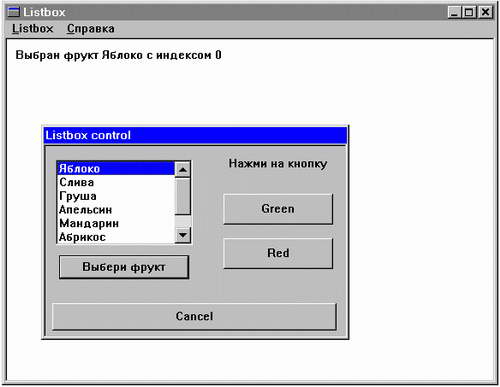
ON\_COMMAND(IDC\_SELFRUIT, OnSelectFruit)

ON\_LBN\_DBLCLK(IDC\_LB1, OnSelectFruit) // Сообщение от списка

END\_MESSAGE\_MAP()

//------------------------------------------------

CApp App;

  
Рис. 8. Пример приложения со списком

Поле ввода

Это последний элемент управления, который мы рассмотрим в качестве введения в диалоги и элементы управления. Поля ввода используются очень широко, так как дают возможность ввести строку по своему усмотрению. Даже зная, как создавать только поля ввода и списки, уже можно писать полезные программы.

Поля ввода принимают многие сообщения и сами могут генерировать несколько типов сообщений. Однако обычно отвечать на большинство из них нет необходимости. Как Вы скоро убедитесь, поля ввода самостоятельно выполняют большинство функций редактирования. Для этого не требуется взаимодействия с программой. Нужно только решить, когда затребовать содержимое поля ввода.

Для получения текущего содержимого поля вода, состоящего из одной строки, используется функция GetWindowText(). Ее прототип таков:

int CWnd::GetWindowText(LPSTR StringVariable,

int MaxStringLen) const;

В результате ее выполнения содержимое поля ввода будет скопировано в строку по адресу StringVariable. Эта функция позволяет получить текст, связанный с любым окном или элементом управления. Применительно к обычному окну функция получает заголовок окна.

В момент создания поле ввода является пустым. Для инициализации его содержимым используется еще одна функция-член класса CWnd - SetWindowText(). Она отображает строку в элементе управления, который вызвал эту функцию. Вот ее прототип:

void CWnd::SetWindowText(LPCSTR String);

Пример программы с полем ввода

**edbox.hpp**

#include "stdafx.h"

#include "resource.h"

// Класс основного окна

class CMainWin: public CFrameWnd {

public:

CMainWin(CString Title, int HowShow);

afx\_msg void OnCommand\_Dialog();

afx\_msg void OnCommand\_Exit();

afx\_msg void OnCommand\_Help();

afx\_msg void OnPaint();

virtual void SetInfoStr(CString str);

private:

CString infoStr;

DECLARE\_MESSAGE\_MAP()

};

// Класс приложения

class CApp: public CWinApp {

public:

BOOL InitInstance();

};

// Класс диалога

class CSampleDialog: public CDialog {

public:

CSampleDialog(char \*DialogName, CWnd \*Owner);

BOOL OnInitDialog();

afx\_msg void OnCommand\_Red();

afx\_msg void OnCommand\_Green();

afx\_msg void OnCommand\_EditOK();

// Здесь читаются данные поля ввода

afx\_msg void OnSelectFruit();

afx\_msg void OnCancel();

private:

DECLARE\_MESSAGE\_MAP()

};

**edbox.hpp**

#include "stdafx.h"

#include <stdlib.h>

#include "Edbox.hpp"

#include "resource.h"

// Реализация

//-------------------------------------------

CMainWin::CMainWin(CString Title, int HowShow)

:infoStr("")

{

RECT rect;

rect.top = 10; rect.left = 50;

rect.bottom = 460, rect.right = 630;

this->Create(0, Title, WS\_OVERLAPPEDWINDOW, rect, 0,

"DIALOGMENU");

this->ShowWindow(HowShow);

this->UpdateWindow();

this->LoadAccelTable("DIALOGMENU");

}

afx\_msg void CMainWin::OnCommand\_Dialog()

{

CSampleDialog sampleDialog("SAMPLEDIALOG", this);

sampleDialog.DoModal();

this->SetInfoStr("Диалог закрыт");

}

afx\_msg void CMainWin::OnCommand\_Exit()

{

this->SendMessage(WM\_CLOSE);

}

afx\_msg void CMainWin::OnCommand\_Help()

{

this->MessageBox("Программа, использующая Editbox\n",

"Editbox", MB\_ICONINFORMATION);

}

afx\_msg void CMainWin::OnPaint()

{

CPaintDC paintDC(this);

paintDC.TextOut(10, 10, infoStr);

}

void CMainWin::SetInfoStr(CString str)

{

infoStr = str;

this->InvalidateRect(0);

}

BEGIN\_MESSAGE\_MAP(CMainWin, CFrameWnd)

ON\_COMMAND(IDM\_DIALOG, OnCommand\_Dialog)

ON\_COMMAND(IDM\_HELP, OnCommand\_Help)

ON\_COMMAND(IDM\_EXIT, OnCommand\_Exit)

ON\_WM\_PAINT()

END\_MESSAGE\_MAP()

//----------------------------------------------

BOOL CApp::InitInstance()

{

m\_pMainWnd = new CMainWin("Editbox", SW\_RESTORE);

return TRUE;

}

//----------------------------------------------

CSampleDialog::CSampleDialog(char \*DialogName, CWnd \*Owner)

:CDialog(DialogName, Owner)

{

}

BOOL CSampleDialog::OnInitDialog()

{

CDialog::OnInitDialog();

CListBox \*listBoxPtr = (CListBox \*) (this->GetDlgItem(IDC\_LB1));

listBoxPtr->AddString("Яблоко");

listBoxPtr->AddString("Слива");

listBoxPtr->AddString("Груша");

listBoxPtr->AddString("Апельсин");

listBoxPtr->AddString("Мандарин");

listBoxPtr->AddString("Абрикос");

listBoxPtr->AddString("Персик");

listBoxPtr->AddString("Ананас");

// Получить объект Editbox

CEdit \*editBoxPtr = (CEdit \*) (this->GetDlgItem(IDC\_EDIT1));

// Установить текст в Editbox

editBoxPtr->SetWindowText("Вася");

return TRUE;

}

afx\_msg void CSampleDialog::OnCommand\_Red()

{

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Red'");

}

afx\_msg void CSampleDialog::OnCommand\_Green()

{

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Нажата кнопка 'Green'");

}

// Здесь читается текст из Editbox

afx\_msg void CSampleDialog::OnCommand\_EditOK()

{

// Получить объект Editbox

CEdit \*pCEdit = (CEdit \*) (this->GetDlgItem(IDC\_EDIT1));

char s[1024];

// Прочитать текст из Editbox

pCEdit->GetWindowText(s, sizeof(s) / sizeof(s[0]) - 1);

// Вывести информацию в окно

((CMainWin \*) (this->GetOwner()))

->SetInfoStr(CString("Введена строка \"")

+ s + "\"");

}

afx\_msg void CSampleDialog::OnSelectFruit()

{

CListBox \*listBoxPtr = (CListBox \*) (this->GetDlgItem(IDC\_LB1));

int index = listBoxPtr->GetCurSel();

if(index == LB\_ERR) {

((CMainWin \*) (this->GetOwner()))

->SetInfoStr("Ни один фрукт не выделен");

}

else {

char s[100];

listBoxPtr->GetText(index, s);

CString infoStr;

infoStr.Format("Выбран фрукт %s с индексом %i", s,

index);

((CMainWin \*) (this->GetOwner()))->SetInfoStr(infoStr);

}

}

afx\_msg void CSampleDialog::OnCancel()

{

this->EndDialog(TRUE);

}

BEGIN\_MESSAGE\_MAP(CSampleDialog, CDialog)

ON\_COMMAND(IDC\_RED, OnCommand\_Red)

ON\_COMMAND(IDC\_GREEN, OnCommand\_Green)

ON\_COMMAND(IDC\_EDITOK, OnCommand\_EditOK)

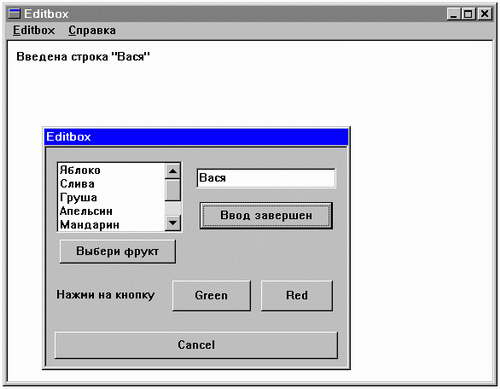
ON\_COMMAND(IDC\_SELFRUIT, OnSelectFruit)

ON\_LBN\_DBLCLK(IDC\_LB1, OnSelectFruit)

END\_MESSAGE\_MAP()

//-------------------------------------------------

CApp App;

  
Рис. 9. Приложение со строкой редактирования в диалоге

Немодальные диалоги

Немодальные диалоги получают сообщения параллельно с основным окном bприложения. То есть, как минимум два окна будут одновременно активными. Поэтому работа с немодальными диалогами требует чуть больше усилий, должны быть выполнены дополнительные операции.

Для создания немодального диалога нужно выполнить два действия. Во-первых, нужно создать "пустой" объект диалога, то есть не связанный с шаблоном из ресурсов. Привязка к ресурсам осуществляется через функцию Create(). Рассмотрим этот процесс подробнее.

Для создания объекта немодального диалога нужно использовать конструктор CDialog::CDialog() без параметров. Он объявлен как protected-член класса. Это означает, что он может быть вызван только изнутри член-функции порожденного класса. Это сделано для того, чтобы программист обязательно определял свой порожденный класс для немодального диалога, и определял в нем дополнительные операции для немодального диалога.

Когда экземпляр создан, он привязывается к ресурсам с помощью функций:

BOOL CDialog::Create(LPCSTR ResourceName, CWnd \*Owner = 0);

BOOL CDialog::Create(UINT ResourceId, CWnd \*Owner = 0);

Первый параметр определяет идентификатор диалога в ресурсах. Второй параметр, как обычно, определяет окно-собственник для диалога.

Необходимо помнить о том, что объект немодального диалога должен существовать в течение всего времени использования диалога. Функция Create() отображает окно и после этого немедленно завершается. А объект окна должен существовать.

В отличие от модальных окон, немодальные не становятся автоматически видимыми при вызове. Чтобы диалог сразу был видимым, нужно в ресурсном редакторе установить опцию Visible. Или можно использовать функцию ShowWindow().

Для закрытия немодального диалога нужно использовать функцию DestroyWindow(). Это означает, что функции OnCancel() и/или OnOK() должны быть переопределены.

Пример программы с немодальным диалогом

**ModelessDialog.hpp**

#include "stdafx.h"

#include "resource.h"

class CMainWin;

// Класс немодального диалога

class CSampleDialog: public CDialog {

public:

CSampleDialog();

// Функция привязки диалога к ресурсам

BOOL Create(LPCSTR DialogName, CWnd \*Owner = 0);

BOOL OnInitDialog();

afx\_msg void OnCommand\_Red();

afx\_msg void OnCommand\_Green();

afx\_msg void OnCommand\_EditOK();

afx\_msg void OnSelectFruit();

afx\_msg void OnCancel();

private:

// Переменная, показывающая, используется ли в

// данный момент диалог

BOOL inUse;

CMainWin \*owner;

DECLARE\_MESSAGE\_MAP()

};

class CMainWin: public CFrameWnd {

public:

CMainWin(CString Title, int HowShow);

afx\_msg void OnCommand\_Dialog();

afx\_msg void OnCommand\_Exit();

afx\_msg void OnCommand\_Help();

afx\_msg void OnPaint();

virtual void SetInfoStr(CString str);

private:

// Экземпляр объекта диалога -

// существует все время, пока существует

// основное окно

CSampleDialog dialog;

CString infoStr;

DECLARE\_MESSAGE\_MAP()

};

class CApp: public CWinApp {

public:

BOOL InitInstance();

};

**ModelessDialog.cpp**

#include "stdafx.h"

#include <stdlib.h>

#include "ModelessDialog.hpp"

#include "resource.h"

static CApp App;

//--------------------------------------------------

CMainWin::CMainWin(CString Title, int HowShow)

// Вызываем конструктор без параметров класса диалога

:dialog(), infoStr("")

{

RECT rect;

rect.top = 10; rect.left = 50;

rect.bottom = 460, rect.right = 630;

this->Create(0, Title, WS\_OVERLAPPEDWINDOW, rect, 0,

"DIALOGMENU");

this->ShowWindow(HowShow);

this->UpdateWindow();

this->LoadAccelTable("DIALOGMENU");

}

afx\_msg void CMainWin::OnCommand\_Dialog()

{

dialog.Create("SAMPLEDIALOG", this);

}

afx\_msg void CMainWin::OnCommand\_Exit()

{

this->SendMessage(WM\_CLOSE);

}

afx\_msg void CMainWin::OnCommand\_Help()

{

this->MessageBox("Программа, использующая немодальный диалог.",

"О программе", MB\_ICONINFORMATION);

}

afx\_msg void CMainWin::OnPaint()

{

CPaintDC paintDC(this);

paintDC.TextOut(10, 10, infoStr);

}

void CMainWin::SetInfoStr(CString str)

{

infoStr = str;

this->InvalidateRect(0);

}

BEGIN\_MESSAGE\_MAP(CMainWin, CFrameWnd)

ON\_COMMAND(IDM\_DIALOG, OnCommand\_Dialog)

ON\_COMMAND(IDM\_HELP, OnCommand\_Help)

ON\_COMMAND(IDM\_EXIT, OnCommand\_Exit)

ON\_WM\_PAINT()

END\_MESSAGE\_MAP()

//----------------------------------------------------

BOOL CApp::InitInstance()

{

m\_pMainWnd = new CMainWin("Приложение с немодальным

диалогом", SW\_RESTORE);

return TRUE;

}

//----------------------------------------------------

CSampleDialog::CSampleDialog()

:CDialog()

{

// Объект создан, но пока не используется

inUse = FALSE;

}

BOOL CSampleDialog::Create(LPCSTR DialogName, CWnd \*Owner)

{

// Если диалог уже используется (был отображен),

// то только отобразить его

if(inUse) {

this->ShowWindow(SW\_SHOW);

return TRUE;

}

inUse = TRUE; // Диалог используется

// Создать диалог и получить результат создания

BOOL success = (CDialog::Create(DialogName, Owner) != FALSE);

owner = (CMainWin \*) Owner; // Собственник

return success;

}

BOOL CSampleDialog::OnInitDialog()

{

CDialog::OnInitDialog();

CListBox \*listBoxPtr = (CListBox \*) (this->GetDlgItem(IDC\_LB1));

listBoxPtr->AddString("Яблоко");

listBoxPtr->AddString("Слива");

listBoxPtr->AddString("Груша");

listBoxPtr->AddString("Апельсин");

listBoxPtr->AddString("Мандарин");

listBoxPtr->AddString("Абрикос");

listBoxPtr->AddString("Персик");

listBoxPtr->AddString("Ананас");

CEdit \*editBoxPtr = (CEdit \*) (this->GetDlgItem(IDC\_EDIT1));

editBoxPtr->SetWindowText("Вася");

return TRUE;

}

afx\_msg void CSampleDialog::OnCommand\_Red()

{

owner->SetInfoStr("Нажата кнопка 'Red'");

}

afx\_msg void CSampleDialog::OnCommand\_Green()

{

owner->SetInfoStr("Нажата кнопка 'Green'");

}

afx\_msg void CSampleDialog::OnCommand\_EditOK()

{

CEdit \*pCEdit = (CEdit \*) (this->GetDlgItem(IDC\_EDIT1));

char s[1024];

pCEdit->GetWindowText(s, sizeof(s) / sizeof(s[0]) - 1);

owner->SetInfoStr(CString("Введена строка \"")

+ s + "\"");

}

afx\_msg void CSampleDialog::OnSelectFruit()

{

CListBox \*listBoxPtr = (CListBox \*)

(this->GetDlgItem(IDC\_LB1));

int index = listBoxPtr->GetCurSel();

if(index == LB\_ERR) {

owner->SetInfoStr("Ни один фрукт не выделен");

}

else {

char s[100];

listBoxPtr->GetText(index, s);

CString infoStr;

infoStr.Format("Выбран фрукт %s с индексом %i", s,

index);

owner->SetInfoStr(infoStr);

}

}

afx\_msg void CSampleDialog::OnCancel()

{

owner->SetInfoStr("Нажата кнопка 'Cancel'");

// Удалить диалог

this->DestroyWindow();

// Больше диалог не используется

inUse = FALSE;

}

BEGIN\_MESSAGE\_MAP(CSampleDialog, CDialog)

ON\_COMMAND(IDC\_RED, OnCommand\_Red)

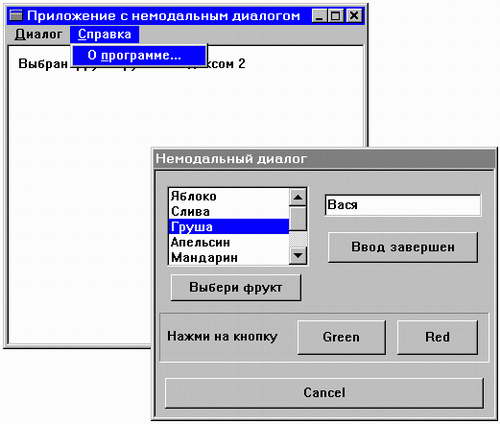
ON\_COMMAND(IDC\_GREEN, OnCommand\_Green)

ON\_COMMAND(IDC\_EDITOK, OnCommand\_EditOK)

ON\_COMMAND(IDC\_SELFRUIT, OnSelectFruit)

ON\_LBN\_DBLCLK(IDC\_LB1, OnSelectFruit)

END\_MESSAGE\_MAP()

  
Рис. 10. Пример приложения с немодальным диалогом