МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра интеллектуальных информационных технологий

Методическое пособие

"ОСНОВЫ СОЗДАНИЯ WINDOWS-ПРИЛОЖЕНИЙ В СИСТЕМЕ MICROSOFT VISUAL STUDIO C++. Процедурный стиль"

для студентов специальности 1-40 03 01 «Искусственный интеллект»

Часть 2

БРЕСТ 2008

УДК 681.3 (075.8)

Целью пособия является знакомство студентов с базовыми понятиями оконных приложений и каркасного программирования в системе Microsoft Visual Studio C++. Издание в 2-х частях. Часть 2.

Рекомендовано к изданию редакционно-издательским советом Брестского государственного технического университета

Составители: Г.Л. Муравьев, доцент, к.т.н.
В.И. Хвещук, доцент, к.т.н.
Ю. В. Савицкий, доцент, к.т.н.

Рецензент: заведующий кафедрой информатики и прикладной математики Брестского государственного университета им. А.С. Пушкина, к.ф-м.н. Савчук В.Ф.

1.ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

1.1. Сообщения. Обработчики сообщений

Через операционную систему Windows проходят и направляются на обработку в приложения разнообразные сообщения о событиях, происходящих в программах, устройствах и т.д. Синхронные сообщения, адресуемые приложению и соблюдающие порядок, очередь обработки, образуют очередь сообщений. Асинхронные сообщения, адресуемые приложению (сообщения таймера, перерисовки, завершения работы и др.), обслуживаются вне очереди. Каждое сообщение сопровождается рядом атрибутов, исчерпывающе характеризующих сообщение и необходимых для его обработки (например, системное время момента наступления события, состояние клавиатуры, "мыши", идентификация источника сообщения и т.д.).

Сообщение, адресуемое приложению, предварительно обрабатывается фрагментом главной функции WinMain с целью отправки на исполнение в соответствующие функции-обработчики. Основные сообщения, их названия и прототипы функций для их обработки приведены в таблице ниже.

Таблица 1. Прототипы функций обработки сообщений

Сообщение	Прототип функции обработки сообщения
WM_CHAR	void Cls_OnChar(HWND hwnd, UINT ch, int cRepeat)
WM_COMMAND	void Cls_OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)
WM_CREATE	BOOL Cls_OnCreate(HWND hwnd, CREATESTRUCT FAR* lpCreateStruct)
WM_DESTROY	void Cls_OnDestroy(HWND hwnd)
WM_GETMINMAXINFO	void Cls_ OnGetMinMaxInfo(HWND hwnd, MINMAXINFO FAR* lpMinMaxInfo)
WM_NITDIALOG	BOOL Cls_OninitDialog(HWND hwnd, HWND hwndFocus, LPARAM lParam)
WM_KEYDOWN	void Cls_OnKeyDown(HWND hwnd, UINT vk, BOOL fDown, int cRepeat, UINT flags)
WM_KEYUP	void Cls_OnKeyUp(HWND hwnd, UINT vk, BOOL fDown, int cRepeat, UINT flags)
WM_KILLFOCUS	void Cls_OnKillFocus(HWND hwnd, HWND hwndNewFocus)
WM_LBUTTONDOWN, WM_LBUTTONDBLCLK	void Cls_OnLButtonDown(HWND hwnd, BOOL fDoubleClick, int x, int y, UINT keyFlags)
WM_LBUTTONUP	void Cls_OnLButtonUp(HWND hwnd, int x, int y, UINT keyFlags)
WM_MOUSEMOVE	void Cls_OnMouseMove(HWND hwnd, int x, int y, UINT keyFlags)
WM_NOTIFY	BOOL Cls_OnNotify(HWND hwnd, INT idCtrl, NMHDR* pnmh)
WM_PAINT	void Cls_OnPaint(HWND hwnd)
WM_QUIT	void Cls_OnQuit(HWND hwnd, int exitCode)
WM_RBUTTONDOWN, WM_RBUTTONDBLCLK	void Cls_OnRButtonDown(HWND hwnd, BOOL fDoubleClick, int x, int y, UINT keyFlags)
WM_RBUTTONUP	void Cls_OnRButtonUp(HWND hwnd, int x, int y, UINT flags)
WM_SETCURSOR	BOOL Cls_OnSetCursor(HWND hwnd, HWND hwndCursor, UINT codeHitTest, UINT msg)
WM_SETFOCUS	void Cls_OnSetFocus(HWND hwnd, HWND hwndOldFocus)
WM_SHOWWINDOW	void Cls_OnShowWindow(HWND hwnd, BOOL fShow, UINT status)
WM_SIZE	void Cls_OnSize(HWND hwnd, UINT state, int cx, int cy)

Продолжение таблицы 1

WM_SYSCHAR	void Cls_OnSysChar(HWND hwnd, UINT ch, int cRepeat)
WM_SYSCOMMAND	void Cls_OnSysCommand(HWND hwnd, UINT cmd, int x, int y)
WM_SYSKEY	void Cls_OnSysKey(HWND hwnd, UINT vk, BOOL fDown, int cRepeat, UINT flags)
WM_TIMER	void Cls_OnTimer(HWND hwnd, UINT id)

1.2. Ресурсы Windows-приложений

В операционной системе Windows специальные графические объекты, имеющие облик и используемые для организации и поддержки графических интерфейсов пользователя, называются ресурсами. Как правило, это окна различных типов. Это диалоговые окна (типовые и пользовательские), служащие подложкой, где могут располагаться различные элементы управления и сами элементы управления. Стандартные диалоговые окна Windows (см. commdlg.h) – часто используемые окна типа Сохранить как, Открыть, Печать, Параметры страницы и др. Основные ЭУ – кнопки, переключатели, списки, комбинированные, графические списки, тексты, групповые рамки, линии прокрутки, линейки с ползунками, окна редактирования, альтернативные кнопки, флажки, индикаторы и т.д.

С пользовательской точки зрения, ресурсы - это изображения графических объектов, формирующих визуальное представление интерфейса приложения. Предназначены для организации канала общения пользователь-приложение, используемого как для ввода информации (данных и управляющих воздействий пользователя) так и вывода информации. Построены в соответствии со стандартом GDI.

С информационной точки зрения, это описания графических объектов, достаточные для генерации соответствующего ресурса. Это текстовые описания (например, описания меню, диалоговых окон), выполненные в терминах специальных декларативных команд. Это описания, представленные в графическом формате (например, пиктограммы, битовые образы). Это описания, представленные настроечными данными в специальных структурах данных (например, окна), достаточными для генерации соответствующего ресурса.

С компонентной (модульной) точки зрения, это файлы описаний ресурсов, не разделяемые с другими приложениями. В текстовом, не откомпилированном виде описания ресурсов хранятся в файле с расширением rc, а в откомпилированном виде в файле с расширением res.

С точки зрения реализация, это автоматически генерируемое программное обеспечение, обеспечивающее визуализацию ресурсов и работу с ними.

1.3. Редакторы ресурсов. Создание ресурсов

В Windows-приложениях для хранения описаний ресурсов используется ресурсный файл типа ResourceScript, который должен быть создан и подключен к приложению командой #include. А сами ресурсы могут создаваться "вручную" (например, путем описания меню в текстовом редакторе), либо с помощью редакторов ресурсов. Последние используются для визуального проектирования ресурсов. Это редакторы меню, диалоговых окон, инструменты для работы со значками, растровыми изображениями и т.п. Доступ к встроенным редакторам ресурсов осуществляется из пункта главного меню Resource.

1. Для создания нового файла ресурсов следует, открыв главное окно среды разработки Visual Studio, выполнить команду добавления в готовый проект соответствующего файла описания ресурсов: пункт меню Project, подпункт Add to Project, New, вкладка Files, тип файла ResourceScript.

2. Для добавления нового ресурса с использованием соответствующего редактора ресурсов следует, открыв главное окно среды разработки Visual Studio, выбрать пункт меню Insert, подпункт Resource. На экран будет выведено окно с перечнем доступных ресурсов (рисунок 1). При выборе типа ресурсов автоматически будет вызван соответствующий редактор ресурсов.

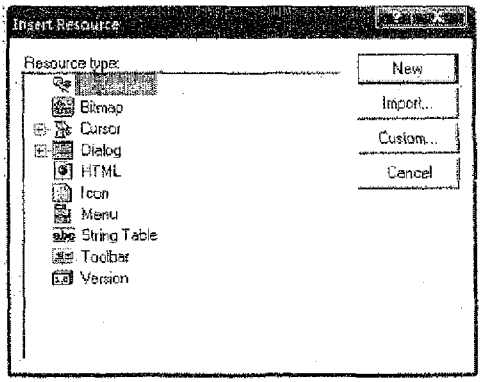

Рис. 1. Типы ресурсов

Это ресурсы следующих типов.

Акселератор (Accelerator) для настройки комбинаций "горячих" клавиш.

Битовый образ (Bitmap) - небольшой по размеру цветной графический объект в виде растрового описания, отображающий окно, часть окна, объекты типа "стрелка", "кисть", "курсор" и т.п. используемый для быстрого вывода соответствующего изображения на экран. Вызываемый при этом редактор графических изображений позволяет создавать и модифицировать растровые изображения пользовательских курсоров, пиктограмм, сохраняемых в файле с расширением *.rc и включаемых в файлы сценариев ресурсов.

Курсор (Cursor), в том числе текстовый для отображения позиции ввода, курсор - указатель "мыши" (см. битовый образ).

Пиктограмма (Icon) – графический объект (см. битовый образ).

Диалоговые окна (Dialog) для описания соответствующих окон и расположенных на них элементах управления. Вызываемый при этом редактор диалоговых окон — это средство разработки графических объектов, позволяющее быстро создавать сложные диалоговые окна с возможностью комбинировать, изменять и настраивать в соответствии с собственными требованиями элементы окна, элементы управления окна. Элементы имеют набор заранее определенных свойств, а их настройка сводится к изменению значений этих свойств.

Текст в формате HTML (HTML).

Меню (Menu) - для создания иерархических пользовательских меню.

Таблица (String Table) - для хранения выводимой текстовой информации. Вызываемый при этом редактор строк предназначен для обработки таблиц строк, представляющих собой ресурс, содержащий список идентификаторов (заголовков), используемых в приложении (одно приложение - одна таблица). Например, здесь могут храниться сообщения, отображаемые в строке состояния. Таблица упрощает изменение языка интерфейса программы, т.к. достаточно перевести на другой язык строки таблицы, не затрагивая код программы.

Панель инструментов (Toolbar).

Информация о версии проекта (Version).

Сами окна (например, с рамкой), включающие клиентскую область, имеют такие собственные ресурсы, как перо, кисть, шрифт. Перо – невидимый до рисования графический объект, применяемый для изображения линий, контуров. Основные атрибуты пера – стиль линии, ее ширина, цвет. Кисть – невидимый до рисования графический объект, применяемый для закраски, заполнения областей изображения. Основные атрибуты кисти – размер, стиль рисования (наполнитель), цвет. Шрифт – графический объект, описываемый как набор символов одного семейства с точки зрения начертания (т.е. одного размера, стиля). Основные атрибуты – толщина, размер, тип начертания (курсив, с подчеркиванием и т.д.).

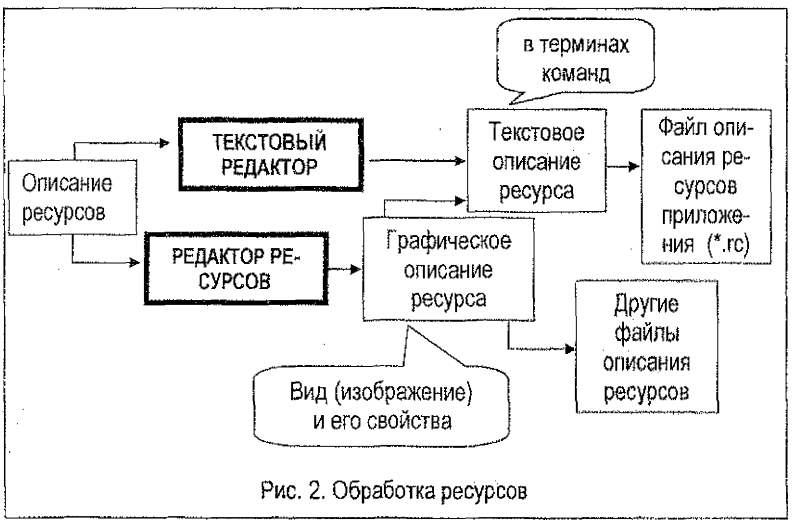

Рис. 2. Обработка ресурсов

3. После того как ресурс создан в редакторе ресурсов или "вручную", компилятор ресурсов считывает ASCII-файл описания ресурсов (*.rc) и создает для компоновщика приложения его двоичный аналог - res-файл.

Порядок обработки ресурсов и компоновки приложения иллюстрируется на рисунках 2, 3, а расширения используемых файлов приведены в таблице 2.

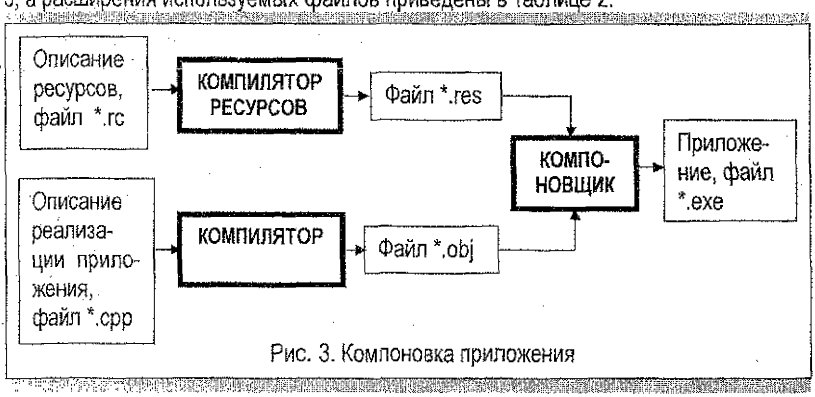

Рис. 3. Компоновка приложения

Таблица 2. Типы файлов, поддерживаемых средой разработки

rc	файл сценария (script)
rct	файл шаблона (template)
res	файл ресурсов
exe	исполняемый файл (редактируется только в операционных системах, построенных на платформе Windows NT. В операционных системах Windows 9x/ME может быть только открыт, но не изменен)
dll	файл динамически подключаемой библиотеки (по возможностям редактирования аналогичен файлу с расширением exe)
bmp	графический файл ("Windows"- формат)
ico, cur	графический файл для хранения изображения пиктограммы

1.4. Ввод и вывод данных

1.4.1. Преобразование типов данных

Поскольку в оконных приложениях при вводе-выводе большинство функций работает с данными, интерпретируемыми как строковые данные, то пользователю нужно самостоятельно приводить данные к нужному типу.

Для преобразования строки в вещественное значение можно использовать функцию **atof** - ПРЕОБРАЗОВАТЬ_В_ВЕЩЕСТВЕННОЕ (*Строка*).

Пример использования функции

```
char MyString[20] = "-123.75";
float MyFloat;
.....
MyFloat = atof (MyString);  .
```

Для преобразования числовых значений в строку можно использовать функцию **wsprintf** - ПРЕОБРАЗОВАТЬ_В_СТРОКУ (*Строка, ШаблонВывода, СписокВывода*).

Пример использования функции

```
char szMyString [80] = " ";
.....
wsprintf (szMyString, "Был год - %d", 1952);  .
```

Для преобразования вещественного значения в строку можно использовать аналогичную функцию **sprintf** либо функцию **_gcvt** ПРЕОБРАЗОВАТЬ_В_СТРОКУ (*ВыводимоеЧисло, ДлинаСтроки, Строка*) с прототипом

char *_gcvt(**double** *Value*, **int** *OunputStringLength*, **char** **OunputString*) .

Пример использования функций

```
#include <stdio.h>
.....
float FloatNumber1 = 12.345;
float FloatNumber2 = -13.345;
char OutputStr[100];
.....
sprintf(OutputStr,"%f",FloatNumber1);
TextOut(hdc, 10,10, 9, strlen(OutputStr));
.....
_gcvt (FloatNumber2, strlen(OutputStr), OutputStr);
TextOut(hdc, 100,100, OutputStr, strlen(OutputStr));  .
```

1.4.2. Ввод данных

Ввод данных для обработки в приложении может производиться, например: - из окошек редактирования (элементов управления) диалоговых окон; - из файлов (с использованием функций традиционных и объектно-ориентированных библиотек ввода-вывода C, C++).

Ввод текста. Выполняется, например, с использованием окошка (поля) редактирования ресурса диалоговое окно и функции **GetDlgItemText** ВВЕСТИ_ТЕКСТ_ИЗ_ПОЛЯ_ДИАЛОГОВОГО_ОКНА (*ДескрипторОкна*, *ДескрипторОкнаРедактирования*, *СтрокаДляВводаДанных*, *ДлинаСтроки*) с прототипом

UINT GetDlgItemText (HWND *hDlg*, **int** *nIDDlgItem*, **LPTSTR** *lpString*, **int** *nMaxCount*),

где используются параметры:

- *ДескрипторОкна диалога*, содержащего окно редактирования *hDlg;*
- *ДескрипторОкнаРедактирования nIDDlgItem;*
- *СтрокаДляВводаДанных lpString;*
- *ДлинаСтроки nMaxCount.*

Пример использования функции в диалоговом окне с hDlg для записи содержимого окошка редактирования IDC_EDIT1, введенного пользователем, в строку RezultString для дальнейшего использования в приложении

GetDlgItemText (hDlg, IDC_EDIT1, RezultString, 15) .

Ввод целых чисел. Выполняется с использованием окошка редактирования ресурса – диалоговое окно и функции **GetDlgItemInt** ВВЕСТИ_ЦЕЛОЕ_ИЗ_ПОЛЯ_ДИАЛОГОВОГО_ОКНА (*ДескрипторОкна*, *ДескрипторПоляРедактирования*, NULL, bool *НаличиеЗнака*) с прототипом

UINT GetDlgItemInt (HWND *hDlg*, **int** *nIDDlgItem*, **BOOL** **lpTranslated*, **BOOL** *bSigned*) ,

где используются аналогичные параметры, а параметр *НаличиеЗнака bSigned* управляет преобразованием числа. Если *НаличиеЗнака* = 1, то результат ввода преобразуется в целое со знаком; если *НаличиеЗнака* = 0, то результат ввода преобразуется в целое без знака.

Пример использования функции

```
int   i;
int   j;
.....
i =  GetDlgItemInt (hDlg, IDC_EDIT1, NULL, 0);
j =  GetDlgItemInt (hDlg, IDC_EDIT1, NULL, 1);  .
```

Ввод вещественных чисел. Производится аналогично вводу строковых данных, но с последующим приведением введенной строки к вещественному типу, например, с помощью функции преобразования atof.

1.4.3. Вывод данных

Типовые средства вывода данных из приложений включают: - функции вывода данных на экран (в пользовательскую, клиентскую часть окна) типа TextOut, DrawText, вывод сообщений в виде окон сообщений MessageBox и другие; - функции вывода данных в файлы (используются традиционные и объектно-ориентированные библиотеки ввода-вывода C, C++).

Ввод-вывод данных требует указания контекста, используемого для этого устройства. Для его хранения используется переменная типа HDC (дескриптор контекста устройства), например, HDC hdc. Так, в функции-обработчике дескриптор получают от ОС Windows при выполнении функции BeginPaint, например,

```
case WM_PAINT:
    hdc = BeginPaint(hWnd, &ps);
```

и освобождают функцией EndPaint(hWnd, &ps). Могут быть использованы также и другие функции, например, GetDC, ReleaseDC. Здесь hWnd – дескриптор окна, &ps – ссылка на структуру типа PAINTSTRUCT (PAINTSTRUCT ps), которая хранит информацию о клиентской части окна.

Вывод текста. Вывод числовых данных требует предварительного преобразования чисел в строковый формат. Вывод выполняется, например, с использованием функции **TextOut** ВЫВЕСТИ_ТЕКСТ (*Окно, КоординатаX, КоординатаY, СтрокаВывода, ДлинаСтроки*).

Пример использования функции

```
char lpszString[ ] = "Строка вывода";
int iStringLength = strlen (lpszString);
int x = 50;
int y = 50;
.....
TextOut (hdc, x, y, lpszString, iStringLength); .
```

Пример использования функции для вывода целых значений от 0 до 10 и их квадратов

```
for (i=0; i<10; i++)
{
    wsprintf (szMyString, " Значение - %d ", i);
    TextOut (hdc, 0, 115 + 15 * (i + 1), szMyString, strlen(szMyString));
    wsprintf (szMyString, "его квадрат - %d",  i * i);
    TextOut (hdc, 140, 115 + 15 * (i + 1), szMyString, strlen(szMyString));
};
```

Вывод окна сообщения. Производится с использованием функции **MessageBox** ВЫВЕСТИ_ОКНО_СООБЩЕНИЯ (*Окно, СтрокаСообщения, НазваниеОкна, СоставОкна*). Прототип функции

int MessageBox (HWND *hWnd*, **LPCTSTR** *lpText*, **LPCTSTR** *lpCaption*, **UINT** MBType**)**.

Параметры:

- *СтрокаСообщения* - указатель на строку С (с нуль-символом), которая содержит выводимое сообщение;
- *НазваниеОкнаСообщения* - указатель на строку С (с нуль-символом), которая содержит заголовок окна сообщения. Если указатель равен NULL, то по умолчанию используется заголовок "Error";
- *СоставОкна* - определяет состав окна сообщения (комбинацию пиктограмм и командных кнопок) и его поведение. Используемые пиктограммы: MB_ICONHAND, MB_ICONSTOP, MB_ICONERROR, MB_ICONQUESTION, MB_ICONEXCLAMATION, MB_ICONWARNING, MB_ICON-ASTERISK, MB_ICONINFORMATION. Используемые комбинации кнопок: MB_ABORTRETRYIGNORE, MB_OK, MB_OKCANCEL, MB_YESNO, MB_RETRYCANCEL, MB_YESNOCANCEL.

Основные значения параметров MessageBox, а также типы возвращаемого результата приведены в ПРИЛОЖЕНИИ 1.

Пример использования функции

```
MessageBox (hWnd, "Пример ", "Демо", MB_OK); .
```

Окно сообщения можно использовать для ввода управляющей информации в виде данных о нажатых кнопках, например,

```
if  (MessageBox (hWnd, " Пример ", "Демо", MB_OKCANCEL) == IDCANCEL)
MessageBox (hWnd, "IDCANCEL", "Демо", MB_OKCANCEL);  .
```

Вывод текста с усиленными возможностями редактирования функцией **DrawText** ИЗОБРАЗИТЬ_ТЕКСТ (*ДескрипторУстройства, АдресСтрокиВывода, ДлинаСтрокиВывода, АдресСтруктурыПрямоугольникаВывода, ФлагиФорматирования*), где *АдресСтруктурыПрямоугольникаВывода* задает прямоугольную часть клиентского окна (описывает ее в структуре типа RECT) для организации вывода. Прототип

int DrawText (HDC *hDC*, **LPCTSTR** *lpString*, **int** *nCount*, **LPRECT** *lpRect*, **UINT** *uFormat*) **.**

Пример использования функции

```
RECT rt;
rt.left = 5;  rt.top = 5;  rt.right = 1000;  rt.bottom = 1000;
.....
GetClientRect (hWnd, &rt);
DrawText (hdc, "Hello", strlen("Hello"), &rt, DT_CENTER);  .
```

2. ОКОННЫЙ ИНТЕРФЕЙС WINDOWS-ПРИЛОЖЕНИЯ

2.1. Использование ресурса "диалоговое окно"

2.1.1. Диалоговое окно в составе главного окна

Интерфейс приложения часто строится на базе масштабируемого окна с рамкой, с клиентской областью (особенно, если предполагается использование клиентской области для вывода текстовой, графической информации), которое может содержать главное меню и обеспечивать запуск других окон, включая диалоговые.

ПРИМЕР. *Создать Windows-приложение, при запуске которого в качестве главного должно выводиться масштабируемое окно с рамкой и клиентской областью, а за ним сразу же автоматически диалоговое окно с кнопками. При этом, главное окно остается на экране, но теряет активность. Нажатие кнопок диалогового окна приводит к его закрытию. Параметры стилей диалоговых окон приведены в* ПРИЛОЖЕНИИ 2. *Примерный вид оконного интерфейса приведен ниже* (рисунок 4).

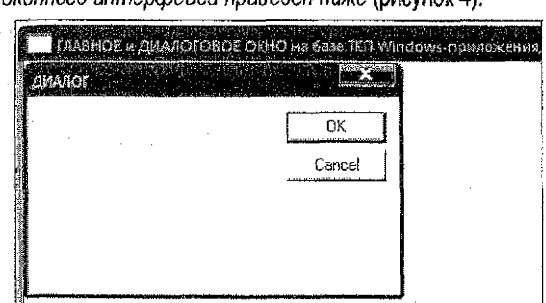

Рис. 4. Оконный интерфейс

ПОРЯДОК (СХЕМА) ВЫПОЛНЕНИЯ ЗАДАЧИ.

1. Создать ТКП (на базе каркаса simple).

2. Описать глобальную переменную (например, HINSTANCE hInstance) для сохранения в ней дескриптора приложения, передаваемого из ОС в функцию WinMain, чтобы обеспечить доступ к дескриптору в любой точке приложения (и за пределами функции WinMain). Убедиться в работоспособности приложения, выполнив его.

3. Создать ресурсный файл и подключить его к проекту:

- создать файл описания ресурсов (или просто файл ресурсов *.rc), для чего выполнить ГМ-Project-Add To Project-New-Files, выбрать тип – Resource Script, а в качестве имени файла задать, например, <ИмяПриложения> (если ИмяПриложения – main, то соответственно после создания файла ресурсов *.rc в папке проекта появятся файлы с названиями main.rc, resource.h);
- подключить файл ресурсов к приложению посредством команды #include "resource.h";
- выполнить приложение (диалоговое окно не появится!).

4. Спроектировать и создать ресурс - диалоговое окно:

- разработать вид и состав окна (здесь окно с названием, системной кнопкой закрытия и двумя пользовательскими кнопками с именами OK и Cancel);
- добавить ресурс в ресурсный файл приложения, используя редактор ресурсов (при необходимости настроить редактор – командой главного меню Tools-Customize вызвать окно Customize, где на вкладке ToolBars включить режим Controls, вызывающий вывод меню элементов управления). Для этого вызвать его командой ГМ-Insert-Resource-Dialog. Создать, используя элементы управления, окно (рисунок 5);

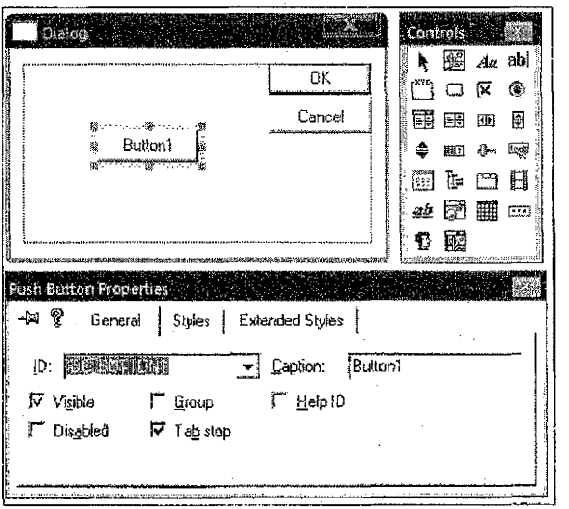

Рис. 5. Редактирование элементов управления

- при необходимости настроить их свойства (здесь три элемента – само окно и две кнопки). Для этого выделить элемент щелчком, а правой клавишей вызвать контекстное меню, в нем режим Properties и сделать необходимые установки;
- каждый элемент имеет свой идентификатор. Чтобы его увидеть, выделить элемент щелчком, а правой клавишей вызвать контекстное меню и в нем режим Properties. Например, в свойствах окна определите его дескриптор - идентификатор ID (это может

быть, например, ID = IDD_DIALOG1, хотя сам дескриптор можно поменять для повышения читаемости программы). Выписать дескрипторы всех элементов управления;

- выполнить приложение (диалоговое окно не появится!).

5. Описать функцию диалогового окна - программу обработчик (имя выбирается произвольно!) его сообщений с прототипом:

LRESULT CALLBACK *ИмяОбработчика* **(HWND, UINT, WPARAM, LPARAM);**

```
LRESULT CALLBACK ИмяОбработчика (HWND   hDlg, UINT   Message,
                                 WPARAM wParam,  LPARAM lParam)
{
    switch (Message)
    {
     case WM_INITDIALOG:
          return FALSE;
      case WM_COMMAND:
          switch (wParam)
          {
          case IDOK:
               EndDialog(hDlg,TRUE);
               break;
          case IDCANCEL:
               EndDialog(hDlg,FALSE);
               break;
           default:
               return FALSE;
          }
          break;
     default:
          return FALSE;
    }
    return TRUE;
};
```

- выполнить приложение (диалоговое окно не появится!).

6. Инициализировать окно и вывести его на экран.

- для этого связать приложение (*ДескрипторПриложения* - переменная hInstance), родительское окно (*ДескрипторОкна*), диалоговое окно (*ДескрипторРесурса* – например, идентификатор IDD_DIALOG1), обработчик сообщений диалогового окна (*ИмяОбработчика*). Все это делается функцией с прототипом

DialogBox(HINSTANCE *ДескрипторПриложения*, **HWND** *ДескрипторОкна*, **LPCTSTR** *ДескрипторРесурса*, **DLGPROC** *ИмяОбработчика*),

вставленной в пользовательскую секцию case WM_PAINT функции главного окна приложения (например, DialogBox (hInstance, (LPCTSTR) IDD_DIALOG1, hWnd, (DLGPROC) TO_PROCESS_DIALOG_BOX));

- выполнить приложение - появится диалоговое окно.

Проверить, что при событиях, ведущих к сообщению WM_PAINT, диалоговые окна выводятся снова – иногда многократно. Диалоговое окно можно сбросить любой кнопкой, но по тем же причинам оно может появиться снова.

Задания для самостоятельного выполнения.

Модифицировать приложение, созданное в § 2.1.1.

1. Добавить вывод подтверждения нажатия кнопок окна.
2. Разрушать диалоговое окно только при нажатии кнопки CANCEL.
3. При запуске диалогового окна разрушать главное окно.
4. При завершении работы с диалоговым окном разрушать главное.

2.1.2. Диалоговое окно в качестве главного окна

Диалоговое окно может использоваться в интерфейсе приложения в роли главного окна. Например, при управлении приложением с помощью набора простых команд, составляющих один уровень иерархии (как "добавить запись", "найти запись", "удалить запись" и т.п.), которые могут быть представлены в окне набором соответствующих кнопок.

ПРИМЕР. *Создать Windows-приложение, при запуске которого в качестве главного должно выводиться диалоговое окно с кнопками. Нажатие кнопок диалогового окна приводит к подтверждению получения соответствующего сообщения, а нажатие Cancel - к его закрытию. Примерный вид оконного интерфейса приведен ниже.*

Назначение приложения. Приложение содержит одно окно – диалоговое, которое в качестве главного окна выводится при его запуске. Демонстрируется реакция приложения на нажатия кнопок окна OK и CANCEL и нажатие левой кнопки "мыши".

Интерфейсные формы. Включают диалоговое окно в роли главного с двумя командными кнопками OK и CANCEL. При нажатии кнопки OK или нажатии левой кнопки "мыши" выводятся окна сообщений (см. рисунки 6, 7). При нажатии кнопки CANCEL приложение завершается.

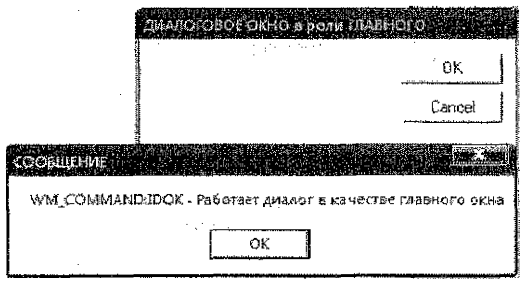

Рис. 6. Фрагмент интерфейса

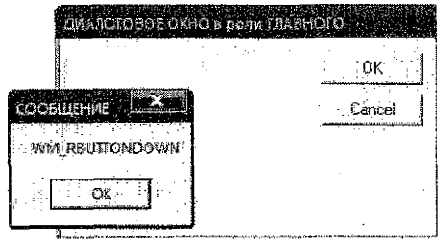

Рис. 7. Фрагмент интерфейса

Обрабатываемые сообщения. Это сообщения, адресуемые главному окну: - сообщение WM_RBUTTONDOWN; - сообщение WM_COMMAND от ресурса IDOK; - сообщение WM_COMMAND от ресурса IDCANCEL.

По нажатии кнопки OK инициируется сообщение WM_COMMAND: IDOK, что приводит к выводу информационного сообщения о приложении (в виде MessageBox).

По нажатии кнопки CANCEL инициируется сообщение WM_COMMAND: IDCANCEL, что при его обработке приводит к разрушению диалогового окна (функцией EndDialog(hDlg,TRUE)) и закрытию приложения путем посылки ОС сообщения PostQuitMessage(0) на закрытие приложения).

По нажатии левой кнопки "мыши" в пределах диалогового окна приложения инициируется сообщение WM_RBUTTONDOWN, что приводит к выводу информационного сообщения о событии (в виде MessageBox).

Порядок разработки.

1. Создать приложение на базе ТКП.

2. Создать ресурс – диалоговое окно (например, с ID - IDD_DIALOG1) и подключить его к приложению командой #include "resource.h".

3. Внести изменения в текст приложения согласно листингу, например:

а) описать пустой обработчик сообщений диалогового окна и его прототип

```
LRESULT CALLBACK  ИмяОбработчика (HWND hDlg, UINT Message,
                                  WPARAM wParam,  LPARAM lParam)
{
    return TRUE;
}  ;
```

б) в главной функции WinMain вместо действий по описанию, регистрации, созданию и визуализации классического главного окна (с рамкой) инициализировать и визуализировать диалоговое окно. Для этого связать текущее приложение (например, HINSTANCE hInst), ранее созданный ресурс - диалоговое окно (например, IDD_DIALOG1) и его обработчик командой DialogBox, например

```
DialogBox(hInst, (LPCTSTR) IDD_DIALOG1, NULL, (DLGPROC) ИмяОбработчика) ;
```

в) описать обработчик

```
LRESULT CALLBACK ИмяОбработчика (HWND   hDlg, UINT   Message,
                                 WPARAM wParam,  LPARAM lParam)
{
    switch (Message)
    {
     case WM_INITDIALOG:
          return FALSE;
     case WM_RBUTTONDOWN:
          < ФрагментПользователя >
          break;
      case WM_COMMAND:
          switch (wParam)
          {
          case IDOK:
               < ФрагментПользователя >
               break;
          case IDCANCEL:
               < ФрагментПользователя >
               break;
           default:
```

```
            return FALSE;
        }
        break;
    default:
        return FALSE;
    }
    return TRUE;
};
```

Задания для самостоятельного выполнения.

1. Реализовать на базе ТКП приложение с диалоговым окном в качестве главного окна. Его функция – один раз создать окно с клиентской областью и передать ему управление. При этом диалоговое окно может остаться на экране или может завершаться при запуске второго окна. В любом случае оно далее не оказывает никаких действий и завершается при закрытии второго окна – при завершении работы с приложением.

Пояснения к выполнению. Конкретный экземпляр окна с рамкой создается функцией hWnd = CreateWindow(...), которая сообщает его дескриптор для дальнейшего использования в разных функциях. Само окно после визуализации активно, но если его закрыть, то конкретный экземпляр окна теряется как и его дескриптор. Поэтому для повторной визуализации окна того же стиля необходимо заново создать его экземпляр функцией CreateWindow и получить новый дескриптор.

Соответственно в WinMain() создать стиль второго окна, создать его экземпляр, создать и активизировать диалоговое окно.

Обрабатываемое сообщение диалогового окна: сообщение WM_COMMAND: IDOK по нажатии кнопки ОК. Действие – визуализация окна с рамкой, созданного в главной функции. Соответственно в обработчике диалогового окна по ОК визуализировать второе окно (команды ShowWindow(*ДескрипторОкнаСРамкой, ПараметрВизуализации*) и UpdateWindow (*ДескрипторОкнаСРамкой*), а секцию закрытия не использовать.

Обрабатываемые сообщения второго окна: сообщение WM_RBUTTONDOWN по нажатии левой кнопки мыши в пределах окна. Действие – вывод информационного сообщения о событии; сообщение WM_DESTROY по закрытию окна. Действие – посылка сообщения на закрытие приложения PostQuitMessage(0). Соответственно в обработчике второго окна реагировать на события, производить пользовательскую обработку и закрытие приложения при свертывании окна.

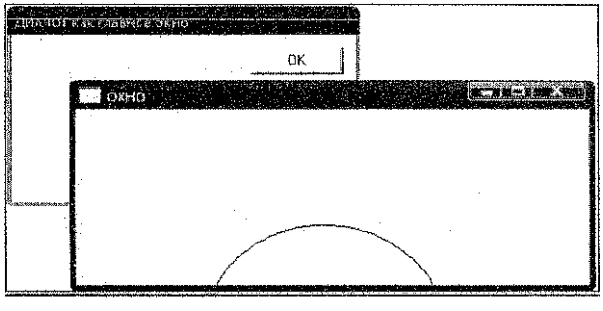

Рис. 8. Фрагмент интерфейса

2. Модифицировать предыдущее приложение с гашением диалогового окна после визуализации второго окна. Для этого сохранять дескриптор диалогового окна в глобальной переменной для последующего использования в обработчике окна с рамкой, например,

```
HWND      hDlgGlobal;
LRESULT CALLBACK ОбработчикДиалоговогоОкна (...)
{
      hDlgGlobal=hDlg;
      ..............
}
```

Для гашения диалога в обработчике окна с рамкой обрабатывать сообщение, например, получение фокуса WM_SETFOCUS - EndDialog(hDlgGlobal,TRUE).

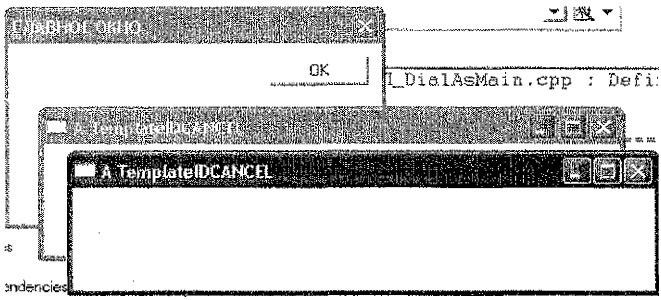

Рис. 9. Фрагмент интерфейса

3. Модифицировать приложение п.1 с многократным запуском окон одного стиля с рамкой по кнопке OK.

4. Модифицировать предыдущее приложение с гашением диалогового окна после визуализации второго окна.

5. Создать приложение с диалоговым окном в качестве главного с системной кнопкой закрытия и с единственной пользовательской кнопкой OK, обеспечивающей многократный запуск окна с рамкой с системными кнопками при сохранении первого. Завершение приложения производить при закрытии любого из окон (родительского или дочерних).

6. Создать приложение с диалоговым окном в качестве главного с системной кнопкой закрытия и с пользовательскими кнопками – OK и CANCEL, а также с реакцией на нажатие правой клавиши мыши. Кнопка OK обеспечивает многократный запуск окна с рамкой с системными кнопками при сохранении первого. Завершение приложения по CANCEL, закрытие любого из дочерних окон – его системной кнопкой.

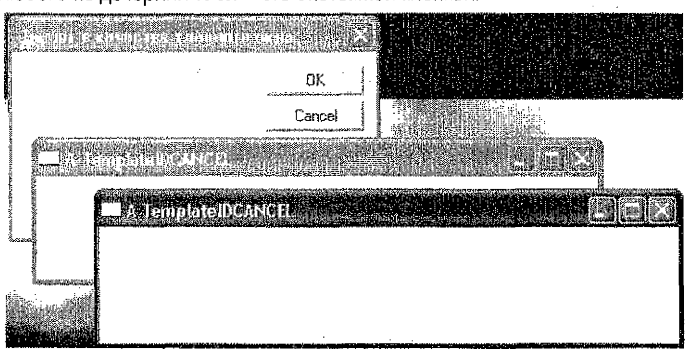

Рис. 10. Фрагмент интерфейса

7. Создать приложение с диалоговым окном в качестве главного с системной кнопкой закрытия и с пользовательскими кнопками – OK и CANCEL, а также с реакцией на нажатие правой клавиши мыши. Кнопка OK обеспечивает многократный запуск окна с рамкой с системными кнопками при сохранении первого. Завершение приложения по CANCEL, закрытие любого из дочерних окон – его системной кнопкой.

2.1.3. Диалоговое окно с окном редактирования в качестве главного окна

При использовании диалогового окна его пространство используется для размещения различных элементов управления, обеспечивающих, в том числе ввод-вывод данных и управление работой.

ПРИМЕР. *Создать Windows-приложение, при запуске которого в качестве главного должно выводиться диалоговое окно (с окнами редактирования, подсказками, кнопками), предназначенное для задания исходных данных и их эхо-вывода. По кнопке Ввести производится ввод значения из окна ввода и эхо-вывод введенного значения.*

Интерфейсные формы. Примерный вид диалогового окна приведен ниже (рисунок 11). Окно включает два окошка редактирования (ЭУ типа Edit Box), два статичных окна (ЭУ типа Static Text), командную кнопку Ввести.

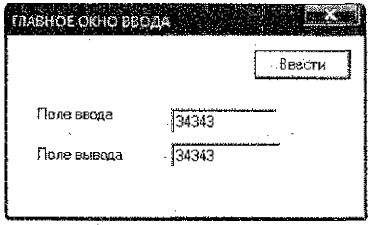

Рис. 11. Фрагмент интерфейса

Обрабатываемые сообщения. Это сообщения, адресуемые диалоговому окну: - сообщение по нажатии кнопки Ввести (например, WM_COMMAND: ID_Enter). Действие – считывание содержимого окна редактирования (например, IDC_EDIT1) в строку (например, InputString) с последующим выводом строки в окно редактирования (например, IDC_EDIT2). Для этого в обработчике окна (в секции case ID_Enter) следует вставить команды GetDlgItemText, SetDlgItemText; - сообщение case WM_COMMAND:IDCANCEL по нажатии системной кнопки завершения работы. Действие – разрушение диалогового окна функцией EndDialog и закрытие приложения путем посылки ОС сообщения PostQuitMessage.

Порядок разработки.

1. Создать приложение на базе ТКП.

2. Создать ресурс – диалоговое окно (например, с ID = IDD_DIALOG1), с окошками редактирования (например, с IDC_EDIT1, IDC_EDIT2 для ввода и вывода соответственно), со статическими окнами (например, с IDC_STATIC1, IDC_STATIC2 соответственно с названиями Caption - "Поле ввода", "Поле вывода"), с кнопкой (например, с ID_Enter). Настроить свойства элементов управления. Например, для поля эхо-вывода с IDC_EDIT2 установить в окне его свойств (Edit Properties) способ выравнивания текста – к левому полю, режим работы – только чтение, автоматическая горизонтальная прокрутка, как показано ниже (рисунок 12)

Проверить работу окна командой главное меню Layout Test, подключив его к приложению командой #include "resource.h5".

3. Внести изменения в текст приложения.

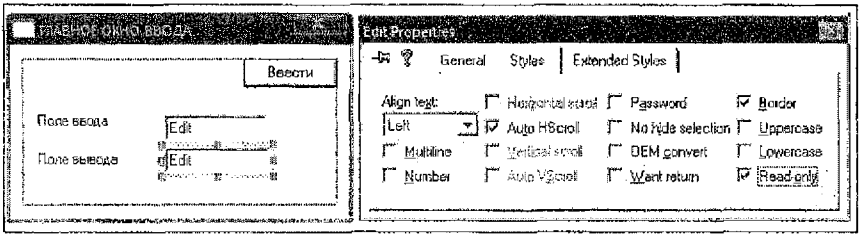

Рис. 12. Редактирование диалогового окна

а) описать пустой обработчик сообщений диалогового окна и его прототип;

б) в главной функции WinMain вместо действий по визуализации окна с рамкой визуализировать диалоговое окно командой DialogBox;

в) описать обработчик. В секции case ID_Enter описать строковую переменную char InputString[256], ввести данные из окна ввода в строку командой

GetDlgItemText(hDlg,IDC_EDIT1, InputString,15),

вывести данные из строки в окно вывода командой

SetDlgItemText(hDlg,IDC_EDIT2, InputString).

В секции IDCANCEL выполнить EndDialog(hDlg,TRUE), PostQuitMessage(0).

4. Компилировать и наблюдать работу окна.

2.1.4. Диалоговое окно со списком и окном редактирования в качестве главного окна

Теоретические сведения. Список отображает набор строк. Строку можно выбрать "мышью", клавиатурой. При этом строка выделяется цветом и посылается сообщение от списка системе о событии. Информацию об элементе управления "список" - List Box можно найти в MSDN (в разделе List Box Controls, местонахождение Technical Articles). Там описаны следующие компоненты списка.

Стили (Styles) списка имеют названия LBS_<НазваниеСтиля>. Например, LBS_STANDARD, LBS_SORT (строки списка располагаются в алфавитном порядке), LBS_MULTIPLESEL (с возможностью выбора произвольного числа строк), LBS_MULTICOLUMN (многоколонковый список с горизонтальной прокруткой), LBS_NOTIFY (с отсылкой родительскому окну сообщения о выборе строки) (рисунок 13).

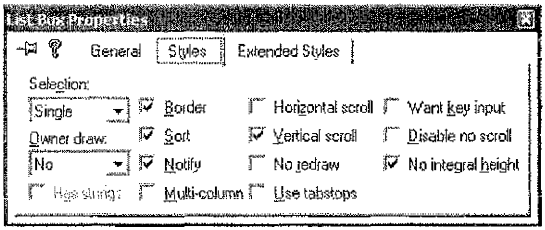

Рис. 13. Настройка списка

Сообщения (Notification Messages), адресуемые своему родительскому окну от списка о происходящих с ним событиях, именуются как LBN_<НазваниеСобытия>. Например, LBN_DBLCLK, LBN_SELCHANGE. Родительское окно списка принимает эти сообщения в виде соответствующих WM_COMMAND сообщений.

Сообщения – команды (Messages), адресуемые списку от приложения именуются как LB_<НазваниеКоманды>. Например, LB_ADDSTRING, LB_DELETESTRING, LB_SETCURSEL, LB_GETCURSEL, LB_GETTEXT, LB_GETCOUNT.

Для посылки сообщения (*КодСообщения* с параметрами *ПараметрыСообщения1, ПараметрыСообщения2*) указанному ЭУ (*ДескрипторЭУ*) указанного диалогового окна (ДескрипторДиалоговогоОкна) в приложениях используется функция SendDlgItemMessage (для списка *ДескрипторЭУ = ДескрипторСписка*, КодСообщения = *НазваниеКоманды*, а значения *ПараметровСообщения* зависят от типа команды)

SendDlgItemMessage (**HWND** *ДескрипторДиалоговогоОкна*, **int** *ДескрипторЭУ*, **UINT** *КодСообщения*, **WPARAM** *ПараметрыСообщения1*, **LPARAM** *ПараметрыСообщения2*).

Примеры использования команды приведены ниже:

- получение текущего размера списка выполняется командой

ЧислоСтрокСписка = SendDlgItemMessage(<ДескрипторОкна>, <ДескрипторСписка>, LB_GETCOUNT, 0, 0) ,

int RecordsAmount = SendDlgItemMessage(hDlg, IDC_LIST1, LB_GETCOUNT, 0, 0).

- добавление строки выполняется командой

SendDlgItemMessage (<ДескрипторОкна>,<ДескрипторСписка>,LB_ADDSTRING, 0, (LPARAM) <ДобавляемаяСтрока>).

SendDlgItemMessage(hDlg, IDC_LIST1, LB_ADDSTRING, 0, (LPARAM) "Иванов").

- удаление строки выполняется командой

SendDlgItemMessage (<ДескрипторОкна>, <ДескрипторСписка>, LB_DELETESTRING, <НомерУдаляемойСтроки>, 0) ,

SendDlgItemMessage(hDlg, IDC_LIST1, LB_DELETESTRING, i, 0).

- установка текущей позиции выделения выполняется командой

SendDlgItemMessage (<ДескрипторОкна>, <ДескрипторСписка>, LB_SETCURSEL, 0, (LPARAM) <ВыделяемаяСтрока>)

SendDlgItemMessage(hDlg, IDC_LIST1, LB_SETCURSEL, 0, (LPARAM) "Иванов");

- получение номера текущей позиции выделения выполняется командой

НомерВыделеннойСтроки = SendDlgItemMessage(<ДескрипторОкна>, <ДескрипторСписка>, LB_GETCURSEL, 0, 0) ,

i = SendDlgItemMessage(hDlg, IDC_LIST1, LB_GETCURSEL, 0, 0);

- получение текста строки по указанному номеру выполняется командой

SendDlgItemMessage (<ДескрипторОкна>, <ДескрипторСписка>, LB_GETTEXT, <НомерСтроки>, (LPARAM) <ПриемнаяСтрока>)

SendDlgItemMessage(hDlg, IDC_LIST1, LB_GETTEXT, i, (LPARAM) Text) .

ПРИМЕР. *Разработать приложение с диалоговым окном в роли главного.*

Цель приложения. Управление списком строк – фамилий (просмотр, выбор, добавление, удаление, редактирование, расположение в алфавитном порядке). При запуске приложения в качестве главного окна выводится диалоговое окно, содержащее пустой список, средства управления списком и кнопку Завершить (Cancel).

Интерфейсные формы. Примерный вид диалогового окна представлен ниже (рисунок 14). Соответственно состав ресурсов: - список (например, с IDC_LIST_NAMES), окно редактирования (например, с IDC_EDIT_NAME), кнопки Добавить, Удалить, Изменить, Завершить (например, с IDC_BUTTON_ADD, IDC_BUTTON_DELETE, IDC_BUTTON_EDIT, IDCANCEL), рамки – ЭУ типа Group Box и т.д.

Обрабатываемые сообщения.

1. Сообщения к окну (целиком):

– (**WM_COMMAND**: **IDCANCEL**) по нажатии кнопки ЗАВЕРШИТЬ приложение функцией EndDialog(hDlg,TRUE) закрывает диалог и функцией PostQuitMessage(0) извещает систему о необходимости завершить приложение. Для этого генерируется и адресуется окну сообщение UINT message = WM_COMMAND с параметром int wmId =

LOWORD(wParam) = IDCANCEL. Оно обрабатывается в обработчике в секции case WM_COMMAND: case IDCANCEL:.

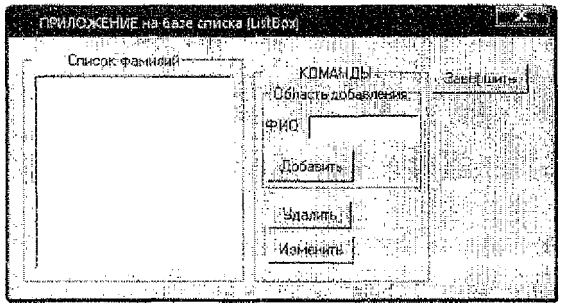

Рис. 14. Интерфейс приложения

- (**WM_INITDIALOG**) при запуске приложения и визуализации окна командой DialogBox возможна подготовка окна к работе, его инициализация (например, начальное заполнение списка константной информацией или считывание информации из файла в список). Для этого генерируется и адресуется окну сообщение UINT message = WM_INITDIALOG. Оно обрабатывается в обработчике в секции case WM_INITDIALOG:.

2. Сообщения, относящиеся к ЭУ (списку) окна:

– (**WM_COMMAND**: **IDC_BUTTON_ADD**) по нажатии кнопки ДОБАВИТЬ считывается функцией GetDlgItemText (hDlg, IDC_EDIT_NAME, Строка, ДлинаСтроки) фамилия, набранная в поле IDC_EDIT_NAME. Если это не пустая строка (функция strcmp), то (через функцию SendDlgItemMessage) приложение генерирует и адресует списку (IDC_<ДескрипторСписка>) сообщение LB_ADDSTRING – добавить к списку указанную в функции строку, считанную ранее из окна редактирования фамилию. Для этого генерируется и адресуется окну сообщение UINT message = WM_COMMAND с параметром int wmId = LOWORD(wParam) = IDC_BUTTON_ADD. Оно обрабатывается в обработчике в секции case WM_COMMAND: case IDC_BUTTON_ADD:.

– (**WM_COMMAND**: **IDC_BUTTON_DELETE**) по нажатии кнопки УДАЛИТЬ приложение считывает (через функцию SendDlgItemMessage) номер выбранной строки списка (как результат обработки адресуемого списку сообщения LB_GETCURSEL). Далее приложение (через функцию SendDlgItemMessage) генерирует и адресует списку (IDC_<ДескрипторСписка>) сообщение LB_DELETESTRING – удалить из списка строку с указанным в функции номером, считанным ранее. Для этого генерируется и адресуется окну сообщение UINT message = WM_COMMAND с параметром int wmId = LOWORD(wParam) = IDC_BUTTON_DELETE. Оно обрабатывается в обработчике в секции case WM_COMMAND: case case IDC_BUTTON_DELETE:.

– (**WM_COMMAND**: **IDC_BUTTON_EDIT**) по нажатии кнопки ИЗМЕНИТЬ приложение считывает (через функцию SendDlgItemMessage) номер выбранной строки списка (как результат обработки адресуемого списку сообщения LB_GETCURSEL). Далее приложение (через функцию SendDlgItemMessage) генерирует и адресует списку (IDC_<ДескрипторСписка>) сообщение LB_GETTEXT – получить из списка строку с указанным в функции номером, считанным ранее. Считанная строка помещается в поле IDC_EDIT_NAME (функция SetDlgItemText) для редактирования с одновременным стиранием этой строки из списка (через функцию SendDlgItemMessage с параметром LB_DELETESTRING) (последующее добавление отредактированной строки к списку инициируется кнопкой ДОБАВИТЬ).

Для обработки рассматриваемого события генерируется и адресуется окну сообщение UINT message = WM_COMMAND с параметром int wmId = LOWORD(wParam) = IDC_BUTTON_EDIT. Оно обрабатывается в обработчике в секции case WM_COMMAND: case IDC_BUTTON_EDIT:.

3. Сообщения, относящиеся к окну (целиком) от ЭУ (списка) окна:

– (**WM_COMMAND**: **IDC_LIST_NAMES**: **LBN_DBLCLK**) при выборе строки в списке фамилий, например, двойным щелчком, генерируется сообщение LBN_DBLCLK. Здесь действие приложения сводится просто к выводу подтверждающего сообщения о происшедшем событии. Для этого генерируется и адресуется окну сообщение UINT message = WM_COMMAND с параметром int wmId = LOWORD(wParam) = IDC_LIST_NAMES с дополнительным параметром int wmEvent = HIWORD(wParam) = LBN_DBLCLK. Оно обрабатывается в обработчике в секции case WM_COMMAND: case IDC_LIST_NAMES: case LBN_DBLCLK:.

– (**WM_COMMAND**: **IDC_LIST_NAMES**: **LBN_SELCHANGE**) при выборе строки в списке фамилий, например, щелчком, клавиатурой генерируется сообщение об изменении выделения LBN_SELCHANGE. Здесь действие приложения сводится просто к выводу подтверждающего сообщения о происшедшем событии. Для этого генерируется и адресуется окну сообщение UINT message = WM_COMMAND с параметром int wmId = LOWORD(wParam) = IDC_LIST_NAMES с дополнительным параметром int wmEvent = HIWORD(wParam) = LBN_SELCHANGE. Оно обрабатывается в обработчике в секции case WM_COMMAND: case IDC_LIST_NAMES: case LBN_SELCHANGE:.

Ниже представлена структура обработчика

```
LRESULT CALLBACK DlgProc(...)
{
    int wmId    = LOWORD(wParam);
    int wmEvent = HIWORD(wParam);
    int RecordsAmount = SendDlgItemMessage(hDlg, IDC_LIST_NAMES, LB_GETCOUNT, 0, 0);
    switch (Message)
    {
    case WM_INITDIALOG:  ... break;
    case WM_COMMAND:
        switch (wmId)
        {
        case IDCANCEL: ... break;
        case IDC_BUTTON_ADD: ... break;
        case IDC_BUTTON_DELETE: ... break;
        case IDC_BUTTON_EDIT: ... break;
        case IDC_LIST_NAMES:
            switch (wmEvent)
            {
            case LBN_SELCHANGE: ... break;
            case LBN_DBLCLK: ... break;
            default: ...
            }
            break;
        default: ...
        }
        break;
    default:
    }
}
```

Порядок создания.

1. Создать приложение на базе ТКП.
2. Создать и подключить ресурс – диалоговое окно.
3. Описать обработку сообщений в соответствии с их спецификациями (см. выше):
- инициализация окна;
- нажатие кнопки ЗАВЕРШИТЬ;
- нажатие кнопки ДОБАВИТЬ;
- нажатие кнопки УДАЛИТЬ;
- нажатие кнопки ИЗМЕНИТЬ;
- выбор строки в списке фамилий двойным щелчком;
- выбор строки в списке фамилий щелчком.

Задания для самостоятельного выполнения.

1. Модифицировать приложение, оформив все секции обработчика в виде системы функций.

2. Модифицировать приложение, обеспечив инициализацию списка заранее заданными фамилиями (строками).

3. Модифицировать приложение, обеспечив сохранение и инициализацию списка введенными фамилиями (строками).

4. Разработать приложение с диалоговым окном в качестве главного окна и списком для обработки телефонов. Примерный вид интерфейса представлен ниже (рисунок 15).

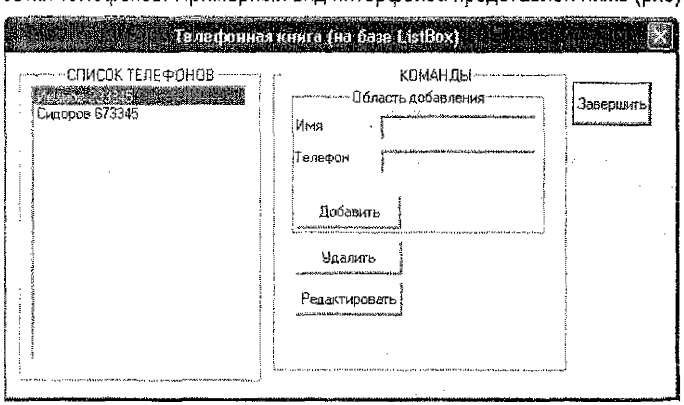

Рис. 15. Интерфейс приложения

2.2. Использование ресурса меню

Окно с рамкой кроме системного меню может содержать и меню, спроектированное и настроенное пользователем. Такое меню является ресурсом окна и сообщения, связанные с выбором пунктов меню, обрабатываются обработчиком окна. Если это главное окно, то они обрабатываются обработчиком главного окна. Само меню может рассматриваться как определенным образом организованная совокупность командных кнопок, которые представляют пункты меню. В основном, это кнопки двух типов - MENUITEM и POPUP. Кнопка типа POPUP определяет пункт главного меню, автоматически вызывающий выпадающее подменю, которое может содержать подпункты типа MENUITEM и POPUP. Кнопка типа MENUITEM определяет конечный пункт меню. При его выборе генерируется сообщение WM_COMMAND, параметр сообщения WPARAM wParam содер-

жит ID выбранного пункта меню. Соответственно для его обработки используются секции в обработчике

```
LRESULT CALLBACK WndProc (HWND hWnd, UINT Message,
                          WPARAM wParam,  LPARAM lParam)
{
    .....
    switch (Message)
    {
    .....
    case WM_COMMAND:  // сообщение от меню
          switch (wParam)  // параметр сообщения – ID выбранного пункта меню
          {
          case IDM_< КонечныйПунктМеню >:
                .....
                break;
    .....
          case IDM_< КонечныйПунктМеню >:
                .....
                break;
          .....
              default:
                return DefWindowProc(hWnd, messg, wParam, lParam);
          }
          break;
          default:
                return (DefWindowProc(hWnd, messg, wParam, lParam));
    }
    return 0;
}
```

При текстовом описании меню соответственно могут использоваться два типа операторов: MENUITEM и POPUP. Оператор MENUITEM определяет конечный пункт меню, а оператор POPUP — выпадающее меню. Операторы имеют следующий формат:

MENUITEM "*НазваниеПункта*", ID_*Пункта* [, *ОпцииПункта*]
POPUP " *НазваниеПункта* " [, *ОпцииПункта*] .

Параметр *НазваниеПункта* задает название пункта меню, например File или Help. Параметр ID_*Пункта* - уникальное целое значение, идентифицирующее пункт меню, посылаемое приложению при выборе данного пункта. Обычно ID-значения хранятся в виде констант в библиотечном файле, который затем включается как в программный файл, так и в rc-файл ресурсов.

ПРИМЕР. *Разработать приложение на базе ТКП с окном с рамкой в качестве главного, содержащее простейшее пользовательское меню.*

Интерфейсные формы. Здесь два пункта типа POPUP – ВВОД, ВЫВОД, четыре пункта типа MENUITEM – Строка 1, Строка 2 (подпункты пункта ВВОД), Строка 1, Строка 2 (подпункты пункта ВЫВОД), соответственно с IDM_inString1, IDM_inString2, IDM_outString3, IDM_outString4. Примерный вид интерфейса показан ниже (рисунок 16, 17).

Обрабатываемые сообщения. Это сообщения, адресуемые главному окну:

- сообщение WM_PAINT – здесь не используется;

Рис. 16. Фрагмент интерфейса

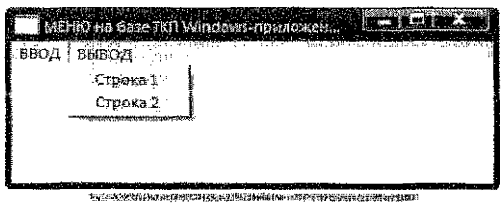

Рис. 17. Фрагмент интерфейса

- сообщение WM_DESTROY при закрытии окна. Действие – завершение работы приложения посылкой сообщения на завершение командой PostQuitMessage(0);

- сообщения WM_COMMAND от конечных пунктов меню, в том числе WM_COMMAND:IDM_inString1,WM_COMMAND:IDM_inString2, WM_COMMAND: IDM_outString3, WM_COMMAND: IDM_outString4.

Порядок разработки.

1. Создать ТКП

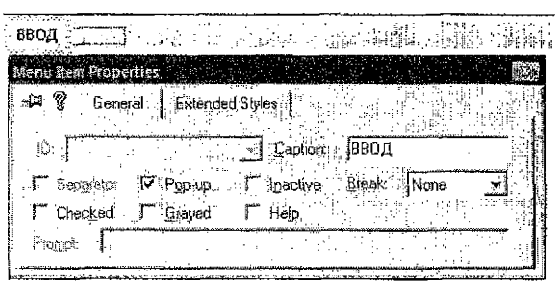

Рис. 18. Свойства окна

2. Создать ресурсный файл и подключить его к проекту командой #include "resource.h"; выполнить приложение (меню не появится!).

3. Спроектировать и описать ресурс:

- разработать вид и состав меню;

- добавить новый ресурс-меню в ресурсный файл приложения командой главного меню Insert-Resource-Menu;

- создать меню в редакторе ресурсов, указывая пункты меню и описывая их свойства. Например, для пункта ВВОД окно свойств выглядит как на рисунке 18. А для пункта Строка 1 окно свойств выглядит как на рисунке 19.

- в свойствах меню определить его дескриптор, идентификатор ID (пусть, например, ID = IDR_MENU1);

- выполнить приложение (меню не появится!).

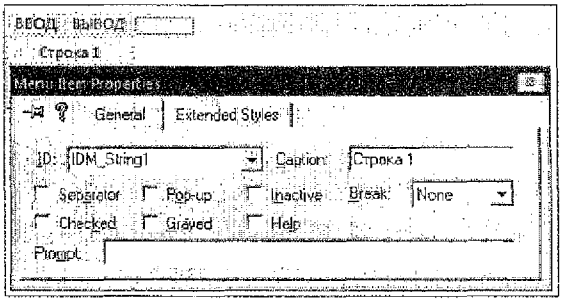

Рис. 19. Свойства окна

4. Присоединить меню (его имя определено как IDR_MENU1) к приложению:
- в функции WinMain() при описании стиля окна в переменной WNDCLASS wcApp определить ссылку на меню, используя поле

wcApp.lpszMenuName = (LPTSTR) IDR_MENU1;

- выполнить приложение (меню визуализируется, но не реагирует на выбор пунктов).
4. Настроить обработку событий выбора пунктов меню, для чего внести изменения в обработчик, добавив к обработке событий WM_PAINT, WM_DESTROY фрагмент реакции на выбор пунктов меню, например

```
LRESULT CALLBACK WndProc (HWND hWnd,  UINT messg,
                          WPARAM wParam,  LPARAM lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    switch (Message)
    {
    case WM_PAINT:
          .....
          break;
    case WM_DESTROY:
          PostQuitMessage(0);
          break;
    case WM_COMMAND:
          switch (wParam)
          {
          case IDM_inString1:
                .....
                break;
          case IDM_inString2:
                .....
                break;
          case IDM_outString3:
                .....
                break;
          case IDM_outString4:
                .....
                break;
```

```
            default:
                return DefWindowProc(hWnd, messg, wParam, lParam);
            }
            break;
            default:
                return (DefWindowProc(hWnd, messg, wParam, lParam));
    }
    return 0;
}
```

Здесь в качестве реакции на выбор пунктов меню выводить подтверждение сделанного выбора командой MessageBox. Выполнить приложение – убедиться в корректности работы меню.

2.3. Совместное использование ресурса меню и диалоговых окон

ПРИМЕР. *Разработать приложение на базе ТКП для многократного ввода-вывода строк и фиксации в окне количества введенных строк. Организовать интерфейс на основе окна с рамкой в качестве главного и меню. Через меню вызывается диалоговое окно для ввода строки, окно сообщения для вывода строки. В клиентскую область окна вывод Количества вводов строки. Оформление секций обработчика как функций, например TO_WRITE_ENTER_AMOUNT.*

Интерфейсные формы. Примерный вид интерфейса показан ниже (рисунок 20). Главное окно содержит пользовательское меню с двумя пунктами ВВОД и ВЫВОД. По пункту ВВОД выводится диалоговое окно (с окошком редактирования и кнопкой ВВЕСТИ), предназначенное для задания вводимой строки. По пункту ВЫВОД производится вывод ранее введенной строки. В нижней части окна выводится подсказка о количестве введенных строк. Пусть меню имеет идентификатор IDR_MENU1, пункты типа MENUITEM – ВВОД, ВЫВОД с идентификаторами IDM_In и IDM_Out, диалоговое окно имеет идентификатор IDD_DIALOG1, окно редактирования - IDC_EDIT1, кнопка ввода - ID_Enter.

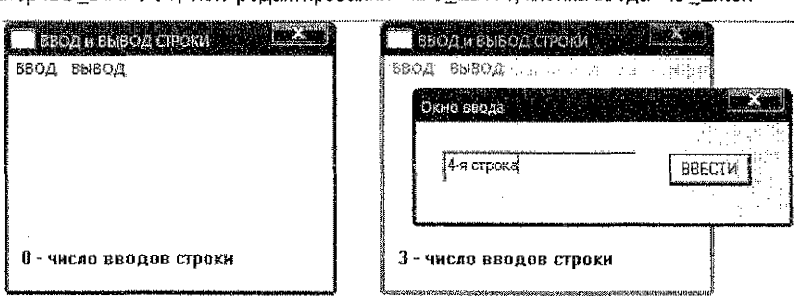

Рис. 20. Фрагмент интерфейса

Обрабатываемые сообщения. Это сообщения, адресуемые главному окну и обрабатываемые его обработчиком:

- сообщение WM_PAINT. Действие - вывести в клиентскую область окна новое значение количества вводов строки;
- сообщение WM_DESTROY. Действие - послать сообщение на завершение приложения;
- сообщение WM_COMMAND: IDM_In. Действие - инициализировать и вывести диалоговое окно ввода;
- сообщение WM_COMMAND: IDM_Out. Действие - вывести строку в окне сообщения.

Это сообщения, адресуемые диалоговому окну и обрабатываемые его обработчиком:

- сообщение WM_COMMAND: ID_Enter. Действия - увеличить количество вводов строки на единицу, ввести следующую строку из окна редактирования, закрыть диалоговое окно, послать сообщение на перерисовку главного окна.

Задания для самостоятельного выполнения.

1. Модифицировать приложение, оформив все секции обработчика в виде системы функций.
2. Модифицировать приложение, реализуя вывод строки через отдельное окно вывода.
3. Модифицировать приложение, реализуя вывод-вывод строки в одном диалоговом окне.
4. Модифицировать приложение, реализуя вывод-вывод массива строк. Вывод строк производить либо в клиентской области окна, либо в ЭУ список диалогового окна.
5. Модифицировать приложение, сохраняя и загружая массив строк из заданного файла.

2.4. Использование стандартных диалоговых окон

Стандартные диалоговые окна позволяют автоматизировать типовые действия пользователя, такие как просмотр структуры папок компьютера, поиск, открытие, сохранение файлов и т.д.

Функции, обеспечивающие работу с типовыми диалоговыми окнами, подключаются с помощью заголовочного файла commdlg.h.

Для навигации по структуре папок и выбора нужного файла для открытия используется функция GetOpenFileName, поддерживающая окно типа "Открыть", а для сохранения файла используется функция GetSaveFileName, поддерживающая окно типа "Сохранить как".

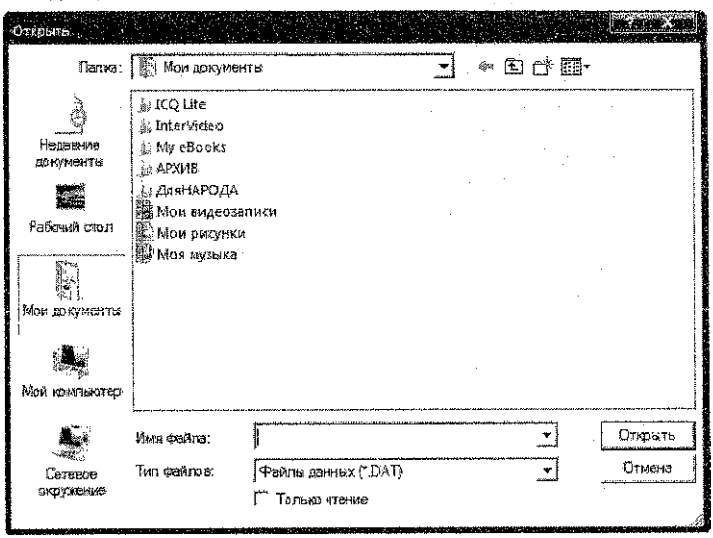

Рис. 21. Окно "Открыть"

Прототип функции GetOpenFileName

BOOL GetOpenFileName(LPOPENFILENAME *СсылкаНаСтруктуру*).

Здесь *СсылкаНаСтруктуру* – указатель типа LPOPENFILENAME на структуру типа OPENFILENAME.

При запуске функции в соответствии с установками структуры типа OPENFILE-NAME инициализируется, визуализируется и активизируется диалоговое окно (рисунок 21), которое ждет выбора пользователя. После сделанного выбора функция автоматически завершает свою работу, закрывает окно и заполняет указанную структуру, которая и содержит информацию о пользовательском выборе файла.

Структура типа OPENFILENAMEW, содержащая информацию для инициализации окна

```
typedef struct tagOFN
{
   DWORD lStructSize;
   HWND hwndOwner;
   HINSTANCE hInstance;
   LPCSTR lpstrFilter;
   LPSTR lpstrFile;
   DWORD nMaxFile;
   LPSTR lpstrFileTitle;
   LPCSTR lpstrTitle;
   DWORD Flags;
   ...
} OPENFILENAME .
```

Основные поля описаны ниже. Здесь:

- lStructSize – размер структуры;
- hwndOwner – дескриптор родительского окна (NULL);
- lpstrFilter – указатель строки, буфера, содержащего определенным образом форматированные пары строк, задающие фильтры окна;
- lpstrFile – указатель на строку, содержащую имя файла, которое зафиксировано в окошке редактирования диалогового окна (или NULL). После успешного завершения функций GetOpenFileName или GetSaveFileName это указатель буфера, содержащего полное имя выбора пользователя;
- nMaxFile – размер буфера, строки lpstrFile;
- Flags – флаги (по завершении работы идентифицируют пользовательский ввод).

Соответственно при использовании типовых окон (например, типа "Открыть", "Сохранить как") необходимо:

1. До начала работы подготовить структуру типа OPENFILENAME. Например,

```
char ИмяФайла [128];
char Фильтр [ ] = "Файлы данных (*.doc) \0 *.dat\0 Все файлы (*.*)\0*.*\0";

OPENFILENAME СтруктураВыбранногоФайла;
memset(&СтруктураВыбранногоФайла, 0, sizeof(OPENFILENAME));
СтруктураВыбранногоФайла.lStructSize = sizeof(OPENFILENAME);
СтруктураВыбранногоФайла.hwndOwner = ДескрипторРодительскогоОкна;
СтруктураВыбранногоФайла.lpstrFilter = Фильтр;
СтруктураВыбранногоФайла.lpstrFile = ИмяФайла;
СтруктураВыбранногоФайла.nMaxFile = sizeof(ИмяФайла);
СтруктураВыбранногоФайла.Flags= OFN_PATHMUSTEXIST | OFN_FILE MUSTEXIST; .
```

2. Выполнить функцию для запуска окна и получения выбора пользователя. Например, GetOpenFileName(&*СтруктураВыбранногоФайла*).

3. Если выбор состоялся, то обработать его, использовав полученное имя файла. Например,

```
if (GetOpenFileName (&СтруктураВыбранногоФайла))
{
        ОБРАБОТАТЬ_ФАЙЛ  ИмяФайла
}
```

ПРИМЕР. *Создать приложение с окном с рамкой. По сообщению WM_ LBUTTONDOWN выводить окно типа "Сохранить как" для получения пользовательского указания места в файловой системе компьютера и имени файла. По сообщению WM_RBUTTONDOWN выводить окно типа "Открыть" для выбора открываемого файла.*

Примерный текст приложения приведен ниже. В качестве обработки следует выводить выбор пользователя в окне MessageBox.

```
#include <commdlg.h>
...
char szFileName[128];
char szFilter[] = "Файлы данных (*.doc) \0*.dat\0Все файлы (*.*)\0*.*\0";
LRESULT CALLBACK   WndProc(HWND, UINT, WPARAM, LPARAM);
...
int  WINAPI WinMain (...) {  ... }
LRESULT CALLBACK WndProc (HWND hWnd, UINT Message,
                          WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    OPENFILENAME ofn;
    switch (Message)
    {
    case WM_RBUTTONDOWN:
        memset(&ofn, 0, sizeof(OPENFILENAME));
        ofn.lStructSize = sizeof(OPENFILENAME);
        ofn.hwndOwner = hWnd;
        ofn.lpstrFilter = szFilter;
        ofn.lpstrFile = szFileName;
        ofn.nMaxFile = sizeof(szFileName);
        ofn.Flags=OFN_PATHMUSTEXIST|OFN_FILEMUSTEXIST;
        if  (GetOpenFileName(&ofn))
        {
            //Команды по обработке файла с именем szFileName
            break;
        }
        else
            break;
    case WM_LBUTTONDOWN:
        memset(&ofn, 0, sizeof(OPENFILENAME));
        ofn.lStructSize = sizeof(OPENFILENAME);
        ofn.hwndOwner=hWnd;
        ofn.lpstrFilter = szFilter;
        ofn.lpstrFile = szFileName;
        ofn.nMaxFile = sizeof(szFileName);
        ofn.Flags=OFN_PATHMUSTEXIST|OFN_FILEMUSTEXIST;
        if (GetSaveFileName(&ofn))
```

```
            {
                //Команды по обработке файла с именем szFileName
                break;
            }
            else
                break;
    case WM_PAINT:
        ...
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    default:
        return (DefWindowProc(hWnd, Message, wParam, lParam));
    }
    return 0;
}
```

ПРИМЕР. *Создать приложение с окном. По сообщению WM_RBUTTONDOWN визуализировать диалоговое окно типа "Открыть" с последующим открытием выбранного файла, по сообщению WM_LBUTTONDOWN визуализировать диалоговое окно типа "Сохранить как" с последующим сохранением информации в выбранном файле.*

Примечание. Для открытия файла используется функция CreateFile. Например, для открытия файла с именем szFileName для выполнения операций чтения

```
hFile = CreateFile(szFileName, GENERIC_READ, 0,NULL, OPEN_EXISTING,0,NULL)
```

для открытия файла с именем szFileName для выполнения операций записи

```
hFile = CreateFile(szFileName, GENERIC_READ | GENERIC_WRITE, 0, NULL,
                   CREATE_ALWAYS, 0, NULL).
```

Для чтения и записи данных используются функции ReadFile, WriteFile. Например, для чтения одной записи - строки из файла szFileName (дескриптор hFile) во внутреннюю строку char Strln

```
ReadFile(hFile, Strln, 9, &dwCount, NULL); ,
```

а для записи строки в файл

```
WriteFile(hFile, StrOut, sizeof(*StrOut), &dwCount, NULL); .
```

Здесь DWORD dwCount.

Задания для самостоятельного выполнения.

Модифицировать приложение для записи в заданный файл и считывания из заданного файла внутреннего константного массива строк (например, строк вида "Запись1", "Запись2", "Запись3", "Запись4", "Запись5").

Модифицировать приложение для записи в заданный файл и считывания из заданного файла произвольного массива строк.

Модифицировать приложение для записи в заданный файл и считывания из заданного файла произвольного массива строк. Управление приложением осуществлять через меню (рисунок 22).

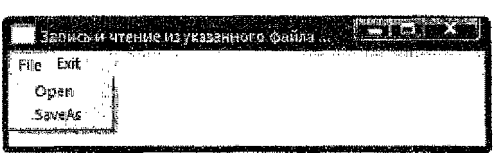

Рис. 22. Фрагмент интерфейса

3. ПОРЯДОК ВЫПОЛНЕНИЯ

Вывод в клиентское окно ТКП текстовой информации и сообщений.

1. Изучить теоретический материал о функциях вывода данных в окно приложения. Справочная информация по функциям вывода - § 1.4.3.

2. Создать приложение (с именем EX_1). Последовательно включить в приложение вывод окна сообщения, например,

MessageBox (NULL, "Работает приложение", "Информационное сообщение", 1),

MessageBox (hWnd, "Работает приложение", "Информационное сообщение", MB_OK),

MessageBox (hWnd, "Работает приложение", "Информационное сообщение", MB_YESNOCANCEL)

и снова выполнить приложение.

3. Создать приложение на базе ТКП (с именем EX_2). Добавить строку приветствия (координаты вывода строки - 0, 0). Для этого вставить в обработчик сообщения WM_PAINT пользовательский фрагмент, например

```
char  szText [25] =  "Работает ТКП";
.....
TextOut (hdc, 0, 0, szText, strlen (szText));  .
```

Организовать вывод данных, например,

```
int  X = 2007;  char  szXasString[100];
.....
wsprintf (szXasString, "Значение X - %d", X);
TextOut (hdc, 0, 130, szXasString, strlen (szXasString));  .
```

Вывести в столбик для значений X от 1 до 10 (с шагом 0,5) пары значений: X - квадрат X.

Запустить приложение. Выполнить свертывание-развертывание окна, перемещение, изменение размеров окна. Убедиться в автоматической перерисовке содержимого окна.

4. Создать приложение на базе ТКП (с именем EX_3), модифицировать предыдущее приложение. Добавить вывод номера перерисовки. Например, при каждой перерисовке окна увеличивать переменную ReDrawNumber++ и выводить номер перерисовки ReDrawNumber. Здесь int ReDrawNumber = 0 описывается как глобальная переменная.

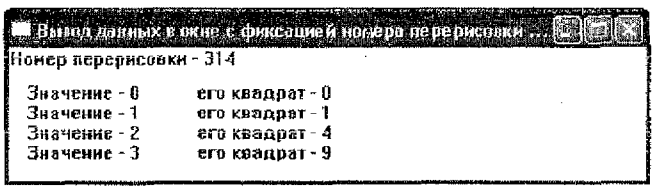

Рис. 23. Фрагмент интерфейса

Увеличить число обрабатываемых чисел до 75. Вид окна для работы с целочисленными значениями приведен на рисунке 23.

5.* Создать приложение (с именем EX_4) на базе ТКП. Протабулировать заданную функцию и вывести ее значения в виде таблицы.

6.* Создать приложение (с именем EX_5) на базе ТКП. Вывести версию ОС (для этого использовать переменную dwVersion типа DWORD и функцию GetVersion для получения ее значения). При преобразовании dwVersion в строковый формат функцией wsprintf использовать форматный код %x.

Вывод в клиентское окно приложения графической информации.

7. Создать приложение на базе ТКП (с именем EX_6). Организовать вывод графических данных.

При рисовании используют: - контекст устройства (переменная типа HDC hdc. Например, hdc = BeginPaint (hWnd, &ps)) и структуру типа PAINTSTRUCT ps.

Основные графические примитивы рисуются с использованием библиотечных функций. Это следующие примитивы и функции: - дуги эллипса Arc, ArcTo; - эллипсы и окружности Ellipse; - линии от текущей точки до точки, указанной в функции LineTo(hdc, x, y); - прямоугольники Rectangle; - полигоны Polygon; - связанные отрезки прямых PolyLine. А также функция рисования точки COLORREF SetPixel (hdc, x, y, COLORREF crColor), функция перемещения курсора к указанной точке с сохранением координаты текущей точки в структуре типа LPPOINT - BOOL MoveToEx(hdc, x, y, LPPOINT pPoint).

Фрагмент использования функций приведен ниже

```
.....
case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        hPen = CreatePen(PS_SOLID,1,RGB (0, 255, 0));
        SelectObject (hdc, hPen);
        Ellipse (hdc, 100, 100, 300, 300);
        MoveToEx(hdc, 50, 50, NULL);
        LineTo(hdc, 400,400);
          .....
        ValidateRect(hWnd,NULL);
        EndPaint(hWnd, &ps);
break;
.....
```

Обработка сообщений в клиентское окно ТКП.

8. Создать приложение (с именем EX_7) на базе ТКП. Включить чувствительность приложения к нажатию левой клавиши "мыши" (сообщению WM_LBUTTONDOWN) и при получении этого сообщения выводить соответствующий MessageBox. Для этого создать в функции-обработчике дополнительную секцию с MessageBox

```
case WM_LBUTTONDOWN:
    .....
    // < ФРАГМЕНТ ПОЛЬЗОВАТЕЛЯ >
    .....
    break;
```

аналогичную

```
case WM_PAINT:
    .....
    break;
```

Повторить задачи п. 3-5, выполняя все действия в секции case WM_LBUTTONDOWN (секция case WM_PAINT должна быть пустой).

Запустить приложение. Выполнить свертывание-развертывание окна, перемещение, изменение размеров окна. Проанализировать перерисовку содержимого окна.

9. Создать приложение (с именем EX_8) на базе ТКП. По сообщению WM_LBUTTONDOWN увеличивать координаты точки вывода X, Y на 50 единиц и инициировать перерисовку, по сообщению WM_RBUTTONDOWN уменьшать координаты

точки вывода X, Y на 50 единиц и инициировать перерисовку, по сообщению WM_PAINT выводить текст "Работает ТКП", начиная с позиции X, Y. Координаты точки вывода X, Y описать как глобальные переменные, начальное значение X = 0, Y = 0. Инициировать перерисовку окна функцией InvalidateRect (hWnd,NULL,TRUE).

Работа с диалоговыми окнами и меню (имена приложений формировать как EXDLG_xx).

10. Изучить теоретический материал по использованию диалоговых окон. Выполнить примеры и задания для самостоятельного выполнения (§§ 2.1.1-4).

11. Изучить теоретический материал по разработке и использованию пользовательских меню. Выполнить примеры и задания для самостоятельного выполнения (§ 2.2).

12. Изучить теоретический материал по использованию диалоговых окон и окон с меню. Выполнить примеры и задания для самостоятельного выполнения (§ 2.3).

13. Изучить теоретический материал по использованию типовых диалоговых окон. Выполнить примеры и задания для самостоятельного выполнения (§ 2.4).

ЛИТЕРАТУРА

1. Б. Страуструп. Язык программирования СИ++. – М.: Радио и связь, 1991. – 352 с.

2. Паппас К., Мюррей У. Visual C++. Руководство для профессионалов: Пер. с англ. - СПб: BHV -Санкт-Петербург, 1996.

3. Орлов С.А. Технологии разработки программного обеспечения: Учебник для вузов. – СПб.: Питер, 2004. – 527 с.

4. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд./ Пер. с англ.. – М.: Издательство БИНОМ, СПб: Невский диалект, 1998. – 560 с.

5. Шилдт Г. Справочник программиста по C/C++, 3-е изд. – М.: Изд. Дом Вильямс, 2003. – 432 с.

6. Паппас К., Мюррей У. Эффективная работа: Visual C++. NET. – СПб.: Питер, 2002. – 816 с.

7. Франка П. C++: учебный курс. – СПб.: Питер, 2005. – 522 с.

8. Одинцов И. Профессиональное программирование. Системный подход. – СПб.: БХВ-Петербург, 2002. – 512 с.

9. Паппас К., Мюррей У. Эффективная работа: Visual C++.NET. – СПб.: Питер, 2002. – 816 с.

10. Мэтт П. Внутренний мир Windows: Пер.с англ. – Киев: НИПФ ДиаСофтЛТД, 1995.

11. Мюррей У., Паппас К. Создание переносимых приложений для Windows. – СПб.: BHV – Санкт-Петербург, 1997. – 816 с.

12. Финогенов К.Г. Win32. Основы программирования. – М.: ДИАЛОГ-МИФИ, 2002. – 416 с.

ПРИЛОЖЕНИЕ 1. Опции функции MessageBox

Таблица 3. Наиболее распространенные значения параметра MBType

Значение	Результат
MB_ABORTRETRYIGNORE	отображаются кнопки Abort, Retry, Ignore
MB_ICONEXCLAMATION	отображается пиктограмма с восклицательным знаком
MB_ICONINFORMATION	отображается информационная пиктограмма с буквой "I"
MB_HAND	отображается пиктограмма с надписью "Stop"
MB_ICONQUESTION	отображается пиктограмма с вопросительным знаком
MB_ICONSTOP	то же, что и MB_HAND
MB_OK	отображается кнопка OK
MB_OKCANCEL	отображаются кнопки OK, Cancel
MB_RETRYCANCEL	отображаются кнопки Retry, Cancel
MB_YESNO	отображаются кнопки Yes, No
MB_YESNOCANCEL	отображаются кнопки Yes, No, Cancel

Таблица 4. Встроенные пиктограммы

Константа	Вид пиктограммы
IDI_APPLICATION	стандартная системная пиктограмма
IDI_ASTERISK	информационная пиктограмма с буквой "I"
IDI_EXCLAMATION	пиктограмма с восклицательным знаком
IDI_HAND	пиктограмма с надписью "Stop"
IDI_QUESTION	пиктограмма с вопросительным знаком
IDI_WINLOGO	пиктограмма с логотипом Windows

Таблица 5. Возвращаемые значения

Нажатая кнопка	Возвращаемое значение	Нажатая кнопка	Возвращаемое значение
Abort	IDABORT	No	IDNO
Retry	IDRETRY	Yes	IDYES
Ignore	IDIGNORE	OK	IDOK
Cancel	IDCANCEL		

ПРИЛОЖЕНИЕ 2. Стили диалоговых окон

Таблица 6. Наиболее распространенные стили диалоговых окон

Значение	Действие
DS_MODALFRAME	создается диалоговое окно с модальной рамкой (этот стиль может использоваться как для модальных, так и для немодальных окон)
WS_BORDER	создается окно с обычной рамкой
WS_CAPTION	создается окно со строкой заголовка
WS_CHILD	окно создается как дочернее
WS_HSCROLL	создается окно с горизонтальной полосой прокрутки
WS_MAXIMIZEBOX	создается окно с кнопкой максимизации
WS_MINIMIZEBOX	создается окно с кнопкой минимизации
WS_POPUP	окно отображается поверх всех других окон
WS_SYSMENU	создается окно с системным меню
WS_TABSTOP	элементы управления окна могут быть выбраны последовательным нажатием клавиши TAB
WS_VISIBLE	окно отображается при вызове
WS_VSCROLL	создается окно с вертикальной полосой прокрутки

ПРИЛОЖЕНИЕ 3. Опции меню

Таблица 7. Опции, наиболее часто используемые при создании меню

Опция	Использование
CHECKED	(опция пункта) для размещения рядом с пунктом меню отметки выбора пункта (для пунктов меню верхнего уровня не используется)
GRAYED	(опция пункта) для пометки пункта меню как не активного (серым, "бледным" цветом). Пункт не может быть выбран пользователем
HELP	(опция пункта) пункт меню может быть связан с командой вызова помощи (применяется только с пунктами типа MENUITEM)
INACTIVE	(опция пункта) пункт меню выводится в списке меню, но не может быть выбран в данных обстоятельствах
MENUBREAK	(опция пункта) то же, что и MENUBARBREAK, но без использования разделительной черты
MENUBARBREAK	(опция пункта) для указания пункта меню, выполняющего роль разделителя других пунктов (в меню верхнего уровня вызывает запись названия нового пункта с новой строки. В выпадающих меню название пункта будет размещено в новом столбце и отделено чертой)
OWNERDRAW	(опция пункта) для указания, что состоянием и изображением пункта меню, включая выделенное, неактивное и отмеченное состояния, отвечает владелец меню
POPUP	(опция пункта) для указания типа пункта меню как POPUP. При выборе этого пункта выводится список пунктов подменю
DISCARDABLE	(опция меню) для указания, что меню может быть удалено из памяти, если больше не используется
FIXED	(опция меню) для указания, что меню постоянно находится в памяти
LOADONCALL	(опция меню) для указания, что меню загружается при обращении

ПРИЛОЖЕНИЕ 4. Листинг каркаса обработчика сообщений диалогового окна (с кнопками OK и CANCEL)

Первый вариант листинга приведен на примере каркаса типа Hello для поддерживаемого им справочно-информационного окна About с одинаковой реакцией на нажатие кнопки OK и нажатие кнопки CANCEL – выход из справки.

```
LRESULT CALLBACK About(HWND hDlg, UINT message,
                       WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_INITDIALOG: return TRUE;
    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK ||
            LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return TRUE;
        }
```

```
        break;
        }
        return FALSE;
}
```

Второй вариант листинга приведен на примере диалогового окна с двумя кнопками OK и CANCEL с раздельными секциями для обработки сообщений – нажатие кнопки OK и нажатие кнопки CANCEL.

```
LRESULT CALLBACK TO_PROCESS_MESSAGE (HWND hDlg, UINT message,
                                     WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_INITDIALOG:
        // < ФРАГМЕНТ_ПОЛЬЗОВАТЕЛЯ >
        return FALSE;
    case WM_COMMAND:
        switch (wParam)
        {
        case IDOK:
            // < ФРАГМЕНТ_ПОЛЬЗОВАТЕЛЯ >
            EndDialog(hDlg,TRUE);
            break;
        case IDCANCEL:
            // < ФРАГМЕНТ_ПОЛЬЗОВАТЕЛЯ >
            EndDialog(hDlg,FALSE);
            break;
        default: return FALSE;
        }
        break;
    default: return FALSE;
    }
    return TRUE;
};
```

ПРИЛОЖЕНИЕ 5. Листинг каркаса обработчика сообщений диалогового окна (с кнопками OK и CANCEL, элементами управления типа Edit, List)

```
LRESULT CALLBACK DlgProc (HWND hDlg, UINT message,
                          WPARAM wParam, LPARAM lParam)
{
  int i; int j =0;
  int wmId, wmEvent;
  wmId    = LOWORD (wParam);
  wmEvent = HIWORD (wParam);
  int RecordsAmount = SendDlgItemMessage (hDlg, IDC_LIST1, LB_GETCOUNT, 0, 0);
  char SelectedText[34];
  switch (message)
  {
```

```
    case WM_INITDIALOG:
        SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0,
                            (LPARAM) "Иванов  233345");
        SendDlgItemMessage (hDlg, IDC_LIST1, LB_ADDSTRING, 0,
                            (LPARAM) "Сидоров  673345");
        SendDlgItemMessage (hDlg, IDC_LIST1, LB_SETCURSEL, 0,
                            (LPARAM) "Иванов  233345");
        break;
    case WM_COMMAND:
        switch (wmId)
        {
        case IDCANCEL:
                EndDialog (hDlg,TRUE);
                PostQuitMessage (0);
                break;
        case IDC_BUTTON_Edit:
                i = SendDlgItemMessage (hDlg, IDC_LIST1, LB_GETCURSEL, 0, 0);
                if ((i >= 0) && (i < RecordsAmount))
                {
                        SendDlgItemMessage(hDlg, IDC_LIST1, LB_GETTEXT, i,
                                           (LPARAM) SelectedText);
                        char Str1[78], Str2[78];
                        sscanf (SelectedText,"%s %s", TheRecord.Name, Str2);
                        SetDlgItemText (hDlg, IDC_EDIT_Name, TheRecord.Name);
                        SetDlgItemText (hDlg, IDC_EDIT_Number, Str2);
                        SendDlgItemMessage (hDlg, IDC_LIST1, LB_DELETESTRING, i, 0);
                }
                break;
        case IDC_LIST1:
                switch (wmEvent)
                {
                case LBN_DBLCLK:
                        //Обработка двойного щелчка по элементу списка
                        break;
                }
                break;
                default:  return FALSE;
        }
        break;
        default:  return FALSE;
    }
    return TRUE;
}
```

ПРИЛОЖЕНИЕ 6. Листинг каркаса обработки сообщений меню (обработчик главного окна с пользовательским меню)

Вариант листинга приведен на примере каркаса типа Hello. В нем поддерживается меню с пунктами File-Exit (выход из приложения) и Help–About (вызов диалогового окна со справочной информацией).

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
                                WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;
    switch (message)
    {
    case WM_COMMAND:
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            switch (wmId)
            {
            case IDM_ABOUT:
                    DialogBox(hInst, (LPCTSTR) IDD_ABOUTBOX, hWnd,
                                        (DLGPROC) About);
                     break;
            case IDM_EXIT:
                    DestroyWindow(hWnd);
                    break;
            default:
                    return DefWindowProc(hWnd, message, wParam, lParam);
            }
            break;
    case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            // <ФРАГМЕНТ_ПОЛЬЗОВАТЕЛЯ>
            EndPaint(hWnd, &ps);
            break;
    case WM_DESTROY:
            PostQuitMessage(0);
            break;
    default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
return 0;
}
```

Содержание

Учебное издание

Составители:
Муравьев Геннадий Леонидович
Савицкий Юрий Викторович
Хвещук Владимир Иванович

Методическое пособие

"ОСНОВЫ СОЗДАНИЯ WINDOWS-ПРИЛОЖЕНИЙ В СИСТЕМЕ MICROSOFT VISUAL STUDIO C++. Процедурный стиль", часть 2

для студентов специальности 1-40 03 01 «Искусственный интеллект»

Ответственный за выпуск: ***Г.Л. Муравьев***
Редактор: ***Т.В. Строкач***
Компьютерная вёрстка: ***Кармаш Е.Л.***
Компьютерный набор: ***Г.Л. Муравьев***
Корректор: ***Никитчик Е.В.***

Подписано в печать 03.10.2008 г. Формат 60x84 1/16.
Бумага "Снегурочка". Усл. п. л. 2,33. Усл. изд. л. 2,5. Тираж 60 экз. Заказ № 976.
Отпечатано на ризографе УО "Брестский государственный технический университет".
224017, Брест, ул. Московская, 267