Лабораторная работа №8 Программирование простейшего приложения с использованием библиотеки классов MFC

Иерархия классов MFC

Библиотека MFC содержит большую иерархию классов, написанных на С++. Структура иерархии приведена на рис. 1. В ее вершине находится класс СObject, который содержит различные функции, используемые во время выполнения программы и предназначенные, в частности, для предоставления информации о текущем типе во время выполнения, для диагностики, и для сериализации.

Информация о типе времени выполнения

Если указатель или ссылка ссылается на объект, производный от класса CObject, то в этом случае предусмотрен механизм определения реального типа объекта с помощью макроса RUNTIME\_CLASS(). Хотя в С++ имеется механизм RTTI, механизм, реализованный в MFC, намного более эффективен по производительности.

Диагностика

Каждый класс, производный от CObject, может по запросу проверить свое внутреннее состояние и выдать диагностическую информацию. Это интенсивно используется в MFC при отладке.

Сериализация

Сериализация - это механизм, позволяющий преобразовать текущее состояние объекта в последовательный поток байт, который обычно затем записывается на диск, и восстановить состояние объекта из последовательного потока, обычно при чтении с диска. Это позволяет сохранять текущее состояние приложения на диске, и восстанавливать его при последующем запуске.

Основные классы

Некоторые классы порождаются непосредственно от CObject. Наиболее широко используемыми среди них являются CCmdTarget, CFile, CDC, CGDIObject и CMenu. Класс ССmdTarget предназначен для обработки сообщений. Класс СFile предназначен для работы с файлами. Класс CDC обеспечивает поддержку контекстов устройств. Об контекстах устройств мы будем говорить несколько позднее. В этот класс включены практически все функции графики GDI. CGDIObject является базовым классом для различных DGI-объектов, таких как перья, кисти, шрифты и другие. Класс СMenu предназначен для манипуляций с меню. От класса CCmdTarget порождается очень важный класс CWnd. Он является базовым для создания всех типов окон, включая масштабируемые ("обычные") и диалоговые, а также различные элементы управления. Наиболее широко используемым производным классом является CFrameWnd. Как Вы увидите в дальнейшем, в большинстве программ главное окно создается с помощью именно этого класса.

От класса CCmdTarget, через класс CWinThread, порождается, наверное, единственный из наиболее важных классов, обращение к которому в MFC-программах происходит напрямую: CWinApp. Это один из фундаментальных классов, поскольку предназначен для создания самого приложения. В каждой программе имеется один и только один объект этого класса. Как только он будет создан, приложение начнет выполняться.

Функции-члены в MFC

Большинство функций, вызываемых в MFC-программе, являются членами одного из классов, определенных в библиотеке. Большинство функций API доступны через функции-члены MFC. Тем не менее, всегда можно обращаться к функциям API напрямую. Иногда это бывает необходимым, но все же в большинстве случаев удобнее использовать функции-члены MFC.

Глобальные функции в MFC

В библиотеке есть ряд глобальных функций. Все они начинаются с префикса Afx. (Когда MFC только разрабатывалась, то проект назывался AFX, Application Framework. После ряда существенных изменений AFX была переработана в MFC, но прежнее название сохранилось во многих идентификаторах библиотеки и в названиях файлов.) Например, очень часто используется функция AfxMessageBox(), отображающая заранее определенное окно сообщения. Но есть и член-функция MessageBox(). Таким образом, часто глобальные функции перекрываются функциями-членами.

Файл AFXWIN.H

Все MFC-программы включают заголовочный файл AFXWIN.H. В нем, а также в различных вспомогательных файлах, содержатся описания классов, структур, переменных и других объектов MFC. Он автоматически подключает большинство заголовочных файлов, относящихся к MFC, в том числе и WINDOWS.H, в котором определены все функции Windows API и другие объекты, которые используются при традиционном программировании на С и "чистом" API.

Каркас MFC-программы

Теперь мы создадим с помощью MFC небольшое приложение. Оно только отображает стандартное окно Windows, и больше ничего не делает. При закрытии окна приложение завершается. Это приложение послужит основой для всех других приложений, которые мы будем рассматривать.

В простейшем случае программа, написанная с помощью MFC, содержит два класса, порождаемые от классов иерархии библиотеки: класс, предназначенный для создания приложения, и класс, предназначенный для создания окна. Другими словами, для создания минимальной программы необходимо породить один класс от CWinApp, а другой - от CFrameWnd. Эти два класса обязательны для любой программы.

Кроме создания вышеупомянутых классов, в программе также должна быть организована обработка всех сообщений, поступающих от Windows. В данном примере программа еще ничего полезного не делает, поэтому отвечать на каждое сообщение не нужно. MFC обработает все сообщения, которые нас не интересуют. Тем не менее в этом примере присутствует карта откликов на сообщения, или просто карта сообщений. Позже мы рассмотрим ее подробнее.

Ниже приведен исходный код программы. Мы всегда будем помещать декларации классов и их реализацию в отдельные файлы. Это соответствует принятой практике программирования на С++.

Исходные тексты примера

simpwin.hpp

#include <afxwin.h>

// Класс основного окна приложения

class CMainWin: public CFrameWnd {

public:

CMainWin();

// Декларирование карты сообщений

DECLARE\_MESSAGE\_MAP()

};

// Класс приложения. Должен существовать только

// один экземпляр этого класса.

// Член-функция InitInstance() вызывается при запуске

// приложения.

class CApp: public CWinApp {

public:

BOOL InitInstance();

};

simpwin.cpp

#include <afxwin.h>

#include <string.h>

#include "SIMPWIN.HPP"

// Создание одного и только одного экземпляра

// приложения

CApp App;

// Реализация

BOOL CApp::InitInstance()

{

// Создание главного окна приложения и его

// отображение.

// Член CApp::m\_pMainWnd - это указатель на объект

// главного окна.

m\_pMainWnd = new CMainWin;

m\_pMainWnd->ShowWindow(SW\_RESTORE);

m\_pMainWnd->UpdateWindow();

// Сигнализируем MFC об успешной инициализации

// приложения.

return TRUE;

}

CMainWin::CMainWin()

{

// Создание окна с заголовком. Используется

// встроенный в MFC

// класс окна, поэтому первый параметр 0.

this->Create(0, "Простейшее приложение на MFC");

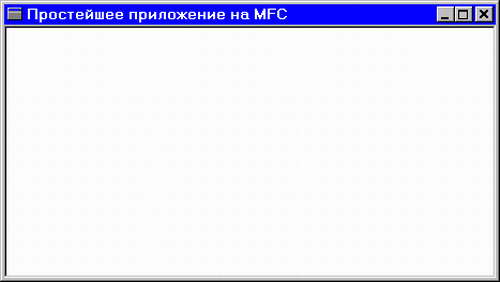
}

// Реализация карты сообщений

BEGIN\_MESSAGE\_MAP(CMainWin /\*класс окна\*/, CFrameWnd

/\*класс-предок\*/)

END\_MESSAGE\_MAP()

  
Рис. 2. Окно, созданное простейшим приложением на MFC

Результат выполнения программы показан на рис. 2. Рассмотрим программу подробнее.

Для создания стандартного окна в приложении должен наследоваться класс от CFrameWnd. В данном примере он называется CMainWin. Он содержит конструктор и макрос DECLARE\_MESSAGE\_MAP(). Макрос на самом деле разворачивается в декларацию карты сообщений, которая определяет, какая член-функция класса должна вызываться в ответ на сообщение Windows. Этот макрос применяется для любого окна, в котором обрабатываются сообщения. Он должен быть последним в декларации класса.

Само окно создается в конструкторе с помощью вызова функции Create(). Эта функция используется почти во всех приложениях. Она выполняет действия по созданию окна. В этом примере приведен самый простой случай ее использования. Пока нам нужно знать, что второй параметр определяет заголовок окна, а первый чаще всего равен NULL.

Класс CApp

приложения порождается от CWinApp. Этот класс отвечает за работу программы. В примере используется член-функция со следующим прототипом:

virtual BOOL CWinApp::InitInstance();

Это виртуальная функция, которая вызывается каждый раз при запуске программы. В ней должны производиться все действия, связанные с инициализацией приложения. Функция должна возвращать TRUE при успешном завершении и FALSE в противном случае. В нашем случае, в функции сначала создается объект класса CMainWin, и указатель на него запоминается в переменной m\_pMainWnd. Эта переменная является членом класса CWinThread. Она имеет тип CWnd\* и используется почти во всех MFC-программах, потому что содержит указатель на главное окно. В последующих двух строчках через нее вызываются функции-члены окна. Когда окно создано, вызывается функция c прототипом:

BOOL CWnd::ShowWindow(int How);

Параметр определяет, каким образом окно будет показано на экране. Наиболее распространенные значения следующие:

|  |  |
| --- | --- |
| Константа | Действие |
| SW\_HIDE | Окно становится невидимым |
| SW\_MAXIMIZE | Окно максимизируется |
| SW\_MINIMIZE | Окно минимизируетс |
| SW\_SHOW | Окно отображается, если было невидимо |
| SW\_RESTORE | Окно приводится к нормальному размеру |

При первом вызове в качестве первого параметра функции можно подставить член m\_nCmdShow, в котором хранится значение кода отображения окна, переданное в программу при запуске (например, в ярлыке Windows 95 пользователь может определить, как показывать окно при запуске приложения).

Функция UpdateWindow() посылает окну сообщение о том, что его содержимое нужно перерисовать. В дальнейшем мы рассмотрим этот процесс подробнее.

В конце программы помещена реализация карты сообщений:

BEGIN\_MESSAGE\_MAP(CMainWin /\*класс окна\*/,

CFrameWnd /\*класс-предок\*/)

END\_MESSAGE\_MAP()

Первый макрос всегда имеет два параметра, первый - класс окна, второй - класс, от которого порожден класс окна. В данном примере карта сообщений пустая, то есть все сообщения обрабатывает MFC.

Подробнее о создании масштабируемых окон

В примере была использована функция Create(), которая на самом деле имеет много параметров. Ее прототип таков:

BOOL CFrameWnd::Create(

LPCSTR ClassName, // Имя Windows-класса окна

LPCSTR Title, // Заголовок

DWORD Style = WS\_OVERLAPPEDWINDOW, // Стиль

const RECT &XYSize = rectDefault, // Область

CWnd \*Parent = 0, //Окно-предок

LPCSTR MenuName = 0, //Имя ресурса меню

DWORD ExStyle = 0, //Расширенные стили

CCreateContext \*Context = 0 // Доп. данные

);

Первый параметр, ClassName, определяет имя класса окна для оконной подсистемы Windows. Обычно его не нужно явно задавать, так как MFC выполняет всю необходимую черновую работу. В данных методических указаниях мы не будем использовать своих классов окон. Параметр Style задает стиль окна. По умолчанию создается стандартное перекрываемое окно. Можно задать свой стиль, объединив с помощью операции "или" несколько констант из приведенных ниже:

|  |  |
| --- | --- |
| Константа | Элемент окна |
| WS\_OVERLAPPED | Стандартное окно с рамкой |
| WS\_MAXIMIZEBOX | Кнопка максимизации |
| WS\_MINIMIZEBOX | Кнопка минимизации |
| WS\_SYSMENU | Системное меню |
| WS\_HSCROLL | Горизонтальная полоса прокрутки |
| WS\_VSCROLL | Вертикальная полоса прокрутки |

В дальнейшем мы не будем так подробно расписывать здесь все возможные значения констант, потому что всегда можно обратиться к справке Visual C++ и получить подробную информацию.

Начальное положение и размер окна определяются параметром XYSize. Это структура, которая определяет экранные координаты левого верхнего и правого нижнего угла прямоугольника, в котором будет отображено окно.

Все остальные параметры, которые мы будем использовать, будут рассмотрены в соответствующих разделах.