ГЛАВА 1

Что такое .NET

В этой главе...

Что такое .NET
Зачем нужны Web-службы на основе XML
Что будет с существующими приложениями MFC
Что делать программистам, работающим с компонентами COM
Подходит ли C++ для написания приложений .NET
Что должны знать начинающие программисты, использующие .NET

Что такое .NET

Независимо от того, являетесь ли вы опытным программистом, разрабатывающим приложения для Windows с помощью Visual C++, Internet-программистом, постоянно следующим последним тенденциям, или же начинающим разработчиком, пока еще не видящим различия между разнообразными программными платформами, я уверена, что приступая к изучению Visual Studio .NET, вы обязательно зададитесь вопросом: что же такое .NET?

Как произносится .NET? Как найти связанную с .NET информацию в Internet?

Символ "." перед словом "NET" произносится как "дот" (от англ. dot — точка. — *Прим. ред.*). В случаях, когда символ "." явным образом не указан (например, в названиях списков рассылок или в начале имени файла), подразумеваемое при этом сочетание ".NET" произносится как "дот нэт". Проводя поиск информации в Internet, в качестве ключевого слова лучше использовать "dotnet", нежели ".NET", поскольку слово "net" часто встречается на Web-страницах, не относящихся к новой технологии Microsoft.

К сожалению, однозначного ответа на этот вопрос не существует. Microsoft использует маркировку .NET на многих продуктах, однако все они вписываются в одну стратегию. Итак, .NET это:

- платформа, на которой выполняется программный код;
- библиотека программного кода, который можно использовать с помощью средств любого языка программирования;
- новые версии инструментов разработки приложений, например, Visual Studio;
- набор серверных продуктов, способствующих продвижению платформы .NET;
- новый способ проектирования и создания приложений, разделяющих выполнение задачи между различными компонентами, которые могут быть расположены даже в Internet.

Платформа .NET не предоставляет средства выполнения приложений, разработанных для Windows или какой-либо другой программной платформы. Вместо этого приложения .NET, скомпилированные в промежуточный язык Microsoft (Microsoft Intermediate Language — MSIL или же просто Intermediate Language — IL), запускаются в рамках общеязыковой среды выполнения (Common Language Runtime — CLR). Данный "механизм выполнения программ" преобразовывает код MSIL в оригинальный программный код, а также предоставляет многие другие функции, включая управление памятью, обеспечение безопасности и поддержку взаимодействия между приложениями (например, использование приложением объекта COM или вызов кода из DLL). На сегодняшний день платформа .NET реализована только на некоторых версиях операционной системы Windows, что, тем не менее, уже спасает многих разработчиков от лишней головной боли.

Вместе с Visual Studio .NET поставляется набор инструментальных средств разработки программного обеспечения .NET (.NET Software Development Kit — .NET SDK), включающий в себя классы .NET, которые можно использовать в рамках любого языка программирования .NET и которые обеспечивают расширенную функциональность для разработчиков. Некоторые из этих средств доступны и Windows-программистам в виде таких библиотек, как MFC, или же в виде служб, предоставляемых в рамках технологий COM и

COM+, однако большая их часть — например, работа с XML или проверка прав доступа — является привилегией .NET.

Visual Studio .NET является новейшей версией Visual Studio, существенно отличающейся от всех предыдущих версий. В прошлом Visual Studio представляла собой набор связанных между собой инструментов, каждый из которых имел свой пользовательский интерфейс, а также свои преимущества и недостатки. Сегодня же Visual Studio поставляется в виде единого продукта, который можно использовать для создания кода на различных языках программирования. И если раньше для написания функции обработки щелчка мышью на кнопке диалогового окна вы могли использовать подход, принятый в Visual Basic, или подход, принятый в Visual C++, то теперь у вас есть единственный подход, принятый во всей Visual Studio. Естественно, вам, как разработчику, может понадобиться изучить новые свойства Visual Studio .NET. Для этого я рекомендую обратиться к приложениям B, "Visual Studio: среда разработки, интерфейс пользователя и система меню", и Г, "Переход к Visual Studio .NET от Visual C++ 6", находящимся в конце данной книги. Следует особо отметить, что информация, представленная в указанных выше приложениях, будет полезной как новичкам, так и опытным программистам Visual C++.

Некоторые языки программирования, используемые в Visual Studio .NET, претерпели существенные изменения в результате адаптирования к платформе .NET. Следует отметить, что язык программирования Visual C++ изменился меньше. Так, в него было добавлено всего лишь несколько ключевых слов, позволяющих разработчикам C++ использовать специфические свойства или возможности .NET при написании кода. Существующие проекты Visual C++ 6 должны открываться и компилироваться в Visual Studio .NET без всяких проблем. Если вам приходилось много слышать о том, что с переходом на .NET языки программирования претерпели существенные изменения, это, скорее всего, относилось к Visual Basic, который действительно был слегка изменен. Программистам же, работающим с Visual C++, беспокоиться на этот счет не стоит.

На рис. 1.1 показано, как различные языки программирования .NET, Web-службы на основе XML, язык XML, библиотека классов .NET и среда выполнения CLR участвуют в создании приложения .NET.

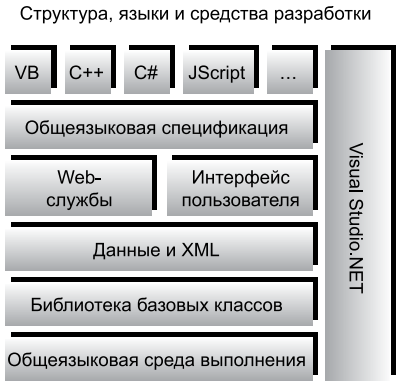

Рис. 1.1. Платформа .NET — это намного больше, чем просто очередная версия компилятора

Помимо всего прочего, компания Microsoft выпускает под маркой .NET и несколько серверных продуктов:

- Microsoft Application Center 2000;

- Microsoft BizTalk Server 2000;
- Microsoft Commerce Center 2000;
- Microsoft Exchange 2000;
- Microsoft Host Integration Server 2000;
- Microsoft Internet Security and Acceleration Server 2000;
- Microsoft Mobile Information 2001 Server;
- Microsoft SQL Server 2000.

Сервер Application Center — это дополнительный модуль для Windows 2000 Server, предназначенный для работы с часто посещаемыми, перегруженными Web-узлами. Он упрощает развертывание узла на несколько серверов. Одним из достоинств сервера Application Center является то, что он позволяет управлять несколькими серверами так же легко, как и одним. По этой причине разработчики подобного Web-узла могут относиться к нему как к обычному узлу, не задумываясь о том, на какой мощной платформе он стоит.

Приложение BizTalk Server значительно упрощает обмен документами XML между приложениями или организациями. Для обработки поступающих документов BizTalk Server применяет Web-службы на основе XML или более традиционные компоненты COM. Большая часть рутинных функций (например, обслуживание очереди документов или направление поступающих документов на соответствующий обработчик) уже встроена в BizTalk Server, поэтому кодировать их вручную больше не нужно. Дополнительный модуль BizTalk Orchestration Designer — это средство, в чем-то похожее на Visio, которое автоматизирует документооборот, вовлекая в процесс автоматизации компоненты COM или объекты .NET.

Пакет Commerce Center — это набор инструментов для создания Web-узлов электронной коммерции. Прежде чем начинать разработку собственной корзины для покупок, средств обслуживания покупателей или анализатора посещаемости сайта, внимательно изучите пакет Commerce Center — думаю, ему всегда будет что вам предложить.

Microsoft Exchange — это гораздо больше, чем просто почтовый сервер; он содержит средства для совместной работы. Многие Web-службы на основе XML в своей работе обращаются к Exchange — к примеру, для планирования расписания встреч.

Сервер Host Integration Server упрощает доступ к существующим системам баз данных, таким, как DB2 для OS/390 и AS/400, передачу файлов для AS/400, AS/36 и VSAM, работу с транзакциями IMS и очередями данных AS/400. Благодаря службам COM+ и MTS вы можете работать с данными точно так же, как если бы доступ к ним осуществлялся через ODBC.

Приложение Internet Security and Acceleration Server представляет из себя кэширующий прокси-сервер и брандмауэр, который одновременно поддерживает функционирование и безопасность корпоративных сетей.

Пакет Mobile Information Server предназначен для поставщиков услуг беспроводной связи, например, операторов мобильной связи и администраторов корпоративных сетей intranet. Он упрощает синхронизацию и передачу информации пользователям беспроводных и мобильных устройств, предоставляя для этого все средства и возможности .NET.

Пакет SQL Server — высокопрофессиональный, масштабируемый и мощный сервер баз данных от компании Microsoft. К последней версии SQL Server была добавлена существенная поддержка XML и .NET.

Самое важное заключается в том, что все перечисленные выше продукты могут работать совместно, обмениваясь документами XML, вызывая методы друг друга или поддерживая очереди сообщений. Благодаря этому создавать новые решения становится все проще и проще — в большинстве случаев решение можно просто "собрать" из суще-

ствующих компонентов. Теперь, прежде чем начинать написание кода нового решения .NET, задумайтесь — а не пытаетесь ли вы в очередной раз изобрести колесо?

Итак, технология .NET имеет все шансы для того, чтобы стать силой, которая вдохновит разработчиков на предложение собственного кода через Internet и использование компонентов, предложенных другими разработчиками. В то же время .NET обеспечивает определенную независимость от платформы (пока что в рамках семейства Windows, а позднее, возможно, это распространится и на другие платформы), а также от используемого языка программирования. Все эти возможности, без сомнения, значительно упростят работу программистов нового поколения (а вот "старичкам" придется много чему научиться).

Полезные Web-узлы

В сети Internet можно найти много полезной информации о .NET, и не только на Web-узлах компании Microsoft. Ниже приведены некоторые Web-узлы, на которых можно найти образцы исходного кода, полезные Web-службы на основе XML и просто последние новости.

- `http://www.usingvisualc.net`. Узел поддержки этой книги. Здесь можно найти образцы исходного кода, исправления, обновления и дополнения, которые не вошли в книгу.
- `http://www.msdn.microsoft.com/net`. Главный узел Microsoft, предназначенный для разработчиков .NET.
- `http://www.discuss.develop.com/dotnet.html`. Архивы форума *DevelopMentor Dotnet* с возможностью поиска. Прежде чем полагаться на сведения той или иной статьи, обращайте внимание на дату ее выпуска, потому что этот форум функционировал еще до официального появления первой бета-версии Visual Studio .NET.
- `http://www.devx.com/dotnet/`. Ссылки на статьи, образцы кода и существующие Web-службы на основе XML.
- `http://www.gotdotnet.com`. Этот узел поддерживается группой разработчиков .NET. Здесь содержатся образцы кода и обучающие примеры, многие из которых были предоставлены не Microsoft.

Зачем нужны Web-службы на основе XML

В основе .NET лежат Web-службы на основе XML. Возможно, вам доводилось посещать презентации Microsoft, где улыбающиеся люди договариваются о посещении доктора со своих компьютеров и мобильных телефонов. Затем они проверяют погоду, заказывают еду и, выполнив все дела, выходят подышать вечерним воздухом. Все это делается с помощью Web-служб на основе XML, как радостно сообщает ведущий презентации, и все это возможно уже сегодня.

Сеть Internet быстро заполняется Web-службами на основе XML. Все они бесплатные (или в крайнем случае недорогие) и легкодоступны. Возможно, вам понятно, почему стоит писать приложения, использующие Web-службы на основе XML: потому что они облегча-

ют программирование. Если приложению нужно конвертировать валюту или получать последний прогноз погоды или спортивные новости, почему бы не воспользоваться для этой цели — заметьте, уже написанными — Web-службами на основе XML? Программировать вручную придется значительно меньше, а приложение станет полезным и популярным. Поистине, Web-службы на основе XML — замечательное изобретение! Более того, они совместимы с разными платформами, разными языками и продуктами разных производителей: к примеру, Web-служба на основе XML, написанная на Visual C++, может быть использована программой, написанной на Perl и функционирующей под операционной системой Unix. Вы можете написать приложение на Visual C++, использующем Web-службу на основе XML, написанную на Java и работающую на компьютере с операционной системой, о которой вы вообще никогда не слышали. Главное, чтобы служба была доступна по сети Internet и соответствовала всем стандартам Web-служб на основе XML — все остальное (в частности, использование конкретного языка программирования и операционной системы) значения не имеет.

Но кто же те добрые души, которые пишут и обеспечивают работу Web-служб на основе XML? И почему имеет смысл присоединиться к этим людям в написании средств, доступных по сети Internet для всех желающих? На это есть четыре причины, весьма существенных с точки зрения коммерции.

- За доступ к Web-службе на основе XML можно взымать определенную плату, поэтому в случае успеха написанная вами служба может стать неплохим источником доходов.
- Бесплатная Web-служба на основе XML популяризирует основную услугу или продукт вашей компании.
- Наличие "частной" Web-службы на основе XML может упростить архитектуру многоярусного приложения, которое в качестве инфраструктуры использует Internet.
- Написание Web-службы на основе XML освобождает от создания визуального интерфейса пользователя.

Существует множество способов получать деньги за пользование Web-службой на основе XML. К примеру, ее можно сделать доступной только для подписчиков за определенную ежемесячную плату. Это способ очень удобен для распространения постоянно обновляющихся сводок погоды или, скажем, сведений о ситуации на дорогах. В этом случае информация, предоставляемая Web-службой, будет отображаться на экране пользователей посредством какого-нибудь другого приложения.

Другой способ предусматривает плату за каждое использование Web-службы. Вне зависимости от способа идентификации пользователя — по IP-адресу или идентификатору, передаваемому Web-службе, — системе легко определить, кто пользовался службой, и обновить выставляемый этому пользователю счет. Таким способом удобно, к примеру, продавать доступ к курсу обмена валют.

Бесплатная Web-служба на основе XML может помочь в продвижении основных продуктов или услуг вашей компании. Если компания, занимающаяся вопросами страхования здоровья, будет использовать Web-службу на основе XML для назначения консультации у врача, последний сразу же обзаведется большим количеством пациентов. Или, скажем, если агенты туристической фирмы будут применять Web-службу на основе XML для заказа столика в ресторане, номера в отеле или аренды машины, дела компаний, предоставляющих эти услуги, пойдут в гору, поскольку агенту станет легче привлекать к ним посетителей.

На создание Web-узла, предназначенного для заказа столиков в ресторане, уходит много времени и усилий. Подобный узел должен иметь интересную графику, легкий интерфейс, быть мощным и обладать высокой пропускной способностью, чтобы все это ве-

ликолепие загружалось мгновенно, не действуя на нервы посетителей. После этого вам придется потратить много денег и усилий на то, чтобы привлечь потенциальных клиентов и внушить им, что заказывать столики нужно именно у вас. Хотя Web-служба на основе XML вообще не нуждается в графическом интерфейсе! Она используется другими приложениями, которым совсем не нужна огромная пропускная способность, потому что у них нет "навороченной" графики, инструкций или страниц типа "Связаться с нами". Единственное, что от вас требуется — это убедить тех самых агентов туристических или других компаний, что они могут бронировать столики с помощью вашей Web-службы. Расскажите им, как получить доступ к вашей службе, и все пойдет как по маслу! Звучит просто и притягательно.

Так же просто разработать Web-службу и для компаний, которые не собираются предоставлять ее в общее пользование. Применение Web-служб на основе XML в процессе разработки intranet- или extranet-проектов может сэкономить массу времени и усилий. Многоярусные приложения могут использовать Internet в качестве инфраструктуры и в то же время быть защищенными от внешних вторжений. Настольное приложение может вызвать Web-службу на основе XML, находящуюся на удаленном сервере, для того чтобы добавить новую запись в базу данных или просмотреть записи, измененные за последнее время, — возможно потому, что посетители подписались на какое-либо событие или разместили новые заказы.

Правительственным организациям, университетам и другим учреждениям время от времени приходится публично оглашать какую-либо информацию. Для привлечения внимания они, как и все остальные компании, вынуждены добавлять к своим презентациям большое количество графики и других "примочек". Однако ограниченные финансовые возможности зачастую приводят к тому, что некоторая информация вообще опускается. Предоставляя же информацию через Web-службу на основе XML, данные учреждения получают возможность переложить всю работу по созданию графического интерфейса на плечи других и сосредоточиться только на подготовке информации, что, естественно, обойдется им гораздо дешевле. В настоящий момент многие разработчики занимаются только тем, что собирают информацию со "скучных" Web-узлов для того, чтобы разместить ее на более привлекательных узлах. Таким разработчикам, несомненно, понравится возможность работы с Web-службами на основе XML.

Что будет с существующими приложениями MFC

Все эти разговоры о Web-службах на основе XML очень хороши, но что делать с уже имеющимися знаниями и навыками? Неужели COM, MFC и ATL больше никому не нужны? Подобные вопросы волнуют всех разработчиков. Но нет, старые технологии еще используются, и знание их все так же ценится. Еще несколько лет назад меня спрашивали: "Неужели появление Java вытеснит C++?". На это я отвечала: "А разве C++ вытеснил Cobol?". Конечно нет. Соответственно и .NET будет использоваться наравне с традиционным программированием для Windows.

Одна из целей, поставленных на этапе проектирования Visual C++ .NET (по словам члена группы разработки компании Microsoft), — "грандиозное обновление", направленное на тех, кто занимался программированием для Windows без использования .NET. К слову, эта цель не стояла для других частей Visual Studio .NET, включая и Visual Basic .NET. Если вы привыкли работать с MFC или ATL, использование этих средств в Visual Studio .NET только увеличит вашу продуктивность. Ни одно из привычных вам средств не

будет скрыто или усложнено. Более того, для поддержки старых приложений или создания новых по типу старых вам даже не придется изучать XML, .NET или еще что-нибудь подобное.

Как ни странно, в некоторых ситуациях все еще лучше придерживаться старой доброй библиотеки MFC и других технологий, применявшихся до появления .NET. То, что в мире .NET называется "классическим", или "неуправляемым", кодом, рекомендуется использовать в следующих случаях.

- У вас есть большое приложение, которое хорошо работает и нуждается только в незначительном обновлении. Как гласит старое изречение: "Не нужно чинить то, что еще не сломалось". Это справедливо и для программного обеспечения. Поддерживайте работу приложения и радуйтесь возможности совмещать управляемый и неуправляемый коды. Возможно, когда-нибудь вам захочется интегрировать свое приложение в среду .NET, однако переписывать его код в любом случае не нужно.
- Ваше приложение имеет ограничения доступа, которые подразумевают невозможность сбора чужой информации и распространения своей информации по Internet с помощью Web-служб на основе XML. Подобное приложение ничего не приобретет от перехода к .NET.
- У вас высокие требования к скорости работы приложения. Неуправляемый код C++ выполняется быстрее, чем управляемый код MSIL (.NET). Некоторым высокоскоростным приложениям вообще не имеет смысла переходить на .NET, а остальные, возможно, перейдут туда позже, когда скорость выполнения управляемого кода повысится.

Большая часть этой книги посвящена традиционному программированию для Windows, которое также называется программированием "не для .NET", "классическим", или "неуправляемым", кодом. Речь идет о следующих главах.

- В главе 2, "Создание первого приложения", рассмотрены различные типы приложений, которые можно создавать с помощью Visual C++.NET, а также рассказано о мастере MFC Application Wizard (Мастер приложений MFC).
- В главе 3, "Взаимодействие с приложением", объясняется, как добавить к приложению диалоговое окно для сбора информации, а также рассматривается концепция сообщений в Windows.
- В главе 4, "Программирование вывода информации в приложении", рассказывается о концепции "документ-представление", а также демонстрируются некоторые текстовые и графические приемы для оформления частей приложения, не связанных с диалоговыми окнами.
- Глава 5, "Печать и сохранение данных", как и следует из ее названия, посвящена двум основным аспектам любого Windows-приложения, предназначенного для работы с документами.
- В главе 6, "Создание полноценного приложения ShowString", подытоживается материал предыдущих глав и рассказывается, как добавить меню и создать простое приложение, на примере которого впоследствии будут продемонстрированы большинство приемов программирования для Windows.
- В главе 7, "Панели инструментов, строки состояния и элементы управления общего назначения", рассмотрено, как обогатить интерфейс приложения дополнительными элементами управления.
- В главе 8, "Справочная система, страницы свойств и мастера", рассказывается, как обеспечить пользователей необходимой справочной информацией о приложе-

нии и помочь им в выполнении стандартных процедур с помощью пошаговых мастеров.

- Глава 9, "Создание компонентов COM+ с помощью библиотеки ATL", посвящена основам концепции COM и созданию простейших компонентов COM.
- В главе 10, "Internet-программирование", рассказано о "старом" подходе к программированию для Internet — т.е. о приемах программирования, применявшихся до появления .NET (и, надо сказать, довольно простых).
- В главе 11, "Программирование взаимодействия приложения с базой данных", речь идет о том, как добавить к приложению поддержку баз данных.
- Глава 12, "Повышение производительности приложений", посвящена проблемам оптимизации кода, утечки памяти и другим аспектам традиционного программирования для Windows, влияющим на производительность работы приложения.
- В главе 13, "Отладка", рассказано о полезном и мощном отладчике Visual Studio, а также о способах быстрого обнаружения ошибок в исходном коде.
- В главе 14, "Создание многозадачных приложений на основе потоков Windows", рассказано о поддержке библиотекой MFC потоков Windows, а также о том, как приложение может выполнять несколько задач одновременно.
- Глава 15, "Особые типы приложений Win32", посвящена библиотекам DLL, консольным приложениям, службам Windows NT и Windows 2000 и другим нестандартным типам приложений.
- Приложение A, "Программирование для Windows и класс Cwnd", содержит краткий обзор программирования для Win32 и рассказывает, как упростить этот процесс с помощью библиотеки MFC.

Что делать программистам, работающим с компонентами COM

Программист, работающий с компонентами COM, обычно сталкивается с одной из следующих задач:

- поддержка приложения, использующего компоненты COM;
- написание приложений, использующих компоненты COM;
- поддержка одного или нескольких компонентов COM;
- написание новых компонентов COM.

Разумеется, большинство программистов, работающих с COM, занимаются сразу несколькими из перечисленных видов деятельности. В любом случае данный способ разбиения работ на четыре отдельных подзадачи удобен для того, чтобы изучить влияние .NET на каждую из них.

Для поддержки приложения, использующего компоненты COM, ничего менять не нужно. При желании, вы можете перевести приложение на платформу .NET и использовать те же компоненты COM, но уже в среде .NET. Приложение также может оставаться "классическим" приложением COM, даже если компоненты COM, с которыми оно работа-

ло, были переизданы как объекты .NET. Эти объекты могут быть использованы в приложении COM точно так же, как если бы они были компонентами COM.

При написании нового приложения вам может захотеться сделать его приложением .NET. Это неудивительно — ведь многочисленные средства .NET позволяют значительно сэкономить время и усилия. Тем не менее, бросив ностальгический взгляд на библиотеку компонентов COM, вы можете отказаться от этой идеи, подумав, что создать приложение .NET, используя компоненты COM, невозможно. И напрасно! Приложение .NET может взаимодействовать с компонентами COM так, как если бы это были объекты .NET.

Если вы уже написали (и, возможно, продали) несколько компонентов COM, вас может обеспокоить вопрос их перевода на платформу .NET. С этим все в порядке — по своей сути компонент COM является объектом .NET (или, по крайней мере, его можно использовать точно так же, как если бы это был объект .NET).

Если вы занимаетесь разработкой нового компонента, вам, возможно, не захочется разрабатывать объект .NET, поскольку большинство программных продуктов, используемых в вашей отрасли, пока еще не поддерживают .NET. Что ж, выбор за вами — вы можете написать компонент COM, поскольку он все равно может использоваться как объект .NET. Однако если вас привлекают новые возможности, предоставляемые .NET, напишите объект .NET. Старые приложения смогут работать с ним как с компонентом COM.

С точки зрения пользователей между компонентами COM и .NET нет никакой разницы, поскольку они взаимозаменяемы. Это оказалось возможным благодаря "чудесной" способности к взаимодействию компонентов COM, что и было главной целью разработчиков .NET. Модель COM приобрела большую популярность, и поэтому новая технология была обязана обеспечить поддержку огромного числа существующих компонентов COM и приложений, работающих с компонентами COM (а также их пользователей).

Если вы потратили много времени на изучение программирования COM и прекрасно разбираетесь в терминах типа `HRESULTS`, `GUID`, `QueryInterface` и т.п., то все это вам пригодится. Однако, если программирование COM всегда казалось вам чем-то вроде ночного кошмара, попробуйте поработать с .NET — все те же компоненты стали доступными в виде объектов .NET. Теперь вы можете создать объект .NET, с которым смогут работать все существующие приложения COM. Более того, вы даже можете написать объект .NET, наследующий компонент COM.

Более подробно о том, как легко взаимодействуют COM и .NET, рассказывается в главе 23, "Взаимодействие COM и .NET".

Подходит ли C++ для написания приложений .NET

В настоящее время довольно много говорится о Visual Basic .NET и о том, как сильно он отличается от Visual Basic 6. Много шума поднято и вокруг нового языка C#. Ну а как насчет Visual C++? Должны ли программисты, работающие с C++, изучать новый язык? Должны ли начинающие программисты пропустить изучение C++ и перейти к другому языку? Ни в коем случае.

В предыдущих версиях Visual Studio интерфейс среды разработки отличался для каждого языка программирования. Например, для выполнения какой-нибудь обычной работы (скажем, для редактирования исходного кода, связанного с кнопкой диалогового окна) программисты Visual Basic должны были дважды щелкнуть на соответствующем элементе, тогда как программисты Visual C++ пользовались для этой цели раскрывающимся

списком. Теперь интерфейс пользователя Visual Studio одинаков для любого языка. Более того, теперь вам не нужно выбирать язык программирования с помощью команды меню Пуск⇨Программы, как это делалось раньше. Сравните подменю, открывающиеся при выборе в меню Программы пункта Microsoft Visual Studio 6 (рис. 1.2) и пунка Microsoft Visual Studio.NET (рис. 1.3).

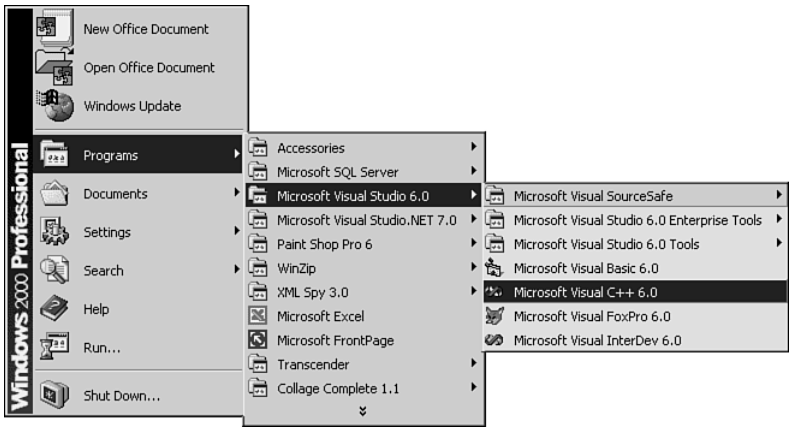

Рис. 1.2. Пакет Visual Studio 6 представлял собой набор взаимосвязанных средств— по одному на каждый язык программирования

Подобный уровень интеграции означает, что вы можете использовать Visual Studio для работы с любым поддерживаемым ею языком программирования. Более того, данная среда разработки позволяет смешивать языки в рамках одного решения. При использовании среды выполнения Common Language Runtime ее средства и службы одинаково доступны для всех языков программирования. Раньше программист Visual Basic 6 не мог использовать всю эффективность библиотеки MFC — ему приходилось дожидаться выпуска библиотеки Visual Basic. Сейчас же программисты, работающие с Visual Basic, Visual C++ и другими языками, поддерживаемыми Visual Studio, в одинаковой мере используют возможности среды Common Language Runtime.

Следует заметить, что языки программирования, используемые в Visual Studio, могут и не быть языками Microsoft. Данную среду разработки можно использовать для редактирования, запуска и отладки приложений, написанных на любом языке, совместимом с Visual Studio. На рис. 1.4 показана отладка в Visual Studio приложения, написанного на языке Perl.

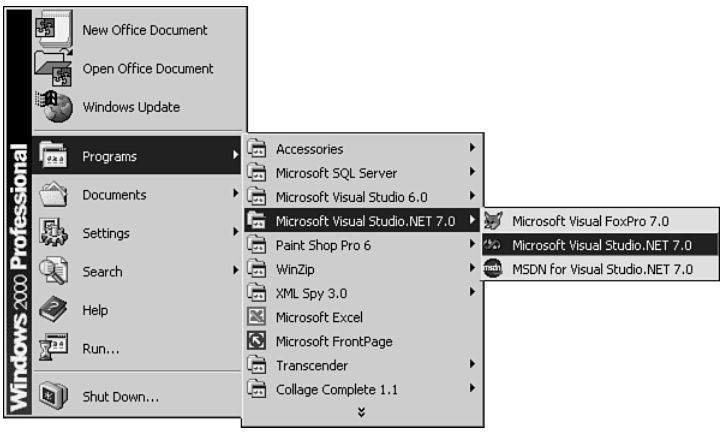

Рис. 1.3. Пакет Visual Studio .NET — единый продукт, предназначенный для работы с несколькими языками программирования

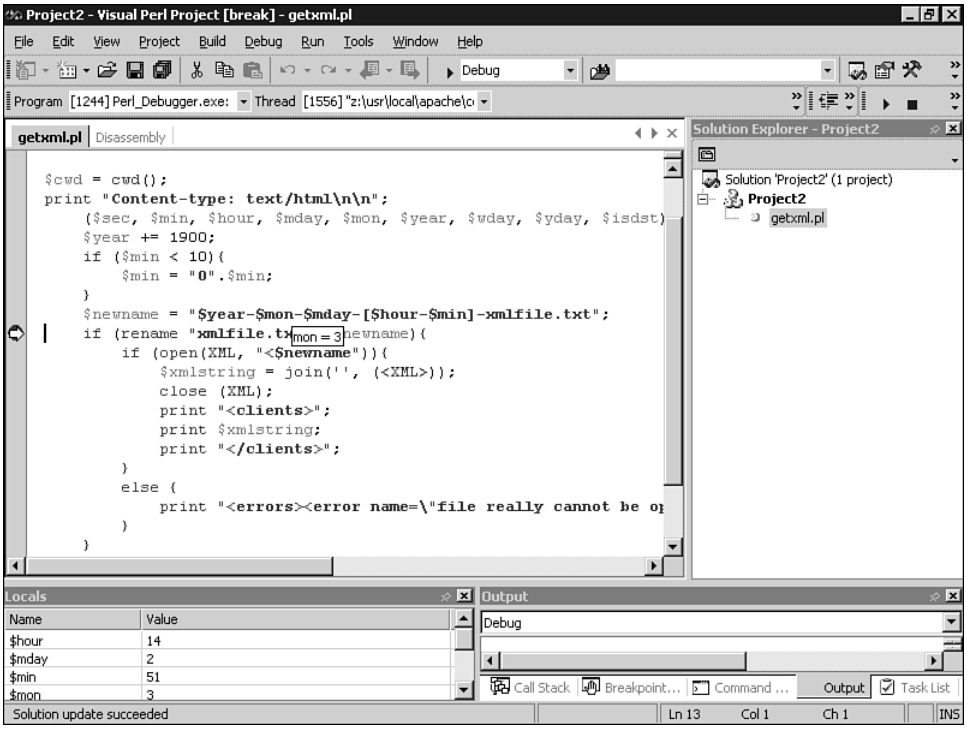

Рис. 1.4. Среда разработки Visual Studio .NET поддерживает и языки программирования производства не Microsoft, такие, как Perl

Работая в среде .NET, вы можете создать базовый класс на Visual C++, породить от него класс на Visual Basic и использовать новый класс в программе на C#. Теперь выбор языка программирования в основном зависит только от удобства, привычек и имеющихся знаний. Зачем учить новый язык, если ты уже знаком с одним из языков .NET?

Впрочем, существует одно исключение, которое, несомненно, обрадует поклонников C++. Говоря в двух словах, мы можем делать то, чего не могут другие. Код C++ может

быть управляемым (что означает возможность полноценного использования всех средств .NET) или неуправляемым. Неуправляемый код обладает меньшей переносимостью между платформами и не может использовать все возможности .NET. Тем не менее, у него есть и очень важное преимущество — он намного быстрее. По этой причине, разрабатывая приложения, для которых скорость выполнения стоит на первом месте, рекомендуется применять неуправляемый код C++ с использованием таких "классических" библиотек, как ATL.

Что должны знать начинающие программисты, использующие .NET

Если вы довольно долго программировали на Visual C++ (скажем, с 2000 года или ранее) и хотите начать работать с .NET, я бы рекомендовала уделить немного внимания той половине книги, которая посвящена программированию для Win32. Попробуйте выполнить то, что вы уже умели делать в предыдущих версиях Visual C++. В новой версии Visual Studio многие из мастеров исчезли или скрыты, а способы связывания элементов приложения с исходным кодом несколько изменились. Вот почему я советую хорошо освоиться с новым интерфейсом Visual Studio, прежде чем приступать к изучению .NET. Большинство вопросов, касающихся обновления интерфейса, рассмотрены в приложении Г, "Переход к Visual Studio .NET от Visual C++ 6". Прочитав его, вы избавите себя от лишних мучений, связанных с поисками в меню той или иной команды, которая была удалена или перемещена.

Если вы никогда не работали с Visual Studio и не собираетесь заниматься поддержкой существующих приложений MFC, я рекомендую начать с той половины книги, которая посвящена непосредственно .NET. Это следующие главы.

- Глава 16, "Общеязыковая среда выполнения", познакомит вас с замечательными средствами, доступными программистам .NET во всех языках.
- Глава 17, "Первое знакомство с .NET", проведет вас по этапам создания объекта .NET — базового "кирпичика" всей концепции .NET.
- В главе 18, "Интеграция с Visual Basic", рассказано, как совмещать исходный код на Visual C++ и на Visual Basic.
- В главе 19, "Интеграция с C#", рассказано, как совмещать исходный код на Visual C++ и на Visual C#. (C# — это новый язык, конкретная реализация которого должна подчиняться определенным стандартам, принятым для его основных элементов. Visual C# является реализацией языка C# от компании Microsoft.)
- В главе 20, "Управляемый и неуправляемый код C++", рассматриваются преимущества и недостатки управляемого и неуправляемого кодов. Вы узнаете, в каких ситуациях следует использовать тот или иной тип кода, а также как их комбинировать.
- В главе 21, "Создание Web-службы на основе XML", показано, как легко и просто предоставить свои компоненты для использования другими приложениями по сети Internet.
- В главе 22, "Доступ к базам данных с помощью ADO.NET", рассматривается новейший способ доступа к данным, разработанный компанией Microsoft, а также

объясняется, в чем состоят преимущества этого способа доступа перед уже существующими.

- В главе 23, "Взаимодействие COM и .NET", рассказано об использовании компонентов COM в качестве объектов .NET и наоборот. Не бойтесь — это гораздо проще, чем можно было подумать.
- В главе 24, "Безопасность и политики", рассматривается, как рабочая среда .NET может уберечь приложение от нападок компьютерных злоумышленников, а также как осуществляется управление безопасностью приложений.

Вся концепция .NET построена на понятии XML. Означает ли это, что для работы с .NET необходимо знать XML? Разумеется, нет. Одним из отличительных свойств .NET как раз и является то, что данная платформа предоставляет в пользование разработчиков все преимущества XML вне зависимости от того, знают разработчики этот язык или нет.

Тем не менее, хотя бы поверхностное знакомство с языком XML может принести еще большую пользу при работе с .NET. Документ XML — это обыкновенный документ, написанный вполне читабельным языком, а вовсе не двоичный файл. Это значит, что его можно просматривать и редактировать даже с помощью Блокнота, а не только какого-нибудь специального приложения. Если вы сможете разобраться в документе XML, сгенерированном вашим приложением, вы быстро определите имеющиеся ошибки и избавите себя от лишней головной боли. Более подробно о языке XML (разумеется, с точки зрения .NET) можно прочитать в приложении Б, "Обзор языка XML".

Что дальше

Пришло время начинать! Если вы хотите сразу же погрузиться в пучины .NET, обратитесь к главе 16, "Общеязыковая среда выполнения", посвященной среде выполнения .NET, в которой запускаются все управляемые приложения. Если же вы предпочитаете немного повозиться с традиционным программированием для Windows с помощью библиотеки MFC, начните с главы 2, "Создание первого приложения". Опытным программистам Visual C++ будет интересно ознакомиться с приложением Г, "Переход к Visual Studio .NET от Visual C++ 6". В случае возникновения каких-либо проблем попробуйте найти интересующую вас тему с помощью предметного указателя. Удачи!

ГЛАВА 2

Создание первого приложения

В этой главе...

Создание приложения Windows

Время поработать самостоятельно

Создание простого диалогового приложения

Создание динамически связываемых библиотек, консольных приложений и пр.

Изменение настройки параметров проекта

Разбор программного кода, генерируемого мастером MFC Application Wizard

Создание приложения Windows

Visual C++ — не просто компилятор программного кода, это генератор программного кода. С его помощью в считанные минуты можно создавать приложения Windows, основанные на библиотеке классов MFC, “заказав” генератору приложений MFC Application Wizard “приготовить” для вас некоторый комплексный обед из уже готовеньких блюд — фрагментов программного кода. MFC Application Wizard — очень эффективный инструмент, позволяющий на ходу создавать программный код, типичный практически для любого приложения Windows. В конце концов, вы ведь не первый программист, которому понадобилось иметь в приложении окно регулируемых размеров с кнопками максимизации и минимизации, а также меню File (Файл), в которое включены команды Open (Открыть), Close (Закрыть), Print Setup (Параметры печати), Print (Печать) и Exit (Выход), не так ли?

Мастера создания приложений, доступные посредством диалогового окна New Project (Новый проект), позволяют создавать много разных типов приложений, но первое, что обычно требуется большинству пользователей, — это исполняемая программа (файл с расширением `.exe`). В этом смысле мастер MFC Application Wizard — как раз то, что им нужно. Кроме того, большинство также хотело бы получить от MFC Application Wizard готовые фрагменты кода — классы, объекты, функции, которые присутствуют едва ли не в каждой порядочной программе. Для того чтобы создать программу подобного типа, выполните команду File⇨New⇨Project (Файл⇨Создать⇨Проект). В области Project Types (Типы проектов) выберите тип проекта **Visual C++ Projects** (Проекты Visual C++), а в области Templates (Шаблоны) — элемент **MFC Application** (Приложение MFC) (рис. 2.1).

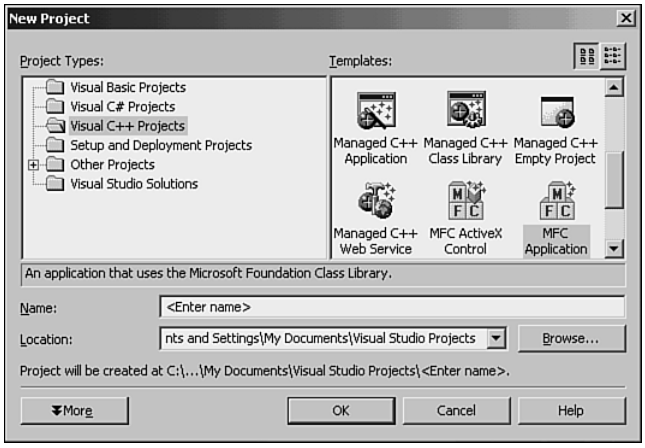

Рис. 2.1. Теперь тип создаваемого приложения выбирается с помощью областей Project Types и Templates диалогового окна New Project

Совет

Если вы хотите создать приложение .NET, переходите к чтению главы 17, “Первое знакомство с .NET”. Настоящая и следующие за ней главы описывают создание классического прил ожения Windows.

Выбрав в качестве шаблона MFC-приложение, задайте имя проекта и щелкните на кнопке OK. В результате этого на экране появится диалоговое окно MFC Application

Wizard, содержащее множество вкладок, предназначенных для настройки создаваемого вами приложения. Вы можете перемещаться по вкладкам, не боясь при этом потерять уже введенную вами информацию. Если же щелкнуть на кнопке Cancel (Отмена), процесс создания приложения вообще будет прерван, а все предыдущие настройки отменены. Оперативная справка по текущей вкладке вызывается на экран с помощью кнопки Help (Справка), а кнопка Finish (Готово) позволяет завершить сеанс настройки приложения. Все вкладки диалогового окна MFC Application Wizard рассматриваются в следующих разделах этой главы.

MFC-приложения используют библиотеку базовых классов Microsoft (Microsoft Foundation Classes — MFC). Описания и ссылки на отдельные компоненты этой библиотеки будут встречаться практически на каждой странице этой книги.

Выбор количества документов, которые будут поддерживаться приложением

Первое, что должен определить программист, приступая к работе с мастером MFC Application Wizard (AppWizard), — сколько окон будет поддерживать будущее приложение, т.е. будет ли оно MDI-приложением, SDI-приложением, простым диалоговым приложением или приложением, поддерживающим множество документов верхнего уровня. Настройки, позволяющие выбрать тип приложения, располагаются на вкладке Application Type (Тип приложения), как показано на рис. 2.2.

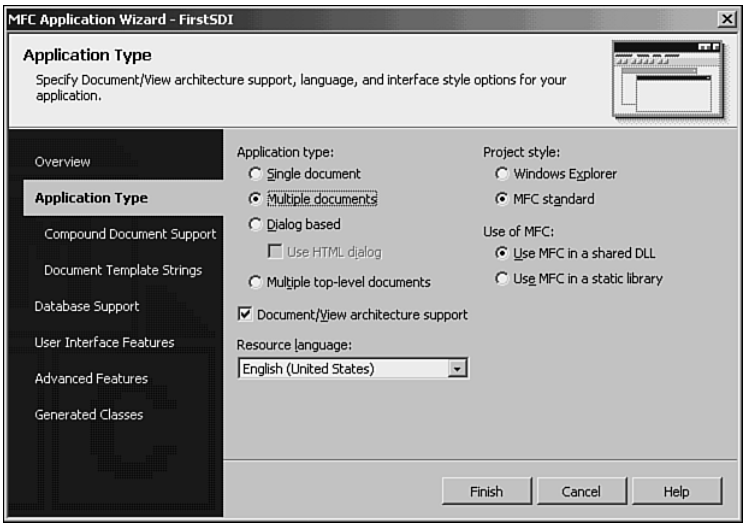

Рис. 2.2. Вторая вкладка мастера создания типового приложения MFC Application Wizard позволяет выбрать тип интерфейса приложения

Для каждого из этих типов приложений MFC Application Wizard по-разному создает программный код и классы.

Подробности о каждом из четырех типов приложений приведены ниже.

- SDI-приложение (Single Document Interface — однодокументный интерфейс) позволяет в каждый момент времени иметь открытым только один документ. Примером может служить известный каждому редактор Блокнот. Если вы выполните в таком

приложении команду File⇨Open, то открытый в текущий момент файл будет закрыт прежде, чем вы сможете открыть новый.

- MDI-приложение (Multiple Document Interface — многодокументный интерфейс) может одновременно поддерживать открытыми несколько документов, каждый из которых представлен отдельным файлом. За примерами ходить далеко не надо — это старые версии Excel, Word и другие хорошо знакомые многим аналогичные приложения. Такие приложения обязательно имеют в главном меню пункт Window (Окна), а в меню File — пункт Close. Одна из особенностей библиотеки MFC состоит в том, что если вам необходимо создать приложение с несколькими представлениями одного и того же документа, то его следует отнести к MDI-приложениям.
- Простое диалоговое приложение вообще не открывает документов. Примером может служить приложение Таблица символов и много других простых приложений, которые входят в базовый комплект Windows. Такие приложения не имеют меню. (Приложение Таблица символов, как правило, находится в папке Стандартные, до которой можно добраться, щелкнув на кнопке Пуск. Возможно, вам понадобится установить это приложение на свой компьютер. Для этого воспользуйтесь аплетом Установка и удаление программ, расположенным на Панели управления.)
- Тип приложений, поддерживающих множество документов верхнего уровня, является отличительной чертой Visual C++ .NET. Приложениями такого типа являются программы пакета Office 2000 — например, Word 2000 или Excel 2000. Каждый отдельный документ такого приложения имеет отдельное меню, строку заголовка и панель инструментов. На рис. 2.3 изображена программа Microsoft Word 2000 с несколькими открытыми документами.

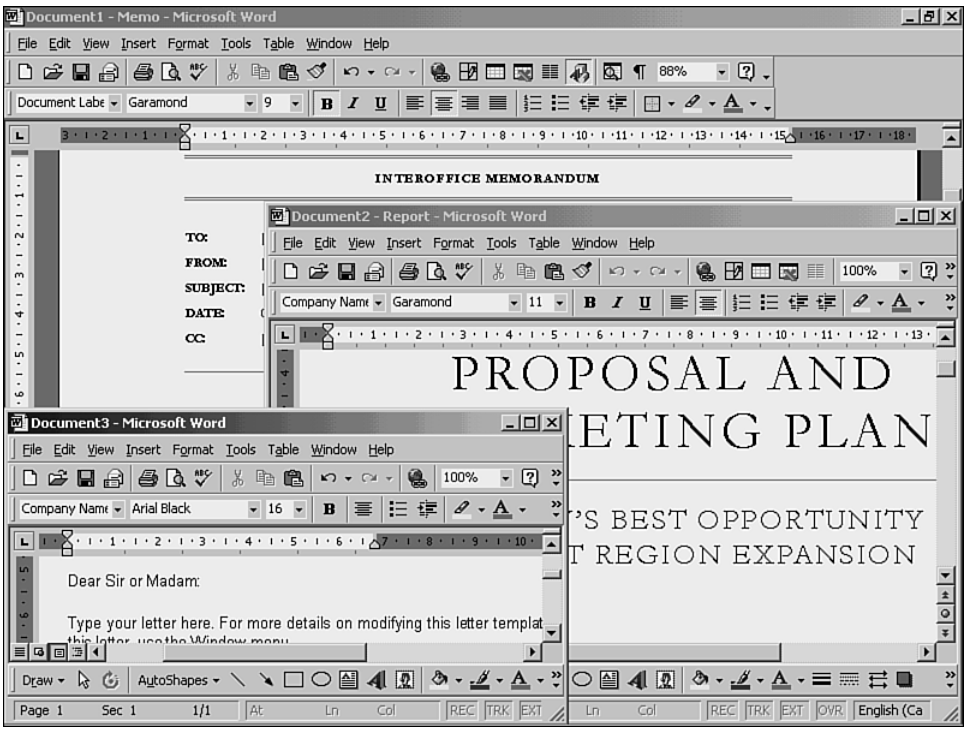

Рис. 2.3. Microsoft Word 2000 — это приложение, которое поддерживает множество документов верхнего уровня

После установки переключателя типа приложения в соответствующую позицию в правой верхней части диалогового окна MFC Application Wizard появится образец вашего будущего “детища”.

Простое диалоговое приложение достаточно сильно отличается от SDI-приложения, не говоря уже о MDI-приложении. При создании простого диалогового приложения некоторые элементы управления окна MFC Application Wizard будут недоступны. Более подробно создание простого диалогового приложения рассматривается в разделе “Создание простого диалогового приложения”, далее в этой главе.

Ниже этой группы переключателей на вкладке Application Type находится флажок Document/View architecture support (Поддержка архитектуры документ/представление). Особенности архитектуры документ/представление будут подробно разъяснены в главе 4, “Программирование вывода информации в приложении”, которая специально посвящена этому вопросу. Пользователи, имеющие большой опыт разработки приложений в среде Visual C++, могут отключить поддержку этой архитектуры, но для большинства разработчиков она будет отнюдь не лишней. Поэтому в дальнейшем, если не будет оговорено особо, считаем, что флажок Document/View architecture support установлен.

Еще ниже на вкладке Application Type находится раскрывающийся список для выбора языка, который вы собираетесь использовать при написании текста программы. Если язык вашей операционной системы отличается от заданного по умолчанию **English (United States)** — американский вариант английского, не забудьте сделать такой же выбор и в списке Resource language (Язык ресурсов). Иначе вы можете в дальнейшем столкнуться с совершенно неожиданными эффектами при работе с окнами Class View (Просмотр классов) и Properties Window (Окно свойств). (Конечно, если вы создаете при-

ложения для заказчика, который пользуется американским английским, у вас не остается иного выбора, как изменить язык операционной системы с помощью аплета Панели управления.)

Переключатель Project Style (Стиль проекта) предназначен для выбора типа интерфейса пользователя для создаваемого вами приложения. Установите переключатель в положение Windows Explorer (Проводник Windows) для выбора интерфейса в стиле обозревателя Web или в положение MFC standard (Стандартное приложение MFC) для выбора более традиционного интерфейса.

Ответ на следующий вопрос не так очевиден. Желаете ли вы, чтобы библиотека MFC была разделяемой динамически связываемой библиотекой (Use MFC in a shared DLL) или статически связываемой библиотекой (Use MFC in a static library)? Динамически связываемая библиотека (Dynamic Link Library — DLL) представляет собой множество функций, используемых самыми разными приложениями. Использование DLL сокращает объем программы, но несколько усложняет ее установку. Если вы просто перенесете на другой компьютер исполняемый файл программы, то, скорее всего, приложение работать не будет, поскольку оно нуждается еще и в соответствующих DLL-файлах. Если же модули библиотеки статически связаны с исполняемым файлом, то приложение легко перемещается с одного компьютера на другой, поскольку весь выполняемый код сосредоточен в одном файле.

Если вы ориентируетесь на заказчиков, которые сами являются разработчиками и имеют на своем компьютере, по крайней мере, одно приложение, использующее модули DLL из библиотеки MFC, или ваши заказчики не против установки DLL-файлов на своем компьютере, смело выбирайте опцию Use MFC in a shared DLL. Выполняемый файл меньших размеров — это всегда хорошо. Если же ваши потенциальные пользователи не столь искушены в компьютерных премудростях, лучше используйте опцию Use MFC in a static library. В результате получится исполняемый файл большего размера, зато вы будете избавлены от многих забот, связанных с сопровождением продукта. И последнее замечание. Если вы разработаете достаточно хорошую программу установки продукта на компьютер пользователя, то почувствуете, что многие сложности с применением динамически связываемых библиотек на самом деле не так уж и страшны.

Поддержка составных документов

Третий этап создания выполняемого приложения Windows с помощью MFC Application Wizard — это выбор уровня поддержки операций с составными документами. Окно MFC Application Wizard при этом будет выглядеть так, как показано на рис. 2.4.

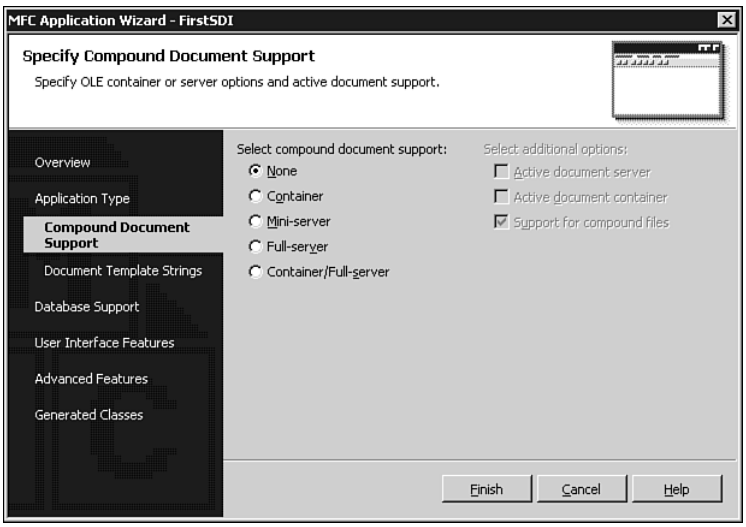

Рис. 2.4. Третья вкладка диалогового окна мастера создания типового приложения MFC Application Wizard позволяет выбрать вариант поддержки технологии составных документов

Технология OLE (Object Linking and Embedding — связывание и внедрение объектов) базируется на технологии COM (Component Object Model — модель компонентных объектов) и представляет собой некоторое подмножество технологии составных документов.

На выбор предлагается пять вариантов поддержки составных документов.

- Если не планируется создание составных документов (как в данной главе), установите переключатель в положение None (Поддержка составных документов отсутствует).
- Если вы хотите, чтобы в приложении использовались связанные или внедренные объекты, такие, как документы Word или рабочие листы Excel, установите переключатель в положение Container (Контейнер).
- Если планируется создание приложения, документы которого могли бы быть внедрены в другое приложение, но при этом само приложение не будет использоваться автономно, установите переключатель в положение Mini-server (Мини-сервер).
- Если ваше будущее приложение станет служить не только сервером для других приложений, но и сможет работать автономно, установите переключатель в положение Full-server (Полноценный сервер).
- И, наконец, если создаваемое приложение должно обладать способностью включать документы других приложений и само обслуживать их своими объектами, установите переключатель в положение Container/Full server (Контейнер/Полноценный сервер).

Если уж вы выбрали какой-либо из вариантов поддержки составных документов, то придется поддерживать и составные файлы. Составные файлы содержат один или более объектов и сохраняются на диске в особом формате, так что один из объектов может быть заменен без необходимости перезаписи всего файла. Таким образом удается сберечь довольно много времени. Для того, чтобы активизировать поддержку составных докумен-

тов, установите флажок Support for compound files (Поддержка составных файлов), расположенный в правой части вкладки Compound Document Support.

Вдобавок к составным файлам вы можете также активизировать поддержку активных документов. Активные документы являются расширением составных файлов. В частности, они позволяют каждому объекту управлять параметрами собственного просмотра и печати. Для того, чтобы указать на необходимость поддержки активных документов, установите флажок Active document container (Контейнер активных документов). Если вы хотите, чтобы документы вашего приложения могли использоваться как активные, установите флажок Active document server (Сервер активных документов). Если ваше приложение будет использоваться как сервер активных документов, вы должны указать значение в поле ввода File extension (Расширение файла), расположенном на вкладке Document Template Strings (Строковые шаблоны документа).

Строковые шаблоны документа

Вкладка Document Template Strings показана на рис. 2.5.

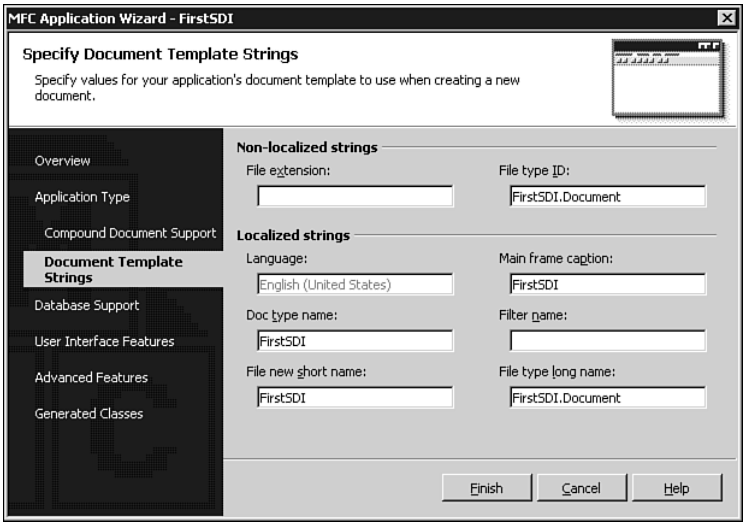

Рис. 2.5. Четвертая вкладка диалогового окна мастера создания типового приложения MFC Application Wizard позволяет выбрать строковые шаблоны

MFC Application Wizard формирует многочисленные идентификаторы, принимая в качестве главной переменной имя приложения, и иногда ему необходимы аббревиатуры этого имени. До тех пор, пока вы не усвоите, как MFC Application Wizard генерирует такие имена, вам придется справляться о них в этом окне. Здесь же их можно при желании откорректировать, а также уточнить надпись, которая будет выведена в строке заголовка главного окна создаваемого приложения. Расширение имени файла (если вы зададите его в поле File extension) будет автоматически добавляться к именам всех файлов, которые сохраняются данным приложением. Аналогично в результате выполнения команды File⇨Open в соответствующем диалоговом окне будут выводиться только файлы с заданным расширением.

Поддержка баз данных

Следующий этап создания приложения Windows с помощью MFC Application Wizard — выбор уровня поддержки операций с базами данных. Окно MFC Application Wizard при этом будет выглядеть так, как показано на рис. 2.6.

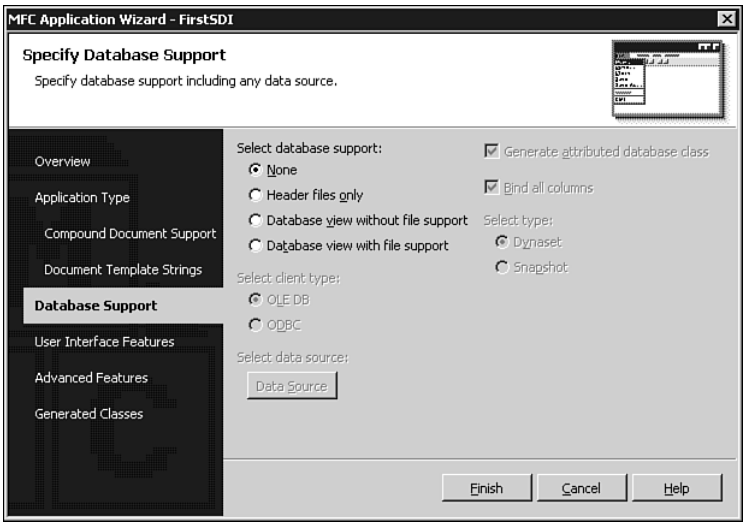

Рис. 2.6. Пятая вкладка диалогового окна мастера создания типового приложения MFC Application Wizard позволяет выбрать способ поддержки баз данных

При создании приложения, взаимодействующего с базой данных, мастер MFC Application Wizard может сгенерировать для вас готовый к использованию класс представления (на основе класса `CRecordView`). Этот класс можно будет применять для извлечения информации из базы данных, при этом практически не создавая программного кода. MFC Application Wizard предлагает на выбор четыре варианта уровня поддержки баз данных.

- Если работа с базами данных в приложении не планируется, установите переключатель в положение None (Поддержка баз данных не предусмотрена).
- Если предполагается иметь доступ к базам данных, но для этого не будут использованы классы представления или меню, сгенерированные мастером MFC Application Wizard, установите переключатель в положение Header files only (Только файлы заголовка).
- Если вы планируете разрабатывать в приложении классы представления базы данных как производные от `CRecordView` и иметь меню Record (Запись), однако не нуждаетесь в средствах сохранения информации на локальном диске, установите переключатель в положение Database view without file support (Генерирование представления базы данных без поддержки файловых операций).
- Если помимо всего, что перечислено в предыдущем пункте, вы планируете иметь возможность сохранения информации на локальном диске (предположим, что это будет одна из опций, доступных пользователям), установите переключатель в положение Database view with file support (Генерирование представления базы данных и поддержка файловых операций).

Если вы выбрали один из вариантов с использованием представления базы данных, то в этом же окне вам следует задать и источник информации. Для этого нужно щелкнуть на кнопке Data Source (Источник данных).

Картинка в правой верхней части диалогового окна MFC Application Wizard меняется после задания любого из предложенных вариантов обращения к базе данных, демонстрируя тем самым последствия сделанного выбора. Изменение типа поддержки работы с базами данных приводит также к активации/деактивации различных элементов управления, расположенных на вкладке Database Support. Более подробно взаимодействие приложения с базой данных рассматривается в главе 11, “Программирование взаимодействия приложения с базой данных”, далее в этой книге.

Определение свойств пользовательского интерфейса

Следующий этап создания приложения Windows с помощью MFC Application Wizard заключается в определении внешнего вида элементов пользовательского интерфейса. Окно MFC Application Wizard при этом будет выглядеть так, как показано на рис. 2.7.

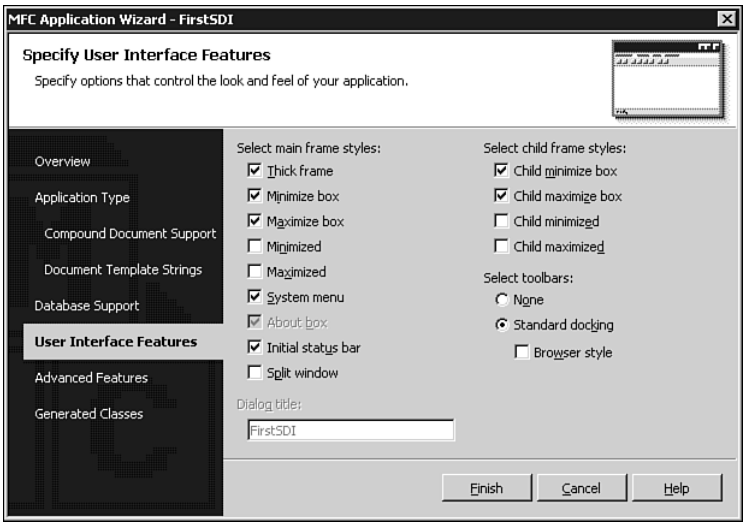

Рис. 2.7. Шестая вкладка диалогового окна мастера создания типового приложения MFC Application Wizard позволяет определить свойства пользовательского интерфейса

Вкладка User Interface Features (Свойства пользовательского интерфейса) содержит много переключателей и флажков, соответствующих предлагаемым параметрам оформления.

Первая группа флажков — Main frame styles (Стили главного фрейма) — позволяет определить различные параметры фреймов. По сути, фрейм является основой окна; системное меню, строка заголовка, кнопки минимизации и максимизации, собственно границы — все это свойства фрейма как объекта. Главный фрейм соответствует всему приложению. MDI-приложение имеет несколько дочерних фреймов (по одному на каждый документ), которые размещаются в пределах главного, родительского, фрейма. Ниже пе-

речислены свойства фрейма, которые можно настраивать с помощью вкладки User Interface Features.

- **Thick frame (Утолщенная граница фрейма).** Граница окна утолщена, при этом его размеры можно изменять стандартным для Windows способом. Снимите этот флажок для того, чтобы запретить изменение размера окна.
- **Minimize box (Кнопка минимизации).** Окно имеет кнопку минимизации в правой части строки заголовка.
- **Maximize box (Кнопка максимизации).** Окно имеет кнопку максимизации в правой части строки заголовка.
- **Minimized (Минимизировать).** При запуске приложения окно сворачивается в пиктограмму.
- **Maximized (Максимизировать).** При запуске приложения окно разворачивается на весь экран.
- **System menu (Системное меню).** В левом верхнем углу окна будет размещена пиктограмма вызова системного меню.
- **About box (О программе).** Мастер MFC Application Wizard создает диалоговое окно, которое содержит настраиваемую информацию о программе, а также пункт в меню Help (Справка), вызов которого приведет к открытию данного диалогового окна. Этот флажок можно снять только в том случае, если вы создаете простое диалоговое приложение.
- **Initial status bar (Панель состояния).** Мастер MFC Application Wizard создает панель состояния, предназначенную для вывода подсказок, соответствующих выбранным пунктам меню, и других сообщений. Позднее можно будет написать специальный программный код для вывода различных индикаторов и других элементов на панель состояния, о чем будет подробно рассказано в главе 9, “Создание компонентов COM+ с помощью библиотеки ATL”.
- **Split window (Разделение окна).** Главное окно приложения будет иметь строку разделения.

Если вы создаете MDI-приложение, то с помощью группы флажков Child frame styles (Стили дочерних фреймов) можете определить свойства дочерних фреймов. Ниже перечислены флажки группы Child frame styles.

- **Child minimize box (Кнопка минимизации дочернего окна).** Каждое дочернее окно имеет кнопку минимизации в правой части строки заголовка.
- **Child maximize box (Кнопка максимизации дочернего окна).** Каждое дочернее окно имеет кнопку максимизации в правой части строки заголовка.
- **Child minimized (Минимизировать дочернее окно).** При создании дочернее окно сворачивается в пиктограмму.
- **Child maximized (Максимизировать дочернее окно).** При создании дочернее окно разворачивается на весь экран.

И, наконец, вы можете указать на необходимость размещения в окне приложения панелей инструментов. Если панели инструментов вам не нужны, установите переключатель Toolbars (Панели инструментов) в положение None (Панели инструментов отсутствуют). Иначе установите соответствующий переключатель в положение Standard docking (Фиксируемая панель инструментов) — в этом случае панель инструментов будет зафиксирована вдоль верхней границы окна. Для того чтобы панель инструментов походила на панель инструментов новых версий обозревателя Internet Explorer, установите флажок

Browser style (Стиль обозревателя). Более подробно панели инструментов рассматриваются в главе 7, “Панели инструментов, строки состояния и элементы управления общего назначения”, далее в этой книге.

Дополнительные параметры

Следующий этап создания приложения Windows с помощью мастера MFC Application Wizard заключается в определении всевозможных дополнительных параметров. Окно MFC Application Wizard при этом будет выглядеть так, как показано на рис. 2.8.

Мастер MFC Application Wizard генерирует весь программный код, необходимый для реализации тех или иных дополнительных свойств приложения. Ниже перечислены параметры, которые можно настраивать с помощью вкладки Advanced Features.

- **Context-sensitive Help (Контекстная справка).** Меню Help в приложении будет иметь пункт Help Topics (Разделы справки), а значительная часть программного кода, необходимого для организации контекстной справки в приложении, будет включена в него мастером MFC Application Wizard. В результате выполнения команды Help Topics на экране появится окно с вкладками Contents (Содержание), Index (Предметный указатель) и Find (Поиск). Этот параметр сложно будет изменить позднее, поскольку реализация справки требует генерации большого объема программного кода. Контекстная справка может иметь два формата: классический формат справки Windows (WinHelp Format) и справка в формате HTML (HTML Help format). С методикой организации контекстной справки в приложении вы сможете познакомиться более подробно в главе 8, “Справочная система, страницы свойств и мастера”.

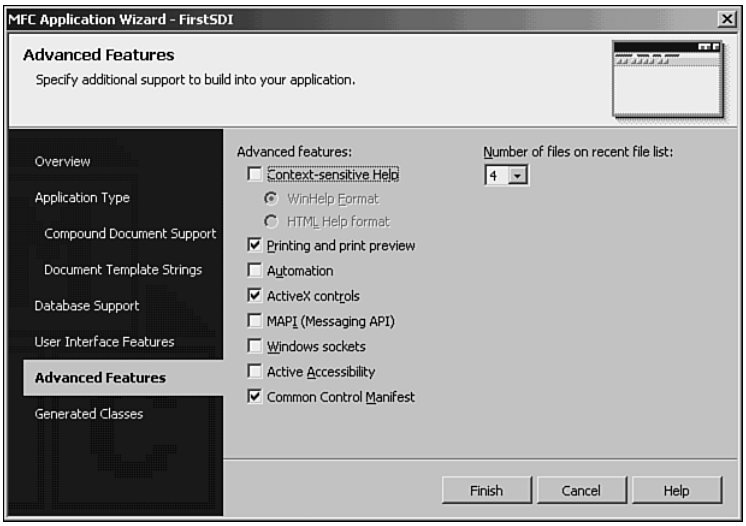

Рис. 2.8. Седьмая вкладка диалогового окна мастера MFC Application Wizard позволяет определить дополнительные свойства приложения

- **Printing and print preview (Печать и предварительный просмотр).** Установка этого флажка приведет к тому, что приложение будет иметь пункты Print и Print preview в меню File. При этом большая часть программного кода, связанного с выполнением этих операций, будет сгенерирована мастером MFC Application Wizard.

Более подробно функции печати рассматриваются в главе 5, “Печать и сохранение данных”.

- **Automation (Автоматизация).** Если вы хотите, чтобы создаваемое приложение могло передавать управление другому приложению через механизм автоматизации, установите данный флажок.
- **ActiveX controls (Элементы управления ActiveX).** Если вы планируете использовать в приложении элементы управления ActiveX, установите этот флажок.
- **MAPI (Messaging API) (MAPI — почтовый интерфейс).** При установке этого флажка приложение сможет отсылать сообщения электронной почты, факсы и др. Обсуждение подробностей реализации MAPI вы найдете в главе 10, “Internet-программирование”.
- **Windows Sockets (Сокеты Windows).** Если этот флажок будет установлен, приложение сможет иметь непосредственный доступ к Internet через такие протоколы, как FTP и HTTP (протокол World Wide Web). Концепция Windows-сокетов обсуждается в главе 10, “Internet-программирование”. Следует отметить, что вы можете создать Internet-программу и без поддержки сокетов Windows, используя для этого классы WinInet (более подробно классы WinInet рассматриваются в главе 10, “Internet-программирование”).

Следует отметить, что вы также можете установить длину списка недавно использовавшихся файлов, который является частью меню File создаваемого приложения. По умолчанию этот параметр имеет значение **4** и менять его не рекомендуется без очень весомых причин.

Имена файлов и классов

И, наконец, последний этап создания приложения Windows с помощью мастера MFC Application Wizard — подтверждение имен классов и имен файлов, которые будут автоматически созданы для вас мастером MFC Application Wizard, как показано на рис. 2.9.

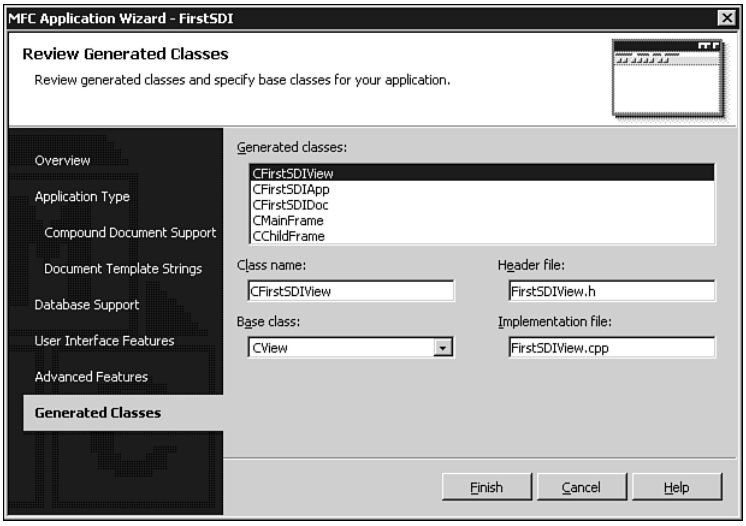

Рис. 2.9. Восьмая, последняя вкладка диалогового окна мастера MFC Application Wizard позволяет подтвердить имена классов и имена файлов, генерируемых мастером MFC Application Wizard

MFC Application Wizard использует для формирования имен классов и имен файлов имя проекта (в данном случае — *FirstSDI*). Нет никакой нужды их изменять. Если в приложении используются классы представления, можно изменить имя класса, наследниками которого являются вновь создаваемые классы. По умолчанию базовым является класс `CView`, но многие разработчики предпочитают использовать класс `CScrollView` или класс `CEditView`. Более подробно классы представления рассматриваются в главе 4, “Программирование вывода информации в приложении”, далее в этой книге.

Создание приложения

Вкладка Overview (Резюме) диалогового окна мастера MFC Application Wizard содержит краткую информацию о типе создаваемого приложения (рис. 2.10).

Если что-либо здесь вас не устраивает, возвратитесь к нужной вкладке и внесите все требуемые изменения. После уточнения настройки снова откройте вкладку Overview, просмотрите то, что у вас получилось, и уж после этого смело щелкайте на кнопке Finish (Готово). Мастер MFC Application Wizard приступит к созданию приложения. Это займет несколько минут. За время, пока вы будете пить кофе, MFC Application Wizard создаст сотни строк программного кода, меню, диалоговых окон, текстов справок, растровых картинок, которые будут записаны в более чем 20 различных файлах. Пусть себе работает...

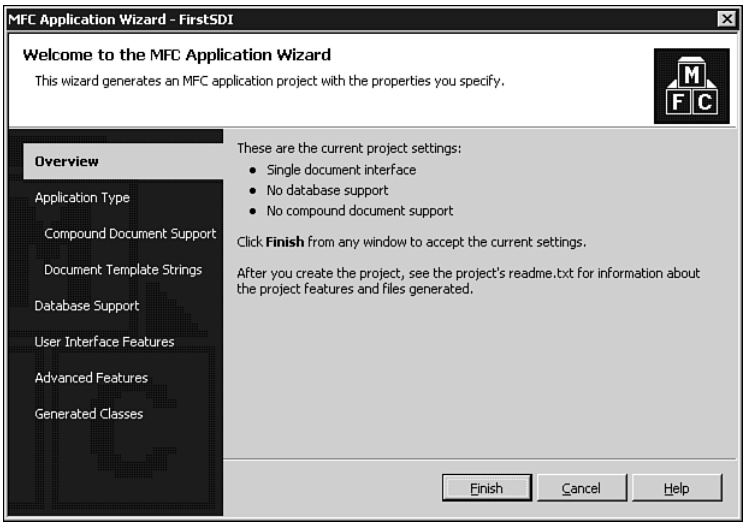

Рис. 2.10. Вкладка Overview диалогового окна мастера MFC Application Wizard содержит краткую информацию о типе создаваемого приложения

Время поработать самостоятельно

Если вы еще не запустили Visual Studio, самое время это сделать. Если прежде вы никогда не пользовались этим продуктом, то его интерфейс может показаться, на первый

взгляд, слегка пугающим. Полное описание всех областей, панелей, меню и комбинаций клавиш приведено в приложении B, “Visual Studio: среда разработки, интерфейс пользователя и система меню”, далее в книге.

Первым делом вызовите мастер MFC Application Wizard. Для этого нужно выполнить команду File⇨New⇨Project. В диалоговом окне New Project укажите **Visual C++ Projects** в качестве типа проекта и **MFC Application** в качестве шаблона приложения. Здесь же введите имя папки, в которой вы собираетесь хранить все файлы создаваемого приложения, — MFC Application Wizard открывает новую папку для каждого нового приложения. В качестве имени проекта введите **FirstSDI**, а затем с помощью мастера MFC Application Wizard пройдите все вкладки с настройками параметров нового приложения. В частности, выберите тип приложения SDI с помощью вкладки Application Type, а на всех последующих вкладках не меняйте ничего. Когда MFC Application Wizard закончит работу над созданием проекта, выполните команду Build⇨Build FirstSDI (Сборка⇨Выполнить сборку FirstSDI) из меню Visual Studio. В результате выполнения этой команды приложение будет откомпилировано и скомпоновано.

Когда этот процесс завершится, выполните команду меню Visual Studio Debug⇨Start (Отладка⇨Запустить). Перед вами появится реальное, работоспособное приложение Windows, показанное на рис. 2.11. “Поиграйте” с ним — попробуйте изменить размеры и положение окна, свернуть его в пиктограмму или развернуть на весь экран.

Поработайте с меню File — выполните команду File⇨Open, и перед вами распахнется до боли знакомое диалоговое окно Windows File Open. Можете смело выбирать любой файл, ваше приложение с ним ничего не сделает. Выполните команду File⇨Exit и закройте приложение. Запустите его вновь и продолжите изучение автоматически сгенерированных возможностей. Установите указатель мыши на одной из кнопок панели инструментов и задержите его там на некоторое время. Появится маленькое окно контекстной справки, содержащее текст, который напомнит вам о назначении этой пиктограммы. Для проверки, действительно ли кнопки панели инструментов дублируют те или иные команды меню, щелкните на кнопке Open и убедитесь, что все дальнейшее в точности повторяет возникшую чуть раньше ситуацию, когда вы выполнили команду меню File⇨Open. Теперь откройте меню View (Вид) и щелкните на отмеченном пункте Toolbar (Панель инструментов). Панель исчезнет с экрана. Выполните команду View⇨Toolbar — и панель инструментов вновь появится на экране. То же самое проделайте и с панелью состояния. Выполните команду Help⇨About; возможно, вы этого не ожидали, но созданное практически без вашего участия приложение имеет даже окно сообщения с информацией об авторских правах (рис. 2.12).

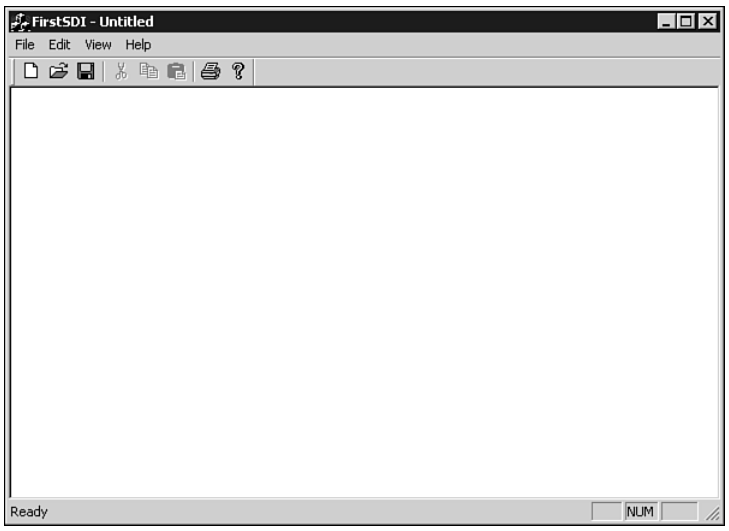

Рис. 2.11. Внешне ваше первое приложение ничем не отличается от любого "целиком оперившегося" приложения Windows

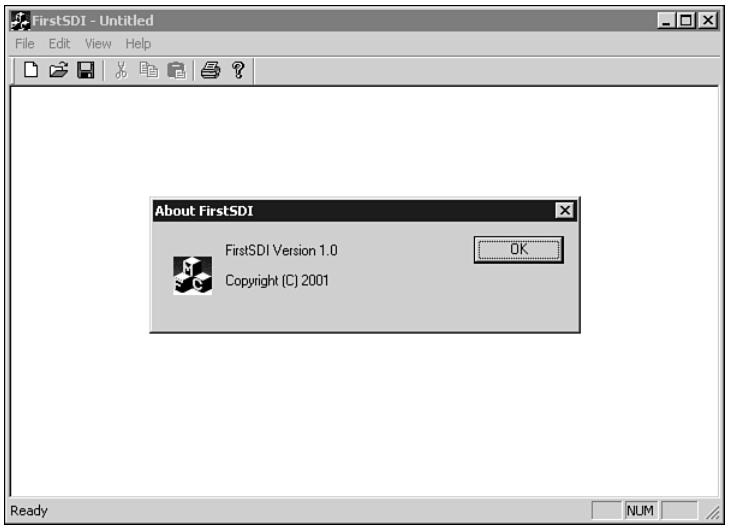

Рис. 2.12. В этом приложении есть даже окно сообщения с информацией об авторских правах

Теперь проведем аналогичный эксперимент с приложением типа MDI, которое мы назовем *FirstMDI*. Отличия в процессе создания приложения касаются только диалогового окна New Project, в котором задается имя проекта, и вкладки Application Type, на которой вам придется установить переключатель в положение Multiple documents. Что касается остальных вкладок, то здесь можно вполне согласиться с настройками, предлагаемыми MFC Application Wizard по умолчанию. После завершения работы MFC Application Wizard повторите уже знакомые операции с Visual Studio и запустите приложение. Вы увидите на экране нечто, весьма близкое к изображенному на рис. 2.13, — MDI-приложение с одним