ПРОГРАММИРОВАНИЕ на C++ ГЛАЗАМИ ХАКЕРА

2-е издание

+CD

Михаил Фленов

Как сделать код маленьким, а программу невидимой
От простых шуточных программ до сложной манипуляции системой
Примеры работы через компоненты C++ и через Windows Sockets
Как работать с портами компьютера и получать информацию о системе
Интересные алгоритмы написания сетевых программ

bhv

Михаил Фленов

ПРОГРАММИРОВАНИЕ на C++ ГЛАЗАМИ ХАКЕРА

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.068+800.92C++
ББК 32.973.26-018.1
Ф69

Фленов М. Е.

Ф69 Программирование на C++ глазами хакера. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 352 с.: ил. + CD-ROM

ISBN 978-5-9775-0303-7

Рассмотрены нестандартные приемы программирования, а также примеры использования недокументированных функций и возможностей языка C++ при разработке шуточных программ и серьезных сетевых приложений для диагностики сетей, управления различными сетевыми устройствами и просто при повседневном использовании интернет-приложений. Во втором издании содержатся новые и переработаны старые примеры, а в качестве среды разработки используется Visual Studio 2008, хотя большинство описываемых примеров работоспособны в более старых версиях и в CodeGear C++ Builder.

Компакт-диск содержит исходные примеры из книги, а также популярные приложения компании CyD Software Labs.

Для программистов

УДК 681.3.068+800.92C++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Кашлакова*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.11.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 28,38.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.003650.04.08 от 14.04.2008 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0303-7

Оглавление

Введение

По своей натуре я заядлый программист и не буду блистать литературными способностями. Зато я постараюсь поделиться своими знаниями и надеюсь рассказать вам что-то новое, раскрыть некоторые из секретов хакеров. Отдельные вещи, которые будут описаны, возможно, вам уже известны, но не все понимают смысл используемого кода. Я же постараюсь описать все как можно подробнее и показать кое-какие известные мне интересные приемы программирования.

Я попытался привести максимальное количество примеров на языке программирования C++. Среди них множество примеров шуточных программ и сетевых приложений. Почему именно на эти две темы я сделаю упор? Если верить моим глазам (а иногда они меня не подводят), именно эти две темы интересует множество читателей, особенно начинающих.

Если вы плохо знакомы с программированием, то лучше начинать изучение с тех примеров и тем, которые представляют наибольший интерес. В этом случае вам будет интереснее постигать новые высоты и получать знания, и обучение будет не в тягость. Чем изучать программирование на скучных задачках, лучше пойти погулять с друзьями.

Чистой теории в книге будет мало, но зато практических занятий — хоть отбавляй. Вы все сможете увидеть своими глазами и сразу же попробовать. Я считаю, что практика является наилучшим учителем. Практические занятия отлагаются в памяти лучше всего. Теория может оказаться бесполезным знанием, если не знать, как применить эту теорию на практике. Поэтому мы будем учиться и узнавать, как решать проблемы.

Если вы ожидали увидеть в данной книге примеры и описания вирусов, то вы сильно ошиблись. Ничего разрушительного мы рассматривать и делать не будем. Я занимаюсь только созиданием. Ни одна из шуточных программ

не будет приводить к печальным последствиям. Шутка хороша, когда она не причиняет боль, а может вызвать у окружающих хотя бы улыбку.

О книге

Для эффективной работы с книгой вам понадобятся минимальные знания C++ и начальные навыки общения с компьютером. Вы должны уметь создать простое приложение, знать, что такое переменные, циклы и как с ними работать. Не помешают знания адресации, указателей и их необходимости. Это позволит вам лучше понимать описываемые примеры. Что касается сетевого программирования, то его я опишу достаточно подробно, начиная с основ и заканчивая сложными примерами. Так что тут начальные знания желательны, но не обязательны.

Я постарался описать все как можно проще, в тексте программ приведено максимум комментариев, чтобы вы наслаждались чтением, и оно не было для вас утомительным.

Эта книга построена не так, как многие другие. В ней нет нудных теоретических рассуждений, а только максимум примеров и исходных кодов. Ее можно воспринимать как практическое руководство к действию. Только на практике вы сможете понять и усвоить все теоретически полученные знания.

Программисты в чем-то похожи на врачей: если врач теоретически знает симптомы болезни, но на практике не может точно отличить отравление от аппендицита, то такого врача лучше не подпускать к больному. Точно так же и программист: если он знает, как функционирует сетевой протокол, но не может с ним работать, то его сетевые программы никогда не будут работать правильно и эффективно.

Ничто не может привести к такому пониманию предмета, как хорошая практика. Когда вы можете "пощупать" все своими руками, то теория запоминается гораздо лучше.

Вот пример из жизни. Я проходил обучение в известном университете на курсах Microsoft по администрированию и программированию сервера баз данных SQL Server. Курсы очень хорошие, и преподаватель старался все очень подробно и доходчиво изложить. Но сам курс был поставлен корпорацией как теоретический, с небольшим добавлением лабораторных работ. В результате нам очень хорошо объяснили, ЧТО может делать сервер. Но когда после курсов я столкнулся с реальной ситуацией, то понял, что не знаю, КАК сделать что-либо. Приходилось снова открывать книгу, которую дали мне в центре обучения (она была на английском языке), и, читая обширный теоретический материал и маленькие лабораторные, разбираться с реальной

задачей. Уж лучше бы я узнал на курсах, как решить проблему, а не что можно теоретически выполнить, потому что такое обучение, по-моему, только пустая трата времени и денег.

Несмотря на это я не противник теории и не пытаюсь сказать, что теория не нужна. Просто нужно описывать, как решить задачу, и рассказывать, зачем мы делаем какие-то определенные действия. После этого, когда вы будете сталкиваться с подобными проблемами, вы уже будете знать, как сделать что-то, чтобы добиться определенного результата.

Именно практических руководств и просто хороших книг с разносторонними примерами не хватает на полках наших магазинов. Я надеюсь, что моя работа хотя бы частично восполнит пробел в этой сфере и поможет вам в последующей работе и решении различных задач программирования.

Итак, книга, несмотря на свое название, сможет пригодиться всем, потому что в ней будут описываться разные программистские приемы, которые используются в каждодневной практике.

Все примеры, которые будут приведены в книге, можно найти на компакт-диске. Тем не менее, я советую все описанное создавать самостоятельно и обращаться к файлам, только если возникли какие-то проблемы, чтобы сравнить и найти ошибку. Когда вы создаете что-то сами, то получаете хорошие практические навыки, а полученная информация откладывается в памяти намного лучше, чем сто прочтенных страниц теории. Это вторая причина, по которой книга построена на основе большого количества примеров и исходных кодов.

Даже если вы решили воспользоваться готовым примером, обязательно досконально разберитесь с ним. Попробуйте изменить какие-то параметры и посмотреть на результат. Попытайтесь модифицировать пример, добавив какие-то дополнительные возможности. Только так вы сможете понять принцип работы используемых функций или алгоритмов.

В качестве среды разработки и для компиляции примеров я использовал Visual Studio 2008, но многие примеры смогут работать даже в Visual Studio 6, не говоря уже о более новых версиях. Единственная проблема, с которой вы можете столкнуться, — файл проекта. Если команды и функции языка C++ изменяются редко (а я постарался не использовать ничего редкого), то формат файла проекта изменился очень сильно и меняется почти в каждой новой версии.

Кто такой хакер? Как им стать?

Прежде чем приступить к практике, я хочу "загрузить" вашу голову небольшим количеством теоретической информации. А именно, прежде чем читать

книгу, вы обязаны знать, кто такой хакер в моем понимании. Если вы будете подразумевать одно, а я другое, то мы не сможем понять друг друга.

В мире полно вопросов, на которые большинство людей не знают правильного ответа. Мало того, люди используют множество слов, даже не догадываясь об их настоящем значении. Например, многие сильно заблуждаются в понимании термина "хакер". Однако каждый вправе считать свое мнение наиболее правильным. И я не буду утверждать, что именно мое мнение единственно верное, но оно отталкивается от первоначального понятия.

Прежде чем начать обсуждать всем известный термин, я должен обратиться к истории и вспомнить, как все начиналось. А начиналось все еще тогда, когда не было даже международной сети Интернет.

Понятие "хакер" зародилось, когда только стала распространяться первая сеть, которая называлась ARPANET. Тогда это понятие обозначало человека, хорошо разбирающегося в компьютерах. Некоторые даже подразумевали под хакером человека, "помешанного" на компьютерах. Понятие ассоциировали со свободным компьютерщиком, человеком, стремящимся к свободе во всем, что касалось его любимой "игрушки". Именно благодаря этому стремлению и тяге к свободному обмену информацией и началось такое бурное развитие Всемирной сети. Именно хакеры помогли развитию Интернета и создали FIDO. Благодаря им появились UNIX-подобные системы с открытым исходным кодом, в которых сейчас работает большое количество серверов.

В те времена еще не было вирусов, и не внедрилась практика взломов сетей или отдельных компьютеров. Образ хакера-взломщика появился немного позже. Но это только образ. Настоящие хакеры никогда не имели никакого отношения к взломам, а если хакер направлял свои действия на разрушение, то это резко осуждалось виртуальным сообществом.

Настоящий хакер — это творец, а не разрушитель. Так как творцов оказалось больше, чем разрушителей, то истинные хакеры выделили тех, кто занимается взломом, в отдельную группу и назвали их крэкерами (взломщиками) или просто вандалами. И хакеры, и взломщики являются героями виртуального мира. И те, и другие борются за свободу доступа к информации. Но только крэкеры, как правило, взламывают сайты, закрытые базы данных и другие источники информации ради собственной наживы, ради денег или минутной славы; такого человека можно назвать только преступником (кем он по закону и является!).

Хакер — это просто гений в некоторой области. В данной книге мы рассматриваем гения в компьютерной сфере, который хорошо знает свой предмет, создает что-то новое и уникальное и при этом помогает другим. Все, кто приписывают этим людям вандализм, сильно ошибаются. Истинные хакеры никогда не используют свои знания во вред другим. Именно к этому я призы-

ваю в данной книге. Только полезная и познавательная информация, которую вы сможете использовать для повышения своего уровня знаний и улучшения качества программирования.

Теперь давайте разберемся, как стать настоящим хакером. Это обсуждение поможет вам больше узнать об этих людях.

- Вы должны знать свой компьютер и научиться эффективно им управлять. А если вы будете еще знать в нем каждую железку, то это только добавит к вашей оценке по "хакерству" большой и жирный плюс.

  Если честно, то еще лет пять назад я знал все железо в компьютерах, разницу в процессорах вплоть до каждого контакта, а сейчас даже не знаю, чем Core2 Duo отличается от своего предшественника. Я просто покупаю ноутбук и даже не пытаюсь думать о том, как работает там процессор, главное, чтобы он работал хорошо и быстро.

  Что я подразумеваю под умением эффективно управлять своим компьютером? Это значит знать все возможные способы выполнения каждого действия и в каждой ситуации уметь использовать наиболее оптимальный из них. В частности, вы должны научиться пользоваться горячими клавишами и не дергать мышь по любому пустяку. Нажатие клавиш выполняется быстрее, чем любое, даже маленькое, перемещение мыши. Просто приучите себя к этому, и вы увидите все прелести работы с клавиатурой. Лично я использую мышь очень редко и стараюсь всегда применять клавиатуру, потому что это намного быстрее.

  Маленький пример на эту тему. Однажды я работал на фирме, где мой начальник всегда копировал и вставлял данные из буфера обмена с помощью кнопок на панели инструментов или команд контекстного меню, которое появляется при щелчке правой кнопкой мыши. Но если вы делаете так же, то, наверно, знаете, что не везде есть кнопки **Копировать, Вставить** или такие же пункты в контекстном меню. В таких случаях мой начальник набирает текст вручную. А ведь можно было бы воспользоваться копированием/вставкой с помощью комбинаций горячих клавиш <Ctrl>+<C>/<Ctrl>+ +<V> или <Ctrl>+<Ins>/<Shift>+<Ins>, которые реализованы практически во всех современных приложениях, на уровне самой ОС и ее компонентов.

  За копирование и вставку в стандартных компонентах Windows (строки ввода, текстовые поля) отвечает сама операционная система, и тут не нужен дополнительный код, чтобы данные действия заработали. Если программист не предусмотрел кнопку, то это не значит, что данного действия нет. Оно есть, но доступно через горячую клавишу.

- Вы должны досконально изучать все, что вам интересно о компьютерах. Если вас интересует графика, то вы должны изучить лучшие графические

пакеты, научиться рисовать в них любые сцены и создавать самые сложные миры. Если вас интересуют сети, то старайтесь узнать о них все. Если вы считаете, что уже знаете все, то купите книгу потолще по данной теме, и вы поймете, что сильно ошибаетесь. Компьютеры — это такая сфера, в которой невозможно знать все!!!

Хакеры — это прежде всего профессионалы в каком-нибудь деле. И это не обязательно должен быть компьютер или какой-то определенный язык программирования. Хакером можно стать в любой области, но я в данной книге буду рассматривать только компьютерных хакеров, или программистов.

- Желательно уметь программировать. Любой хакер должен знать как минимум один язык программирования. А лучше знать даже несколько языков. Лично я для начала рекомендую всем изучить Delphi или C++. Delphi достаточно прост, быстр, эффективен, а главное, это очень мощный язык. C++ — признанный стандарт во всем мире, но немного сложнее в изучении. Но это не значит, что не надо знать другие языки. Вы можете научиться программировать на чем угодно, даже на языке Basic (использовать его не советую, но знать не помешало бы).

  В связи с последними изменениями на рынке средств разработки и непонятной ситуацией с компанией разработчиком Delphi, я все больше начинаю рекомендовать использовать C++.

  Хотя я не очень люблю Visual Basic за его ограниченность, я видел несколько великолепных программ, которые были написаны именно на этом языке. Глядя на них, сразу хочется назвать их автора хакером, потому что это действительно виртуозная и безупречная работа. Создание из ничего чего-то великолепного — как раз и есть искусство хакерства. А Visual Basic .NET уже ничем не отличается по возможностям и мощности от C#.

  Хакер — это человек, который что-то создает. В большинстве случаев это относится к коду, но можно создавать и графику, и музыку. Все это тоже относится к искусству хакера. Но даже если вы занимаетесь компьютерной музыкой, знания программирования повысят ваш уровень. Сейчас создавать свои программы стало уже не так сложно, как раньше. С помощью таких языков, как C#, и благодаря платформе .NET можно создавать простые утилиты за очень короткое время, и при этом вы не будете ни в чем ограничены. Так что не поленитесь и изучите программирование.

  На протяжении всей книги я буду рассказывать о том, что необходимо знать программисту-хакеру, и покажу множество интересных приемов и примеров на языке C++. Если вы еще плохо знакомы с этим языком, то книга поможет познакомиться с программированием с помощью интересных и познавательных примеров.

- Не тормозить прогресс. Хакеры всегда боролись за свободу информации. Если вы хотите быть хакером, то тоже должны помогать другим. Хакеры обязаны способствовать прогрессу. Некоторые делают это через написание программ с открытым кодом, а кто-то просто делится своими знаниями.

  Открытость информации не означает, что вы не можете зарабатывать деньги. Это никогда не возбранялось, потому что хакеры тоже люди. Самое главное — это созидание. Вот тут проявляется еще одно отличие хакеров от крэкеров: хакеры создают, а крэкеры уничтожают. Если вы написали какую-нибудь уникальную шуточную программу, то это вас делает хакером. Но если вы написали вирус, который уничтожает диск, то это вас делает крэкером, я бы даже сказал — преступником.

  В борьбе за свободу информации может применяться даже взлом, но только не в разрушительных целях. Вы можете взломать какую-нибудь программу, чтобы посмотреть, как она работает, но не убирать ее систем защиты. Нужно уважать других программистов, не нарушать их авторские права, потому что защита программ — это их хлеб.

  Представьте себе ситуацию, если бы вы украли телевизор. Это было бы воровство и преследовалось бы по закону. Многие люди это понимают и не идут на преступления из-за боязни наказания. Почему же тогда крэкеры спокойно ломают программы, не боясь закона? Ведь это тоже воровство. Лично я приравниваю взлом программы к воровству телевизора с полки магазина и считаю это таким же правонарушением.

  Я также отказался от нелегального софта и уже много раз делился размышлениями на эту тему в своем блоге **http://www.flenov.info**.

  Я сам программист и продаю свои программы. Но я никогда не делаю сложных систем защиты, потому что это мешает законопослушным пользователям, а крэкеры все равно взломают. Какие только системы защиты ни придумывали крупные корпорации, чтобы защитить свою собственность, но большинство взламывалось еще до официального выхода программного продукта на рынок. В цивилизованном мире программа должна иметь только простое поле для ввода какого-то кода, подтверждающего оплату, и ничего больше. Не должно быть никаких активаций и сложных регистраций. Но и пользователи должны быть честными, потому что любой труд должен оплачиваться. А то, что какой-то товар (программный продукт) можно получить бесплатно, не значит, что вы должны это делать.

- Не изобретать велосипед. Тут опять действует созидательная функция хакеров. Они не должны стоять на месте и обязаны делиться своими знаниями. Например, если вы написали какой-то уникальный код, то поделитесь им с другими, чтобы людям не пришлось создавать то же самое. Вы можете не выдавать все секреты, но должны помогать другим.

Ну а если к вам попал код другого человека, то не стесняйтесь его использовать (с его согласия!). Не выдумывайте то, что уже сделано другими и обкатано пользователями. Если каждый будет создавать колесо, то никто и никогда не создаст повозку.

- Хакеры — не просто отдельные личности, а целая культура. Но это не значит, что все хакеры одеваются одинаково и все на одно лицо. Каждый из них — это отдельный индивидуум и не похож на других. Не надо копировать другого человека. Удачное копирование не сделает вас продвинутым хакером. Только ваша индивидуальность может сделать вам имя.

  Если вас знают в каких-то кругах, то это считается очень почетным. Хакеры — это люди, добывающие себе славу своими знаниями и добрыми делами. Поэтому любого хакера должны знать.

  Как вам узнать, являетесь ли вы хакером? Очень просто: если о вас говорят как о хакере, то вы и есть хакер. Жаль, что этого добиться очень сложно, потому что большинство считает хакерами взломщиков. Поэтому, чтобы о вас заговорили как о хакере, нужно что-то взломать. Но это неправильно, и не надо поддаваться этому соблазну. Старайтесь держать себя в рамках и добиться славы только добрыми делами. Это намного сложнее, но что поделаешь. Никто не говорил, что будет просто.

- Чем отличаются друг от друга программист, пользователь и хакер? Программист, когда пишет программу, видит, какой она должна быть, и делает так, как он видит. Пользователь не всегда знает, что задумал программист, и использует программу так, как понимает.

  Программист не всегда может предугадать действия своих клиентов, да и программы не всегда тщательно оттестированы. Пользователи имеют возможность ввести параметры, которые приводят к неустойчивой работе программ.

  Хакеры намеренно ищут в программе лазейки, чтобы заставить ее работать неправильно или необычно. Для этого требуется воображение и нестандартное мышление. Вы должны чувствовать исполняемый код и видеть то, чего не видят другие.

  Если вы хотите быть хакером, то должны уметь сделать из абсолютно безобидной вещи что-то интересное и веселое. Для этого вы должны включать свое воображение. Именно оно позволит нам создавать шуточные программы и писать эффективные алгоритмы.

Некоторые считают, что правильно надо произносить "хэкер", а не "хакер". Это так, но только для английского языка. У нас в стране это слово обрусело и стало "хакером". Мы — русские люди, и давайте будем любить свой язык и признавать его правила.

Напоследок советую почитать статью "Как стать хакером". Эта статья написана знаменитым хакером по имени Eric S. Raymond и отражает дух хакерства. Я бы дал ссылку в Интернете, но боюсь, что она может измениться к моменту выхода книги, как это произошло с первым изданием. Поэтому я понадеюсь, что вы умеете пользоваться поисковой машиной (например, yandex.ru) и сможете найти статью самостоятельно.

Тут же возникает вопрос, почему же автор относит к хакерскому искусству написание шуточных и сетевых программ. Попробую ответить на этот вопрос. Во-первых, хакеры всегда пытались доказать свою силу и знания методом написания каких-либо интересных, веселых программ. К этой категории я не отношу вирусы, потому что они несут в себе разрушение, хотя они тоже бывают с изюминкой и юмором. Зато простые и безобидные шутки всегда ценились в узких кругах. Таким способом хакер показывает не только свои знания особенностей операционной системы, но и старается заставить улыбнуться. Не секрет, что многие хакеры обладают хорошим чувством юмора, и он поневоле ищет своего воплощения. Я советую шутить с помощью безобидных программ.

Ну, а сетевое программирование неразделимо с понятием "хакер" с самого его рождения. Хакеры получили распространение благодаря сети, пониманию ее функционирования и большому вкладу в развитие Интернета. Именно поэтому данная тема будет рассматриваться достаточно подробно и, как всегда, с нестандартной точки зрения.

И последнее. Я уже сказал, что любой хакер должен уметь писать программы на каком-нибудь языке программирования. Некоторые заведомо считают, что если человек хакер, то он должен знать и уметь программировать на языке ассемблера. Это не так. Знание этого языка желательно, но не обязательно. Я люблю Delphi, который позволяет мне сделать все, что я захочу. А главное, что я могу сделать это быстро и качественно. Но я также использую C++ и ассемблер.

Языки программирования надо использовать с умом. Несмотря на мою любовь к Delphi должен признать, что он удобен для написания практически любых программ, кроме игр. Тут всегда властвовал и будет властвовать C++. При сетевом программировании иногда может оказаться удобнее C++, а иногда Delphi. Но писать большие приложения на языке ассемблера даже глупо.

Вы также должны понимать необходимость использования технологий. Тут же приведу простейший пример. Сейчас все программисты вставляют в свои продукты поддержку XML. А ведь не всем пользователям этот формат нужен, и не во всех программах он востребован. Следование рекомендациям Microsoft не означает правильность действий, потому что заказчик — не Билл Гейтс, а ваш потребитель. Поэтому надо всегда делать то, что требуется конечному пользователю.

Я рекомендую не обращать особого внимания на авторитет корпорации Microsoft (хотя некоторые разработки гениальны). Сколько технологий доступа к данным придумала Microsoft? Просто диву даешься: DAO, RDO, ODBC, ADO, ADO.NET, и это еще не полный список. Корпорация Microsoft регулярно выкидывает на рынок что-то новое, но при этом сама этим не пользуется. При появлении новой технологии все программисты бросаются переделывать свои программы под новый стандарт и в результате тратят громадные ресурсы на постоянные переделки. Таким образом, конкуренты сильно отстают, а Microsoft движется вперед, потому что не следует своим собственным рекомендациям и ничего не переделывает. Если программа при создании использовала для доступа к данным DAO, то можно спокойно оставить ее работать через DAO и не переделывать на ADO, потому что пользователю все равно, каким образом программа получает данные из базы, главное, чтобы данные были.

Могу привести более яркий пример — интерфейс. В программе MS Office постоянно меняется интерфейс, и при этом всем говорят, что именно он самый удобный для пользователя. Все бегут переводить свои программы на новый внешний вид меню и панелей, а тот же Internet Explorer и все остальные программы выглядят как 10 лет назад. В них практически ничего не меняется, и Microsoft не тратит на это время, а конкуренты тратят месяцы на переписывание множества строчек кода.

Да, следование моде придает вашим программам эффектность, но при этом вы должны сохранить индивидуальность.

Возможно, сложилось впечатление, что я противник Microsoft, но это не так. Мне очень нравятся продукты этой фирмы, например, Windows или MS SQL Server, но я не всегда согласен с ее методами борьбы с конкурентами. Это жестокий бизнес, но не до такой же степени!

Программисты и хакеры навязывают другим свое мнение о любимом языке программирования как единственно приемлемом, обычно добиваясь успеха, ведь заказчик часто не понимает в программировании. На самом же деле заказчику все равно, на каком языке вы напишете программу, его интересуют только сроки и качество. Лично я могу обеспечить минимальные сроки написания приложения, сохраняя хорошее качество, только работая на Delphi. Такое же качество на C++ я (да и любой другой программист) смогу обеспечить только в значительно большие сроки.

В данной книге будут описываться примеры именно на Visual C++, потому что этот язык является наиболее распространенным и многими признан стандартом для программирования. Однако существует книга, в которой решаются те же проблемы, но с использованием примеров на языке Delphi.

Вот когда заказчик требует минимальный размер или наивысшую скорость работы программы, тогда я берусь за ассемблер и C (не путать C и C++). Но это бывает очень редко, потому что сейчас носители информации уже практически не испытывают недостатка в размерах, и компьютеры работают в миллионы раз быстрее своих предков. Таким образом, размер и скорость программы уже не являются критичными, и на первый план ставятся скорость и качество выполнения заказа.

Итак, на такой деловой ноте мы закончим вводную лекцию и перейдем к практическим упражнениям по воинскому искусству, где часто главное — скрытность и победа минимальными силами.

Благодарности

Первым делом хочу поблагодарить своих родителей за то, что они произвели меня на свет. Если бы ни они, не было бы этой книги, по крайней мере, в моем исполнении. Свою жену Ирину, которая поддерживала меня. Свою дочку Юлю и сына Кирилла, которые иногда давали мне время для работы, а иногда отнимали его, из-за чего случалось отдыхать от компьютера.

Редакторов журналов, где публиковались мои статьи. Благодаря этому я получил большой опыт журналистики, который впоследствии помог мне написать мои книги. Я технический специалист, и еще некоторое время назад не мог связать двух слов, не говоря уже о том, чтобы красиво и доступно изложить свои мысли на компьютере. Но благодаря большой журналистской практике в журнале "Хакер" я немного научился писать.

Отдельное спасибо издательству "БХВ-Петербург" за то, что поверили в меня как в автора и помогли в издании моих книг. Благодарю всех редакторов издательства за то, что редактируют мой бред и исправляют грамматические ошибки, которых иногда бывает очень много.

Дальше я могу еще очень долго перечислять людей, которым благодарен в своей жизни, поэтому коротко: воспитателям в детсаде, учителям в школе, преподавателям в институте, всей команде журнала "Хакер" и всем моим друзьям за то, что они меня терпят, вероятно, любят, и по возможности помогают. Ну, и самое главное, я хочу поблагодарить всех моих читателей, потому что вдохновляют на новые свершения.

А вам отдельное спасибо за то, что приобрели эту книгу, отдельное спасибо, если именно приобрели, а не скачали в Интернете. Надеюсь, что вы не разочаруетесь и не будете жалеть о потраченных деньгах и времени. Чтобы этого не случилось, я постарался наполнить информацией не только страницы книги, но и компакт-диск, где находится много дополнительной и полезной информации.

ГЛАВА 1

Оптимизация

В этой главе мы будем говорить об оптимизации, причем — с неожиданных сторон. Мы будем оптимизировать видимость окон, мы будем уменьшать размер программы, повышать скорость работы кода и даже защищать его. Какое отношение имеет защита к оптимизации? Иногда эти темы являются антиподами, и именно поэтому я решил объединить эти две темы под одной крышей, а точнее — в одной главе.

Что самое главное при написании программ-приколов? Ну, конечно же, невидимость. Программы, созданные в этой и следующих главах, будут незаметно сидеть в системе и выполнять нужные действия при наступлении определенного события. Это значит, что программа не должна отображаться на Панели задач или в списке запущенных программ, в окне, появляющемся при нажатии <Ctrl>+<Alt>+<Del>. Поэтому, прежде чем начать что-то писать, нужно узнать, как спрятать свое творение от чужого глаза.

Кроме этого, программы-приколы должны иметь маленький размер. Приложения, создаваемые Visual C++ с использованием современных технологий MFC (Microsoft Foundation Classes, базовые классы от Microsoft), достаточно "весомые". Даже простейшая программа, выводящая одно окно, займет достаточно много места на диске. Если вы захотите отослать такую шутку по электронной почте, то отправка и получение письма с вашей программой отнимут лишнее время у вас и получателя. Это не очень приятно, поэтому в этой главе мы познакомимся с тем, как можно уменьшить размер программ, создаваемых в Visual C++.

1.1. Сжатие исполняемых файлов

Самый простой способ уменьшить размер приложения — использование программы для сжатия файлов. Раньше я любил использовать ASPack (**http:/**

www.aspack.com), но в последнее время перешел на использование открытой библиотеки UPX. Эта библиотека распространяется в исходных кодах и может быть найдена по адресу **http://upx.sourceforge.net**. Для работы с библиотекой лучше иметь какую-нибудь визуальную программу. Тут я предпочитаю разработку Абдульманова Рафаэля Рахимовича (**http://rafsoft.narod.ru**) — UPX Control. В принципе, программа простая, но удобная. Главное окно программы можно увидеть на рис. 1.1.

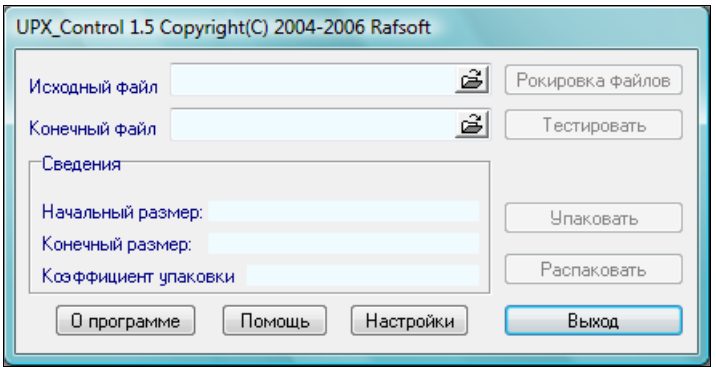

Рис. 1.1. Главное окно утилиты управления UPX

Окно простейшее, ибо нужно только выбрать исполняемый файл, который вы хотите сжать, результирующий файл — и нажать кнопку **Упаковать**. Этой же программой можно вернуться к изначальному состоянию, нажав кнопку **Распаковать.**

Теперь давайте разберемся, как работает сжатие. Сначала весь код программы сжимается архиватором. В программе используется алгоритм сжатия LZWA, оптимизированный для сжатия двоичного кода. В конец сжатого кода добавляется код разархиватора. И в самом конце ASPack изменяет заголовок исполняемого файла так, чтобы при старте сначала запускался разархиватор, который будет во время выполнения распаковывать программу в первоначальное состояние прямо в памяти и запускать распакованный вариант на исполнение.

В UPX алгоритм сжатия очень хороший, а разархиватор достаточно маленький (меньше 1 Кбайт), поэтому сжатие происходит очень сильно, а к результирующему файлу добавляется только один килобайт. Таким образом, программа может сжать файл размера в 1,5 Мбайт в 300–500 Кбайт.

Теперь, когда вы запускаете сжатую программу, сначала заработает разархиватор, который разожмет бинарный код программы и аккуратно поместит его в памяти компьютера. Как только этот процесс закончится, разархиватор передаст управление вашей программе.

Некоторые считают, что из-за расходов на распаковку программа будет работать медленней!!! Я бы сказал, что вы не заметите разницу. Даже если и будут какие-то потери, то они будут неощутимы (по крайней мере, на современных компьютерах). Это происходит потому, что архивация хорошо оптимизирована под двоичный код. И по сути дела, распаковка происходит только один раз и в дальнейшем никакого влияния на работу программы не оказывает. В результате, потери в скорости из-за сжатия будут неощутимы.

Программа без сжатия перед началом выполнения все равно грузится в память. В случае сжатого кода во время загрузки происходит разархивирование кода. В данном способе есть две стороны: происходят затраты времени на распаковку, но программа меньше занимает места на диске и быстрее считывается с него. Жесткий диск — одно из самых медленных звеньев персонального компьютера, поэтому чем меньше надо загружать, тем быстрее программа может приступить к выполнению. Именно вследствие этого итоговая потеря в скорости запуска программы незначительная.

При нормальном программировании с использованием всех современных возможностей типа визуальности и объектного программирования код получается большим, но его можно сжать специальным архиватором на 60–70%. А писать такой код намного легче и быстрее.

Еще одно "за" использование сжатия — заархивированный код труднее взломать, потому что не каждый дизассемблер сможет прочитать упакованные команды. Так что помимо уменьшения размера вы получаете защиту, способную остановить большинство взломщиков. Конечно же, профессионала этим не отпугнешь. Но взломщик средней руки не будет возиться со сжатым двоичным кодом.

Тут нужно быть честным и вспомнить, что и описанная ранее утилита работы с UPX умеет разархивировать, а значит, хакер тоже сможет ей воспользоваться. Декомпиляторы и отладчики тоже не из каменного века, и хорошие разработки умеют определять алгоритм сжатия и распаковывать код автоматически при открытии файла без дополнительных программ.

Коммерческие программы сжатия, такие как ASPack, имеют функции шифрования, которые могут усложнить жизнь хакерам. Опять же, только усложнить, но не испортить вовсе. Для исполнения код в любом случае будет расшифрован, а в арсенале взломщиков есть утилиты, позволяющие снимать код из памяти для дальнейшего анализа. Этот процесс не из легких, но вполне возможный.

1.2. Без окон, без дверей...

Следующий способ уменьшить размер программы заключается в ответе на вопрос, из-за чего программа, созданная в Visual C++, получается большой.

Ответ очень прост: C++ является объектно ориентированным языком. В нем каждый элемент представляет собой объект, который обладает своими свойствами, методами и событиями. Любой объект вполне автономен и многое умеет делать без ваших указаний. Это значит, что вам нужно только подключить его к своей форме, изменить нужным образом свойства — и приложение готово! И оно будет работать без какого-либо прописывания его деятельности.

Но в объектном программировании есть и свои недостатки. В объектах реализовано большое количество действий, которые вы и пользователь сможете производить с ними. Но реально в любой программе мы используем два-три из всех них. Все остальное — для программы лишний груз, который никому не нужен.

Но как же тогда создать компактный код, чтобы программа занимала минимум места на жестком диске и в оперативной памяти? Тут есть несколько вариантов:

- не использовать библиотеку MFC, которая упрощает программирование на языке C++. В этом случае придется больше писать кода вручную и работать только с Windows API (Windows Application Programming Interface, прикладной программный интерфейс). Программа получается очень маленькой и быстрой. Результирующий код будет меньше, чем при использовании MFC в сочетании с самым большим сжатием. Но таким образом вы лишаетесь простоты визуального программирования и можете ощутить все неудобства программирования с помощью чистого WinAPI.

  Для большей оптимизации размера файла можно использовать ассемблер, но он относительно сложен, да и писать программу на нем намного дольше, чем даже на чистом C. Хотя кому как. Именно поэтому данная тема не рассматривается в этой книге;

- сжимать готовые программы с помощью компрессоров. Объектный код сжимается в несколько раз, и программа, созданная с использованием MFC, может превратиться из монстра в 300 Кбайт в скромного по размерам "зверя", занимающего на диске всего 30–50 Кбайт. Главное преимущество состоит в том, что вы не лишаетесь возможностей объектного программирования и можете спокойно забыть про неудобства WinAPI.

Второй метод мы уже обсудили *(см. разд. 1.1)*, поэтому остается только описать первый, и самый интересный, вариант. Полученный код уже получается маленьким, и его можно еще дополнительно сжать с помощью утилиты UPX. Результат будет потрясающим.

Если вы хотите создать действительно компактную программу, то необходимо забыть про все удобства. Вы не сможете подключать визуальные формы или другие удобные модули, написанные фирмой Microsoft для упрощения

жизни программиста. Нельзя использовать классы или компоненты ActiveX. Только функции API самой ОС Windows — и ничего больше.

Теперь переходим к практическим занятиям. Запустите Visual C++ и создайте новый проект. Для этого нужно выбрать команду меню **File | New | Project** (Файл | Новый | Проект). Перед вами откроется окно создания нового проекта (рис. 1.2). Слева расположено дерево типов проектов. Нас интересует C++, поэтому выберите пункт **Visual C++**. Этот пункт мы будем выбирать при написании абсолютно всех примеров из данной книги. С правой стороны в списке **Templates** (Шаблоны) варианты создания различных проектов. Выберите пункт **MFC Application** (Приложение MFC).

Внизу окна расположены две строки ввода. В первой вы должны указать имя создаваемого проекта. Оно будет использоваться в качестве имени запускаемого файла и имени файла, который вы будете в дальнейшем открывать для редактирования. Давайте здесь укажем: `TestMFC`.

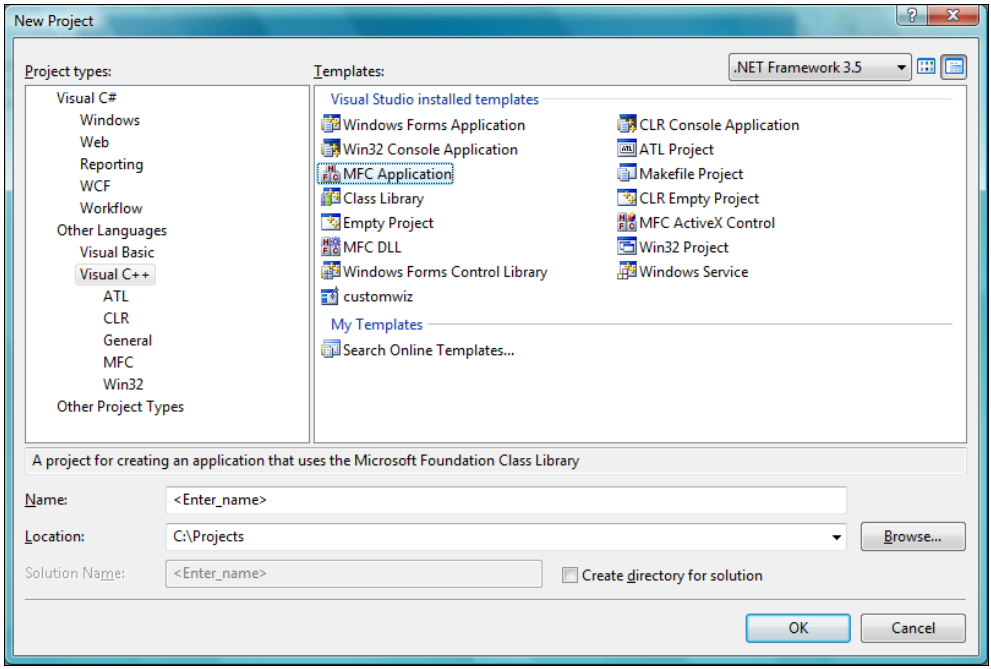

Рис. 1.2. Окно выбора типа проекта

В строке **Location** (Расположение) нужно указать путь к папке, в которой среда разработки создаст необходимые проекту файлы. Я рекомендую завести свою папку с именем, например, My C++ Projects (или любым другим на

выбор, но понятным по смыслу), где будут размещаться все проекты. Выберите эту папку и нажмите **ОК**. По окончании работы мастера у вас в папке My C++ Projects появится еще одна папка с именем TestMFC, в которой и будут находиться все файлы данного проекта.

Как только вы нажали **ОК** в окне создания нового проекта (см. рис. 1.2), перед вами откроется окно Мастера создания нового MFC-приложения (рис. 1.3).

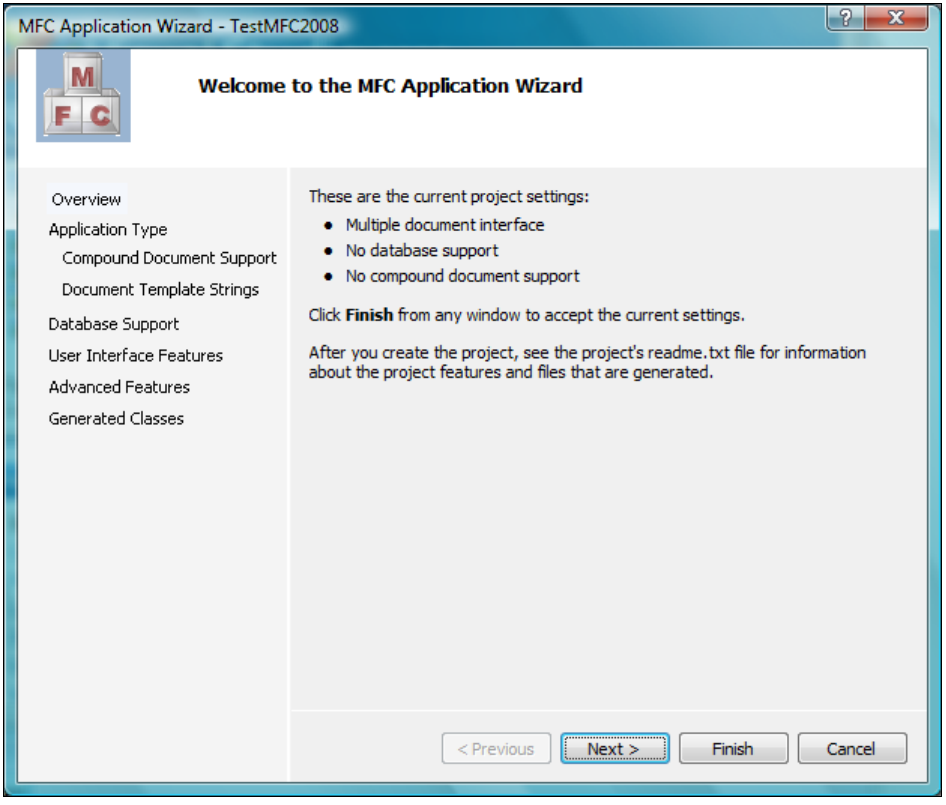

Рис. 1.3. Окно Мастера создания проекта

Вы можете сразу нажать кнопку **Finish**, чтобы завершить его работу с параметрами по умолчанию, или предварительно указать свои настройки. Наша задача — создать маленькое приложение, поэтому на данном этапе постараемся оптимизировать то, что может создать для нас Мастер.

С левой стороны окна расположены разделы. Выделяя их, вы можете настраивать соответствующие параметры. Давайте просмотрим все разделы и установим необходимые значения, а заодно познакомимся с теми параметра-

ми, которые будут использоваться для создания приложений при рассмотрении последующих примеров:

- **Application Type** (Тип приложения) — в этом разделе мы задаем тип создаваемого приложения. Давайте укажем здесь следующие значения:
  - **Single Document** (Документ с одним окном) — нам достаточно будет только одного окна. Многодокументные приложения мы не рассматриваем, поэтому большинство примеров с использованием MFC будет основываться на этом типе приложений или на основе диалоговых окон (**Dialog based**);
  - **Project style** (Стиль проекта) — во всех приложениях будем использовать стиль по умолчанию (**MFC стандарт**);
  - **Document | View architecture support** (Поддержка архитектуры Документ | Просмотр) — это значение нас пока не интересует, поэтому оставим по умолчанию;
- **Advanced Features** (Дополнительные возможности) — в этом разделе в будущем нас будет интересовать только параметр **Windows socket** (поддержка сокетов Windows), который позволит нам писать примеры для работы с сетью.

Во всех остальных разделах оставляем значения по умолчанию, потому что мы не будем использовать базы данных или документы. В большинстве случаев нам будет достаточно окна и меню. А первое время постараемся обходиться вообще без MFC.

Нажмите кнопку **Finish**, чтобы завершить работу мастера. По завершении его работы вы увидите главное окно среды разработки Microsoft Visual C++, представленное на рис. 1.4. Это окно мы будем использовать достаточно часто, поэтому в процессе работы уточним все детали.

Сейчас нас интересуют параметры проекта. Мы должны отключить все, что нам будет мешать. При сборке проекта в Visual C++ по умолчанию могут использоваться два вида настроек: debug и release. Первый необходим на этапе разработки, и в этом режиме Visual C++ создает запускаемый файл, который содержит слишком много дополнительной информации. Она будет необходима вам в среде разработки в дальнейшем при отладке программы и поиске ошибок. Во втором режиме эта информация отключается, и запускаемый файл будет меньшего размера.

В верхней части окна на панели с кнопками найдите выпадающий список, в котором написано **Debug**. Измените это значение на **Release.**

Среда разработки Visual C++ может создавать запускаемые файлы, использующие MFC-библиотеки двух типов: статические и динамические. По умолчанию используется динамическая сборка. В таком режиме запускаемый

файл получается меньшего размера, но он не будет работать без динамических библиотек, таких как mfcXXX.dll, где XXX — это номер версии среды разработки.

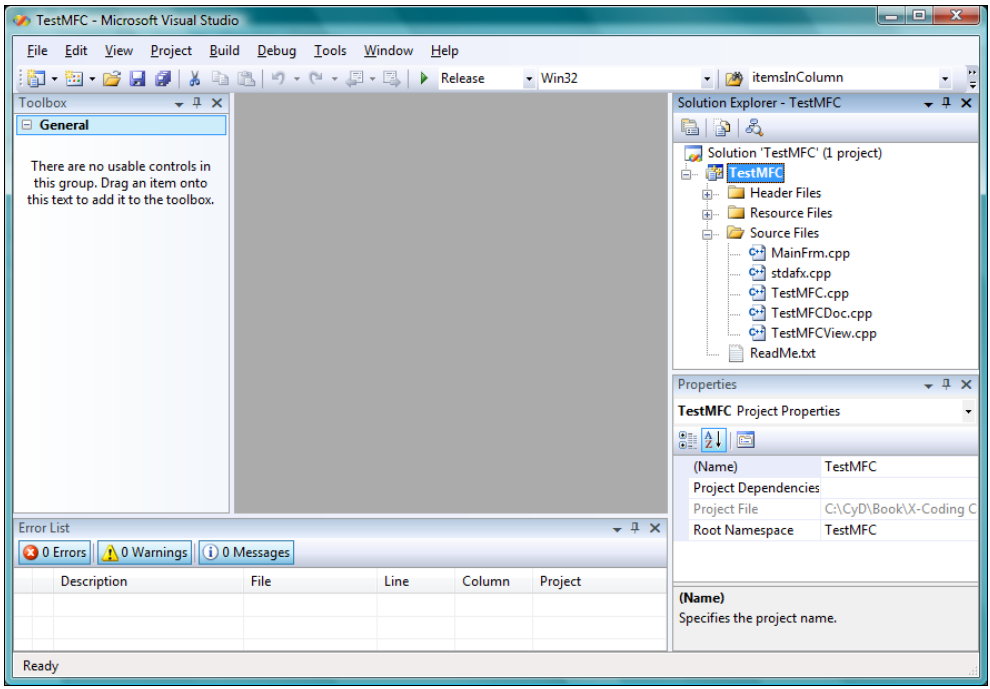

Рис. 1.4. Окно среды разработки

В этом случае, чтобы кто-то смог запустить наш проект, мы должны отослать ему не только запускаемый файл, но и библиотеки. Это неудобно и неприлично. Лучше использовать статическую компиляцию, при которой результирующий файл будет намного больше, зато все будет содержать внутри себя. При таком подходе не потребуются дополнительные библиотеки.

Чтобы изменить тип использования MFC, в окне **Solution Explorer** сначала выберите имя вашего проекта, а затем в меню команду **Project/Properties**. На рис. 1.5 представлено окно свойств проекта.

Примечание

После выхода первой книги я получил много вопросов, когда пользователи не смогли скомпилировать проекты. Это случилось у тех, у кого по умолчанию в IDE выбирается Multibyte-строки. Я не использовал Unicode-строки, поэтому если компилятор будет ругаться, то в параметре **Character Set** выберите значение **No Set**.

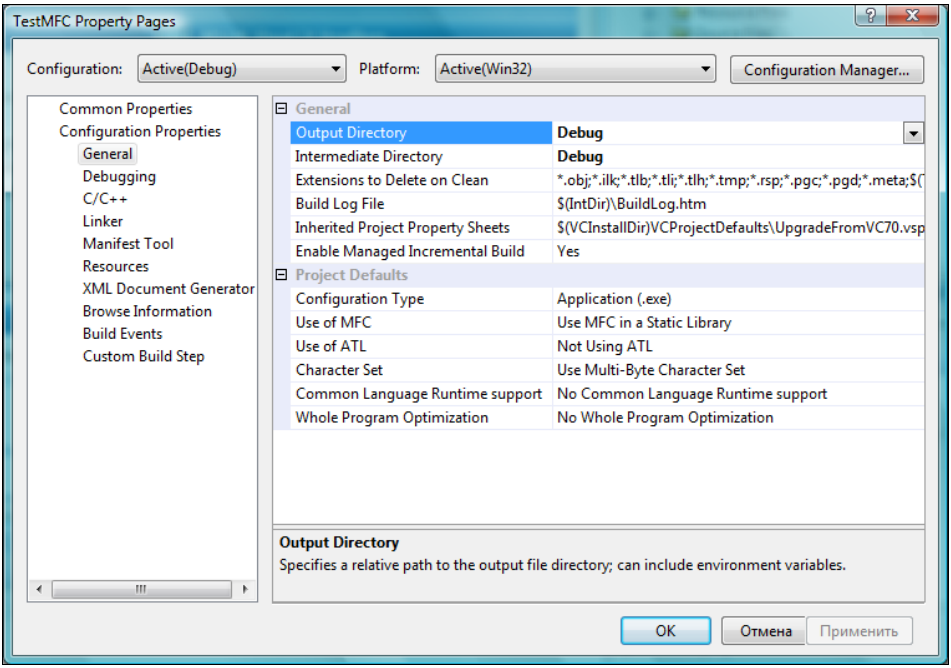

Рис. 1.5. Окно свойств проекта

Слева в окне расположены разделы свойств. Нас будет интересовать раздел **General** (Основные). Выделите его, и в основном окне появится список соответствующих свойств. Найдите свойство **Use of MFC** и измените его значение на **Use MFC in a Static Library**. Нажмите кнопку **OK**, чтобы закрыть окно и сохранить изменения.

Теперь соберем наш проект в готовый исполняемый файл. Для этого нужно выбрать команду меню **Build | Build solution** (Построить | Построить проект). Внизу главного окна, в панели **Output** (Вывод), будет появляться информация о ходе сборки. Дождитесь, пока не появится сообщение типа:

```
---------------------- Done ----------------------
    Build: 1 succeeded, 0 failed, 0 skipped
```

Теперь перейдите в папку, которую вы выделили под хранение проектов, и найдите там папку TestMFC. В ней расположены файлы с исходным кодом нашего проекта, сгенерированные мастером. Тут же должна быть папка Release, в которой среда разработки создала во время компиляции промежуточные и исполняемый файлы. Выделите файл TestMFC.exe и посмотрите его свойства (надо щелкнуть правой кнопкой мыши и выбрать в появившемся меню пункт **Свойства**). Размер нашего пустого проекта у меня в Visual

Studio 2008 составил 446 Кбайт. От такого размера не только звезда будет в шоке, но и простые программисты, как мы! Это очень много. В Visual Studio 2003 это было около 380 Кбайт.

Попробуйте открыть его в программе UPX и сжать. У меня сжатый исполняемый файл составил 209 Кбайт. Сжатие составило более 50%, и это уже более или менее приемлемый размер для шуточной программы.

Примечание

Пример этой программы вы можете увидеть на компакт-диске в папке /Demo/Chapter1/TestMFC. Чтобы открыть этот пример, выберите команду меню **File | Open solution**. Перед вами появится стандартное окно открытия файлов. Перейдите в нужную папку и выберите файл с именем проекта и расширением .vcproj или .sln.

Чтобы сделать программу еще меньше, необходимо отказаться от MFC и писать на чистом C. Это немного сложнее и не так удобно, но для небольших проектов вполне приемлемо.

Для того чтобы создать маленькую программу без использования MFC, нужно снова использовать меню **File | New | Project** и здесь выбрать уже тип создаваемого проекта **Win32 Project**. В качестве имени давайте укажем `ctest`, а путь оставим тот же.

Если у вас все еще открыт предыдущий проект, то под строкой ввода пути для проекта есть переключатели: **Add to solution** (Добавить в решение) и **Close solution** (Закрыть решение). Если вы выберете первый из них, то текущий проект будет добавлен в уже открытый. Если выбрать закрытие, то текущий проект будет закрыт, и для вас будет создано новое рабочее поле.

После нажатия кнопки **OK** перед вами откроется окно Мастера. Первый шаг чисто информационный, поэтому выберите раздел **Application Settings** (Настройки приложения). Перед вами откроется окно, как на рис. 1.6.

Нас интересует простое приложение Windows, поэтому вы должны выбрать в разделе **Application type** (Тип приложения) переключатель **Windows application**. Больше ничего, чтобы мастер не добавлял ничего лишнего. Нам необходим только самый минимум. Нажмите кнопку **Finish**, и будет создан новый проект.

Здесь также нужно изменить **Debug** на **Release**, чтобы создать проект без дополнительной информации. В настройках проекта ничего менять не надо, потому что созданный мастером шаблон не использует MFC, и ему не нужны динамические библиотеки. Можете зайти в свойства проекта и убедиться, что в свойстве **Use of MFC** стоит **Standard Windows Libraries** (Использовать стандартные библиотеки Windows). Это значит, что нет MFC, и ничего до-

полнительного программе не надо, только стандартные библиотеки Windows и прямой вызов API операционной системы.

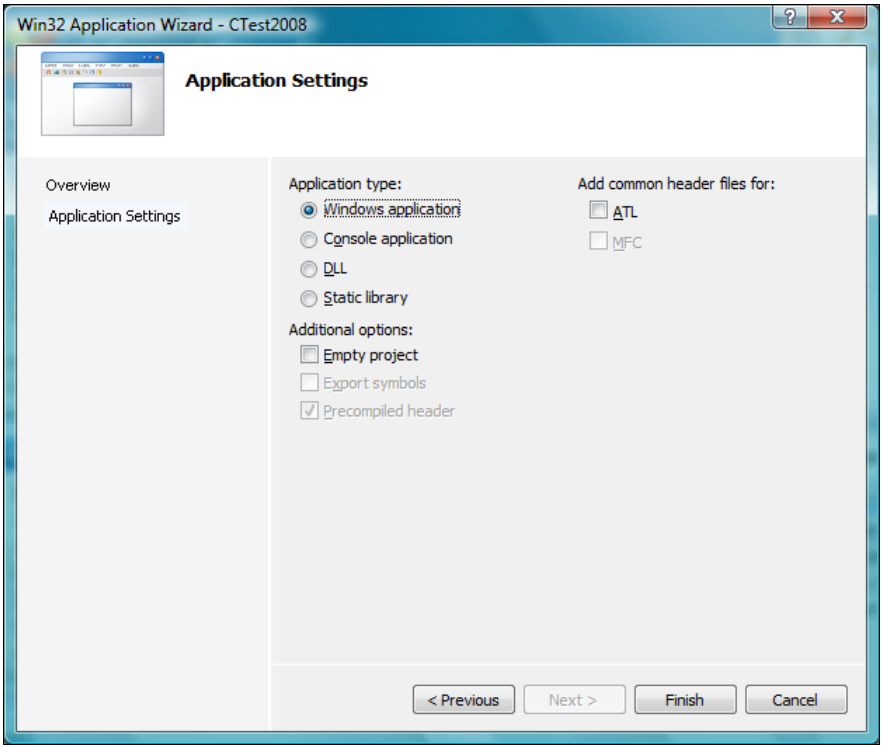

Рис. 1.6. Параметры приложения Win32

Попробуйте откомпилировать проект. Для этого выберите команду меню **Build | Build solution**. По окончании сборки перейдите в папку ctest/Release каталога, где вы создавали проекты, и посмотрите на размер результирующего файла. У меня в Visual Studio 2008 получилось 86 Кбайт. В Visual Studio 2003 размер составлял 81 Кбайт. Если сжать файл, то получится менее 70 Кбайт. Вот такая программа уже скопируется по сети достаточно быстро.

Программу можно еще больше оптимизировать и убрать некоторые вещи, которые не используются, но отнимают драгоценные килобайты, однако мы этого делать уже не будем. Сделаем только пару выводов: размер файла зависит от используемых функций и может отличаться в зависимости от используемого компилятора. Каждый компилятор может использовать разные функции и методы оптимизации.

Если вы не имеете опыта программирования, то я бы порекомендовал сейчас отложить книгу и прочитать GetStarted.doc на компакт-диске в каталоге Doc и