Михаил Фленов

ПРОГРАММИРОВАНИЕ В Delphi ГЛАЗАМИ ХАКЕРА

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2007

УДК 681.3.068
ББК 32.973.26-018.1
Ф69

Фленов М. Е.

Ф69 Программирование в Delphi глазами хакера. — 2-е изд., перераб. и доп.— СПб.: БХВ-Петербург, 2007. — 480 с.: ил. + CD-ROM

ISBN 978-5-9775-0081-4

Рассмотрены нестандартные приемы программирования, а также примеры использования недокументированных функций и возможностей языка Delphi в ОС Windows при разработке шуточных программ и серьезных сетевых приложений для диагностики сетей, управления различными сетевыми устройствами и просто при повседневном использовании интернет-приложений.

Компакт-диск содержит исходные коды примеров и откомпилированные программы, а также популярные приложения компании CyD Software Labs.

Для программистов

УДК 681.3.068
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Инны Тачиной*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.09.07.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 38,7.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04
от 11.11.2004 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 978-5-9775-0081-4

Оглавление

Введение

Замечания ко второму изданию

Это второе издание книги, но, в отличие от других книг, у меня второе издание отличается от первого как собака от кошки. Я переписал больше 80% книги, и не только исправил ошибки, но и добавил просто непристойное количество нового материала. Устаревшую информацию, которая может быть использована только в Windows 95, я убрал из книги вовсе. Те темы, которые могут быть еще интересными, я перенес в виде статей в PDF-формате на компакт-диск.

Компакт-диск заслуживает отдельного внимания. В данном издании я постарался сделать его максимально интересным. В каталоге Документация вы найдете множество интересных документов и статей. Поэтому если вы читаете эту страницу еще в магазине, то обязательно требуйте компакт-диск. Его содержимое вас приятно удивит. Наверно, никто еще не давал такого количества дополнительной документации своим читателям.

Благодарности

Первым делом хочется поблагодарить своих родителей за то, что они произвели меня на свет. Если бы не они, не было бы этой книги, по крайней мере, в моем исполнении. Свою семью, жену, детей я просто обязан отблагодарить за их терпение к моей работе и за частые исчезновения в виртуальной реальности.

Давыдова Дениса — главного редактора "Игромании". Когда-то он был редактором игровой части в журнале "Хакер", и мне повезло, что я начинал свою писательскую деятельность в журнале именно с ним. Журнал тогда только начинал свое развитие, и Денис уделял мне достаточно много времени. Я программист и выжать из себя пару литературных строк в то время вообще не мог (да и сейчас я в этом деле не гений, у меня в школе по русскому и литературе всегда было 3 с большим минусом), а Денис мне дал пинок, от которого я, можно сказать, начал свою карьеру.

Покровского Сергея aka SINtez — главного редактора журнала "Хакер". После того как Давыдов Денис ушел из журнала, я какое-то время стал меньше публиковаться, но потом меня, можно так сказать, взял под крыло Сергей.

Сначала я стал вести рубрику "Hack-Faq", а потом и "Кодинг", которая очень быстро развилась и получила популярность.

Хочется поблагодарить всех друзей, которые меня поддерживают и помогают мне в моей работе и в жизни. Я могу еще очень долго перечислять людей, которым я благодарен в своей жизни, поэтому перечислю 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 за то, что они читают мои опусы.

Отдельная благодарность редакторам, которые делают мои книги лучше, находят и помогают исправить мои ошибки, опечатки и недочеты, редакции издательства "БХВ-Петербург" за длительное и плодотворное сотрудничество.

О книге

Как вы, наверное, уже поняли, я помешанный на программировании человек и в течение всей книги не буду блистать литературным стилем. Зато я постараюсь поделиться своими знаниями и надеюсь рассказать вам что-то новое.

Когда вышел в свет первый вариант моей работы о Delphi глазами хакера, то книга вызвала бурю эмоций, как положительных, так и отрицательных. Я прекрасно понимаю, что такая книга мало кого оставит равнодушным, но надеюсь, что положительных эмоций у вас останется намного больше.

В книге я расскажу вам о программировании для хакера. Я буду достаточно часто использовать один термин — "кодинг". Что это такое? Под этим словом мы будем, как и все, подразумевать слово "программирование". А вот под словом "хакер" лично я подразумеваю немного иной смысл, чем другие. Я считаю, что хакер — это профессионал в компьютерной сфере, но не обязательно доставляющий много неприятностей другим людям своими знаниями. Так вот в этой книге я постарался показать много интересных вещей с точки зрения сетевого программиста-профессионала, а не взломщика. Более подробно о понятии "хакер" вы найдете в следующем разделе.

Хакер — понятие обширное, и затронуть все стороны просто нереально. Я специализируюсь на сетях, их безопасности, Web-программировании, Web-безопасности и шутках над системой. Драйверами и нулевым кольцом я занимаюсь не очень серьезно. Да, я знаю основы популярных ошибок, я знаю основные проблемы, но не собираюсь посвящать этим неприятностям свою жизнь.

В данной работе я попробую привести как можно больше нестандартных приемов программирования, недокументированные функции и возможности, а главное — продемонстрирую вам приемы работы с сетью в операционной системе Windows.

В книге приведено максимальное количество примеров на языке программирования Delphi. Для этого я написал множество шуточных программ и сетевых приложений. Чистой теории будет мало, зато практических занятий — хоть отбавляй. Практические примеры позволят вам лучше понять тему, потому что лучше один раз увидеть результат собственными глазами, чем сто раз услышать или прочитать. Есть еще один плюс у большого количества кода — вы можете использовать мои заготовки в ваших собственных приложениях. Да, код шуточных программ особо нигде не пригодится, а вот примерам работы с сетью и написанным мною функциям вы сможете найти применение.

Для понимания книги вам понадобятся хотя бы начальные знания о среде Delphi и сносное умение общаться с компьютером и мышью. Что касается сетевого программирования, то его я опишу полностью, начиная от основ и заканчивая сложными примерами. Так что тут начальные знания желательны, но не обязательны. Если вы начинающий программист, то могу посоветовать для получения основ прочитать мою книгу "Библия Delphi"[1] и посетить мой сайт **www.vr-online.ru**, где выложено достаточно много полезной информации.

Если вы ожидали увидеть в данной книге примеры и описания вирусов, то вы сильно ошиблись. Ничего разрушительного я делать и рассказывать не буду. Я занимаюсь созиданием, а не разрушением. Чего и вам советую. Меня больше интересует безопасность, а не взлом. Да, я знаю, как взламываются программы и сайты, потому что все, кто хочет защищаться, должен знать, как вас могут атаковать.

Для эффективной работы с книгой вам понадобятся хотя бы начальные знания Delphi. Вы должны уметь создавать простое приложение, знать, что такое циклы и как с ними работать. Не помешают знание адресации, указателей, и для чего они нужны.

Эта книга построена не так как многие другие. В ней нет длинных и нудных теоретических рассуждений, а только максимум примеров и исходного кода. Ее можно воспринимать как практическое руководство к действию. Чтобы книга не была занудной, я постарался описывать все простым и неформальным языком и в некоторых случаях даже в веселой форме.

Я постарался облегчить вам задачу, описав все как можно проще. Большинство кода расписано очень подробно, и в тексте программ вы найдете мак-

[1] Фленов М. Библия Delphi. — СПб.: БХВ-Петербург, 2004.

симум комментариев, которые помогут получать наслаждение от чтения кода вместо обычной головной боли. Некоторые меня критикуют за то, что я уж слишком сильно все разжевываю, и остается только проглотить, но мне кажется, что во время чтения книги, вы не должны слишком сильно отвлекаться на изучение кода. Таким образом, изучать код можно будет даже в метро, без наличия под рукой компьютера. Но для закрепления материала, я бы рекомендовал все же писать вам код самостоятельно и пробовать воссоздать мои примеры без книги или хотя бы без обращения к исходным кодам на компакт-диске. Это позволит еще лучше усвоить материал и глубже понять проблему. Пробуйте, экспериментируйте и читайте.

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

Это сравнение приведено здесь не просто так. В 2002 году я попал в больницу с температурой и болями в области живота. Меня положили в хирургическое отделение и хотели вырезать аппендицит. Я пролежал три дня, и ни один врач не решался меня отправить на операцию, но в то же время никто не знал, откуда у меня боли, и почему температура под вечер поднимается до 39 градусов.

На третий день вечером я сбежал из больницы, потому что у моей мамы был день рождения. На нем присутствовал знакомый врач (по специализации акушер), который, осмотрев меня, сказал пить по 1 таблетке через каждые 12 часов (не будем уточнять, что это был за препарат, но это от желудка) и выписываться из больницы. Может кто-то не поверит, но после первой таблетки температура упала, а после второй я вообще плясал, как Борис Моисеев. Результат: врачи перепутали отравление с аппендицитом и чуть не лишили меня моего аппендикса. А ведь могли же вырезать по ошибке или просто ради интереса.

Уже после выхода первой версии Delphi глазами хакера, в начале 2006 года меня жена снова отправляет в больницу с температурой и болями в животе. В этот день по скорой дежурила другая больница, и в ней снова решили, что у меня аппендицит. Здесь врачи не стали рисковать, а сразу же отправили меня в общий наркоз видеть сны, и среди ночи сделали мне в пузе две дырки, чтобы обследовать внутренности. Конечно же, аппендикс был в порядке, и ничего не удалив, меня зашили. Кстати, дело было в субботу и оперировали меня дежурные врачи, и, конечно же, за деньги, несмотря на нашу бесплатную медицину. В понедельник, когда на работу вышли все врачи, на обходе главный хирург выписал меня, чтобы я с температурой никого не заразил.

Вот так вот во второй раз я заплатил деньги за то, что мне сделали две дырки, ничего не нашли, и выкинули через 36 часов после операции, когда я только начал понемногу ходить. Могли бы хоть перевести в другое отделение, и найти, что же со мной было.

Этот случай еще больше закрепил мое отношение к практике. Ничто не может привести к такому пониманию предмета, как хорошее практическое занятие, потому что когда вы можете ощутить все своими руками, то никакая теория не нужна.

Еще один пример из компьютерной жизни. В 2000 году я проходил обучение в МГТУ им. Баумана на нескольких курсах Microsoft SQL Server. Курсы были очень хорошие, и преподаватель старался все очень подробно и легко преподнести. Но сам курс был поставлен корпорацией как теоретический с небольшим добавлением лабораторных работ. В результате, нам очень хорошо объяснили, ЧТО может делать SQL Server. Но когда после курса я столкнулся с реальной проблемой, то понял, что я не знаю КАК сделать что-либо. Приходилось снова открывать книгу, которую выдали в центре обучения (она была предоставлена Microsoft и, конечно же, на английском языке), и, читая обширную теорию и маленькие практические примеры, разбираться с реальной задачей. Уж лучше бы я узнал на курсах, как практически выполнять примеры из жизни, 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 восполнит хотя бы небольшой пробел в этой сфере и поможет вам в последующей работе и решении различных задач программирования.

Все примеры, которые описаны в книге, можно найти на компакт-диске в каталоге Примеры. Несмотря на это, я советую все описанное создавать самостоятельно и обращаться к готовым файлам, только если возникли какие-то проблемы, чтобы сравнить и найти ошибку. Любая информация после практической работы откладывается в памяти намного лучше, чем прочтенные сто страниц теории.

Даже если вы решили воспользоваться готовым примером, обязательно досконально разберитесь с ним. Попробуйте изменить какие-то параметры и посмотреть на результат. Можете попытаться улучшить программы, добавив

какие-то возможности. Только так вы сможете понять принцип работы используемых функций или алгоритмов.

Помимо этого, на компакт-диске можно найти следующие каталоги:

- Headers — здесь находятся все необходимые заголовочные файлы, которые нужно будет подключать к Delphi для компиляции некоторых примеров;
- Source — здесь я выложил исходные коды своих простых программ, чтобы вы могли ознакомиться с реальными приложениями. Их немного, но посмотреть стоит;
- Документация — дополнительная документация, которая может понадобиться для понимания некоторых глав. Документации достаточно много, поэтому будет что почитать на досуге. Если вы скачали книгу из Интернета, и содержимого диска нет, то могу только посочувствовать;
- Программы — программы, которые пригодятся при программировании. Среди них Header Convert — программа, которая конвертирует заголовочные файлы с языка C на Delphi, и ASPack — программа сжатия исполняемых файлов.

Кто такой хакер и как им стать?

Слово "хакер" в названии книги — это не дань моде и не рекламный ход. Это стиль жизни, но чтобы понять, что я вкладываю в это понятие, следует узнать, кто такой хакер. Многие вкладывают в слово "хакер" совершенно другой смысл, поэтому могут воспринять данную работу неправильно.

Прежде чем приступить к практике, я хочу "загрузить" вашу голову небольшой теорией. Нет, теорией не по программированию, а просто познавательной информацией. 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". Почему я уверен, что заблуждаются? Да потому что мнений много, но только одно из них может быть верным. Однако каждый вправе считать свое мнение наиболее правильным. И я не буду утверждать, что именно мое мнение единственно верное, но оно отталкивается от действительно корректного и правильного понятия, с которого все начиналось.

Но для начала, посмотрим на перевод этого слова, который дает нам ABBYY Lingvo, и оказывается, что один из вариантов — дилетант, непрофессионал (рис. В1). Оригинально, не правда ли? И как это слово связано с компьютерным взломщиком?

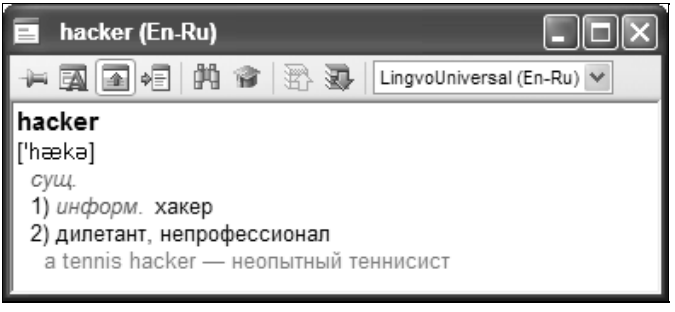


Рис. В1. Перевод слова "hacker" в ABBYY Lingvo

А теперь посмотрим на перевод слова "cracker" (рис. В2). Это слово также очень часто связывают со взломщиками, и оно уже лучше отражает суть (дробилка, босяк), а иногда и слова хвастун или лжец тоже верно отражают суть.

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

Понятие "хакер" зародилось, когда только начинала распространяться первая сеть 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 к свободному обмену информацией и началось такое бурное развитие Всемирной сети. Именно хакеры помогли развитию Интернета и создали FIDO. Благодаря им, появилась UNIX — система с открытым исходным кодом, на котором 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 не приветствовалось виртуальным сообществом.

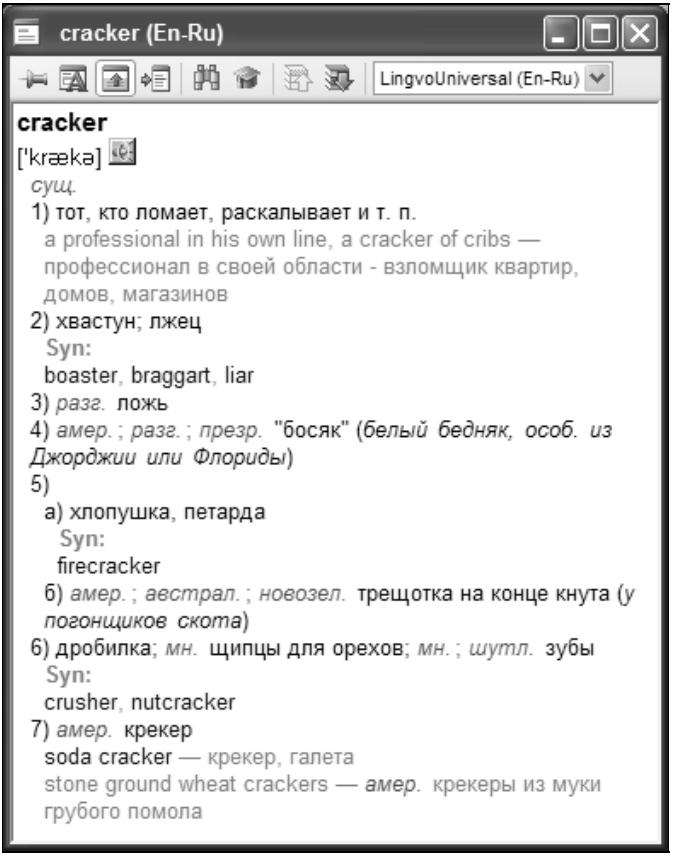


Рис. В2. Перевод слова "cracker" в ABBYY Lingvo

Настоящий хакер — это творец, а не разрушитель. Так как творцов оказалось больше, чем разрушителей, то истинные хакеры выделили тех, кто занимается взломом, как отдельную группу и назвали их крекерами (взломщиками) или просто вандалами. И хакеры, и взломщики являются гениями виртуального мира. И те и другие борются за свободу доступа к информации. Но только крекеры взламывают сайты, закрытые базы данных и другие источники информации. Если крекер совершает взлом исключительно ради свободы информации, то это еще можно простить, но если это делается ради собственной наживы, ради денег или минутной славы, то такого человека можно назвать только преступником (кем он по закону и является!).

Существует еще одно деление, когда хакером называют интернет-взломщика, а крекером — взломщиков программ. При этом White Hat (белая

шапка) хакеры являются "добрыми", т. е. взламывают сети только для изучения и улучшения нашей жизни, а Black Hat (черная шапка) — это все те же вандалы. Только журналистам в основном все равно, кто перед ним — белая или черная шапка. Они готовы любого назвать взломщиком и хакером, лишь бы раздуть сенсацию и получить за это деньги. Именно поэтому я склоняюсь к классическому делению на хакеров, без каких-либо шапок и крекеров. Это уже разные слова, и тут что-то спутать сложнее, а значит, меньше путаницы с понятиями.

Как видите, хакер — это просто человек, который в чем-то хорошо разбирается, который живет знаниями и безопасностью. Все, кто приписывает этим людям вандализм, сильно ошибаются. Истинные хакеры никогда не используют свои знания во вред другим.

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. Если вы будете еще и знать в нем каждую железку, то это только добавит к вашей оценке по "хакерству" большой жирный плюс.

  Что я подразумеваю под умением эффективно управлять своим компьютером? Это значит знать все возможные способы каждого действия и в каждой ситуации уметь использовать наиболее оптимальный. 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.

  Маленький пример на эту тему. Мой начальник всегда копирует и вставляет из буфера с помощью кнопок на панели инструментов или команд контекстного меню, которое появляется при щелчке правой кнопкой мыши. Но если вы делаете также, 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 "горячих" клавиш <Ctrl>+<C>/<Ctrl>+<V> или <Ctrl>+<Ins>/<Shift>+<Ins>, которые достаточно универсальны и присутствуют практически во всех современных приложениях.

- 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 по данной теме по-

толще и вы поймете, что сильно ошибались. Компьютеры — это такая вещь, в которой невозможно знать все!!!

Хакеры — это, прежде всего, профессионалы в каком-нибудь деле. И тут даже не обязательно должен быть компьютер. Хакером можно стать в любой области, но я буду рассматривать только компьютерных хакеров.

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

  Хотя я не очень люблю Visual Basic за его ограниченность, неудобность и сплошные недостатки, я видел несколько великолепных программ, который 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.

  Хакер — это созидатель, 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 Delphi можно создавать простенькие утилиты за очень короткое время. Так что не поленитесь и изучите программирование. А я на протяжении всей книги буду показывать то, что необходимо знать программисту-хакеру, и покажу множество интересных приемов и примеров.

- Не тормози 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.

  Открытая информация — не значит, что вы не можете зарабатывать деньги. Это никогда не возбранялось, потому что хакеры тоже люди и тоже хотят кушать и должны содержать свою семью. Но деньги не должны быть главным в вашей жизни. Самое главное — это созидание, процесс создания. Вот тут проявляется еще одно отличие хакеров от крекеров — хакеры "создают", а крекеры "уничтожают" информацию. 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 крекером, а 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 с нее систем защиты. Нужно уважать труд других программистов, потому что защита программ — это их хлеб. Если в программах не будет защиты, то программисту не на что будет есть.

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 крекеры спокойно ломают программы, не боясь закона? Ведь это тоже воровство. Лично я приравниваю взлом программы к воровству телевизора с полки магазина и считаю это одним и тем же.

- Не изобретай велосипед. Тут опять действует созидательная функция хакеров. Они не должны стоять на месте и обязаны делиться своими знаниями. Например, вы написали какой-то уникальный код, 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 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 выглядят как китайцы — все на одно лицо. Каждый из них — это отдельный индивидуум и не похож на других. Не надо копировать другого человека. Тем, что вы удачно скопируете кого-то, вы не станете 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.

- Ну и самое главное — нужно учиться, читать, пробовать и тестировать. Лично я себя хакером не считаю, хотя у меня достаточно большой опыт в программировании и в безопасности. Мне приходилось взламывать программы и сайты, но глядя на всю ту информацию, которую еще необходимо впитать, изучить и проверить, так становится понятно, что мои знания по сравнению с тем, что можно знать — это капля в море. Поэтому я продолжаю все время что-то изучать.

  Но даже если на секунду представить, что вы смогли изучить абсолютно все, что вас интересовало, то ощущение полноты будет недолгим. ИТ-сфера развивается очень быстро, и через год знания могут очень сильно

устареть. Знания постоянно нужно обновлять, следить за изменениями и изучать все новое.

Как вам узнать, являетесь ли вы хакером или нет? Очень просто, если о вас говорят, как о хакере, то вы он и есть. Да, некоторые непосвященные в компьютерах иногда называют хакерами своих друзей, которые умеют банально установить Windows, или что-то поменять в системном блоке, например заменить плату.

Жаль, что реального признания 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 на этот соблазн. Старайтесь держать себя в рамках и добиваться славы только добрыми делами. Это намного сложнее, но что поделаешь... Никто не говорил, что будет просто.

В нашей стране получить признание еще сложнее. Может быть это наш менталитет, а может недостаток воспитания. Когда кто-то становится популярным, то его стараются опозорить или даже унизить. Всеми уважаемый хакер — Кевин Митник для своего времени был настоящим гением. Да, его знания немного устарели за время пребывания в тюрьме, но старые заслуги от этого не становятся пустышкой. В нашей же стране большинство считает его просто ламером. А жаль, ведь это простое неуважение к другим и показ собственного плохого воспитания.

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

Напоследок советую почитать статью "Как стать хакером" на сайте **www.sekachev.ru**. Эта статья написана знаменитым человеком в ИТ-области по имени Eric S. Raymond. С некоторыми его взглядами я не согласен, но в большинстве своем она также отражает дух хакерства.

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. В эту категорию 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. Этим хакер показывает не только свои знания особенностей операционной системы, но и старается заставить ближнего своего 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, понимания ее функционирования и большому вкладу в развитие Интернета.

Ну и последнее. Я уже сказал, что любой хакер должен уметь программировать 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. Знания ассемблера желательно, но не обязательно. Я люблю Delphi, и он позволяет мне сделать все, что я захочу. А главное, что я могу сделать это быстро и качественно.

Я по образованию экономист-менеджер и 6 лет проучился в институте по этой специальности. Но даже до этого я знал, что заказчик всегда прав. Почему-то в компьютерной области стараются избавиться от этого понятия. Например, Microsoft делает упор на программистов, пытаясь воспитывать их писать определенные программы, не объясняя, зачем это нужно пользователям. Многие тупо следуют этим рекомендациям и не задумываются о необходимости того, что они делают.

Тут же приведу простейший пример. Сейчас все программисты вставляют в свои продукты поддержку XML, и при этом никто из них не задумывается о необходимости этого. 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 требует конечный пользователь.

Я вообще рекомендую не обращать внимания на корпорацию Microsoft, потому что считаю ее только тормозом прогресса. И это тоже можно доказать на примере. 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 кидаются переделывать свои программы под новый стандарт и в результате тратят громадные ресурсы на постоянные переделки. Таким образом, конкуренты сильно тормозят, а Microsoft движется вперед, потому что не следует 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. Несмотря на то, что уже давно появились ADO и ADO.NET, некоторые утилиты Microsoft продолжают использовать старые интерфейсы, и никто их не собирается переписывать, чтобы не тратить на это время.

Программисты и хакеры навязывают другим свое мнение о любимом языке программирования как о единственно приемлемом и делают это обычно успешно, потому что заказчик очень часто ничего 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 вкупе с хорошим качеством, только работая на Delphi. Такое же качество на VC++ я (да и любой другой программист) смогу обеспечить только в значительно бóльшие сроки. Ну что поделаешь, если объектное программирование проигрывает компонентному в скорости разработки. Недаром у Microsoft появилась компонентная платформа .NET и компонентный язык C#.

Вот когда заказчик требует минимальный размер или наивысшую скорость работы программы, тогда я берусь за ASM 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 первых своих предков. Таким образом, размер и скорость программы уже не являются критичными и на первый план ставится 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.

Обратная связь

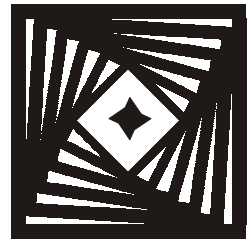
Первое издание этой книги содержало некоторые ошибки, неточности и недостатки. Эта версия тоже может содержать ошибки, потому что я полностью переработал материал. Если вы что-то нашли, то обязательно сообщите мне через мой сайт **http://www.vr-online.ru** или на e-mail **horrific@vr-online.ru**. Лучше сообщать через сайт, потому что его я посещаю каждый день.

Каждая работа может содержать ошибки, и я прекрасно это понимаю. Недавно я прочитал книгу "Совершенный код"[2], там автор признался, что после выхода первого издания он нашел около 200 ошибок и неточностей. Но, несмотря на это, книга была бестселлером, и ошибками не пытались оскорбить автора, как это делают в нашей стране. Если вы нашли ошибку, то сообщите мне о ней.

Как я уже говорил, все ошибаются, и все могут совершить ошибку. Я тоже человек, поэтому нормально отношусь к критике и всегда жду отзывов читателей.

---

[2] Макконелл С. Совершенный код. — СПб.: Питер, 2005.



Глава 1

Минимизация, скорость и безопасность

Что самое главное при написании шуточных программ? 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 появляться на панели задач или в списке запущенных программ в окне, выдаваемом при нажатии комбинации клавиш <Ctrl>+<Alt>+<Del>. Да, в Windows 2000/XP и других ОС есть возможность увидеть запущенные процессы, и от них спрятаться труднее, но это уже не такая большая проблема. Так что, прежде чем начать что-то писать, нужно узнать, как спрятать свое творение от чужого глаза.

Помимо этого, программы-приколы должны иметь маленький размер. Приложения, создаваемые Delphi, достаточно "весомые". Даже простейшая программа, выводящая одно окно, занимает более 200 Кбайт на диске. Если вы захотите отослать такую шутку по электронной почте, то отправка и получение письма с вашей программой отнимет лишнее время у вас и получателя. Это не очень приятно, поэтому в данной главе я познакомлю вас с тем, как можно уменьшить размер программ, создаваемых в Delphi.

1.1. Методы минимизации

Чтобы понять, какие методы есть для уменьшения размера программы, необходимо ответить на вопрос: "Из-за чего программа, созданная Delphi, получается большой?" Ответ очень прост, Delphi является объектным языком, а если точнее — компонентным. *Компоненты* — это следующий шаг в программировании. В Delphi каждый элемент выглядит как объект или компо-

нент, который обладает своими свойствами, методами и событиями. Любой объект вполне автономен, он многое уже умеет делать без ваших указаний и без необходимости в написании дополнительного кода. 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.

Простой пример автономности объектов можно увидеть на примере какого-либо визуального компонента. Например, кнопка обладает множеством свойств: положение, текст, тип кнопки, рисунок и т. д. Вам не нужно заботиться о том, как будет отображаться текст или картинка на кнопке, все это уже реализовано в самом компоненте. Вам не надо заботиться о том, чтобы кнопка правильно визуально реагировала на действия пользователя при наведении фокуса или нажатии, об этом уже позаботились разработчики, которые создавали кнопку.

Но в объектном программировании есть и свои недостатки. В объектах реализовано большое количество действий, которые вы и пользователь сможете производить с ним. Но реально в любой программе мы используем два-три из всех этих свойств. Все остальное — для программы лишний груз, который никому не нужен, но от которого невозможно избавиться. Каждый объект — неделимое целое, из которого просто нельзя вырвать определенные свойства и методы.

А если вспомнить про такую возможность объектного программирования, как наследование, то получается, что у кнопки может быть несколько предков и весь их код, все их возможности также должны быть включены в исполняемый файл. В сложных библиотеках (а библиотека VCL, используемая в Delphi очень сложная) очень много различных объектов, и в большинстве случаев иерархия наследования в библиотеках древовидная. В этом дереве все объекты и компоненты (ветви дерева) происходят от одного объекта (ствол), что немного экономит количество исходного кода и упрощает жизнь, но сэкономить результирующий код все равно не позволяет.

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

- Сжимать готовые программы с помощью компрессоров. Объектный код сжимается в несколько раз, и программа, созданная с использованием VCL, может превратиться из монстра в 300 Кбайт в скромного по размерам "зверя", "весящего" всего 30—50 Кбайт. Главное преимущество состоит в том, что вы не лишаетесь возможностей объектного программирования и можете спокойно забыть про неудобства WinAPI.
- Не использовать визуальную или объектную библиотеку, которая упрощает программирование. В Delphi такой библиотекой является VCL, а в

Visual C++ — это MFC. В этом случае весь код придется набирать вручную и работать только с WinAPI, что достаточно не просто. Программа в таком случае получается очень маленькой и быстрой. Но таким образом вы лишаетесь простоты визуального программирования и можете ощутить все неудобства программирования с помощью чистого WinAPI. Небольшую утилиту написать с использованием только Windows API еще можно, но большую программу — нереально.

- Использовать визуальные библиотеки, которые не сильно влияют на размер результирующего исполняемого файла. Для Delphi есть такая библиотека — это связка MCK-KOL.
- Использовать динамическую компиляцию библиотеки. Коротко рассмотрим, как это помогает нам. В этом случае, в исполняемый файл попадают только логика работы программы и ваши собственные объекты или компоненты. Стандартная библиотека языка не попадает в исполняемый файл, а подключается динамически из dll-файлов, и должна поставляться совместно с программой или быть заранее установленной на компьютере пользователя.

В данной главе мы рассмотрим все эти методы уменьшения размера. Причем вы можете использовать даже сочетания из двух методов, например, написать программу без использования визуальной библиотеки, а потом еще и сжать ее с помощью специализированного компрессора.

1.2. Сжатие исполняемых файлов

Самый простой способ уменьшить размер приложения — использование программы для сжатия файлов. Лично я очень люблю ASPack, которую вы можете найти на компакт-диске в каталоге Программы (файл установки называется ASPack.exe). Она прекрасно сжимает исполняемые exe-файлы и динамические dll-библиотеки.

Я не буду подробно описывать процесс установки ASPack, потому что там абсолютно нет ничего сложного. Только одно нажатие на кнопке **Next**, и все готово! Теперь запустите установленную программу, и вы увидите окно, изображенное на рис. 1.1. Главное окно состоит из нескольких вкладок:

- **Open File**;
- **Compress**;
- **Options**;
- **About**;
- **Help**.

На вкладке **Open File** есть только одна кнопка — **Open**. Нажмите ее и выберите файл, который вы хотите сжать. Как только вы выберете файл, программа перейдет на вкладку **Compress** и начнет сжатие (рис. 1.2).

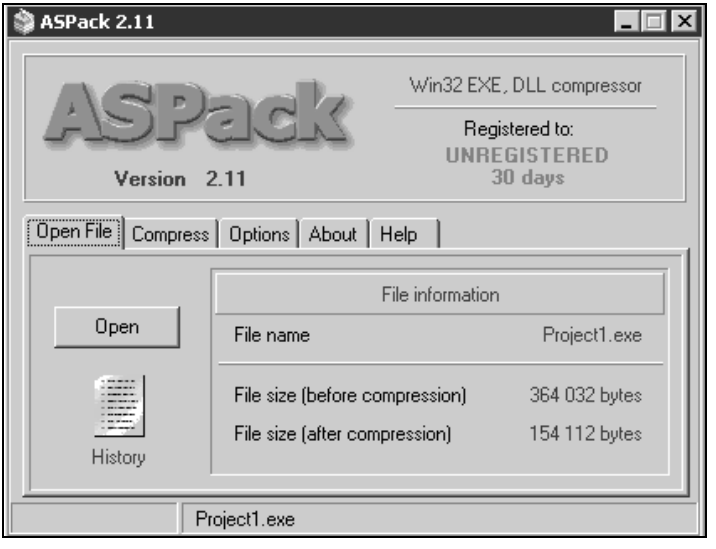


Рис. 1.1. Главное окно ASPack

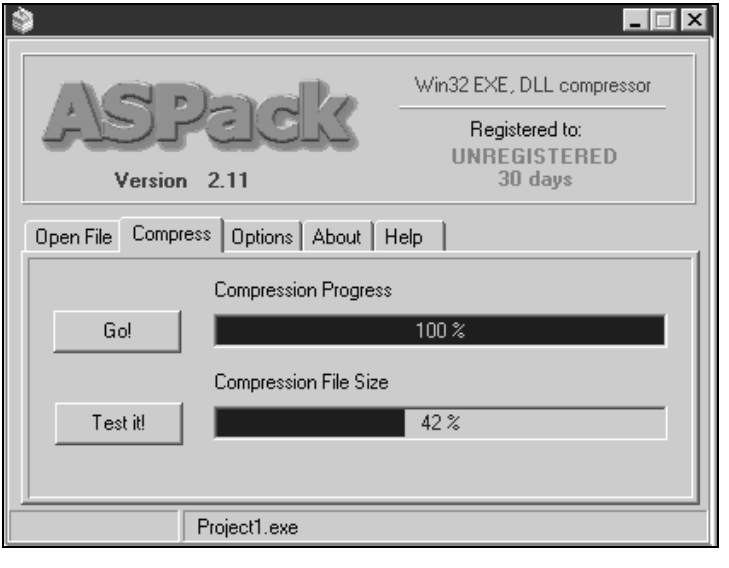


Рис. 1.2. Сжатие файла

Сжатый файл сразу перезаписывает существующий, а старая несжатая версия сохраняется на всякий случай под тем же именем, но с расширением bak. Настроек у ASPack не так уж много (рис. 1.3), и с ними вы сможете разобраться без моей подсказки. Лучше я расскажу вам, как это работает.

Рис. 1.3. Настройки ASPack

Давайте разберемся, как работает сжатие. Сначала весь код программы сжимается архиватором. Если вы думаете, что он какой-то "навороченный", то сильно ошибаетесь. Для сжатия используется обычный архиватор, только оптимизированный для сжатия двоичного кода. После этого, в конец сжатого кода добавляется код разархиватора, который будет программу распаковывать обратно. И в самом конце ASPack изменяет заголовок исполняемого файла так, чтобы при старте сначала запускался распаковщик.

Теперь, когда вы запустите сжатую программу, сначала заработает распаковщик, который "развернет" бинарный код программы и аккуратно разместит 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 окажутся неощутимыми (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 и только на этапе запуска исполняемого файла.

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

Еще один аргумент в пользу применения сжатия — заархивированный код труднее взломать, потому что не каждый дизассемблер сможет прочитать упакованные команды. Так что, помимо уменьшения размера вы получаете защиту, способную отпугнуть большинство взломщиков. Конечно же, профессионала не отпугнешь даже этим, но взломщик средней руки не будет мучиться со сжатым двоичным кодом.

Еще одна утилита для сжатия файлов, получившая большую популярность — UPX. Ее преимуществом является то, что она абсолютно бесплатна и даже распространяется в исходных кодах. Скачать ее можно здесь: **http://upx.sourceforge.net**. Но тут есть одна проблема — утилита выполняется из командной строки, а в наши времена, когда миром правят курсор мыши и простые визуальные окна, люди отвыкли от черного экрана с курсором.

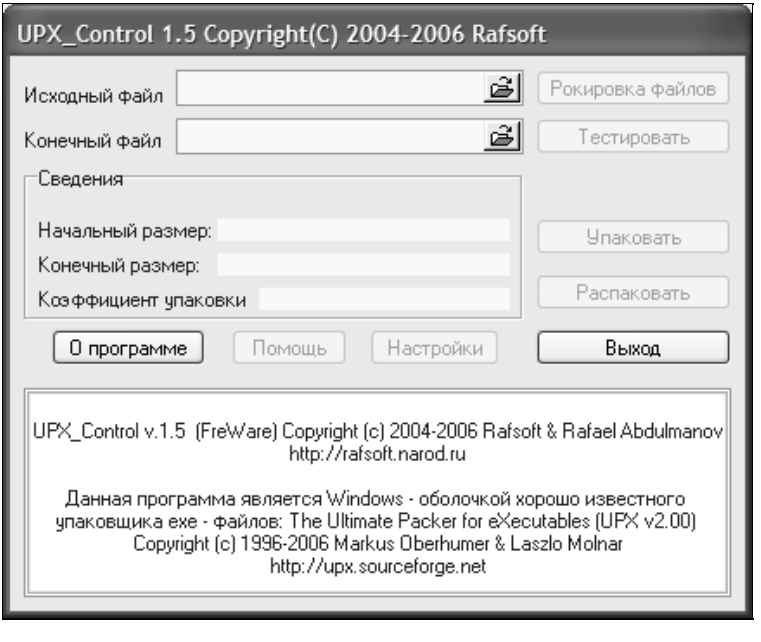


Рис. 1.4. Главное окно программы UPX_Control

Для тех, кто любит визуальность и понятность, могу посоветовать небольшую утилиту UPX_Control, которую можно скачать здесь: **http://rafsoft.narod.ru**. Главное окно еще проще, чем у UPX, а возможности не хуже (рис. 1.4).

В этой программе нужно выбрать исходный файл — исполняемый файл, который необходимо сжать, и нажать кнопку **Упаковать**.

Существует множество программ для сжатия исполняемых файлов. Они отличаются алгоритмом сжатия и возможностями. Некоторые программы позволяют еще шифровать код и предоставлять механизмы, предотвращающие взлом утилит, но это уже необходимо платным программам, а эту тему мы не рассматриваем. К тому же, для большинства универсальных утилит защиты уже давно есть универсальные программы взлома, так что платить за такую защиту деньги — смысла нет, ведь она не эффективна.

1.3. Без окон, без дверей...

Если вы хотите создать программу действительно маленького размера, то необходимо забыть обо всех удобствах. Вы не сможете подключать визуальные формы или другие удобные модули, написанные фирмой Borland для упрощения жизни программиста. Только API-функции самой Windows — и ничего больше.

Для того чтобы создать маленькую программу в Delphi, нужно создать новый проект (по умолчанию система Delphi при открытии сама создаст новый файл проекта, но вы всегда можете самостоятельно создать новое приложение, выбрав в меню **File | New | Application**) и зайти в менеджер проектов (меню **View | Project Manager**). Здесь нужно удалить все модули и формы (пункт **Unit*N***, он выделен на рис. 1.5), чтобы остался только файл самого проекта (по умолчанию его имя Project1.exe). Никаких модулей в проекте не должно быть.

Теперь нужно щелкнуть правой кнопкой мыши на имени проекта и выбрать из появившегося контекстного меню пункт **View Source** или в главном меню **Project** выбрать пункт **View Source**. В редакторе кода откроется файл проекта Project1.dpr. Если вы уже удалили все модули, то его содержимое должно быть таким:

```
program Project1;

uses
  Forms;

{$R *.res}
```

```
begin
  Application.Initialize;
  Application.Run;
end.
```

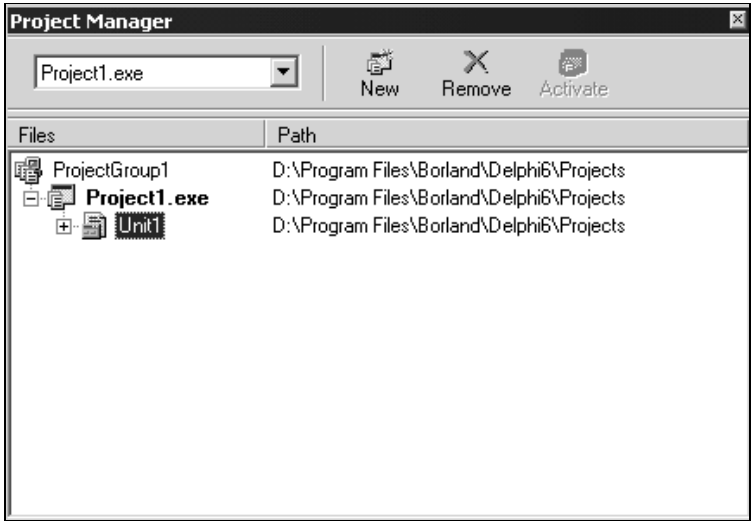


Рис. 1.5. Project Manager

Теперь можно скомпилировать абсолютно пустой проект. Для этого надо выбрать в меню **Project** пункт **Compile Project** или нажать комбинацию клавиш <Ctrl>+<F9>. После компиляции следует выбрать в меню **Project** команду **Information for Project1**. Появится окно с информацией о проекте. Окно, которое появилось передо мной, вы можете увидеть на рис. 1.6.

В правой части окна должны быть описаны используемые пакеты. Так как вы все удалили, значит, там должна красоваться надпись **(None)**. А вот с левой стороны должна быть описана информация о скомпилированном коде. Самая последняя строка показывает размер файла, и у меня он равен 370 688 байтов. Ничего себе "пустая программа"!!! Мы же ничего еще не написали. Откуда же тогда такой большой код?

Давайте разберемся, что осталось в нашем проекте, чтобы обрезать все то, что еще не обрезано. Сразу обратите внимание, что в разделе `uses` подключен модуль `Forms`. Это объектный модуль, написанный "дядей Борландом", а значит, его использовать нельзя, потому что именно он увеличивает размер нашей программы. Между командами `begin` и `end` используется объект

`Application`. Этот объект тоже указывать не надо, потому что он использует объектные возможности и не заботится о "фигуре" программы.

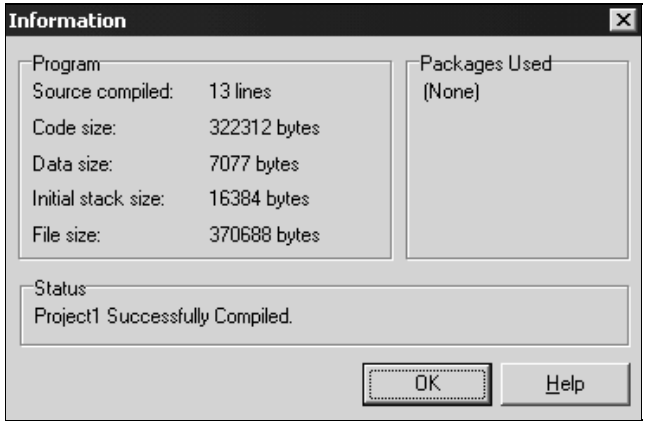


Рис. 1.6. Окно информации о проекте

Большой объем, который появляется даже у пустой программы, как раз и связан с объектом `Application`, который объявлен в модуле `Forms`. Хотя мы использовали только два метода — `Initialize` и `Run`, — при компиляции в exe-файл попадает весь объект `TApplication`, а он состоит из сотен, а может и тысяч строчек кода. Помимо этого, вместе с `TApplication` в исполняемый файл потянется целая куча и еще маленький вагончик дополнительного мусора, который реально в данном проекте не используется.

Чтобы избавиться от накладных расходов, нужно заменить модуль `Forms` на `Windows`, который описывает только функции Windows API и не связан с объектами Delphi. Его подключение является обязательным, иначе вы не сможете вызвать ни одной функции из набора WinAPI. А между `begin` и `end` вообще все можно удалить. В итоге, самый минимальный (с учетом использования модуля `Windows`) код программы будет выглядеть так:

```
program Project1;

uses Windows;

begin

end.
```

Снова откомпилируйте проект. Откройте окно информации и посмотрите на размер получившегося файла. У меня получилось 8192 байта (рис. 1.7). Вот это уже по-человечески. Стоило только отказаться от библиотеки VCL, как размер файла сократился в несколько раз. Есть еще приемы, которые позволят сократить размер исполняемого файла до 6 Кбайт или даже менее, но это уже не важно. Я не вижу смысла мучаться из-за 2—4 Кбайт, которые не сыграют большой роли.

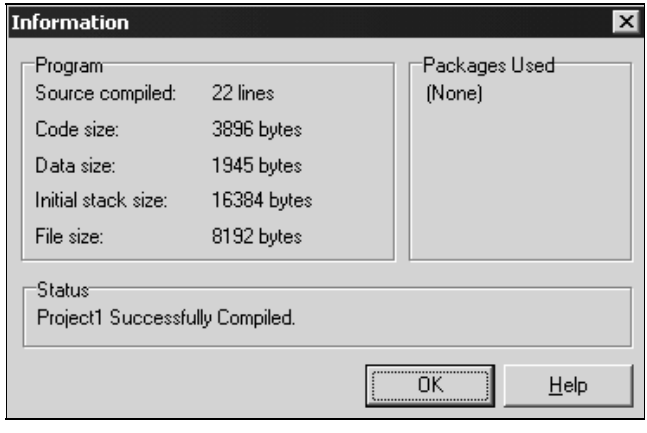


Рис. 1.7. Информации о проекте после удаления ненужного кода

Заготовка минимальной программы с использованием WinAPI готова. Теперь вы можете смело добавлять свой код. Мне нужно только объяснить вам, какие модули можно подключать к своему проекту в раздел `uses`, а какие — не стоит, дабы "жирность" исполняемого файла резко не увеличилась. Тут все очень просто и не займет много времени.

Если при установке Delphi вы не отключали копирование исходных кодов библиотек, то перейдите в каталог, куда вы установили Delphi. Здесь перейдите в папку Source, затем в Rtl и, наконец, в Win. Если вы отключили копирование исходников, то вставьте компакт-диск с Delphi и поищите эти каталоги там. В них расположены исходные коды модулей, в которых описаны все API-функции Windows. Именно эти модули вы можете подключать к своим проектам, если хотите получить маленький код. Если вы подключите что-то другое, то я уже не гарантирую минимум размера вашей программы (хотя, есть и исключения). Самое опасное — это подключение модулей из каталога Delphi7\Source\Vcl\.

Сразу же рассмотрим пример. Если вы хотите, чтобы в вашей программе были возможности работы с сетью, то вам нужно подключить к нему биб-