


# НЕСТАНДАРТНЫЕ ПРИЕМЫ ПРОГРАММИРОВАНИЯ НА DELPHI



РЕЗИДЕНТНЫЕ  
ПРОГРАММЫ В WINDOWS

ИНСТАЛЛЯЦИЯ  
И ДЕЙНСТАЛЛЯЦИЯ  
ПРОГРАММ

ПОИСК В ДОКУМЕНТАХ

РАБОТА С ГРАФИКОЙ  
В WINDOWS

ЛЮБИТЕЛЬСКАЯ  
КРИПТОГРАФИЯ

РАБОТА С COM-  
И USB-ПОРТАМИ

**PRO**

ПРОФЕССИОНАЛЬНОЕ  
ПРОГРАММИРОВАНИЕ

**Юрий Ревич**

**НЕСТАНДАРТНЫЕ ПРИЕМЫ  
ПРОГРАММИРОВАНИЯ НА  
DELPHI**

Санкт-Петербург  
«БХВ-Петербург»  
2005

УДК 681.3.06  
ББК 32.973.26-018.2  
P32

**Ревич Ю. В.**

P32      Нестандартные приемы программирования на Delphi. — СПб.: БХВ-Петербург, 2005. — 560 с.: ил.

ISBN 5-94157-686-2

Книга призвана помочь программистам разрабатывать полноценные, профессиональные Windows-приложения в Delphi. Показано, как предотвращать повторный запуск приложения, работать с нестандартными окнами, перехватывать нажатие клавиш, создавать резидентные программы в Windows, а также инсталляторы и деинсталляторы программ, осуществлять поиск в документах, работать с COM- и USB-портами, шифровать текст и многое другое. Рассмотрены примеры решения этих и многих других проблем, которые встают при создании программы, ориентированной на долговременное использование и распространение. Приведены приемы работы с Windows API. Изложение ведется на примерах поэтапного создания реально работающих практических приложений. Компакт-диск содержит исходные тексты разобранных в книге примеров.

*Для программистов*

УДК 681.3.06  
ББК 32.973.26-018.2

**Группа подготовки издания:**

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Екатерина Капалыгина</i>
Компьютерная верстка	<i>Ольга Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Игоря Цырульникова</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 21.09.05.

Формат 70×100<sup>1</sup>/<sub>16</sub>. Печать офсетная. Усл. печ. л. 45,16.

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 5-94157-686-2

© Ревич Ю. В., 2005

© Оформление, издательство "БХВ-Петербург", 2005

# Оглавление

## Введение

<i>О чем и для кого написана эта книга</i> .....	9
Зачем все это?.....	10
Что можно найти в книге?.....	12
Знания и умения.....	15
Кто такие хакеры?.....	16
Как пользоваться книгой.....	18

## Глава 1. Ликбез

<i>Некоторые сведения о программировании, Windows и Delphi</i> .....	21
О Delphi и Windows.....	23
О пользовательских интерфейсах компьютерных программ.....	28
Страна советов.....	33
Совет 1 — о справке.....	34
Совет 2 — о комментариях и именах переменных.....	34
Совет 3 — об исключениях.....	35
Совет 4 — о функциональности.....	36
Совет 5 — об интерфейсе.....	37
Совет 6 — о пользовательских установках.....	40
Совет 7 — об украшениях.....	41
Совет 8 — об автоматизации.....	41
Немного о стилях программирования.....	43

## Глава 2. Начинаем работу

<i>Создаем типичное приложение</i> .....	47
Начало.....	49
Компоненты.....	49
Свойства.....	52
Меню, таймер и диалог.....	55
Открытие файла.....	56
Перелистывание.....	60

### Глава 3. Окна настезь

#### *Нестандартное закрытие и восстановление окна программы.*

<i>Иконка в Tray Bar</i> .....	<b>65</b>
Сворачивание приложения в Tray Bar при потере фокуса .....	66
Сворачивание приложения в Tray Bar вместо закрытия .....	71
Сворачивание приложения в Tray Bar вместо минимизации .....	74

### Глава 4. Погрузочно-разгрузочные работы

#### *Предотвращение повторного запуска и загрузка с заставкой* ..... **77** |

Предотвращение повторного запуска приложения .....	77
Демонстрация заставки .....	82
Сворачивание в Tray Bar при запуске .....	85

### Глава 5. Чертик из табакерки

#### *Как установить и использовать горячую клавишу* ..... **89** |

Горячая клавиша с вызовом всплывающего меню .....	89
Простая программа в виде иконки — отладочный пример .....	91
Резидентная программа для исправления текста в неправильной раскладке .....	98
Заготовка .....	98
Попытка первая — в лоб .....	100
Вариант второй — посложнее .....	101
Вариант третий — ура! .....	102

### Глава 6. Давим на клавишу

#### *Некоторые особенности работы с клавиатурой.*

#### *Клавиатурный шпион и использование hook* ..... **109** |

Как все это устроено .....	110
Клавиатурный шпион .....	116

### Глава 7. Язык мой — враг мой

#### *Резидентный переключатель раскладки* ..... **125** |

Самый простой переключатель раскладки .....	127
Переключатель с заменой системной иконки — промежуточный вариант .....	134
Переключатель с установками .....	141

### Глава 8. Unicode и другие звери

#### *Как работать с документами в различных кодировках* ..... **153** |

О кодировках .....	154
Unicode .....	159
Unicode и Win32 .....	160
Программа преобразования Unicode в чистый текст .....	163
Преобразование "вручную" .....	164
Преобразование через WideString .....	168

Проблема автоматического переключения раскладки в RichEdit .....	170
Автоматическое определение кодировки текстовых файлов .....	174
Форматы в буфере обмена (попытка доработки перекодировщика) .....	189

## **Глава 9. Vis-a-vis**

<i>Организация диалогов, операции "один обработчик — много действий", передача фокуса ввода и другие хитрости</i> .....	<b>193</b>
Особенности работы с клавиатурой в Delphi .....	193
Диалог типа MessageBox .....	194
Диалог для установки таймера в SlideShow .....	198
Диалог с установкой нескольких параметров и сохранение установок через INI-файлы .....	202

## **Глава 10. Графика и Windows**

<i>Приемы отображения и преобразования растровых изображений</i> .....	<b>211</b>
Растровые изображения в Windows .....	213
BMP .....	221
Иконки .....	222
Преобразование BitMap в Icon .....	225
Приложение-термометр с иконкой в Tray .....	240
Термометр .....	240
Приложение .....	243

## **Глава 11. Возобновляемые ресурсы**

<i>Как работать с ресурсами исполняемого файла</i> .....	<b>257</b>
Наглядная агитация .....	260
Заставка и номер версии в SlideShow .....	264
Номер версии в приложении без формы .....	269
Произвольные ресурсы .....	270

## **Глава 12. Бабушка в окошке**

<i>Нестандартные окна</i> .....	<b>273</b>
Красивая заставка в SlideShow .....	276
Прозрачная форма и окно flystyle .....	278

## **Глава 13. Приставание с намеком**

<i>Прокрутка колесиком, режим Drag&amp;Drop, работа с ProgressBar и другие мелочи</i> .....	<b>283</b>
Прокрутка в компоненте ScrollBox .....	284
Полный Drag&Drop .....	286
Программа для поиска файлов .....	287
О работе с индикаторами длительности процесса .....	296

## Глава 14. Читать умеете?

<i>Доработка программы Trase</i> .....	<b>297</b>
Составление списка вложенных папок .....	299
Поиск заданной строки .....	303
Полируем почти до блеска .....	310
Запуск файлов из приложения .....	315
Оптимизация чтения через memory mapped files .....	319
Настройки .....	325

## Глава 15. Вася, посмотри, какая женщина!

<i>Доделиваем SlideShow</i> .....	<b>331</b>
Процедура составления списка файлов с картинками .....	333
Демонстрация картинок по списку .....	341
Музыка без медиаплеера .....	346
Демонстрация "превьюшек" .....	351

## Глава 16. About help

<i>Справка и окно O программе</i> .....	<b>363</b>
Основы основ HTML .....	369
Справка и пункт <i>O программе</i> для Trase .....	375
Справка для переключателя клавиатуры .....	379
Справка в SlideShow .....	382

## Глава 17. Регистрируем и устанавливаем

<i>Как создать инсталлятор и деинсталлятор самостоятельно</i> .....	<b>389</b>
---	------------

## Глава 18. Читаем документы Word

<i>Технология OLE Automation</i> .....	<b>405</b>
Работа с Word через объект <i>Word Basic</i> .....	408
Работа с Word через объект <i>VBA</i> .....	411
Доработка программы Trase .....	415

## Глава 19. Любительская криптография

<i>Приемы простейшего шифрования и стеганографии</i> .....	<b>421</b>
Операция XOR и простейшее шифрование файлов .....	425
Стеганография на коленке .....	430

## Глава 20. Последовательные интерфейсы COM и USB

<i>И немного о программах реального времени под Windows</i> .....	<b>441</b>
Передача данных через COM-порт .....	442
О программах реального времени .....	443
Прием и передача одного или нескольких байтов .....	448
Прием и передача в реальном времени .....	459

Прием и передача данных с помощью параллельного потока.....	460
Прием и передача данных с помощью компонента <i>AsyncFree</i> .....	469
Программа для чтения данных с GPS-навигатора.....	473
Эмуляция COM-порта через шину USB.....	480

## **Глава 21. Массивы и память**

<b>Работа с большими массивами информации.....</b>	<b>483</b>
Различные способы организации динамических массивов.....	483
Строка типа <i>PChar</i> .....	484
На каждую хитрую гайку... или нетипизированные указатели, как способ организации массивов.....	485
Динамические массивы, строки и <i>TMemoryStream</i> .....	490
Произвольный доступ к большим массивам данных.....	493

## **Приложение 1. О системах счисления..... 501**

Позиционные системы.....	502
Двоичная система.....	505
Шестнадцатеричная система.....	506
Представление чисел в формате BCD.....	509
Модуль <i>Arithm</i> .....	510

## **Приложение 2. Виртуальные и скан-коды для 101/104-кнопочной клавиатуры..... 513**

## **Приложение 3. Коды символов..... 519**

## **Приложение 4. Последовательные порты компьютера COM и USB..... 525**

Принципы передачи информации по интерфейсу RS-232.....	525
Установка линии RTS в DOS и Windows.....	531
Приемы программирования UART в микроконтроллерах на примере AVR.....	533
Преобразователи уровня UART/RS-232.....	536
Схема для преобразования USB/RS-232.....	539

## **Приложение 5. Описание компакт-диска..... 543**

## **Литература..... 547**

## **Предметный указатель..... 551**



# Введение

## О чем и для кого написана эта книга

Я вообще всю жизнь полагал, что ничего хорошего из исполнения чаяний неких гипотетических потребителей, "народа", никогда не выходило. Писать (стихи, романы, пьесы, сценарии и... программы) надо исключительно для себя. Ну, и для близкого круга друзей.

*Е. Козловский, "Ниоткуда с любовью"*

По классификации ученого-химика А. Шкроба, создателя хорошего сайта о науке под названием "VivosVoco", программисты делятся на любителей, дилетантов и профессионалов<sup>1</sup>. Любители пишут программы для развлечения, дилетанты пишут программы по необходимости, профессионалы пишут программы для заработка. Вероятно (кто бы провел такое исследование?), любителей и дилетантов больше, чем профессионалов. Мало того, с распространением Интернета и появлением онлайн-сообществ грани между любителями, дилетантами и профессионалами при разделении их по признаку цели все больше и больше стираются — к какой категории, например, отнести добровольных членов сообщества создателей Linux? Сейчас самодеятельное (объединим таким названием область деятельности любителей и дилетантов) программирование у периодических компьютерных изданий несколько не в моде — просто в силу того, что значительную часть ниши, которую раньше занимали программы непрофессионалов (точнее, программирующих специалистов), ныне занимают универсальные фирменные продукты. Однако, как заявляют авторы одного старого пособия по программированию: *"...прикладные программы, созданные программирующим профессионалом (т. е. "дилетантом" в нашей классификации — Ю. Р.), с точки зрения профессионального программиста зачастую выглядят неуклюжими и неизящными. Но зато они обладают одним общим достоинством — они действи-*

---

<sup>1</sup> А. Шкроб. Я не любитель, я другой... — Компьютерра, № 24—25, 1998 (<http://www.computerra.ru/offline/1998/252/1439/>).

*тельно работают...*"<sup>2</sup>. Дополним данную мысль — это происходит потому, что у дилетантов нет выхода: они пишут программы для себя, и плохо работающие им просто не нужны.

Но согласно хорошему определению, услышанному автором этих строк от одного профессионального фотографа, профессионал отличается от любителя тем, что любитель всегда ищет ответ на вопрос "как", а профессионал — "зачем". Любый любитель или дилетант в конце концов доходит до той стадии, когда ему позарез требуются некоторые функции, которые Delphi (а большинство непрофессионалов использует именно Delphi) сама по себе дать либо не может, либо их осуществление не описано в обычных учебниках и пособиях. Причем среди таких функций есть очень распространенные и необходимые. Вот таким любителям и дилетантам и адресована эта книга. Не ждите от нее последовательного изложения основ объектно-ориентированного программирования (ООП) или построения Windows API. Подобно тому, как можно грамотно писать по-русски, не понимая разницы между существительным и сказуемым, создавать вполне работоспособные программы можно без глубокого знания ООП. Правда, как и в случае грамотности, ваши умения будут ущербными в том смысле, что выйти за рамки конкретных образцов вам будет сложновато, но на основе изложенного в этой книге материала вполне можно научиться делать программы не хуже фирменных — ну, а если вас программирование интересует, как самостоятельный предмет, то для этого нужно читать совсем другие пособия и, как правило, не на русском языке.

## Зачем все это?

Индия планирует довести экспорт программного обеспечения к 2008 году до 50 миллиардов долларов — почти в два раза больше объема российского экспорта нефти. И, хотя официальные данные по экспорту ПО из России отличаются от этой цифры примерно на два порядка (полмиллиарда в 2004 году, по официальным данным), на самом деле есть основания полагать, что официальная статистика врет — некоторое представление о реальности может дать тот факт, что около 10% shareware в мире делается в России и Украине. К тому же индусам завидовать вообще не очень хочется: они не программисты, а кодеры, рабочие, которые лишь кладут кирпичи в здание, возводимое другими. Самостоятельно в Индии не создано, вероятно, ни одной хоть сколько-нибудь известной программы, мне, по крайней мере, о таких программах слышать не приходилось.

---

<sup>2</sup> Сташин В. В. и др. Проектирование цифровых устройств на однокристалльных микроконтроллерах. — М.: Энергоатомиздат, 1990.

"Большое" программирование — индустрия, и без вот таких кодеров-пролетариев обойтись не в состоянии. Но как существование индустрии звукозаписи не отменяет музыкальное творчество, так и производство коммерческого программного обеспечения не противоречит существованию программиста-творца. На западе таких обычно, не особенно разбираясь, именуют "хакерами", но как мы увидим позже, это не совсем точное определение. Но дело не в терминах — такой программист вполне может добиться успеха и признания не будучи коммерсантом: свидетелем тому множество. Главному разработчику нашумевшего браузера Firefox Блейку Россу было всего 19 лет, когда версию 1.0 этого достойного конкурента Internet Explorer скачали за первый же месяц 5 миллионов человек. Большинство вошедших в обиход программных новинок последнего времени сделаны молодыми людьми в том же возрасте или чуть постарше, причем отнюдь не в рамках плановых разработок крупных софтверных компаний. Файлообменные сети Шона Фэннинга, сервис LiveJournal Брэда Фитцпатрика, проигрыватель WinAmp Джастина Франкеля, поисковая система Google Сергея Бриана и Ларри Пейджа — список можно продолжать и продолжать. Придется сделать вывод, что время "компьютерных гениев" и "гаражных стартапов" (start up) отнюдь не прошло. Кстати, один из самых успешных отечественных программных продуктов — распознаватель текстов FineReader — также был создан под руководством совсем молодого тогда физика Давида Яна. Кто знает, может ваше имя недолгое время спустя будет звучать в этом ряду?

Но занятие под названием "программирование" — отнюдь не прерогатива только молодого поколения, с пеленок привыкшего держаться за мышку. Существует достаточное количество профессий, в которых молодые имеют преимущество благодаря своей высокой активности и стремлению сделать карьеру, и, наоборот, есть области деятельности, куда молодым до поры до времени дорога заказана. Программирование — как раз то занятие, где возраст не имеет никаких преимуществ, ни в ту, ни в другую сторону — было бы желание. Молодежь легко и быстро осваивает новое, зато человек постарше берет свое обстоятельностью и стремлением докопаться до глубин.

Единственное, что нужно заметить по этому поводу — особенности отечественного технического образования и, главное, содержание деятельности советского инженера были таковы, что он привык все делать сам, и даже "доводить напильником" то, что не было доделано разработчиками. Это с одной стороны хорошо — кругозор намного шире, и такой человек в отдельных случаях может создавать много лучше работающие программы, чем обычный среднестатистический западный специалист. Но в подавляющем большинстве случаев стремление сделать все самому от начала до конца, не затрудняя себя изучением того, что уже сделано до тебя, может только навредить — современные ОС чаще всего этого просто не позволят и правильно сделают. А почему — вы узнаете уже из первой главы этой книги.

## Что можно найти в книге?

Предполагается, что читатель уже обладает некоторыми навыками создания приложений в среде Delphi и хочет расширить функциональность своих программ, придать им удобный интерфейс, законченность и профессиональные скоростные качества. Просто поместить на форму компонент Delphi, к примеру, `MainMenu` и написать обработчик щелчка мыши для пункта **Файл | Открыть** может каждый после десятиминутного ознакомления с соответствующим разделом в учебнике. Поэтому мы посвятим одну главу созданию проекта "с нуля", а в дальнейшем сосредоточимся на реализации функций, которые вы часто встречаете в различных программах, но приемы программирования для них в учебниках по Delphi не описаны вообще или вызывают трудности при практической реализации. Некоторые из таких функций могут служить просто для красоты или удобства, другие необходимы, если вы пишете программу не на один раз, а собираетесь ею часто пользоваться и тем паче распространять. Как предотвратить повторный запуск программы? Что такое потоковое чтение файлов и зачем оно нужно? Как правильно написать программу без окна, в виде иконки? Как перехватить нажатие клавиш? Как создать инсталляционный пакет? На подобные простые и не очень простые вопросы мы постараемся ответить в дальнейшем. Причем не удивляйтесь, если вы встретите подробное обсуждение некоей проблемы, которая может показаться совершенно второстепенной, скажем, как убрать с глаз долой текстовый курсор в компонентах-редакторах — хорошо сделанная программа отличается от плохо сделанной в первую очередь подобными мелочами.

В процессе создания программ мы постараемся избежать недозволенных приемов и недокументированных возможностей. В Windows 9x вам в принципе никто не запрещает, например, печатать документ прямым обращением к LPT-порту или использовать прямое обращение к регистрам последовательного порта. Но разработчики из MS не рекомендуют этого делать, и поступают совершенно правильно — с нашей точки зрения, хотя бы потому, что при переходе к семейству NT такие программы окажутся совершенно неработоспособными и их придется заново переписывать. В случае же, если мы все делаем, как рекомендовано инструкцией, мы, по крайней мере, снимаем с себя ответственность за происходящее. Это, конечно, шутка — с точки зрения пользователя, программы должны работать, а кто за это несет ответственность — дело десятое. И мы увидим, что при всех наших стараниях, нам все же не удастся следовать рекомендациям разработчиков в полной мере.

Кроме этого, я постараюсь, чтобы при использовании в своих программах примеров из этой книги вам не потребовалось бы использовать ничего, выходящего за рамки свежееустановленной Delphi. За некоторыми отдельными ис-

ключениями — это касается, во-первых, использования готового компонента AsyncFree для работы с COM-портом (но будет также и подробно рассказано, как все сделать без него), во-вторых, я для сокращения записи буду часто пользоваться собственной процедурой преобразования числа в HEX-форму (также будет показано, как ее можно легко заменить стандартными средствами).

В книге вы нередко встретите изложение личных предпочтений автора и его взглядов на ту или иную проблему, включая некоторые особенности Windows вообще. По мнению автора, в Windows есть много неоправданно сложных и просто ошибочных механизмов, для которых можно и нужно обсуждать целесообразность и границы их применимости. Сам автор отчетливо понимает, что его воззрения не есть истина в последней инстанции, и относится к этому в стиле "это мое мнение, и я его разделяю", но старается предоставить читателю самому судить о том, насколько автор прав в тех или иных случаях. В частности, реконструкция логики разработчиков языковых интерфейсов Windows (см. главу 8) представляет именно такой случай — вполне возможно, что среди профессионалов найдутся люди, которые считают эту систему стройной и логичной, находя в ней положительные стороны, ускользнувшие от автора этих строк. Прошу читателя иметь в виду, что дискутировать с автором в подобных вопросах совершенно не возбраняется.

Следуя за автором в его критике отдельных сторон Windows, следует иметь в виду одно обстоятельство — как и в любом другом деле, голоса критикующих здесь всегда звучат громче. В защите Windows не нуждается — ее доминирующая роль на рынке ПК сама по себе есть надежная защита. Однако справедливости ради следует отметить, что в продуктах Microsoft есть много положительных качеств — это и реализации некоторых функций Windows (например, буфера обмена), очень неплохая и изначально весьма продвинутая система работы с таблицами в Word, удобная работа с макросами в Office (кажется, вообще не имеющая аналогов), и т. п. Большинство же недостатков Windows (исключая откровенные недоработки) проистекает из того факта, что эту систему создавали и продвигали на рынок в условиях, весьма далеких от тепличных, в которых создавалась и продвигалась, например, MacOS. Обеспечить бесперебойную работоспособность ОС на миллионах различных конфигураций "железа" от неизвестных производителей — задача в принципе не решаемая, и, надо сказать, в Microsoft с ней справились не так уж и плохо. Это доказывает тот факт, что Linux при установке создает на порядок больше проблем (пока?), и с этим спорить трудно. Главный недостаток Windows — ее негибкость и ориентация на некоего "среднего" пользователя (так, как его себе представляют в Microsoft) — есть продолжение этих достоинств. Не надо забывать и другое — если бы не Windows, собиравшая и аккумулировавшая в себе все чужие передовые идеи, иногда искажая их до неузнаваемости,

но всегда доводя до состояния, относительно пригодного к использованию даже последними "чайниками", никакой компьютерной революции бы не произошло, компьютер так и остался бы дорогой игрушкой для инженеров и бизнесменов. Мало кто помнит, что правило, согласно которому последняя модель PC всегда стоит \$1500, стало соблюдаться только с началом эпохи Windows. Первый IBM PC (с весьма ограниченными возможностями) был выпущен на рынок по запредельной для 1981 года цене примерно \$3000, а продвинутая модель IBM PC AT (на основе 286-го процессора) в середине 80-х стоила порядка \$5000—7000.

Есть и еще несколько обстоятельств, во многом извиняющих разработчиков из MS. Например, подавляющее большинство современных программ вполне может выполняться на "железе" трех-, пятилетней давности, и уже почти забыто, что еще лет пять-шесть назад в балансе аппаратных и программных средств аппаратные средства решительно отставали — очередные версии ОС требовали последних и достаточно дорогих конфигураций компьютера. Так как никому не хотелось тратить лишние деньги, был период, когда в эксплуатации одновременно находился почти весь парк компьютерного "железа", начиная с клонов IBM PC XT и до первых моделей Pentium MMX. Поэтому гораздо более остро стоял вопрос совместимости софта — пересаживаясь на новые компьютеры, пользователь хотел более комфортабельной работы, но вовсе не желал расставаться с любимыми Norton Commander и Lexicon (автор своими глазами наблюдал этот мучительный процесс). И в этом вопросе корпорации Microsoft, которая не стала ломать ситуацию "через колено", можно сказать только большое-большое спасибо. Но эти требования совместимости всего и вся имели и другую сторону — первые версии Windows оказались весьма далеки от идеала, который обычно обозначается словами "многозадачная операционная система".

Некоторые подробности о Windows и Delphi будут излагаться отдельно, в виде "заметок на полях", но хочется еще раз подчеркнуть, что книга эта ни в коем случае не призвана заменить учебник программирования, справку по Windows API или по работе в Delphi. Вам как минимум потребуется учебник-справочник по языку Object Pascal, приемам работы в Delphi IDE (Integrated Development Environment, интегрированная среда разработки) и основным компонентам. Некоторые подобные издания, а также ссылки на отдельные удачные, по субъективному мнению автора, интернет-ресурсы приведены в списке литературы. Отмечу, что и печатных пособий, и тем более сайтов, посвященных Delphi, так много, что никакой подобный список не может претендовать на полноту, так что эта задача даже и не ставилась. В качестве основного источника сведений по Windows API вполне достаточно официального сайта Microsoft MSDN [14] или даже встроенной в Delphi справки (файл win32.hlp, который располагается в папке C:\Program Files\Common Files

\Borland Shared\MSHelp\); может быть вызван и через меню **Пуск | Программы | Borland Delphi 7 | Help | MS SDK Help files | Win32 Programmer's Reference** — правда, в некоторых отношениях эта справка устарела). Есть и множество русскоязычных ресурсов, той или иной степени полноты, представляющих собой простой или комментированный перевод материалов с сайта Microsoft MSDN. Наиболее полные и систематизированные переводы принадлежат Владимиру Сокоикову [16,18], другие вы без труда отыщете сами, просто набрав в Яндексе название той или иной функции API.

## Знания и умения

Есть знания и умения. Обучение чтением теоретических курсов — приобретение знаний, как таковых — изобретение нового времени и составляет основное содержание классной системы образования, возникшей в период Реформации. До этого обучение происходило исключительно на личных примерах и в процессе реальной работы. И сейчас в тех областях, где закончивший некий курс обучения должен выйти в мир, обладая именно практическими умениями (как, например, в медицине), основное содержание обучения составляет практика. А есть области, в которых теоретических знаний не требуется вообще или нужно настолько мало, что без специального изучения теории можно обойтись — например, вождение автомобиля или, скажем, плотницкие работы. С другой стороны, такие области, как теоретическая физика или математика, наоборот, в значительной степени основаны именно на знаниях (хотя тоже далеко не полностью). Любопытно, что программирование, при том, что оно опирается на самые абстрактные области человеческого знания, по сути своей — типичное умение, ремесло. Можно спокойно начинать программировать, обладая математическим багажом в пределах средней школы — необходимые знания приобретаются в процессе отработки умений. А умения приобретаются только на практике.

Автора этой книги часто спрашивали: "Как можно научиться работать на персональном компьютере?" Ответ на этот вопрос единственный: надо его приобрести. Остальное сделает ваше желание научиться — если оно имеется, конечно. Точно так же и навыки программирования нельзя приобрести при чтении даже самых умных книжек. Включите компьютер, подберите себе кресло поудобнее и начинайте работать. Только если вы хотите стать Настоящим Программистом, не забывайте, что это очень особая профессия. Сама специфика деятельности программиста такова, что здесь не могут удержаться люди, привыкшие к делению жизни на работу с 9 до 17 и развлечения все остальное время. Этой особенности профессии посвящена добрая половина программистского фольклора:

У жены программиста спросили:  
— А как он за тобой ухаживал?  
Жена, после минутного раздумья:  
— Компьютер показал...

Но, чтобы писать вполне работоспособные программы, Настоящим Программистом становиться вовсе необязательно. В этой книге я ориентировался на тех, кто делает программы, предназначенные для практической работы, а не для развлечения, постаравшись сосредоточиться на том, чтобы все "навороты" служили одной цели: сделать программу удобной и приятной для пользователя. Впрочем, грань между серьезным и развлекательным провести трудно. Никому сами по себе не нужные "приколы", несомненно, иногда неплохо помогают иллюстрировать тот или иной прием. Да и одно из основных наших приложений — SlideShow, которое мы будем создавать на протяжении почти всей книги, по своему назначению есть программа в основном развлекательная.

## Кто такие хакеры?

Несколько слов об упомянутом ранее понятии "хакер". В компьютерной прессе не устают подчеркивать, что настоящий хакер — это не преступник, который рыщет по Интернету в поисках способов взломать сайт Пентагона или своровать базу кредитных карт клиентов онлайн-аукциона eBay. Точнее, часть таких преступников-"крякеров" — и любителей и профессионалов, занимающихся промышленным шпионажем — может относиться к хакерам, но уже обратное неверно: далеко не все хакеры, и даже не заметная их часть — преступники. Однако, когда доходит до объяснений, а кто же собственно такой хакер, дело обычно ограничивается следующей характеристикой: это человек, помешанный на компьютерах, желающий досконально знать, как все в них устроено, который отличается виртуозным владением соответствующими инструментами, позволяющими ему выделывать с компьютерами и сетями разные необычные штуки. Между тем, это определение является недостаточным: кроме виртуозного владения компьютером, принадлежность к хакерскому течению предполагает еще и определенный склад ума и особую социальную философию левацкого толка. Развитие хакерской субкультуры связано с Массачусетским технологическим институтом и появлением в 1961 году компьютеров PDP-1. Позднее, с появлением первой сети ARPAnet и, в дальнейшем, Internet (Интернет), хакерство интернационализировалось. Главными практическими результатами деятельности представителей хакерской субкультуры для всего остального мира стало рождение ОС Unix и языка C, а позднее сообщества "свободного софта" и соответствующего мировоз-



зрения. Характерной особенностью Unix (как и ее знаменитого клона — Linux) было то, что она была сделана полностью в рамках частной инициативы, без какого-то заказа — в дальнейшем это стало основным отправным пунктом идеологии "свободного софта" вообще, даже когда подобные проекты делаются в коммерческих целях, дух все равно тот же: "потому что интересно".

Основным в хакерстве является именно вот это либертарианское, отчасти анархистское мировоззрение, совпавшее по времени возникновения с расцветом движения хиппи со всеми сопутствующими атрибутами — цветочками в волосах, драными джинсами и способностью работать когда хочется, а не когда этим занимаются все остальные. Хакеры — это технологические хиппи, те из них, кто обрел, как ему показалось, именно в компьютерах желанную свободу от этого кошмарного мира "власти чистогана". Хакерское движение, как и молодежные бунты 60-х вообще, вне зависимости от практических результатов, сделало очень много хорошего. Прежде всего, в политическом смысле они стали действительно реальной оппозицией консервативной бизнес-верхушке — заняли то место, которое ранее безуспешно пытались занять коммунисты-догматики. У них это получилось потому, что, в отличие от политиков, они не болтали и не устраивали революций, а делали реальные дела ("Не пишите манифесты, пишите код!"), и чуть ли не впервые в истории показали всем, что, оказывается, серьезные вещи можно делать и так — без корпоративных структур, без формального подчинения, на основе добровольности и открытости. Современный компьютерный — и не только компьютерный — мир очень и очень многое у них заимствовал, как в плане методологии ведения разработок, так и в плане конкретных идей.

Эту небольшую политинформацию я привел вот зачем: хакер — совсем не синоним программиста-виртуоза, хакерство — течение социальное, а не профессиональное, поэтому термин этот без нужды примерять к себе не следует. Далеко не все профессиональные программисты и даже взломщики программ называют себя хакерами и являются ими. Вообще большинство из тех, кто профессионально "ломает" программы и средства защиты, являются серьезными учеными-криптографами, обладателями всяких академических званий и степеней, или, по крайней мере, составили себе имя публикациями на соответствующие темы. И взломом они занимаются не просто так, а потому что без их труда невозможно правильно оценить эффективность работы тех или иных алгоритмов. Согласитесь, в этом подходе есть некоторые отличия от очередных "подвигов" взломщиков сайтов, по какой-то странной причине называющих себя "хакерскими группами". Тем более, совсем не каждый, кто считает, что хорошо владеет компьютером, вправе считать себя хакером. Добавлю специально для тех, кто усматривает некую романтику в образе "крякера"-подпольщика: все громкие деяния, типа кражи исходного кода

Windows или остроумно написанного вируса, заразившего очередной миллион компьютеров, относятся к настоящему хакерству, примерно так же, как поведение водителя-лихача к манере вождения шофера-профессионала.

## Как пользоваться книгой

Но вернемся к тому, с чего начали — играть лучше в практически полезные дела. Исходя из этих соображений, книга эта построена так, чтобы каждый пример, по возможности, являлся неким законченным проектом (или его частью), который может пригодиться на практике. Некоторые примеры (но не все, конечно) будут даны в развитии — вы увидите, как приложение постепенно отлаживается, с промежуточными экспериментами по использованию того или иного приема или функции. Все примеры (по главам) имеются на прилагаемом диске в том виде, в котором они описаны в тексте. Эти примеры в идеале нужно изучать следующим образом — вы берете исходный текст примера, такой, какой он есть к началу соответствующего раздела, и повторяете все, что описано в этом разделе по ходу мысли автора. А потом сравниваете то, что получилось, с окончательным вариантом, который имеется на диске. Если варианты совпали — замечательно, если нет — при сравнении легко установить, что именно (и у кого именно) не так. Все примеры тестировались на совместимость с Windows 98 и Windows XP (в последнем случае без установленных сервис-паков). При обнаружении каких-то особенностей в работе под указанными ОС, это оговаривается в тексте. Отразить в оглавлении все имеющиеся в книге приемы возможности, конечно, не было, поэтому в конце книги размещен предметный указатель в форме FAQ, где перечислены по возможности все имеющиеся в книге ответы на вопросы "Как?".

Запускать проекты прямо с диска неудобно, вам придется долго настраивать среду и вы не сможете вносить изменения, поэтому их нужно сначала перенести на жесткий диск вашего компьютера. При переносе, возможно, придется изменить путь к папке с проектом, который указан в исходном DSK-файле. Если вы устанавливали Delphi в режиме по умолчанию и папка Program Files находится на диске C:, то можно ничего не менять — просто скопируйте папку, относящуюся к данной главе, в папку Projects. В этом случае путь будет такой: C:\Program Files\Borland\Delphi7\Projects, и он зафиксирован в DSK-файлах проектов на диске. В противном случае можно вовсе не копировать DSK-файл в новую папку, но при этом вы также потеряете и все настройки. Лично я открываю DSK-файл в Блокноте и вношу исправления через пункт **Заменить | Заменить все**. Перенос можно осуществить и отдельным копированием через пункт **Save as** сначала проекта, потом каждого модуля программы по очереди.

Перед тем как приступить к делу, автор считает себя обязанным выразить благодарности:

- Валерию Васильевичу Фаронову — за его замечательные учебники по Pascal и Delphi;
- Евгению Сергеевичу Голомину, глубоко верующему человеку, за выложенные им для всеобщего ознакомления исходные тексты к прекрасной программе "Опечатка" (<http://come.to/golomin>);
- всем без исключения посетителям форумов по программированию в Рунете — без них эта книга вообще бы не увидела света.

Со всеми вопросами и пожеланиями пишите на **revich@homepc.ru**.

# ГЛАВА 1



## Ликбез

### Некоторые сведения о программировании, Windows и Delphi

Создайте систему, которой сможет пользоваться каждый дурак, и только дурак захочет ею пользоваться.

*Принцип Шоу*

Компьютерная программа выполняет то, что вы приказали ей делать, а не то, что вы бы хотели, чтобы она делала.

*Третий закон Грида*

Современный персональный компьютер — устройство необычайно сложное. Причем рядовой ПК образца 2004 года отличается от IBM PC двадцатилетней давности гораздо больше, чем, к примеру, последняя модификация Ford Focus от легендарного Ford T образца 1913 года. И дело тут не в самой по себе скорости работы, которая возросла примерно на три порядка. Гораздо важнее принципиально возросшая функциональность — ни о какой 3D- и даже обычной многоцветной 2D-графике тогда и речи не шло, голосовые интерфейсы существовали разве что в фантастических романах, а предсказать нечто подобное Интернету, да еще и мобильному, фактически не смог никто. Если продолжить аналогию с первыми автомобилями, то современный ПК в сравнении с IBM PC скорее следует уподобить, если и не реактивному истребителю, то, по крайней мере, пассажирскому лайнеру.

Казалось бы, управлять такой сложной машиной — учиться и учиться. Но ПК не обосновались бы настолько прочно в наших домах и офисах, если бы разработчики не придумали программные средства, сводящие сверхсложные операции к двум-трем щелчкам мыши. Причем на сегодняшний момент сложилась довольно парадоксальная ситуация: разнообразие компьютерного "железа" настолько велико, что *правильно* подобрать комплектующие и *правильно* настроить современный ПК невозможно без определенного багажа специальных знаний. А вот технологии программирования ПК, наоборот, предельно упростились. Правда, это в полной мере справедливо только для

случая создания пользовательских программ — системные программы, разумеется, требуют для своего создания специальных знаний. Показательно, что в популярной книге С. В. Зубкова "Assembler для DOS, Windows и Unix" [13], выдержавшей несколько переизданий, программированию под Windows и Unix посвящено менее 100 страниц из почти 700 — настолько это простое по сути занятие.

### ***Заметки на полях***

Попробую пояснить эту парадоксальную мысль. Для начала хочу подчеркнуть разницу между понятиями "сложный" и "громоздкий" — очень простая по сути программа может быть очень громоздкой и наоборот. Например, расшифровать заголовок файла, содержащего изображение в формате BMP, и воспроизвести на экране содержащуюся в нем картинку — достаточно сложное занятие. Программа, которая это делает из-под DOS, окажется и сложной и достаточно громоздкой, учитывая еще тот факт, что для воспроизведения True Color придется где-то искать и присоединять к программе драйвер имеющейся в наличии видеокарты. Излишне говорить, что в силу большого разнообразия "железа" последняя задача может доставить и программисту и пользователю массу хлопот. А вот в Windows ни один из этих вопросов не стоит вообще — все драйверы уже установлены заранее, а для вывода на экран картинки формата BMP не требуется даже знать, что у нее есть какой-то там заголовок — даже на ассемблере это делается вызовом одной-двух функций. Вместе с тем, если простейшая ассемблерная программа под DOS может состоять из одной-единственной команды процессора, то любая программа под Windows обязана содержать некий минимум команд, так что решение упомянутой ранее задачи отображения картинки по объему кода может даже превысить DOS-вариант. Что делает Windows-программы более громоздкими, но отнюдь не более сложными, потому что этот необходимый минимум повторяется из программы в программу и заново его "изобретать" каждый раз не требуется (см. [13]).

Так получилось потому, что большинство функций в современных операционных системах скрыто за оболочкой, носящей название *пользовательские программные интерфейсы* (Application Programming Interface, API). Собственно процесс программирования сводится к тому, чтобы вовремя и правильно вызвать нужную функцию API. Поэтому такие крайне сложные для "обычного" программирования операции, как, к примеру, вывод на экран полноцветной графики или создание текстового редактора с поддержкой форматирования, шрифтов, операций с буфером обмена и прочих необходимых свойств, сводится к вызову нескольких функций API. Довольно большой прогресс по сравнению с прерываниями DOS, которые программисты в большинстве случаев старались обойти, не видя необходимости в лишних прослойках между программой и BIOS или "железом", не так ли?

Однако на этом разработчики современных систем не остановились. С помощью средств так называемого *визуального программирования* процесс вызова функций API максимально автоматизирован. Вам уже не нужно выполнять рутинную работу по написанию кода программы целиком. Достаточно

перетаскать мышью нужный компонент на форму, и соответствующий код включается в текст программы автоматически. В результате процесс программирования стал воистину творческим занятием: всю "грязную" работу берет на себя компьютер (точнее, выбранный пакет программирования), а вам остается только правильно обработать возникающие в программе события. Специальных знаний об устройстве компьютера и построении ОС здесь требуется не больше, чем для обычного квалифицированного пользователя.

Разумеется, как в любом другом деле, за удобства приходится платить: вы полностью привязаны к существующим программным интерфейсам, которые писали для вас добрые дяди из Microsoft и других фирм-производителей системного софта, и шаг вправо-влево тут означает если и не расстрел, то, во всяком случае, необходимость приобретения *очень специальных* знаний. Однако такой выход за пределы стандартных функций в подавляющем большинстве случаев и не требуется: "добрые дяди" постарались предусмотреть по максимуму все, что требуется среднестатистическому пользователю. Другое дело, что поиск нужной функции и способа ее правильного использования может быть очень непростым занятием — ведь далеко не все можно сделать, перетаскивая компоненты мышью на форму. Именно в этом деле и призвана помочь книга, которую вы держите в руках.

## О Delphi и Windows

Кстати, а почему именно Delphi? В принципе все современные пакеты визуального программирования позволяют делать одно и то же, а если какие-то вещи делать удобнее в одном пакете, а другие — в другом, то никто не мешает использовать их совместно. Скажем, библиотеки VCL (Visual Components Library) от Borland являются общими для Visual C++ и Delphi и написаны в основном на Object Pascal, а Windows API, наоборот, большей частью написаны на C (или C++), что не мешает использовать их в любой среде. Но Delphi, безусловно, является на сегодняшний день наиболее универсальной средой программирования, которая позволяет без лишних сложностей создавать как самые простые пользовательские программы, так и навороченные профессиональные пакеты.

Эта книга написана с расчетом на выполнение примеров в среде Delphi 7.0. Седьмая версия — последний релиз Delphi для платформы Win32, на которой основываются все версии Windows от 95-й до XP. В настоящее время Microsoft переходит на платформу .NET — на ней будет полностью основана Windows Longhorn, которую планируется выпустить на рынок в 2006 году. Частично на .NET переходит уже обновленная 64-разрядная Windows XP, готовящаяся к выходу в 2005 году. Между прочим, базовый язык программирования для платформы .NET под названием C# разрабатывает Андерс

Хейлсберг — человек, которому корпорация Borland во многом обязана самим своим существованием.

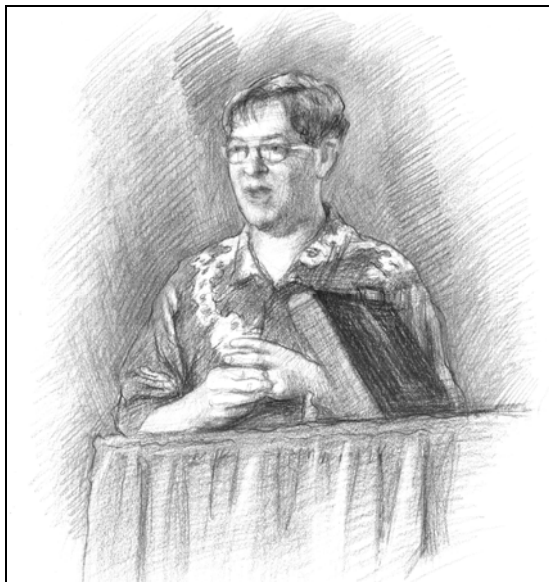


Рис. 1.1. Андерс Хейлсберг (рисунок Александры Дрофиной)

Андерс Хейлсберг (Anders Hejlsberg) — создатель Turbo Pascal и один из главных архитекторов Borland Delphi с момента ее возникновения, человек, разработавший продукты, выведшие корпорацию Borland в ряды ведущих поставщиков программного обеспечения. Также главный архитектор языка C# для платформы Microsoft .NET. О себе Андерс рассказывает так (из интервью журналу "Домашний компьютер", #1, 2004):

*Я родился в 1960 году в Копенгагене, Дания. Свое тихое и милое детство я провел в пригороде Копенгагена. Потом учился на инженера по электротехнике в Техническом университете Дании. В 1987 году переехал в США, а в 1994 — женился. Живу в Сиэтле. С 1983 по 1996 я работал в компании Borland, а теперь в Microsoft. В 1979 я основал компьютерную компанию в Дании под названием PolyData. Это было время, когда персональных компьютеров еще не существовало. Мы продавали компьютерные комплексы и писали для них программное обеспечение. Я написал такие вещи, как ассемблер, дисассемблер, небольшую операционную систему и несколько расширений для Microsoft ROM-Basic. Моим самым первым большим проектом стал компилятор с языка Pascal и редактор, который мог заменить ROM-Basic. После этого я написал еще одну реализацию Pascal для операционной системы CP/M. Она называлась PolyPascal. В 1983 году мы объединились с*

*ребятами, которые только что основали компанию Borland, они лицензировали наш компилятор Pascal, добавили туда свой собственный редактор и назвали все это Turbo Pascal. Я помню, как думал, что они сумасшедшие: эти парни продавали новый продукт по цене 49 долларов 95 центов, в то время как он стоил 500 долларов! Но достаточно быстро выяснилось, что я ошибался — Turbo Pascal стал очень популярным. Мы продали его столько, что в начале было невозможно представить.*

Для того чтобы читатель немного сориентировался, приведем краткую сравнительную характеристику различных сред обработки от Borland, базирующихся на языке Pascal.

В начале всего был Turbo Pascal 1.0, вышедший на рынок 20 ноября 1982 г. Об искусстве Андерса Хейлсберга может говорить тот факт, что интегрированная среда разработки, встроенный редактор и библиотека времени выполнения умещались в файле turbo.com размером 33 280 байт. Правда, эта версия сама могла делать только COM-программы (если кто помнит, то в DOS существовал такой формат исполняемых файлов, отличавшийся от привычного EXE тем, что занимал только один 64-килобайтный сегмент памяти), но зато работала она на медленных ПК того времени очень быстро и давала очень компактный код, благодаря чему сразу вывела компанию Borland в лидеры. В дальнейшем эволюция Turbo Pascal привела к версии 7.0 (1992) — до сих пор популярному средству разработки программ под DOS. Благодаря непревзойденной по удобству среде разработчика (Integrated Development Environment, IDE) и простому для освоения языку, Turbo Pascal стал особенно популярным в непрофессиональной и полупрофессиональной среде. Строго говоря, последняя версия Turbo Pascal, как такового, называлась 6.0, седьмая версия называлась Borland Pascal и включала в себя две разных среды: Borland Pascal for DOS 7.0 (базирующийся на шестой версии с некоторыми существенными доработками) и Borland Pascal for Windows. Если вам удастся достать какое-нибудь пособие по последнему (например, раритетную книгу В. В. Фаронова "Паскаль и Windows"), то вы увидите, что разработка приложений под Windows "вручную" — довольно громоздкое занятие, требующее хорошего знания приемов объектно-ориентированного программирования (ООП). Чтобы облегчить это занятие, в Borland (под руководством все того же Хейлсберга) к 1995 году была разработана Delphi 1.0 — первая визуальная среда программирования, ориентированная на разработку 16-разрядных приложений под Windows 3x.

Начиная же с версии 2.0, Delphi ориентировалась на 32-разрядные версии Windows (95 и выше), использующие платформу Win32. Все использующиеся на практике на момент выхода этой книги версии Windows ориентируются



именно на эту платформу, поэтому в подавляющем большинстве случаев вам вообще не нужно думать о том, какая версия Windows у вас установлена. И все же многие API могут быть специфичными для того или иного семейства ОС либо той или иной версии. Поэтому работу программ, особенно тех, которые предназначены для распространения, следует проверять под различными версиями. Для этого удобно установить на компьютере сразу несколько систем, благо XP это позволяет без лишних сложностей.

На момент написания данной книги уже имеются версии Delphi 8.0 и Delphi 2005, ориентированные на .NET. Для совместимости в комплект поставки Delphi 8.0 входит Delphi 7.1, подобно тому, как в Borland Pascal входил Turbo Pascal, а Delphi 2 комплектовалась версией 1. Разница только в том, что вторая версия Delphi в свое время вышла с опозданием, уже после выхода Windows 95, а восьмая версия отчасти опережает события. Заметим, что наименование "Delphi 8.0" следует признать неудачным — это другой продукт для другой платформы (изменили же когда-то марку Turbo Pascal на Borland Pascal — почему бы и не следовать этому принципу и в дальнейшем?). Поэтому большинство примеров, которые вы встретите в этой книге, в среде Delphi 8.0 (и выше), скорее всего, просто не будут компилироваться — но сами по себе готовые программы будут пригодны еще долгие и долгие годы, в будущей Longhorn декларирована совместимость с Win32 (подобно тому, как версии Win95/98/ME были совместимы с DOS-программами).

Что же касается более ранних версий, то большинство примеров из этой книги совместимы с версиями Delphi, начиная с 4.0, в крайнем случае 6.0. При установке следует позаботиться, чтобы у вас оказались установленными файлы справки по Win32. Эта справка, к сожалению, практически не обновлялась с версии 3.0, так что можно просто скопировать файл win32.hlp в отдельный каталог и пользоваться им автономно, независимо от версии пакета.

Примечание специально для тех, кто только осваивает программирование: на пиратских развалах якобы появилась русифицированная версия Delphi. Ее устанавливать *не следует*, и не только потому, что она неизвестного происхождения, но и потому, что все без исключения пособия и интернет-ресурсы оперируют с английской версией, и в результате вы потратите гораздо больше времени на обратный перевод русских названий пунктов меню и текстов сообщений, чем на то, чтобы один раз выучить английские.

Несколько слов о том, каковы особенности работы программ под теми или иными версиями Windows. Главное отличие семейства 9x (95/98/ME) от семейства NT (NT/2000/XP) заключается в реализации многозадачности. В забытых ныне 16-разрядных версиях Windows 3.x многозадачность называлась кооперативной — фактически весь процесс работы заключается в выполнении последовательного ряда событий Windows, и пока одно событие

не обрабатывается, все остальные вынуждены ждать своей очереди. Таким образом, малейшая ошибка в одном приложении полностью подвешивает всю систему. Все программы, в том числе служебные, видимы друг для друга, модуль, содержащий ошибки обращения к памяти, может легко испортить ее содержимое, принадлежащее другому процессу.

В версиях семейства NT реализована настоящая — вытесняющая — многозадачность (напомним, что NT появилась раньше, чем 95-я). Такая система распределяет процессорное время и память между процессами (process) и потоками (thread)<sup>1</sup>, как будто бы каждый из них выполняется в отдельном компьютере, поэтому "зависшая" программа теоретически не оказывает влияния на все остальные. Но за это приходится платить — в первую очередь тем, что прямой доступ к "железу" практически исключен, а если вы попытаетесь этот запрет обойти тем или иным способом, то такая попытка ничем хорошим ни для вашей программы, ни для системы не кончится. То есть требования к качеству программного кода сильно повышаются и можно смело утверждать, что все случаи "обрушения" Windows XP связаны именно с неправильно (как говорят программисты — "криво") написанными приложениями.

А в версиях 9x в целях совместимости с 16-разрядными приложениями (включая DOS-программы) реализовано что-то вроде промежуточного варианта. Все 32-разрядные прикладные программы выполняются в соответствии с моделью вытесняющей многозадачности. На уровне таких приложений формально все работает независимо, как и NT. Однако в области модели памяти (в основном из-за требований совместимости с 16-разрядными приложениями DOS и Win3x) есть серьезные дыры. Как и в 3x, ничто не может помешать программе, содержащей ошибку обращения к памяти, произвести запись в адреса, принадлежащие системным DLL, и вызвать крах всей системы.

В Windows 9x ситуация, когда выполняющийся поток все никак не освобождает ресурсы компьютера, случается не так уж редко. Даже формально правильное приложение может быть источником неприятностей: так, многим знакома картинка, когда ресурсоемкая программа (архиватор, файловый менеджер при копировании больших объемов информации, или, например, поисковая программа) настолько "оккупирует" все ресурсы компьютера, что даже картинка рабочего окна не успевает прорисовываться. Один из механизмов таких задержек — но не единственный, конечно — связан с тем, что многие подобные процессы выполняются через 16-разрядные функции, а

---

<sup>1</sup> Сами по себе процессы ничего не делают, а лишь предоставляют ресурсы и контекст для выполнения потоков. Любой процесс должен иметь, по крайней мере, один поток, который и выполняет код процесса. Причем планирование переключения задач в Windows осуществляется на уровне потоков, а не процессов.

только один поток может обращаться к 16-разрядным DLL в каждый момент времени, потенциально затормаживая другие процессы, которым нужен к ним доступ. Другой — с неправильным распределением приоритетов одновременно выполняющихся потоков. Борьба с такими неприятностями можно только внутри самой программы, периодически искусственно вызывая обработчик системных сообщений, но самое правильное — по возможности не использовать потенциально "тормозных" процедур вообще.

## О пользовательских интерфейсах компьютерных программ

Крупнейший в мире специалист по компьютерным интерфейсам Джеф Раскин (Jef Raskin, 1943—2005 гг.) говорил примерно так (за точность цитаты не ручаюсь): *"Пользователь обычно не понимает, насколько ему неудобно, механизм обратной связи от потребителя к производителю не работает, и единственное, что может заставить проектировщика создавать по-настоящему хороший интерфейс — это его совесть"*. Я не могу тут не остановиться на истории пользовательских интерфейсов, потому что непонимание, насколько неудобно работать с современными программами, к сожалению, характерно для программистов в еще большей степени, чем для пользователей. Невозможно делать удобные программы, если вы не очень хорошо представляете, с чем вам придется иметь дело.

Точкой роста почти всех идей, реализованных впоследствии в так называемом графическом интерфейсе пользователя (Graphic User Interface, GUI), стал Xerox Palo Alto Research Center (Xerox PARC), возникший на рубеже 60—70 годов. Любопытно, что одним из его основателей, а с сентября 1970 года — и руководителей, стал Боб Тейлор (Robert W. Taylor), который пришел в Xerox из DARPA, где в 1966—69 гг. возглавлял проект ARPAnet, предшественника Интернета. Одной из идей, родившихся в этом центре, была так называемая парадигма WIMP (Windows, Icons, Menus, Point-and-Click — "окна, пиктограммы, меню, укажи и щелкни"), которая переросла позже в концепцию GUI и продолжает эксплуатироваться в настоящее время. Эти разработки связывают с именем Алана Кея (Alan Key), также известного, как автора SmallTalk (первого объектно-ориентированного языка программирования). В 1975 году в Xerox была начата разработка нового проекта, закончившегося в апреле 1981 года представлением системы Xerox 8010 Professional Workstation, более известной под торговой маркой Xerox Star. Именно с нее и началось победное шествие GUI, внедрение которого, как промышленного стандарта, было произведено в 1988 году усилиями Sun Microsystems, Xerox и AT&T. Xerox Star мы также обязаны такими вещами, как иконки, окна, меню, двухбайтовые многоязычные шрифты (современный Unicode — см. главу 8), режим

WYSIWYG и др. За подробностями я отсылаю читателя к [25], а здесь нам важно, что распущенный после рыночного провала Xerox Star примерно в 1983 году коллектив ее разработчиков в основном оказался в Apple — за важными исключениями, о которых поговорим отдельно.

В числе прочих инноваций в интерфейсе Star было впервые использовано представление экранного пространства в виде "рабочего стола" (Desktop). Эта метафора основывается на том, что пользователь якобы не имеет представления о существовании, скажем, программы под названием "текстовый редактор", а просто "открывает документ". Альтернативой является концепция "инструментов" (Tools) или "приложений" (Applications), где пользователь запускает нужный инструмент (приложение), и с его помощью открывает документ, причем тип его идентифицируется обычно по расширению имени файла. Ответственность за результат, например, загрузки файла, содержащего изображение, в текстовый редактор при этом целиком ложится на пользователя. Разработчики Star сознательно шли на потерю универсальности, присущую инструментальной модели, в то же время предупреждая об ограниченности сферы применимости метафоры "рабочего стола" исключительно офисными системами. (Дуглас Энгельбарт, изобретатель мышинного интерфейса, позднее иронизировал: *"Весь мир был увлечен идеей "офисной автоматизации", будучи уверен, что "настоящие пользователи" компьютеров — это секретари, чьи задачи необходимо автоматизировать."*)

В 1978 году упомянутый ранее Джеф Раскин работал в Apple, где начал новый проект под названием Macintosh, однако уже в 1982 г. разругался с Джобсом и покинул фирму. Впрочем, это неудивительно — ни один из его проектов так и не стал успешным в коммерческом смысле, притом что выдвинутые им теории обязательно изучают на всех соответствующих курсах в мире. Основная заслуга Раскина в том, что он одним из первых осознал: самое важное ментальное ограничение человека — ограниченность внимания. Фокус внимания у человека один. Интерфейсы, переключающие на себя внимание человека, — это одна из ключевых причин неэффективности взаимодействия с машинами.

### **Заметки на полях**

Вот пример: когда мне диктуют телефонный номер, я хочу сразу записать его, пока он не вылетел из памяти. В типичной современной ОС для этого требуется запустить специальное приложение (по крайней мере, переключиться на него), после чего создать новый контакт (это же надо еще знать, что то, что вам нужно, называется именно "контактами", см. главу 16), найти на появившейся форме подходящее поле, вписать туда номер... Автор этих строк и в бумажной-то телефонной книжке не вел записи по алфавиту, предпочитая для ускорения записывать их "внавал", и сейчас фиксирует номера, адреса и прочие полезные сведения в маленькой программке в стиле картотеки из Windows 3x. И даже забросил ведение адресной книги в The Bat! — слишком много усилий приходится

предпринимать для систематизации записей и извлечения оттуда нужного адреса, проще найти корреспондента через "Поиск".

Раскин выступил с радикальным предложением: отказаться от зоопарка разных приложений и заменить его на единую рабочую среду, которая всегда ведет себя одинаково. В такой среде, например, нажатие клавиш всегда приводило бы к вводу текста, поэтому человеку, желающему записать телефон, достаточно было бы просто его набрать. Разумеется, определить введенный номер в "правильное" место нужно было бы и в этой среде, но в ней это можно сделать после ввода номера, что принципиально удобнее. В результате Раскин придумал интерфейс, суть которого состоит в следующем: все, что хранится в компьютере, представляет собой единый документ, отдельные приложения — это команды и модули, запускаемые пользователем или подключающиеся автоматически. Причем все управление осуществляется с помощью клавиш — мыши Раскин не признавал, действительно, ведь для работы с текстом мышь в принципе не требуется, так зачем лишние сущности? Несмотря на понятные ограничения (а кроме инструментальной модели, предоставляющей пользователю полный доступ к "потрохам" системы, вплоть до ее перепрограммирования, любая другая модель является ограниченной и пригодной лишь в определенной области) эта система для тех, кто работает именно с документами, была бы, вероятно, более удобной. И в конце 80-х она даже была осуществлена на практике в компьютере Canon Cat, который разошелся в количестве 20 тыс. экземпляров, но затем проект был свернут.

Подход Раскина можно интерпретировать так: вся среда в компьютере есть одно универсальное приложение. Легко заметить, что черты этого подхода можно встретить, например, в MS Office, в попытках интегрировать все действия пользователя — или большинство их — в единую среду. Я не знаю, что могла бы представлять доведенная до логического завершения концепция Раскина (в Canon Cat был осуществлен фактически лишь текстовый редактор с функциями электронных таблиц), но есть сильное подозрение (основанное на практическом опыте), что ни один программист, даже такой корифей, как Джеф Раскин, не сумел бы сделать интегрированную среду во всех нюансах так, чтобы в ней было бы удобно работать всем без исключения.

Но вернемся к существующим интерфейсам. Для платформы PC (уже превратившейся к тому времени в Wintel) "рабочий стол" был реализован только спустя пятнадцать лет — с появлением Windows 95. Казалось бы, было время все продумать и учесть недоработки Xerox и Apple, но в Windows была принята эклектичная попытка сохранить все преимущества инструментальной модели, как наиболее гибкой, но рассчитанной на специально обученных пользователей, и в то же время склонить на свою сторону армию "чайников", обучаться не желающих или не имеющих возможности. В результате пророческие слова Раскина о том, что *"пользователь обычно не понимает, на-*

сколько ему неудобно" относятся к Windows, как ни к чему другому. Буквально любое незнакомое действие в Windows, несмотря на все попытки унификации и стандартизации, требует от "чайника" консультаций с более продвинутым собратом по несчастью<sup>2</sup>. Непоследовательное использование Desktop-метафоры может приводить к просто катастрофическим последствиям: т. к. первичным признаком для отнесения файла к тому или иному типу в Windows по-прежнему служит расширение его имени, то при неправильном переименовании файл может быть просто потерян для непросвещенного пользователя. При попытке скрыть расширения, как это происходит по умолчанию в Windows, часто может возникать забавная ситуация, когда в одной папке оказываются несколько абсолютно одинаковых с виду значков, что приводит к недоразумениям — одно дело послать по электронной почте многомегабайтный TIFF, другое — компактный JPEG, а ведь все иконки, относящиеся к изображениям (если вы не зададите специально обратного), будут связаны с одним приложением, и потому и обозначатся одинаково.

Рискну предположить, что правильным решением всех этих проблем был бы выпуск множества относительно автономных модификаций одной и той же ОС с интерфейсами, "заточенными" под нужды бухгалтера, секретаря, технического писателя, ученого, фотохудожника, журналиста, ребенка, наконец, с возможностью установки их в любой комбинации и переключения между ними. Глядишь, и модель Раскина вполне тогда вписалась бы в коммерческие продукты, и любителям прыгающих по экрану кнопок или "скрепок-помощников" тоже было бы где развернуться...

Резюмируем то, что сказано ранее, попробовав сформулировать основную цель любого интерфейса — не только компьютерного. Подавляющее большинство действий совершается человеком бессознательно. Часть подобных действий человек умеет совершать от рождения, таких, как глотательные рефлексы или отдергивание руки от горячего предмета. Среди этих действий есть очень сложные *программы поведения* — например, половое поведение или стайные инстинкты. Другая часть — инстинкты и программы, приобретенные в процессе взросления и обучения, это навыки ходьбы, речи, письма и счета, или социальное поведение: стыдливость, способность к состраданию и т. п. Наконец, сравнительно небольшую часть действий составляют те, что контролируются сознанием. Однако именно эти действия требуют повышенного внимания, они человека, пользуясь расхожим выражением, "поглощают целиком". Ни один человек не может обдумывать две

---

<sup>2</sup> Причем по мере развития одни неудобства устраняются (так, реализация Plug&Play в Windows XP принципиально лучше всех предыдущих версий), зато множатся другие (в той же XP — непонятные проблемы с языком, атрибутами папок, активацией и т. п.).