http://radio-mir.com

индексы: 48925, 45995

радиомир

12 • 2003

Ваш компьютер

ABBYY SOFTWARE HOUSE

СЛОВАРЬ: АНГЛО-РУССКИЙ РУССКО-АНГЛИЙСКИЙ

Lingvo • 8.0

ОБИМЫЕ МУЛЬТФИЛЬМЫ

БАРТОК ВЕЛИКОЛЕПНЫЙ

MPEG 4

Немецкий язык в 3 приёма

тивный курс обучения

COSMOPOLITAN

Virtual Makeover 2003

Deluxe Edition

НА РУССКОМ ЯЗЫКЕ

Windows 95/98/2000/NT

>>>>>>>>>>>>>>>> Windows XP

РУССКАЯ ВЕРСИЯ

Microsoft Windows xp Professional Corporate

Загрузочный диск

PETROSOFT

Windows 2003 Enterprise Server

MSDN RTM Build

Киностудия на твоем компьютере

5000 ЛОГОТИПОВ

НОВАЯ РЕДАКЦИЯ-НОВАЯ РЕДАКЦИЯ-НОВАЯ РЕДАКЦИЯ

KENZO

Coca-Cola

100 PIPERS

ABSOLUT Country of Sweden VODKA

ВСЕ ЛОГОТИПЫ

U.S.Robotics

COMPAQ

DANONE

GRINDERS

Dirol XYLITOL Effect

DOS EQUIS

Gillette

FedEx

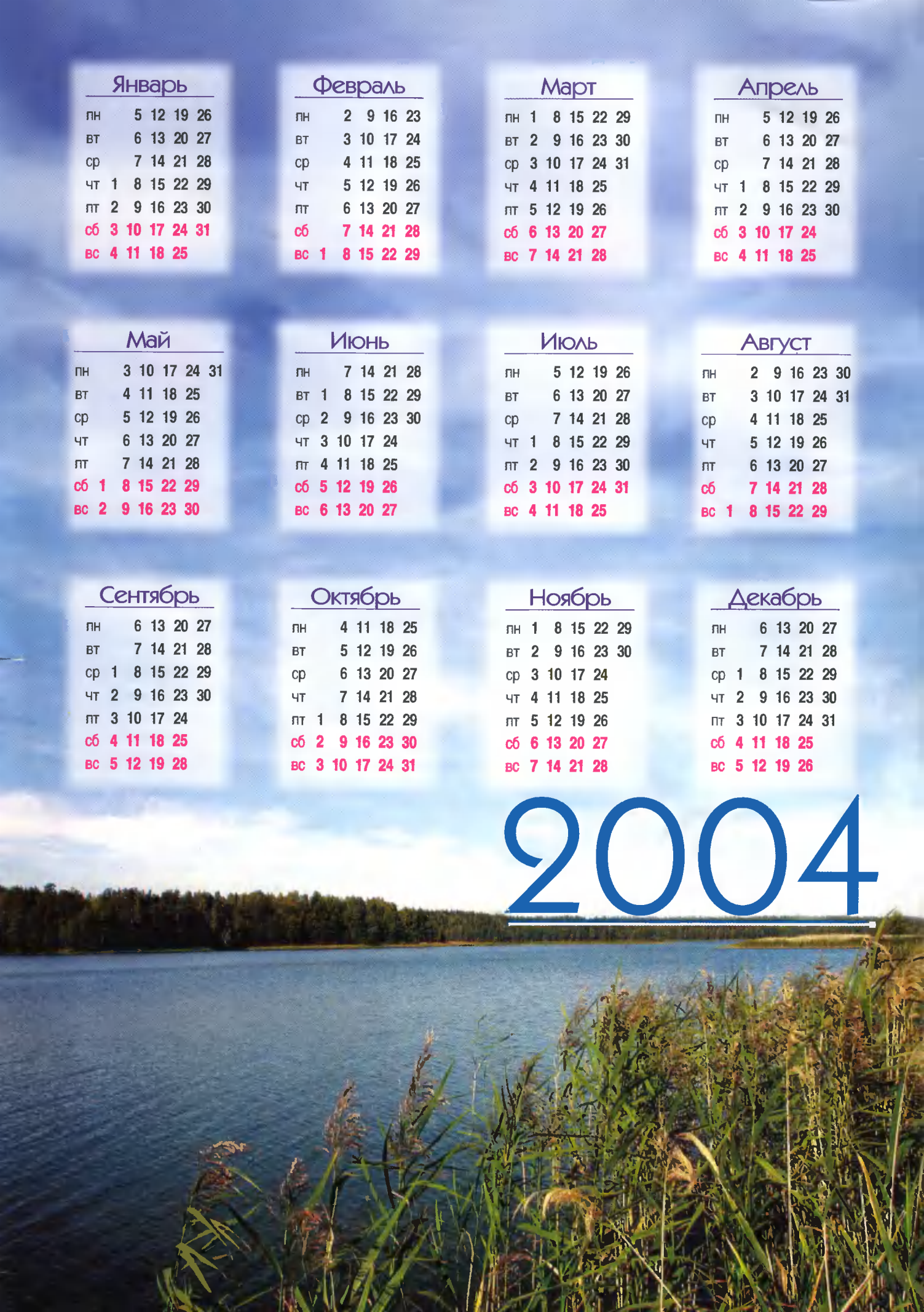
Dr. Martens AirWair

НОВАЯ РЕДАКЦИЯ-НОВАЯ РЕДАКЦИЯ-НОВАЯ РЕДАКЦИЯ

Полная CD версия

Mixman DJ Megamix

Вы слышали, как DJ'и заводят тояпу в клубах ? Теперь у Вас есть шанс самому испытать свои возможности и воплотить свои идеи!



Январь

пн		5	12	19	26	
вт		6	13	20	27	
ср		7	14	21	28	
чт	1	8	15	22	29	
пт	2	9	16	23	30	
сб	3	10	17	24	31	
вс	4	11	18	25		

Февраль

пн		2	9	16	23	
вт		3	10	17	24	
ср		4	11	18	25	
чт		5	12	19	26	
пт		6	13	20	27	
сб		7	14	21	28	
вс	1	8	15	22	29	

Март

пн	1	8	15	22	29	
вт	2	9	16	23	30	
ср	3	10	17	24	31	
чт	4	11	18	25		
пт	5	12	19	26		
сб	6	13	20	27		
вс	7	14	21	28		

Апрель

пн		5	12	19	26	
вт		6	13	20	27	
ср		7	14	21	28	
чт	1	8	15	22	29	
пт	2	9	16	23	30	
сб	3	10	17	24		
вс	4	11	18	25		

Май

пн		3	10	17	24	31
вт		4	11	18	25	
ср		5	12	19	26	
чт		6	13	20	27	
пт		7	14	21	28	
сб	1	8	15	22	29	
вс	2	9	16	23	30	

Июнь

пн		7	14	21	28	
вт	1	8	15	22	29	
ср	2	9	16	23	30	
чт	3	10	17	24		
пт	4	11	18	25		
сб	5	12	19	26		
вс	6	13	20	27		

Июль

пн		5	12	19	26	
вт		6	13	20	27	
ср		7	14	21	28	
чт	1	8	15	22	29	
пт	2	9	16	23	30	
сб	3	10	17	24	31	
вс	4	11	18	25		

Август

пн		2	9	16	23	30
вт		3	10	17	24	31
ср		4	11	18	25	
чт		5	12	19	26	
пт		6	13	20	27	
сб		7	14	21	28	
вс	1	8	15	22	29	

Сентябрь

пн		6	13	20	27	
вт		7	14	21	28	
ср	1	8	15	22	29	
чт	2	9	16	23	30	
пт	3	10	17	24		
сб	4	11	18	25		
вс	5	12	19	28		

Октябрь

пн		4	11	18	25	
вт		5	12	19	26	
ср		6	13	20	27	
чт		7	14	21	28	
пт	1	8	15	22	29	
сб	2	9	16	23	30	
вс	3	10	17	24	31	

Ноябрь

пн	1	8	15	22	29	
вт	2	9	16	23	30	
ср	3	10	17	24		
чт	4	11	18	25		
пт	5	12	19	26		
сб	6	13	20	27		
вс	7	14	21	28		

Декабрь

пн		6	13	20	27	
вт		7	14	21	28	
ср	1	8	15	22	29	
чт	2	9	16	23	30	
пт	3	10	17	24	31	
сб	4	11	18	25		
вс	5	12	19	26		

2004

Учредитель и издатель:
ООО "Радиомир Пресс"
Свидетельство о регистрации
ПИ №77-9841 от 17.09.2001

Главный редактор
Ольга СТРЫЖАНКОВА

Зам. гл. редактора
Иван БЕЛЬСКИЙ

Редколлегия:
Сергей ДРОЗДОВСКИЙ,
Алексей КОНДРАШОВ,
Анатолий КОРБИТ,
Владимир КУЦЕНКО,
Елена ЛЕВИТМАН,
Николай МЕДВЕДЬ,
Геннадий ПЕЧЕНЬ,
Геннадий ШУЛЬГИН

Контактные телефоны:
в Москве (095) 105-99-89,
в Минске (017) 249-41-47

Компьютерная верстка —
Янина БЕЛЬСКАЯ

Техническая графика —
Наталья БЕЛЬСКАЯ,
Александр ГОРАЮТИН,
Мерия КАЛАБУХОВА

Оформление обложки —
Наталья БЕЛЬСКАЯ,
Надежда БОГОМОЛОВА

Адреса для писем:
119454, РФ, г.Москва, а/я 37,
факс (095) 131-97-17;
220095, РБ, г.Минск-95, а/я 199

E-mail: rl@radiopage.by
rm@radio-mir.com
WWW: http://radio-mir.com

Наши платежные реквизиты:
получатель: ООО "Радиомир Пресс",
ИНН 7729404741, КПП 772901001,
р/с 40702810900010000084
в ООО КБ "Агропромкредит"
ф-л Центральный, г.Москва,
корр. счет 30101810500000000109
БИК 044525109
Адрес банка: 125315, г.Москва,
Ленинградский пр., дом 76, корп. 4

Материалы для публикации принимаются в рукописном, печатном и электронном вариантах

Требования к графическим материалам рекламного характера в электронном виде: CorelDRAW до 10.0, все шрифты в кривых; bitmaps 300 dpi; TIFF 300 dpi; CMYK. Приложить печатную копию.

За достоверность рекламной и другой публикуемой информации несут ответственность рекламодатели и авторы. Мнение редакции не всегда совпадает с мнениями авторов

Адрес редакции: 119454, РФ, г.Москва, ул.Кошгоянца, 6 — 233, тел. (095) 105-99-89

Подписано к печати 28.10.2003 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная.

Отпечатано в типографии
ЗАО "Красногорская типография".
Заказ №3247.

Радиомир
Ваш компьютер

ЕЖЕМЕСЯЧНЫЙ МАССОВЫЙ ЖУРНАЛ
С 1995 г. по 6/2001 г. издавался под названием "Радиолюбитель. Ваш компьютер"

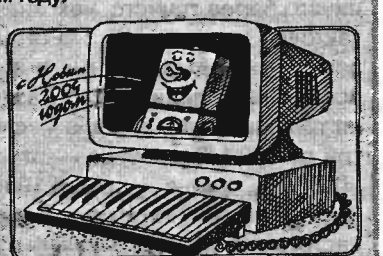
№12/декабрь/2003

ЧИТАЙТЕ В НОМЕРЕ

Дорогие друзья!

Подводя итоги 2003 г., мы по традиции называем авторов, чьи материалы определяли "лицо" нашего журнала в уходящем году:

А.Горячкин (г.Кыштым, Челябинской обл.),
Е.Илясов (г.Балашов, Саратовской обл.),
В.Куц (г.Пушкин, Ленинградской обл.),
И.Рощин (г.Москва),
М.Долинский (г.Гомель),
Р.Карпач (г.Минск),
С.Рюмик (г.Чернигов),
коллектив авторов HI-TECH.



Скромная награда за их труд — бесплатная подписка на 2004 г.

Желаем и нашим авторам, и нашим читателям творческих успехов и реализации всех задумок.

С.РЮМИК,
г.Чернигов.

"Cell" — ячейка для "PlayStation 3"

Идя в будущее, у прошлого спроси дорогу.
Народная мудрость

Как сопоставить между собой возможности игровых приставок? Лет пять назад такой проблемы не существовало — чем выше разрядность центрального процессора и больше объем памяти, тем приставка лучше. Первый тревожный звонок прозвучал в 1998 г., когда специалисты не смогли однозначно определить разрядность "Dreamcast" ф.SEGA. Рекламные заявления о "первой в мире 128-битной" приставке, на проверку оказались блефом.

ЦП современных приставок представляет собой "компьютер в компьютере" и содержит большое число разнородных регистров. Более того, в процессе работы они могут конфигурироваться в блоки переменной длины. Понятие "разрядность" стало больше маркетинговым, чем техническим термином. Два малоизвестных факта. В ЦП "истинно" 128-разрядной "PlayStation 2" имеются регистры длиной 16-, 32- и 128 битов. Этот же ЦП посылает в графический синтезатор слова блоками по 64 бита, которые в дальнейшем попадают на обработку в 2560-разрядную (это не ошибка!) магистраль.

Технические параметры игровых приставок следует рассматривать в комплексе, включая пропускную способность системной шины, объем кэш-памяти, производительность графического процессора, параметры аудиосистемы и периферийных устройств. И все же, должен быть один, легко запоминающийся параметр, по которому любой геймер мог бы получить общее представление о потенциальных возможностях приставки. На сегодняшний день не видно ничего более точного, чем значение тактовой частоты ЦП. По этому параметру все выпущенные и анонсированные модели можно разделить на три поколения: первое — с частотой до 100 МГц, второе — от 100 до 1000 МГц, третье — свыше 1000 МГц (см. **таблицу**). Понятно, что верхняя частота у третьего поколения приставок, когда она станет известной "дефакто", определит нижнюю частоту четвертого поколения, и т.д.

Сведения о приставках третьего поколения окружены ореолом таинственности, даже их названия доподлинно неизвестны. В гонке собираются участвовать три фирмы. Первой об этом в 2001 г. заявила ф.Sony, спланировав свою работу на пять лет вперед. Затем представители ф.Microsoft пообещали выпустить к 2006 г. "лучшую в мире приставку". В январе 2003 г. президент ф.Nintendo Сатори Ивата (Satoru Iwata) объявил, что его фирма также намечает разработку приставки нового поколения.

Одно известно точно — все новые модели будут иметь тактовую частоту ЦП выше 1 ГГц. Такую планку установила ф.Sony, пообещав увеличить ее на порядок по сравнению с "PlayStation 2".

Фирмы, входящие в "железный треугольник", выделили на разработку "Cell" 400 млн. долларов, организовали исследовательскую группу на базе производства IBM в Остине, шт.Техас, и запланировали построить отдельный завод по производству этих процессоров.

В августе 2002 г. был завершен исследовательский этап работ. Теперь очередь за инженерами производства, которым предстоит не только воплотить теоретические изыскания в "железо", но и отработать технологию промышленного изготовления "кремний на изоляторе". Важный момент — если в 2001 г. для "Cell" планировались нормы допуска 0,1 мкм, то в 2002 г. они были скорректированы в сторону уменьшения до 50...60 нм.

Игровая приставка	Год выпуска	Тактовая частота, МГц	Фирма
Первое поколение			
"NES" ("Dendy")	1983	1,7	Nintendo
"Mega Drive"	1988	7,6	SEGA
"PlayStation"	1994	33	Sony
"Nintendo-64"	1996	94	Nintendo
Второе поколение			
"Dreamcast"	1998	200	SEGA
"PlayStation 2"	2000	295	Sony
"GameQube"	2001	485	Nintendo
"X-box"	2001	733	Microsoft
Третье поколение			
"PlayStation 3"	2005...2007	3000...5000	Sony
"GameQube 2"			Nintendo
"X-box 2"			Microsoft

Немного истории

12 марта 2001 г. ф.Sony, IBM и Toshiba объединились в альянс по разработке принципиально нового процессора, получившего кодовое название "Cell" (в переводе с англ. "ячейка, клетка"). В техническом задании ф.Sony фигурировала цифра "1 Тфлопс" — один терафлопс или один триллион операций с плавающей запятой двойной точности в секунду. Такой производительностью, по мнению Кена Кутараги, идеолога разработки "PlayStation" и "PlayStation 2", должна обладать игровая приставка нового поколения. Для сравнения, это почти в 100 раз выше, чем обеспечивает Pentium4 с тактовой частотой 2,5 ГГц!

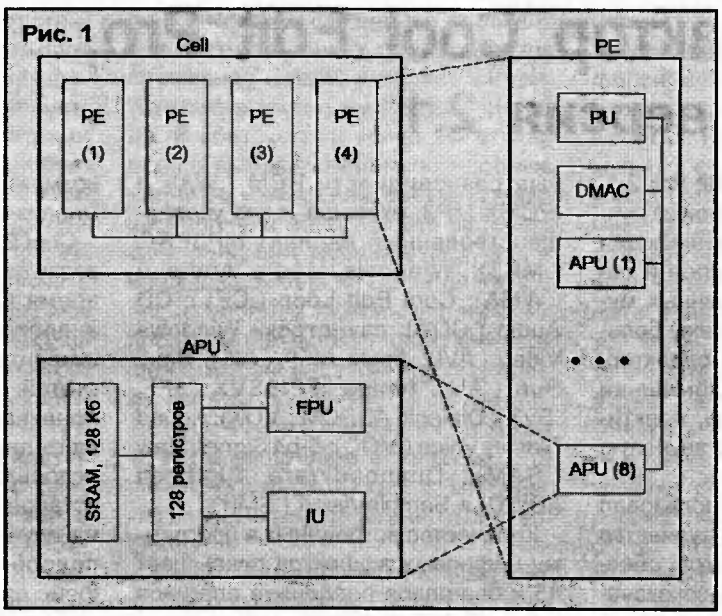
В конце 2002 — начале 2003 г. ф.Sony оформила патенты на архитектуру "Cell", а также лицензировала у ф.Rambus технологию "YellowStone" и интерфейс "Redwood" для быстродействующего ОЗУ и широкополосных коммуникаций. Появились первые технические подробности проекта.

На **рис.1** приведена обобщенная схема внутреннего устройства процессора "Cell" согласно слайд-шоу Пауля Зимонса (Paul Zimmons, http://home.btconnect.com/cell/CELL.ppt). Разработчики альянса называют свое детище не иначе как "суперкомпьютер на чипе". Внутри СБИС находится комплексная электронная схема, состоящая из нескольких вычисли-

тельных ядер, специализирующихся на выполнении разных задач.

"Cell" может содержать от 4 до 16 процессорных элементов PE (Processor Element) в зависимости от сложности выполняемых задач. Каждый PE включает в себя центральный процессор PU (Processing Unit), контроллер прямого доступа в память DMAC (Direct Memory Access Controller) и от 1 до 8 сопроцессоров APU (Attached Processing Unit). В свою очередь, каждое APU содержит статическое ОЗУ SRAM объемом 128 Кб, 128 регистров общего назначения, а также 4 вычислителя с плавающей запятой FPU и столько же — для целочисленных операций IU.

Рис. 1



Основой PU является процессор PowerPC с кэш-памятью. На место любого APU может быть установлен специализированный графический ускоритель Visualizer. Внутренние шины "Cell" имеют размерность 128, 384, 1024 бита, разрядность регистров общего назначения — 128 битов. Общее число транзисторов на чипе — около 1 млрд.

Поскольку все вычислители работают независимо друг от друга, то за счет параллельного выполнения задач суммарная производительность одного APU составляет 32 Гфлопс. Если "Cell" содержит 4 PE, то в них будет 32 APU общей мощностью 1 Тфлопс. Синхронность выполнения вычислений обеспечивается специальным сетевым протоколом, где каждый вычислитель и каждый банк памяти имеет свой IP-адрес.

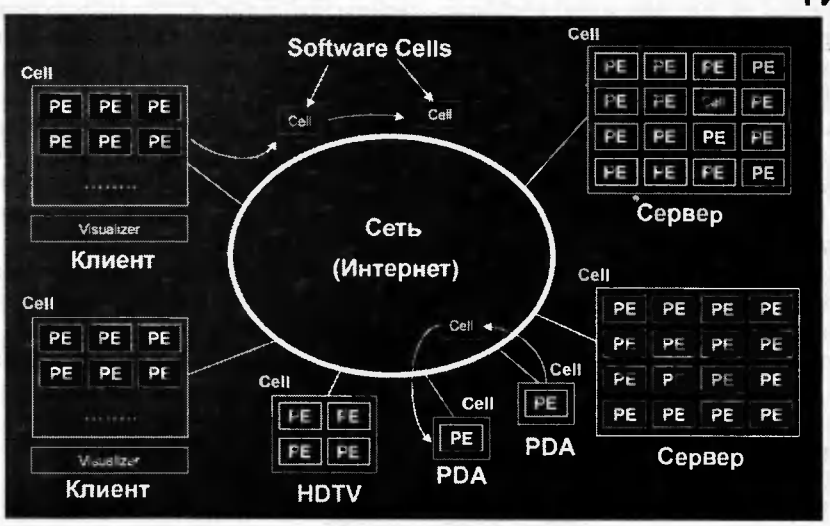
Идея процессора с несколькими вычислительными ядрами не нова. В середине 80-х годов ф.Inmos выпускала транспьютеры — 32-разрядные процессоры, которые производили параллельные вычисления в матрице себе подобных устройств. Однако технология того времени еще не позволяла "упаковывать" всю систему на один кристалл.

Ближайшими аналогами "Cell" считаются чипы POWER4 (ф.IBM) и MAJC (ф.Sun Microsystems). Первый из них содержит связку двух 64-разрядных процессоров PowerPC и применяется в мощных серверах. Второй предназначен для обработки мультимедиа в профессиональных графических станциях.

компоненты ячеек объединяются вместе для решения определенных задач. Например, наладонный персональный помощник PDA, имеющий "Cell", можно будет подключить к широкоформатному телевизору высокой четкости HDTV, также имеющему "Cell", и этот тандем будет работать вместе по своему внутреннему протоколу, без необходимости переналадки.

Насколько реальны "планы Наполеона" — покажет будущее. Разработчики уже столкнулись со сложностями, которые

Рис. 2



Преимущества "Cell" — увеличенное количество процессорных ядер, универсальность (он может быть сконфигурирован как для обычных, так и для графических вычислений), сетевой протокол внутренних обменов, возможность исправления ошибок в пакетах передачи данных, гибкая маршрутизация.

Параллельно с разработкой аппаратной начинки ф.Sony ведет работу над собственной операционной системой Cell-OS и над адаптацией программного обеспечения к другим ОС, включая Linux. В начале 2003 г. ведущие разработчики игр (Capcom, Electronic Arts, Activision) заявили о поддержке платформы Cell и о подготовке к выпуску первых игровых программ.

Процессор "Cell" может входить не только в состав "PlayStation 3". На его основе планируется создать новую расширяемую вычислительную платформу (**рис.2**). Эта система подобна улею, где

отодвинули, по крайней мере, до 2006 г. появление "PlayStation 3". В Интернет-форумах, посвященных этой приставке, до сих пор с недоверием относятся к цифре "1 Тфлопс", считая ее рекламным трюком. Многие сомневаются, что игровая приставка стоимостью менее 1000 USD сможет превзойти по вычислительной мощности ныне действующие многомиллионные суперкомпьютеры "Cray". В противовес хочется заметить, что техническая поддержка проекта держится на авторитете ф.IBM, которая обладает технологией, позволяющей делать транзисторы с затворами, состоящими всего лишь из нескольких десятков атомов. Кроме того, идеологией ф.Sony всегда была постановка (и выполнение!) амбициозных сверхзадач, начиная от первого в мире транзисторного приемника, первого видеомагнитофона, первого переносного проигрывателя CD, первого электронного домашнего робота-собаки "AIVO".

Аудиоредактор Cool Edit Pro, версия 2.1

В.КУЦ,
E-mail: victor_kootz@mail.ru

Звуковой редактор Cool Edit Pro 2.1, без сомнения, является одной из лучших программ, предназначенных для редактирования аудиофайлов и создания на их базе полноценных музыкальных композиций. Скажу больше, используя только этот редактор, можно превратить свой домашний компьютер в полноценную, практически профессиональную звуковую студию!

Каждый, кто хоть раз использовал свой домашний компьютер в качестве личного музыкального центра, обеспечивающего не только высококачественное (в той или иной степени) воспроизведение аудиофайлов, но и их хранение, ощутил насущную потребность в записи, элементарном редактировании, сохранении или преобразовании форматов сокровищ своей музыкальной коллекции. Все эти нехитрые возможности способен предоставить практически любой простейший звуковой редактор. Но только полупрофессиональные редакторы, к одним из лучших образцов которых по праву относится Cool Edit Pro, кроме этого, обеспечивают большое количество дополнительных функций, которые, хоть и используются любителями относительно редко, тем не менее, рано или поздно оказываются востребованными — ведь по мере роста мастерства самодеятельных звукорежиссеров возрастают и их потребности.

Так, редактор Cool Edit Pro позволяет сводить аудио на 128 дорожках с применением эффектов реального времени, обладает удобным инструментарием для редактирования и конвертирования аудио (включая пакетный режим), поддерживает большое число форматов аудиофайлов. В их число входят практически все современные разновидности PCM (.WAV и .PCM); MP3, включая и его усовершенствованный вариант MP3PRO (.MP3); Windows Media Audio 9 (.WMA); Cool Edit Loop (.CEL); CD Audio (.CDA); саундтреки Windows Video (.AVI); Apple AIFF (.AIF); Next/Sun (.AU); Amiga IFF/8SVX (.IFF, .SVX); Dialogic ADPCM (.VOX); Sound Blaster Voice (.VOC); 8-bit Signed Raw (.SAM); DiamondWare Digitized (.DWD) и SampleVision (.SMP).

Количество встроенных в программу цифровых эффектов превышает 45 и благодаря поддержке плагинов может быть увеличено практически до бесконечности. Отдельной группой стоят достаточно сложные в использовании, но предоставляющие самые широкие возможности так называемые научные фильтры. На фоне такого богатого окружения отходят на второй план инструменты, составляющие гордость многих других аналогичных продуктов: встроенные в редактор аудиориппер, тон-генератор, анализаторы спектра и фазы, инструменты для представления звуковой дорожки в виде спектра или осциллограммы, управление аудиопотоком через MIDI-интерфейс, возможность "рисования" сигнала и многое-многое другое.

Cool Edit Pro, являясь дальнейшим развитием очень популярного в свое время Cool Edit 96, отличается от аналогичных продуктов других производителей, в первую очередь, простотой и доступностью интуитивно понятного интерфейса, позволяющего даже начинающему любителю использовать редактор сразу после установки, не тратя много времени на изучение Help'a, кстати, весьма подробного. К сожалению, интерфейс программы англоязычный, да и на просторах Интернета пока не были замечены какие-либо ее русификаторы. Поэтому для тех, кто не в ладах с чужими языками, данная статья может послужить своеобразным путеводителем, пусть и весьма ограниченным, облегчающим первоначальное знакомство с этой интересной, но и, вместе с тем, достаточно сложной для новичка программой.

Основные элементы рабочего окна

Главное рабочее окно редактора Cool Edit Pro (**рис.1**) включает в себя

Рис. 1

расположенную в самом верху стандартную панель меню, содержащую основные команды управления редактором, а под ней — панели инструментов с кнопками, дублирующими большинство из этих команд и обеспечивающими легкий доступ к ним. Жаль только, что панели инструментов не настраиваемые, хотя и имеется возможность включать/отключать группы кнопок, соответствующих или пунктам меню, или их важнейшим подразделам. Ниже панели инструментов располагается линейный индикатор, на котором цветом (по умолчанию — зеленым) отмечается положение отображаемого в данный момент на экране фрагмента обрабатываемого файла относительно его полного размера. Захватив "ладошкой", в которую превращается курсор при попадании на индикатор, выделенный участок, и нажав левую кнопку мыши, можно без труда перейти к любому необходимому участку файла. А с помощью правой кнопки можно изменить масштаб выбранной части изображения. Сам файл музыкального произведения, вернее, его графическое отображение, занимает большую часть рабочего окна. Естественно, отображаются и обрабатываются оба стереофонических канала одновременно, хотя, при необходимости, можно оставить только один из них, щелкнув мышью соответственно в верхней половине левого (верхнего на экране) или в нижней половине — правого (нижнего) каналов. Сам звуковой сэмпл может отображаться в двух графических режимах — Waveform и Spectral. Установленный по умолчанию режим Waveform позволяет получить графическое изображение частотно-временной характеристики звукового сигнала в виде кривой, на вертикальной оси которой располагается шкала громкости звучания отдельной выборки, а на горизонтальной — временная шкала. В режиме Spectral мы имеем амплитудно-частотную характеристику исследуемого сигнала, т.е. данные о том, какие частоты в определенный момент времени наиболее сильны. Переключаться между этими режимами можно простым нажатием кнопки **Toggle between Waveform and Spectral views** в верхней панели инструментов.

Оцифрованные шкалы по краям изображений значительно облегчают оценку количественных параметров сигнала. Снизу располагается временная шкала, а справа — вертикальная шкала амплитуды сигнала. В контекстном меню этих шкал можно выбрать наиболее удобные единицы измерения соответствующих величин. Левую часть рабочего окна занимает появившаяся еще в предыдущей версии Cool Edit Pro 2.0 очень удобная панель управления проектом (Organizer), обеспечивающая быстрый доступ к любому файлу, как уже открытому, так и находящемуся в папке Избранное, а также ко всем доступным эффектам. Эта панель может быть скрыта, впрочем, как и любые другие элементы интерфейса.

В нижней части рабочего окна располагаются (слева направо) следующие элементы:

- 10 кнопок транспортной панели, имеющих обозначения, аналогичные клавишам управления бытовой аудиоаппаратуры и выполняющих те же функции;
- 8 кнопок управления масштабом отображения в горизонтальной и вертикальной плоскостях;
- окно, отображающее текущее положение маркера курсора музыкального файла, или, проще говоря, время записи/воспроизведения этого файла. Это окошко можно сделать плавающим и поместить в любое место экрана, хотя не совсем понятно, для чего это может быть нужно;
- группа из 6 полей, отображающих начальное и конечное время воспроизведения файла, а также длительность как всего файла, так и его выделенного участка.

Вот такой получился длинный ряд элементов управления, но и это еще не все. Еще ниже на все окно размахнулся пиковый индикатор уровня сигнала. По умолчанию полный размах его шкалы составляет 120 децибел, что чаще всего является некоторым излишеством — в домашних условиях вряд ли будут обрабатываться файлы, имеющие такой динамический диапазон. Рекомендуется подстроить шкалу, исходя из своих конкретных потребностей, воспользовавшись контекстным меню.

Завершает всю эту картину строка состояния, а куда ж от нее денешься? И, как мы уже привыкли, посредством ее контекстного меню можно определить, какая информация будет в ней отображаться.

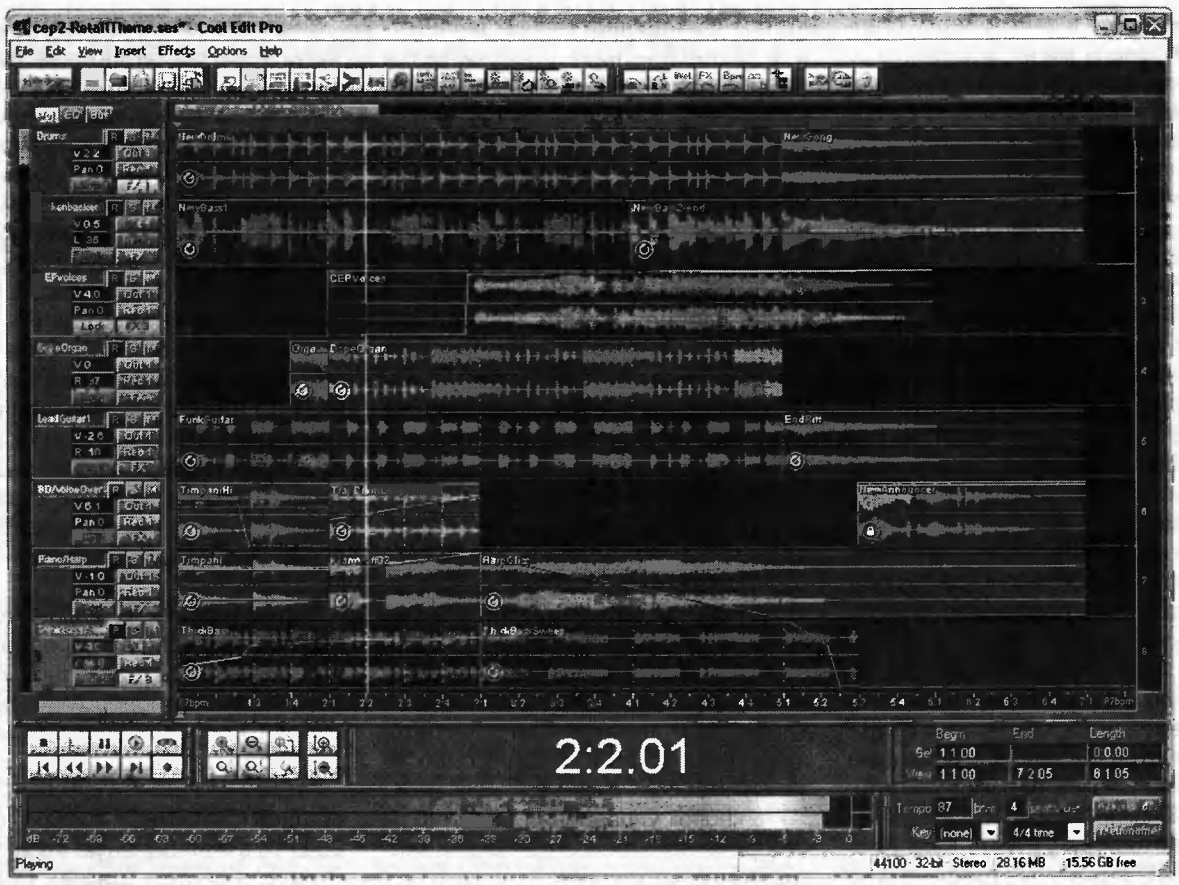
Как я уже упоминал, одной из отличительных особенностей полупрофессиональных звуковых редакторов, к которым по праву принадлежит Cool Edit Pro, является возможность мультитрекового редактирования звуковых сэмплов. Если каждый из них имеет до двух аудиодорожек (стерео), то такое редактирование подразумевает наличие множества редактируемых треков. В Cool Edit Pro их может быть до 128, чего вполне достаточно для решения самых сложных задач. Редактирование одиночного трека или нескольких одновременно — задачи принципиально разные, и требуют разных средств. Поэтому в рассматриваемом редакторе существует два режима работы — Edit и Multitrack.

Первый из них — Edit — установлен по умолчанию и предназначен для редактирования одной стереофонической аудиодорожки. Именно о нем и будет весь наш дальнейший разговор.

Второй режим — Multitrack — превращает звуковой редактор Cool Edit Pro в полноценную 4-канальную звуковую студию (**рис.2**). Применение многоканальной студии может быть обусловлено необходимостью комбинирования сигналов из нескольких источников, выравнивания их уровней, синхронизации начала и конца их различных участков и, в конце концов, сведения всех фрагментов в единую стереофоническую фонограмму.

Рабочее окно студии отличается, в первую очередь, наличием 4 дорожек, на которых располагаются фрагменты будущей композиции. С помощью мышки их можно перемещать, копировать, пересылать для дополнительной обработки в звуковой редактор. В правой части каждой из дорожек появилась дополнительная панель управления, кнопки на которой позволяют активизировать или наоборот — запрещать отдельные дорожки, устанавливать некоторые параметры фрагментов и готовить их к сведению в единую фонограмму. При одновременной обработке нескольких больших звуковых файлов в этом режиме необходимо иметь в виду, что требования к мощности компьютера, особенно к размеру доступной оперативной памяти, существенно возрастают, и на слабых машинах возможны задержки в процессе обработки, порой весьма значительные.

Рис. 2



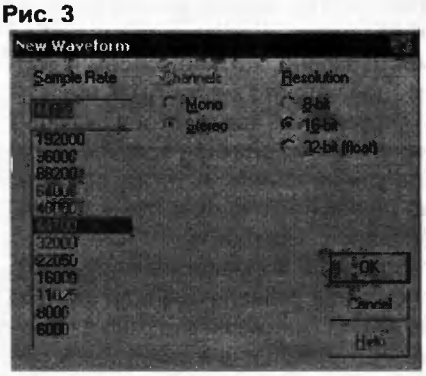
Самое элементарное — запись и первичное редактирование звуковых файлов

Для записи нового звукового файла нужно создать новый документ — либо через меню **File**, либо нажав на кнопку **Create new wave**. В появившемся окошке (**рис.3**) необходимо задать параметры будущего файла. В поле **Sample Rate** следует выбрать частоту дискретизации. Величина этого параметра впрямую определяет качество полученной записи — чем выше частота, тем лучше качество; но, вместе с тем, существенно увеличивается и размер итогового файла. В большинстве случаев принятой по умолчанию частоты 44100 Гц (стандартной для CD-аудио) для записи музыкальных композиций вполне достаточно. Для записи обычной речи вполне подойдут и минимальные значения — 6000...8000 Гц.

В следующем поле надо указать вид сигнала — моно или стерео. Для записи с обычного монофонического микрофона лучше выбрать монотрек, но и стереорежим может пригодиться, ведь в этом режиме можно сигнал с одного микрофона записывать последовательно на обе дорожки звукового сэмпла, имитируя режим мультитрекового редактирования.

Последний параметр — Resolution — позволяет выбрать разрядность числа, определяющего амплитуду сигнала. К примеру, 8-битовое число позволяет

Рис. 3



описать амплитуду только 256 значениями, что приемлемо только для записи речи, да и то с крайне невысоким качеством. 16-битовое разрешение, являющееся стандартом для аудио-CD, обеспечивает достаточно высококачественную запись сэмпла. 32-битовое значение амплитуды может потребоваться в отдельных случаях, когда важно высочайшее качество записываемого сигнала любой ценой.

Определив параметры нового Wav-файла, остается только нажать соответствующую кнопку — и все, запись пошла.

Для самого примитивного редактирования музыкального трека в меню **Edit** имеются команды редактирования текущего файла. Нужный фрагмент выделяют простым перетаскиванием мышью с нажатой левой кнопкой. Эти операции лучше выполнять в режиме Waveform, так как в нем наиболее точно можно определить начало и конец отдельных звуков, а точнее — точки, где их громкость минимальна. Кроме обычных команд **Copy, Cut, Paste** и т.д., представляет интерес команда **Set Current Clipboard**, позволяющая установить в качестве рабочего или один из 5 имеющихся в программе собственных буферов обмена, или стандартный буфер Windows. Клипы из этих буферов можно вставлять в существующий файл или создать на их основе новый. Другие команды позволяют удалить выделенный участок файла или удалить все, кроме выделения; убрать участки фонограммы без сигнала; конвертировать формат музыкального файла. А такая интересная команда как **Zero Crossings** позволяет автоматически сдвигать границы выделенной области к участку сигнала, имеющему нулевой уровень, что позволяет избежать характерных, но крайне неприятных щелчков при монтаже различных участков фонограммы.

(Продолжение следует)

MATLAB. ПРИБЛИЖЕНИЕ ФУНКЦИЙ. ИНТЕРПОЛЯЦИЯ

Б.КИСЕЛЕВ,
г.Минск,
БГАТУ, каф.ВТ.

В этой статье будут рассмотрены простейшие приемы интерполирования для аппроксимации функций, а также использование для этой цели системы MATLAB. Мы попытаемся изложить суть вопроса без строгих выкладок и показать ряд примеров, а за подробностями отошлем читателя к [1, 2] и любой другой литературе по численным методам.

Аппроксимация

Аппроксимация — это замена одних математических объектов другими, более простыми и в том или ином смысле близкими к исходным (например, замена кривой линии ломаной).

Аппроксимация функций часто применяется в тех случаях, когда функция $f(x)$ задана таблично, т.е. в нескольких точках x_i (**табл.1**):

Табл. 1

X	x_0	x_1	x_2	...	x_i	...	x_n
Y	y_0	y_1	y_2	...	y_i	...	y_n

Здесь: i — номер узла функции, (x_i, y_i) — узловые точки. Значения x_i различны между собой. Такие таблицы получаются, например, при проведении опыта, когда мы задаем значения аргумента x и измеряем значения функции y. Необходимо подобрать такую, непрерывную на $[x_0, x_n]$ функцию $\varphi(x)$, чтобы она проходила близко к узлам или точно через них, т.е. $\varphi(x)$ должна приближенно отражать функцию $f(x)$. В результате, по функции $\varphi(x)$ можно приближенно определить значения y в любой точке отрезка $[x_0, x_n]$, а не только в узлах. Каждый "на глаз" решал такие задачи в школе, когда рисовал кривые по результатам измерений. Аппроксимация дает возможность получить формулы для вычисления этих кривых.

Аппроксимацию можно осуществить различными функциями. Однако чаще всего используются многочлены (полиномы), т.к. они являются самыми простыми функциями, имеют непрерывными все производные, и с их помощью можно с любой заранее заданной точностью представлять практически все имеющие прикладной интерес функции.

На практике используют два вида аппроксимации — *интерполяцию* и *подбор эмпирических моделей*.

Интерполяция

Интерполяция имеет место в том случае, когда аппроксимирующая функция $\varphi(x)$ *точно проходит* через заданные узлы табличной функции. Интерполяцию можно применять в различных случаях, но чаще всего ее используют для замены исходной функции более простой (вычисляют $f(x)$ в нескольких точках и проводят $\varphi(x)$ через эти точки), для восстановления непрерывной функции по заданной таблице, а также при замене экспериментальных кривых теоретическими (на экспериментальной кривой выбирают несколько точек и через них проводят $\varphi(x)$).

В качестве $\varphi(x)$ обычно берут многочлен:

$$\varphi(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n, \qquad (1)$$

коэффициенты которого можно определить, решив систему линейных уравнений. Эта система формируется, исходя из условий равенства значений функции $f(x)$ и многочлена $\varphi(x)$ в узлах. Например, построим уравнения для интерполяции второй степени (квадратичной), т.е. когда (1) является многочленом второй степени:

$$\varphi(x) = a_0 + a_1x + a_2x^2.$$

Пусть необходимо вычислить значение функции в точке x^*, которая лежит между пятым и шестым узлами. Для построения системы уравнений выберем узлы x_5, x_6, x_7, т.к. точка x^* должна лежать между узлами. Тогда:

$$\begin{cases} a_0 + a_1x_5 + a_2x_5^2 = y_5, \\ a_0 + a_1x_6 + a_2x_6^2 = y_6, \\ a_0 + a_1x_7 + a_2x_7^2 = y_7. \end{cases}$$

Матрица коэффициентов при неизвестных a_i будет иметь вид:

$$X = \begin{pmatrix} 1 & x_5 & x_5^2 \\ 1 & x_6 & x_6^2 \\ 1 & x_7 & x_7^2 \end{pmatrix}.$$

Вектор неизвестных $A = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$.

Вектор правых частей $Y = \begin{pmatrix} y_5 \\ y_6 \\ y_7 \end{pmatrix}$.

В матричной форме записи система уравнений примет вид: $X\,A = Y$.

Тогда:

$$A = X^{-1}Y. \qquad (2)$$

В [3] уже рассматривался вопрос, с помощью каких функций в MATLAB лучше решать систему (2).

Теперь, зная коэффициенты a_i, определим функцию в точке x^*:

$$\varphi(x^*) = a_0 + a_1x^* + a_2x^{*2}.$$

Это значение и считаем за приближенное значение y в точке x^*. Заметим, что можно было выбрать узлы x_4, x_5, x_6. Выбор узлов может повлиять на точность, поэтому в общем случае желательно, чтобы точка x^* лежала ближе к узлам.

Если необходимо определить значение функции в другой точке, например, в x^{**}, и эта точка лежит за пределами отрезка $[x_5, x_7]$, то коэффициенты a_i необходимо вычислить заново, для другого набора узлов. В нашем примере имеется три узла, и степень многочлена равна двум, т.е. число коэффициентов на единицу больше, чем степень многочлена. Следовательно, для построения многочлена степени n необходим n+1 узел. Причем, всегда имеется только единственное решение [1].

Чтобы не решать системы уравнений n+1-го порядка, пользуются готовыми формулами, например, формулами Лагранжа, Ньютона. Если функция $f(x)$ известна, то можно определить погрешность от замены ее многочленом [1].

В системе MATLAB для интерполяции функции одной переменной обычно используется функция **interp1**. Общая форма обращения к ней:

yi = interp1(x,y,xi,method).

Функция возвращает вектор $\mathbf{y_i}$ — значения интерполяционного многочлена в точках $\mathbf{x_i}$ (**x** и **y** — векторы узлов из табл.1). Опция **method** задает метод интерполяции:

- **'nearest'** — интерполяция нулевого порядка (ступенчатая);
- **'linear'** — интерполяция первого порядка (линейная, задается по умолчанию);
- **'cubic'** — интерполяция третьего порядка (кубическая).

Рассмотрим несколько примеров реализации интерполяции в системе MATLAB.

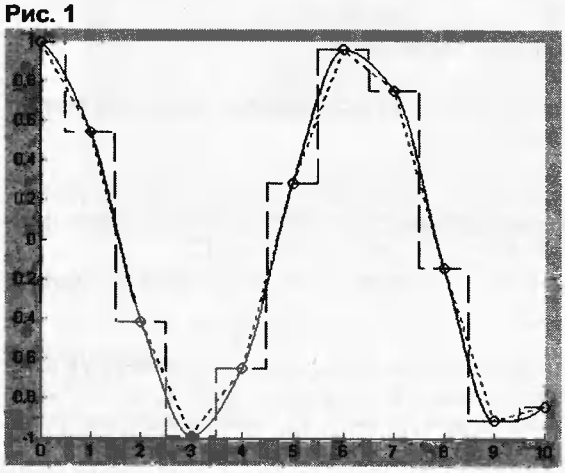
Пример 1

Необходимо построить интерполяционные многочлены для функции косинуса. Программа для MATLAB, решающая эту задачу:

```
% Интерполирование
%=================
x=0:10;y=cos(x);
xi=0:0.01:10;
% интерп. 0-й степеин
yi=interp1(x,y,xi,'nearest'); hold on
plot(xi,yi,'r');
% интерп. 1-й степеин
yi=interp1(x,y,xi);
plot(x,y,'o',xi,yi,'g');
% интерп. 3-й степени
yi=interp1(x,y,xi,'cubic');
plot(xi,yi,'k'), hold off;
```

Первая строчка программы формирует табл.1 из одиннадцати узлов для $x = (0, 1, 2, \ldots, 10)$. Вторая строчка создает вектор точек x_i, в которых вычисляется интерполяционный многочлен. Эти точки заданы с мелким шагом, чтобы графики имели относительно гладкий вид (**рис.1**). Кстати, заметим, что сама функция **plot** использует линейную интерполяцию, т.е. все точки графика соединяются отрезками прямых.

Рис. 1



Параметры функции **plot** описаны в [4], здесь лишь отметим, что кружочками показаны узловые точки, штриховая линия — ступенчатая интерполяция, пунктирная — линейная, сплошная — кубическая.

Как видно из рис.1, косинусоиду больше всего напоминает кубическая интерполяция, однако повышение степени многочлена не всегда улучшает качество аппроксимации.

Пример 2

Необходимо построить интерполяционный многочлен 9-го порядка для функции:

$$y = \frac{1}{\sqrt{(1-x^2)^2 + 0.04x^2}}, \quad \text{где } x \in [0,\ 2.25]. \qquad (3)$$

Для построения интерполяционного многочлена произвольного порядка можно воспользоваться функцией **polyfit**, которая определяет коэффициенты многочлена заданной степени по узлам табл.1 (этот вопрос будет рассмотрен в следующей статье):

R = polyfit(X,Y,n).

Здесь: **X, Y** — соответствующие массивы табл.1, **n** — степень многочлена, **R** — массив коэффициентов многочлена в порядке уменьшения его степеней.

Полученный с помощью этой функции многочлен будет интерполяционным, т.е. пройдет точно через узлы, если его степень на единицу меньше числа узлов таблицы.

Имея массив коэффициентов многочлена, можно вычислить его значения с помощью функции:

p = polyval(R,x).

Здесь: **R** — массив коэффициентов многочлена; **x** — массив точек, в которых требуется вычислить многочлен; **p** — вычисленный массив значений многочлена в точках x.

Выполнив команды:

```
% Задание таблицы значений интерполируемой функции (3)
X=(0:0.25:2.25)';
Y=1./sqrt((1-X.^2).^2+0.04*X.^2);
% Вычисление коэффициентов интерполяционного
% многочлена 9-го порядка
R=polyfit(X,Y,9);
% Построение графиков
x=0:.001:2.25; % массив абсцисс для графиков
p=polyval(R,x); % массив точек интерполяционного многочлена
yt=1./sqrt((1-x.^2).^2+0.04*x.^2); % массив точек функции (3)
%
% График интерполяционного многочлена 9-го порядка и функции (3)
plot(x,yt,'k',X,Y,'ko',x,p,'k');grid on;
%
% График линейного многочлена и функции (3)
figure;
plot(x,yt,'k',X,Y,'ko',X,Y,'k');grid on;
```

Рис. 2

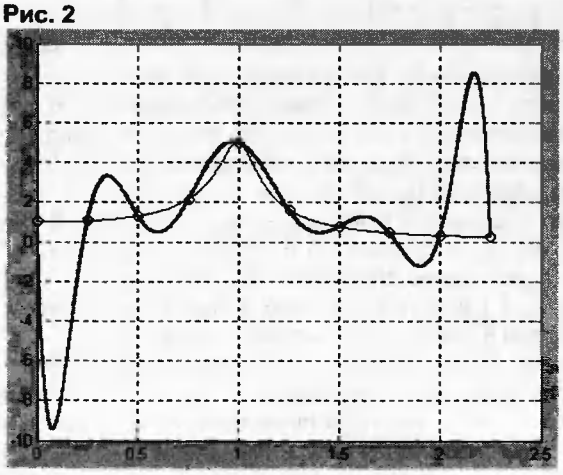
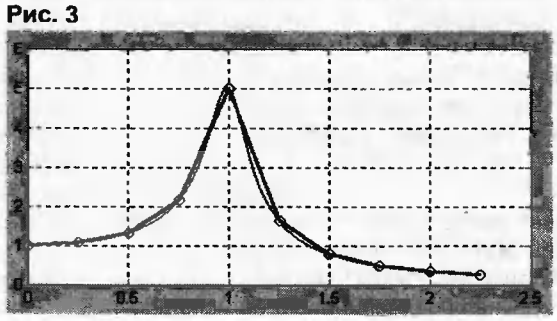


Рис. 3



получим результат, представленный на **рис.2** и **3**.

На рис.2 кружочки обозначают узлы интерполяции, жирной линией обозначен график интерполяционного многочлена 9-го порядка, а тонкой — функции (3).

На рис.3 жирной линией обозначен график линейного интерполяционного многочлена, остальные обозначения такие же.

Рис.1...3 иллюстрируют тот факт, что интерполяционные многочлены низкого порядка несущественно проигрывают в точности (а в некоторых случаях выигрывают (рис.3)) многочленам высокой степени, зато они намного проще. Сильная "колебательность" характерна для многочленов высокой степени.

В инженерной практике для интерполяции обычно используют многочлены не выше третьего порядка, поэтому и в системах компьютерной математики, как правило, нет специальных функций для интерполяционных многочленов степени выше трех. В тех случаях, когда нет информации о поведении функции между узлами, тем более нет смысла применять интерполяцию высокой степени.

Пример 3

На **рис.4** приведена регуляторная характеристика дизельного двигателя, снятая опытным путем,. Если эта характеристика будет использоваться в расчетах, то необходимо получить функцию данной кривой. Для этой цели

Рис. 4

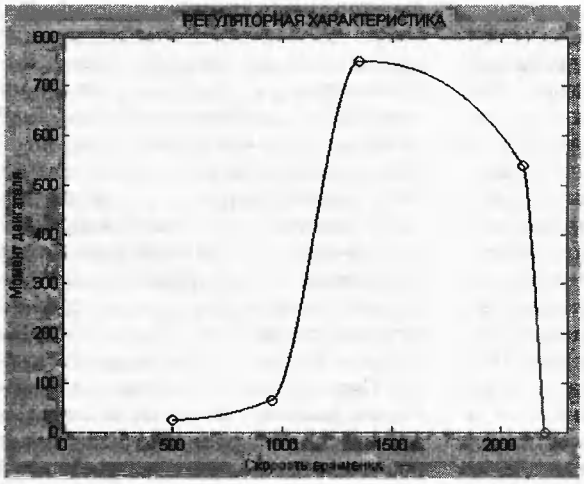


Рис. 5

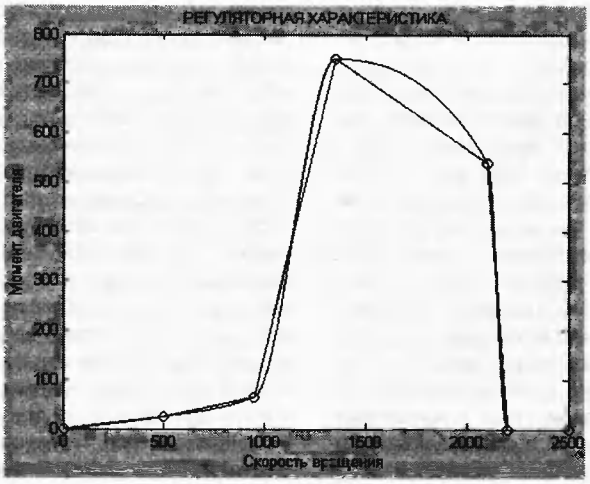


Табл. 2

φ(об/мин)	500	950	1350	2100	2200
$M_{дв}$ (н/м)	25	65	750	540	0

используем интерполяционный многочлен первого порядка. В качестве узлов интерполяции выберем точки, представленные в **табл.2**, (на рис.4 они обозначены кружочками).

Добавим слева узел (0; 0), а справа узел (2300; 0) на тот случай, если при каких-то вычислениях расчеты будут попадать за пределы табл.2. Тогда при расчетах истинная кривая заменится кривой $\varphi(x) = a_0 + a_1 x$, где a_0 и a_1 будут рассчитываться для каждого из участков на **рис.5**.

Если необходимо, чтобы в узлах интерполяции многочлен имел непрерывные производные, то применяют интерполяцию сплайнами [2]. В рассмотренной выше функции **interp1** для этого следует выбрать метод **"spline"**.

MATLAB использует для интерполяции еще несколько функций:

- **interp2** — интерполяция функций двух переменных;
- **interp3** — интерполяция функций трех переменных;
- **interpn** — интерполяция функций *n* переменных;
- **interpft** — интерполяция периодических функций рядом Фурье и другие.

Отметим еще одно интересное свойство интерполяции — удобство вычислений обратных функций. В узлах значения функции рассматриваются как аргумент, а аргументы — как значения функции, затем вычисляют интерполяционный многочлен.

Экстраполяция

Если интерполяционный многочлен, построенный на краю отрезка, использовать для вычисления значений за пределами этого отрезка, то такая процедура будет называться *экстраполяцией*.

Упражнения

1. Вычислить функцию *sin*(5*x*) в точках *x*, использованных для функции (3), и построить интерполяционный многочлен 9-го порядка. Сравнить графики многочлена и функции *sin*(5*x*).
2. Выбрать шесть узлов для функции $y = 2 + 3x$ на отрезке [0,10] и определить коэффициенты интерполяционного многочлена пятого порядка.
3. По данным примера 1 построить интерполяционный многочлен обратной функции и результат сравнить с функцией $x = \arccos(y)$.

Вопросы для самопроверки

1. Что означает выражение "Функция задана таблично"?
2. Пояснить суть интерполяции.
3. Сколько нужно задать узлов, чтобы построить многочлен степени k?
4. Как зависит вид интерполяционного многочлена от значений функции между узлами?
5. Чему равна минимальная степень интерполяционного многочлена?
6. Чему равна максимальная степень интерполяционного многочлена?
7. Как вычислить обратную функцию с помощью интерполяции?

Литература

1. Демидович Б.П., Марон И.А. Основы вычислительной математики. — М.: Наука, 1966.
2. Калиткин Н.Н. Численные методы. — М.: Наука, 1978.
3. Киселев Б.М. MATLAB. Работа в командном режиме. — Радиомир. Ваш компьютер, 2003, №№ 6, 7.
4. Киселев Б.М. MATLAB. Визуализация результатов вычислений MATLAB. — Радиомир. Ваш компьютер, №8, С.4.

Худая грамота только душе пагуба

(интернет-мечтателям посвящается)

Р.КАРПАЧ,
E-mail: Commander-Norton@tut.by
www.fdd5-25.narod.ru

Увы, мой стих не блещет новизной,
Разнообразьем перемен нежданных.
Не поискать ли мне тропы иной,
Приемов новых, сочетаний странных?
В.Шекспир

Вызывает ли у вас раздражение слово портал? Несведущим оно, конечно, не будет резать слух и воскрешать разнообразные эмоции. В лучшем случае они вспомнят игру Warcraft 2 или интернет-сайт со стандартным силиконовым видом, форумом PHPBB и каким-нибудь аномальным чатом. Может быть, в воображении еще нарисуется on-line-газета, увешанная баннерами, как новогодняя елка, в которой, как всегда, очень много мусора и очень мало полезных вещей... Как это до боли знакомо, не так ли, дорогой читатель? Ты сам себе говоришь: "Да я смог бы сделать лучше! Мой портал мог бы стать самым посещаемым!" В итоге в Сети появляется еще один ресурс, как клон Norton Commander похожий на остальные. И вот уже другой пользователь, наткнувшись в поисковике на этот портал, скрипя зубами, нажимает ALT+F4. Потому

что в очередной раз он нашел то, чего как раз и не искал.

"Рожки да ножки"

Вот и я, дорогой читатель, как-то загорелся благородной идеей создания портала, где будет все. Зависнув однажды в одном благородном чате, мне удалось познакомиться с неким человеком под ником Маршал78. Он представился создателем и автором портала, который через несколько дней, месяцев или лет должен был открыться. Это произвело на меня неизгладимое впечатление, ведь я, по правде говоря, считал, что создатели коммерческих проектов в Сети — люди серьезные, очень "продвинутые" и коммуникабельные. Маршал78, недолго думая, предложил мне встретиться и договориться о совместной работе над проектом. В тот период у меня уже имелся кое-какой опыт общения с подобным контингентом. Воображение само по себе рисовало человека с красивой распальцовкой и золотой цепью на килограмм. Я не мог представить себе, чтобы обычный простой смертный начал делать, вы только подумайте, свой собственный портал. От этой мысли даже начинало немного штормить в голове. Но вот долгожданный день встречи с Маршалом78 наступил. "Стрелка", а по-другому и не скажешь, была назначена возле Академии наук РБ, видимо, это должно было прибавить респектабельности. Хотя сейчас, глядя в прошлое, она больше напоминала встречу двух агентов спецслужб во вражеской стране — кодом служили усы и органайзер. Я долго ползал в переходе, ища правильный выход к зданию Академии. Погода стояла ненастная, небо было затянуто серыми тучами, и дул пронизывающий ветер. Возле здания Академии наук РБ стояло два человека. Они ругались замудреными словами. Я подошел, представился и попытался включиться в разговор. Но слышал только непонятные для себя реплики. Так мы простояли некоторое время, пока не подошел еще один создатель проекта. Как это ни парадоксально, мы пошли искать кафе, в котором сможем посидеть. Маршала78 все не устраивали цены — то дорого, то еще дороже. Так мы бродили, пока не зашли в студенческую столовую, цены здесь оказались приемлемыми, мы, наконец, сели, и началось. На мою голову обрушились новые компьютерные термины — хостинг, php, perl, mysql, mssql, asp и многое другое. Один из пришедших клялся и божился, что ему написать форум и чат для портала MyBasnya.net не составит никакого труда. Второй бил себя в грудь и кричал, держа в руках стакан с пивом, что он сможет обеспечить информацией все вдоль и поперек. Мне приходилось молчать, попивая чай, следуя народной мудрости "молчи — за умного сойдешь". Вмиг на столе выросла груда бумаги со схемами, идеями, набросками. Я был просто в шоке от такого, на мой взгляд, хакерского подхода. Маршал78 был не то что проникнут идеей его полного будущего превосходства в Сети, но и называл месячный оборот в 150 000 USD, от чего мне стало совсем дурно. "Съезд" закончился спустя три часа, и я, "загруженный", поехал домой...

Прошло несколько месяцев. Те двое, что приезжали на встречу, отказались от проекта. Я и Маршал78 остались без программистов. Время шло, а портала все не было — MyBasnya.net продолжал "висеть в воздухе", словно воздушный шар. В своем стремлении чего-то достичь Маршал78 напоминал Наполеона — он создавал один дизайн за другим и никак не мог определиться, что же он хочет видеть на портале. А хотел он многого. За это время он достал все конторы и фирмы Минска своими предложениями о сотрудничестве. Но везде ему давали ответ: "Сделаешь, тогда и будем говорить." Время шло, но толку было мало. Маршал78 собрал новую встречу и нашел новых программистов, 16-ти и 17-ти лет соответственно. Это ввергло меня в недоумение, но я промолчал. Надо — значит, надо. У Маршала78 созрел новый план. Он решил сделать on-line-газету, искренне полагая, что у него все получится без проблем. Замысел был очень прост — своей информативностью и гениальностью задушить все компьютерные СМИ республики. "Мама, дорогая, да он просто гений!" — подумал я, и, конечно, согласился. Но новые программисты были людьми ограниченными — вместо дохода в 150 000 USD их интересовал банальный чат для подростков, который Маршал78 повесил у себя. Они глотали слюни и хотели только одного — стать модераторами. "Боже мой, с кем он связался?" — подумал я и бросил MyBasnya.net, перед этим снявшись на телевидении вместе с создателями портала в одной из компьютерных передач. С того "веселого" времени ничего не изменилось — портал MyBasnya.net находится в полусонном состоянии вот уже два года. Программисты ушли и организовали свой чат с помощью чужих исходников, выданных за свои. Маршал78 старается изо всех сил. А я решил подойти к созданию портала с другой, более прагматичной точки зрения.

Идея и хостинг созданы друг для друга

Рассуждая сам с собой о выбранном нами с Маршалом78 пути создания портала, я открыл для себя две непреложные истины. Они заключаются в идее уникальности и независимости от чего- или кого-либо. Как можно создавать серьезный проект, если каждый занимается и всем, и ничем? Первое, что оказалось нужно — это команда, которая займется хоть чем-то. Второе — это идея и концепция портала. Не имею привычки раскрывать свои карты, выскажу лишь мнение, что заняться нужно тем, чего еще нигде нет. Условно назовем идею проекта "Любители хачапури". Вы знаете, что это такое? Вы когда-нибудь ели хачапури? Ели! То-то же, в этом и есть самая соль. Хачапури есть, чай есть, а портала про это нет. Нет форума хачапури, чата хачапури, рассылки хачапури. И, следовательно, условный контингент любителей этого блюда испытывает в Сети огромные психологические проблемы. Конечно же, из-за отсутствия соответственного портала. Вы скажете: "Что за бред ты пишешь?" Поясню. Сколько сайтов посвящено компьютерной тематике? Много! Не то слово! Их сотни, если не тысячи, тех, что называют себя порталом. И если вы откроете такой ресурс, на вас не обратят внимания или скажут, что вы — еще один из многих. Это жестокая правда жизни. А вот если вы у себя будете рассказывать про хачапури, каждый, даже если таких ресурсов станет много, скажет, что именно здесь у вас родилась великая идея. Для престижа найдите еще людей, которые смогут вести свои рубрики, тех, кому будет просто интересно. Пусть это будет один человек, но он должен быть окрылен вашей идеей. Затем мы посмотрим вокруг и поймем, что везде и все дышит однообразием — PHPBB-форум, тот же PERL-чат для подростков с кучей смайликов. Скучно... Но зато у вас будет хачапури! Оно выведет портал из стандартного силиконового однообразия. А если вем начнут выказывать свое недовольство, это свидетельствует лишь о том, что вас заметили. Дело сделано! Проблема теперь в другом. Как заработать? Это дело ваше, я и Маршал78 в этом вам не советчики, не повторяйте моих ошибок.

Все это было лирическое отступление. Ну а теперь серьезно. Самая большая проблема для всех начинающих порталов заключается в выборе местопребывания, или, проще говоря, хостинга.

Что же такое хостинг? Хостинг — это выделяемый пользователю на сервере (компьютере, постоянно подключенном к сети) раздел на диске в виде каталога, в котором хранится вся информация вашего сайта. Все — от графики до скриптов. Хостинг бывает платный и бесплатный (условно). С бесплатным хостингом бывает много проблем. Обычно там "живет" куча рекламы, которую размещают, чтобы оплатить свои безвозмездные услуги и еще заработать. Это могут быть баннеры, текстовые ссылки, рекламные фреймы, всплывающие окна и тому подобное. Всегда существует ограничение на закачку файлов. Но самое главное не это. На бесплатном хостинге нельзя заниматься коммерческой деятельностью, а следовательно, он больше подходит для домашних страниц, либо сайтов жад-

ных компьютерных фирм.

Так что если вы не жадная компьютерная фирма и не домашняя страничка, то я рискну вам предложить рецепты выбора хостинга. Первое, на что мы обратим внимание — это предоставляемое дисковое пространство. Разброс цен и объема поистине огромен. Зарубежные компании могут предложить и 500 Мб всего за 15 USD в месяц. Наши же товарищи более скупые, т.к. они чаще всего бывают субхостерами, т.е., проще говоря, перепродают услуги. Поэтому первое правило, возникшее само собой в процессе рассуждения, подскажет нам, что хостинг нужно покупать у головных компаний, а не у посредников. Ну а если дело идет уже на принцип, и вы ориентируетесь на аудиторию только из СНГ или наоборот, только из НАТО, то следует решить, нужна ли вам русская служба поддержки, которую можно доставать целыми днями, или лучше сэкономить некоторую сумму. Решать вам. После того как найден приемлемый по цене вариант, нужно выяснить, что же может предоставить хостер. Важный фактор — дополнительные возможности: собственная cgi-bin-директория, где можно запускать свои скрипты; поддержка Perl, Php, SSI, баз данных и т.д. Возможно, новичкам эти сервисы и ни к чему, но поверьте, что без них вы никогда не сделаете интерактивный сайт с форумами, гостевыми книгами, поисками и прочими полезными наработками. Не лишним будет узнать у хостера, возможна ли прямая закачка с вашей матрицы или другого накопителя на сервер. Ведь многие до сих пор пользуются dial-up, и закачивать собственный сайт будет очень тяжело. И еще, если уж вы решили покупать платный хостинг, то обязательно просите тестовый аккаунт на 3...7 дней, чтобы проверить сервис, или ищите в правилах строчку о возвращении денег в течение 30 дней, если вас что-то не устраивает. Обязательно заключайте договор, чтобы на крайний случай был юридически оформленный документ. Есть еще одна тонкость у платных хостингов. Читая правила и тарифы, следует внимательно вникать во все, что касается затрат. Нужно убедиться, что не будет скрытых платежей, кроме тех, что делаются сразу. Некоторые хитроумные товарищи любят за все взимать денежки. Например, помимо обычной помесячной оплаты, с вас могут взять вначале некую сумму в 20...50 условных единиц за установку и настройку вашего аккаунта — ведь они трудятся дни и ночи на благо пользователей, а вы должны это оценить. Кроме того, могут быть платными услуги парковки и поддержки ваших доменов 2-го и 3-го уровня, дополнительно могут стоить e-mail, субдомены, открытия баз данных. Могут также взиматься дополнительные деньги за перерасход трафика. Последнее, кстати, обычно оправдано, ибо есть разница, содержите вы страничку с посещаемостью 100 человек в день или весьма посещаемый портал, обозреваемый десятками тысяч пользователей ежедневно. Немаловажное значение имеет служба поддержки, которая в идеале должна оперативно, в течение нескольких часов после запроса, отвечать на ваши вопросы и давать консультации, исправлять ошибки и улаживать прочие мелочи, возникающие в процессе пользования сервером. Ради интереса можно позвонить или задать администратору какой-нибудь каверзный вопрос, чтобы посмотреть его реакцию. Конечно, претендовать на истину в вопросе выбора хостинга я не могу. Каждый должен решать сам, что ему нужно, а что нет.

Вспомним все?

Когда мы с Маршалом78 делали свой портал, мы и не знали, чем же исторически Perl отличался от PHP. Да если бы и захотели узнать, нам бы это не понадобилось. Ведь создатели порталов — люди продвинутые, и нагружать себя разной полезной информацией они не должны. Тем более, когда они еще настраивают PHPBB-форум. Каждый раз, когда мы встречались и обсуждали работу, мы невольно употребляли различные термины, связанные с хостингом. И бывало, что вновь пришедшим была непонятна тема нашей беседа или очень скучна. Всякий раз, когда речь заходила о PHP, PERL, SQL, возникал вопрос: "Что это такое?" Хотя меня всегда терзала другая проблема: "Зачем это нужно?" Скажите, для чего были изначально разработаны PERL, PHP, HTML? И вновь мы натыкаемся на стену из вопросов. Так может, стоит ее проломить?

PERL (Practical Extraction and Report Language — практический язык извлечений и отчетов)

Perl был разработан Ларри Уоллом (Larry Wall) в 1986 г., когда тот являлся системным администратором одного проекта UNIX, связанного с созданием безопасной многоуровневой сети. Работа была выполнена, но потребовалось создание отчетов на основе большого числа файлов с многочисленными перекрестными ссылками между ними. Первоначально Ларри предполагал использовать для этих целей фильтр awk, но оказалось, что последний не мог управлять открытием и закрытием большого числа файлов на основе содержащейся в них же самих информации о расположении файлов. Его первой мыслью было написать специальную системную утилиту, решающую поставленную задачу. Но вспомнив, что до этого ему уже пришлось написать несколько утилит для решения задач, не "берущихся" стандартными средствами UNIX, он принял кардинальное решение — разработать язык программирования, который сочетал бы в себе возможности обработки текстовых файлов (sed), генерации отчетов (awk), решения системных задач (shell) и низкоуровневое программирование. Результатом этого решения и явился язык Perl, интерпретатор для которого был написан на C.

По утверждению самого Ларри Уолла, при создании языка Perl им двигала лень — не в прямом смысле, а в смысле того, что для решения стоявшей перед ним задачи следовало бы написать большое количество программ на разных языках, входящих в состав инструментальных средств UNIX, а это достаточно утомительное занятие. Новый язык программирования сочетал в себе возможности системного администрирования и обработки файлов — две основные задачи, решаемые обычно при программировании в системе UNIX. Причем следует отметить, что язык Perl появился из практических соображений, а не из-за желания создать еще одно "красивое" средство для работы в UNIX. Поэтому, когда Ларри Уолл предоставил Perl широкому кругу пользователей, тот получил широкое распространение среди системных администраторов. С появлением языка Perl стало возможно решать задачи с помощью одного инструмента, и не тратить время на изучение нескольких языков среды программирования UNIX. Perl решал задачи быстрее, чем sed или awk, и быстро стал использоваться не только для решения задач системного администрирования. В дальнейшем сам Ларри Уолл позаимствовал у Генри Спенсера (Henry Spencer) пакет для работы с регулярными выражениями и модифицировал его для языка Perl. Другие функциональные возможности были разработаны не только Ларри Уоллом, но и его друзьями и коллегами, а затем включены в состав языка. Опубликование в Internet привело к появлению сообщества единомышленников, которые не только эксплуатировали, но и развивали язык. Он и до настоящего времени продолжает интенсивно развиваться за счет разработки пакетов, реализующих новые применения языка к развивающимся информационным технологиям.

PHP/FI — Personal Contents Page/Forms Interpreter. PHP появился как развитие другого продукта — PHP/FI, созданного Rasmus'ом Lerdorf'ом в 1995 г. сначала как простой набор Perl-скриптов для отслеживания доступа к его собственному online-резюме. Он назвал этот набор скриптов "Personal Contents Page Tools". Поскольку требовалась большая функциональность, Rasmus написал расширенную реализацию C, которая могла работать с базами данных, и дал пользователям возможность разрабатывать простые динамические Web-приложения. Исходный код PHP/FI был опубликован для широкого доступа, чтобы

любой мог использовать, расширять и улучшать его. PHP/FI имел некоторую основную функциональность того PHP, который известен нам теперь — Perl-подобные переменные, автоматическую интерпретацию переменных форм и синтаксис, внедренный в HTML. Сам по себе синтаксис напоминал Perl, хотя и более ограниченный, упрощенный и неполный. PHP/FI 2.0, который был официально выпущен только в ноябре 1997 г., имел несколько тысяч поклонников по всему миру и был установлен приблизительно на 50000 доменов, что составляло примерно 1% всех доменов Internet. PHP 3.0 был первой версией, похожей на сегодняшний PHP. Его создали Andi Gutmans и Zeev Suraski в 1997 г. как полностью переписанный язык. Одной из сильных сторон PHP 3.0 была возможность его расширения. Кроме того, предоставляя конечным пользователям прочную инфраструктуру для различных БД, протоколы и APIs, возможности расширения PHP 3.0 побуждали десятки разработчиков поставлять новые модули расширения. Возможно, именно в этом был секрет ошеломляющего успеха PHP 3.0 — в конце 1998 г. он стал базой для десятков тысяч пользователей и сотен тысяч Web-сайтов. На пике своего успеха PHP 3.0 был установлен приблизительно на 10% Web-серверов Internet. PHP 4.0, оснащенный большим количеством новых возможностей, был официально выпущен в мае 2000 г., спустя почти два года после своего предшественника. Помимо значительно возросшей производительности, в этой новой версии были такие новые ключевые возможности как поддержка большого количества Web-серверов, HTTP-сессий, буферизации вывода, более безопасные способы работы с пользовательским вводом.

HTML (Hyper Text Markup Language) — язык разметки гипертекста. Начало истории его развития следует отнести к далекому 1986 г. когда международная организация по стандартизации ISO приняла стандарт ISO-8879, озаглавленный "Standard Generalized Markup Language (SGML)". Стандарт посвящен описанию SGML — обобщенного метаязыка, позволяющего строить системы логической, структурной разметки любых разновидностей текстов. Создатели SGML стремились максимально абстрагироваться от проблем представления электронного текста в разных программах, на разных компьютерных платформах и устройствах вывода. Так, если с помощью SGML размечается документ, содержащий заголовки, идеология языка запрещает указывать, что такой-то заголовок должен набираться, скажем, шрифтом Times полужирного начертания. SGML в таком случае требует ограничиться указанием на уровень заголовка и его место в иерархической структуре документа. Язык SGML — это типичное детище академической науки, изящная игрушка теоретиков, его создание не было вызвано насущной практической необходимостью. Однако в 1991 г. SGML был выбран в качестве основы нового языка разметки гипертекстовых документов для системы передачи гипертекстовой информации через Интернет. Этот язык, самое известное из приложений SGML, был назван HTML. Изначально HTML, как и положено SGML-приложению, разделял все особенности старой идеологии. Из сорока с небольшим тегов HTML версии 1.2 (июнь 1993 г.) всего три, да к тому же и не рекомендованных к использованию, осмеливались намекать на физические параметры представления документа. Вся разметка была чисто логической. Первым графическим браузером в те далекие времена была программа Mosaic, разработанная, как и сам WWW, в научном учреждении — Национальном центре суперкомпьютерных приложений США (National Center for Supercomputer Applications — NCSA). HTML неторопливо развивался, оставаясь в рамках парадигмы структурной разметки, и в апреле 1994 г. началась подготовка спецификации следующей версии языка — 2.0. Этим занимался образованный в том же году консорциум W3. Однако в 1994...1995 гг. его членами были почти исключительно университеты и научные учреждения. Столь "академический" состав W3C сказывался как на самих документах, публикуемых консорциумом, так и на процедуре их принятия. Достаточно сказать, что окончательный вариант HTML 2.0, единственным серьезным усовершенствованием в котором был механизм форм для отсылки информации с компьютера пользователя на сервер, был окончательно утвержден лишь в сентябре 1995 г., когда в W3C уже полным ходом шло обсуждение HTML 3. Пожалуй, этот проект — самая яркая и неоднозначная страница в истории языка. Работа над ним началась в марте 1995 г., и первоначальный вариант стандарта включал в себя много интересных нововведений: теги для создания таблиц, разметки математических формул, вставки обтекаемых текстом рисунков, примечаний и другое. Но самое главное, HTML 3.0 был попыткой разрешить уже достаточно очевидное к тому времени противоречие между идеологией структурной разметки и потребностями пользователей, заинтересованных в первую очередь в гибких и богатых возможностях визуального представления. Чтобы разрешить это противоречие, не оскверняя заветов отцов-основателей HTML, авторы версии 3 ввели в нее поддержку нового средства иерархических стилевых спецификаций (Cascading Style Sheets, CSS). Система CSS формально независима от HTML, имеет совершенно иной синтаксис, не наследует никаких идеологических ограничений и позволяет, уже в совершенно иных терминах, задавать параметры визуального представления для любого тега HTML. С помощью CSS автор может, наконец, с чистой совестью указать, каким шрифтом набирать заголовки такого-то уровня. В конце 1995 г. ситуация в мире HTML была довольно смутной. Популярность браузера Netscape неуклонно росла, программисты этой фирмы готовили к выпуску версию 2.0, которая должна была утвердить господство Netscape на вечные времена благодаря неслыханному набору новшеств. К этому времени W3C окончательно завяз в своем проекте HTML 3, который был слишком сильно оторван от реального мира, и на завершение которого у консорциума попросту не хватало средств. Этот язык, по сравнению с HTML 2.0, был важным шагом вперед, однако он развивался по-прежнему в рамках идеологии структурной разметки, а инструмент, дающий возможность выйти за эти рамки, система CSS, была еще далека от завершения. В этот тяжелый момент в игру вступил новый участник — неугомонная корпорация Microsoft. Долгое время эта компания, привыкшая монопольно владеть своим сектором рынка, недооценивала перспективы Интернета и не собиралась как-либо участвовать в развитии этой информационной среды, считая ее, по-видимому, слишком демократичной и непредсказуемой. Однако невероятный взлет Netscape (число копий браузера Navigator измерялось к этому времени уже десятками миллионов) заставил Microsoft изменить свое мнение. И именно на браузерном фронте, где господство Netscape оставляло меньше всего шансов конкурентам, корпорация Microsoft нанесла свой главный удар. Поначалу мало кто верил, что браузер Microsoft Internet Explorer, который тогда существовал в версии 2.0 и не представлял собой ничего выдающегося, сможет составить конкуренцию Netscape. Тем не менее, выпущенная летом 1996 г. версия Internet Explorer 3.0, которая поддерживала почти все расширения Netscape и обладала оригинальным и привлекательным интерфейсом, вызвала настоящий бум и очень быстро утвердилась в качестве "второго главного браузера". Сейчас Microsoft и Netscape делят рынок браузеров почти поровну, и окончательный исход их битвы не берется предсказать никто.

Язык **Java** зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов. Реализация проекта была начата на языке C++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента — языка программирования. Стало очевидным, что необходим неза-

висимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры. Рождению языка Java предшествовала довольно интересная история. В 1990 г. разработчик ПО компании Sun Microsystems Патрик Нотон (Patrick Naughton) понял, что ему надоело поддерживать сотни различных интерфейсов программ, используемых в компании, и сообщил исполнительному директору Sun Microsystems и своему другу Скотту МакНили (Scott McNealy) о своем намерении перейти работать в компанию NeXT. МакНили, в свою очередь, попросил Нотона составить список причин своего недовольства и выдвинуть такое решение проблем, как если бы он был Богом и мог исполнить все, что угодно.

Нотон, хотя и не рассчитывал на то, что кто-то обратит внимание на его письмо, все же изложил свои претензии, беспощадно раскритиковав недостатки Sun Microsystems, в частности, разрабатываемую в тот момент архитектуру ПО NeWS. К удивлению Нотона, его письмо возымело успех — оно было разослано всем ведущим инженерам Sun Microsystems, которые не замедлили откликнуться и высказать горячую поддержку своему коллеге и одобрение его взглядов на ситуацию в Sun Microsystems. Обращение вызвало одобрение и у высшего руководства компании, а именно, у Билла Джоя (Bill Joy), основателя Sun Microsystems, и Джеймса Гослинга (James Gosling), начальника Нотона.

В тот день, когда Нотон должен был уйти из компании, было принято решение о создании команды ведущих разработчиков, с тем чтобы они делали что угодно, но создали нечто необыкновенное. Команда из шести человек, с кодовым названием Green, ушла в добровольное изгнание, погрузившись в исследования бытовых устройств, таких как Nintendo, Game Boys, устройств дистанционного управления. Команда Green пыталась найти средство, с помощью которого можно было бы установить взаимодействие между этими устройствами. Вскоре стало ясно, что такие электроприборы как видеомагнитофоны, проигрыватели лазерных дисков, стереосистемы были реализованы на разных процессорах. Это означало, что если производитель захочет добавить телевизору или видеомагнитофону дополнительные функции или характеристики, он будет зажат в рамках средств, зашитых в аппаратное обеспечение. Эта проблема, в сочетании с ограниченностью памяти микросхем этих устройств, выдвинула новый подход к программированию, который должен был стать ведущим на рынке бытовой электроники. Команда приступила к разработке нового объектно-ориентированного языка программирования, который был назван Oak (дуб), в честь дерева, росшего под окном Гослинга. Вскоре компания Sun Microsystems преобразовала команду Green в компанию First Person. Новая компания обладала интереснейшей концепцией, но не могла найти ей подходящего применения. После ряда неудач ситуация для компании неожиданно резко изменилась. Был анонсирован Mosaic — так родился World Wide Web, с которого началось бурное развитие Internet. Нотон предложил использовать Oak в создании Internet-приложений. Так Oak стал самостоятельным продуктом. Вскоре были написаны Oak-компилятор и Oak-браузер "WebRunner". В 1995 г. компания Sun Microsystems приняла решение объявить о новом продукте, переименовав его в Java. Когда Java оказался в "руках" Internet, стало необходимым запускать Java-апплеты — небольшие программы, загружаемые через Internet. WebRunner был переименован в HotJava, а компания Netscape встала на поддержку Java-продуктов.

SQL (Structured Query Language, язык структурированных запросов) предназначен для создания и работы с реляционными базами данных (relational database), которые представляют собой наборы связанных данных, хранящихся в таблицах. Для управления реляционными базами данных используются программы, называемые системами управления базами данных, СУБД (database management systems, DBMS). На рынке предлагается большое разнообразие таких программ: DB2 Universal Database, Oracle, Microsoft SQL Server, Microsoft Access, Sybase Adaptive Server Enterprise, SQL Anywhere Studio, Informix Information Server, Ingres, MySQL, mSQL и другие. Для выполнения операций с базами данных эти СУБД используют SQL. Благодаря своей элегантности и машинной независимости, а также поддержке промышленными лидерами в технологии реляционных баз данных, SQL был признан стандартным языком и в обозримом будущем сохранит свои позиции. Стандарт SQL является совместной разработкой ANSI (American National Standards Institute) и ISO (International Organization for Standardization). С 1986 г. эти организации опубликовали серию стандартов SQL, каждый последующий из которых является объединением предыдущих. Эти стандарты имеют тенденцию опережать развитие компьютерной индустрии на несколько лет. В 2000 г. большинство коммерческих продуктов соответствовало стандарту SQL92. Даже сейчас еще не все элементы этого стандарта широко реализованы. Впрочем, многие коммерческие СУБД расширяют SQL за рамки определений ISO, добавляя полезные возможности. В любой из версий отклонения от стандарта должны описываться в документации на программный продукт.

И все у нас с Маршалом78 было бы хорошо, если бы мы слишком много не говорили о создании портала при минимуме знаний о работе в веб. О каком платном хостинге можно было говорить, когда я толком не знал, как настроить веб-сервер? Чувствую, читатель уже почесывает свои компьютерные руки, чтобы написать мне письмо: "APACHE ставить нужно было!" А я с этим не согласен. Проследите мою логику. Большинство пользователей работают с Windows. Так зачем им устанавливать UNIX-программу, с жуткой консолью, которая может только напугать пользователя? Хотя уже имеется отличная графическая надстройка, ее по-прежнему нужно настраивать. А мне, ленивому Windows-юзеру, это нужно? Конечно, нет. Под любимые или ненавистные, кому как, Окна существует множество веб-серверов. Но почему-то все упорно ставят APACHE.

Вот некоторые популярные Windows-сервера.

Falcon Web Server (www.blueface.com) — "сокол", именно так переводится его название. Веб-сервер способен работать на небольшом сайте, с посещаемостью 50...80 загрузок в минуту. Программа умеет выполнять ISAPI и приложения WinCGI в виртуальных директориях. Сервер Falcon разрабатывался так, чтобы быть по возможности совместимым с IIS. Следовательно, вы, как разработчик, можете разрабатывать и тестировать свои скрипты в Falcon, а затем развертывать их в любое время без модификаций.

Local web (http://www.intranet-server.co.uk/) — простой приятный веб-сервер, который поддерживает Perl. Имеет вполне доходчивый графический интерфейс. Все настройки производятся буквально парой кликов мышки.

Omni httpd (http://www.omnicron.ca) — так же как и все win-сервера, имеет очень приятный вид и простой интерфейс. Поддерживает работу как с PHP, так и с Perl. Я считаю его наиболее удачной программой для домашних условий.

Единственное "НО", которое может возникнуть у вас при работе с пречисленными утилитами — это регистрация. Но, как говорят: "Где наше не пропадало?" В завершение этих заметок мне остается лишь тяжело вздохнуть по поводу Маршала78 и его портала. Еще раз хочется вспомнить те хорошие времена, когда я не знал ничего, но хотел очень многого. Ведь прав был Шекспир?

Как часто эти найденные строки
Для нас таят бесценные уроки.

При работе над статьей была использована информация с сайтов http://dimexio.hoha.ru, http://sql.itsoft.ru, http://www.codenet.ru, http://z3950.uiggm.nsc.ru, http://www.weberam.ru и материал Дмитрия Кирсанова http://www.kirsanov.com.

Браузер Opera: еще несколько советов

И.РОЩИН,
г.Москва,
Fido: 2:5020/689.53
ZXNet: 500:95/462.53
E-mail: bestview@mtu-net.ru
WWW: http://www.ivr.da.ru

Введение

В этой статье я продолжаю начатый в [1] рассказ о некоторых настройках и приемах работы с Opera 5.12б, которые я использую. Сразу замечу, что для других версий этого браузера изложенные здесь сведения могут не подойти. И, конечно, чтение этой статьи ни в коей мере не может заменить изучения прилагаемого к Opera 5.12 подробного (около 600 Кб) описания.

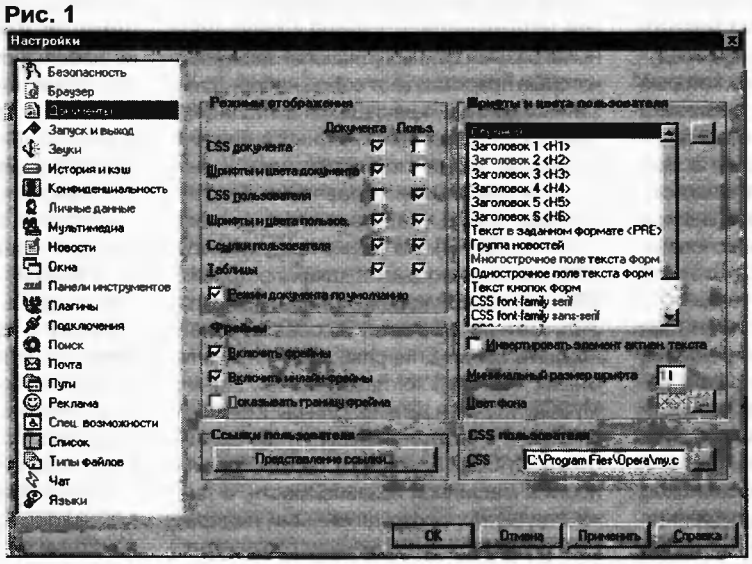
Настройка цветов

Иногда попадаются web-страницы с не очень удачным сочетанием цветов текста и фона. В результате — или текст плохо различим, или текст хорошо различим (например, черный текст на белом фоне), но читать все равно неудобно из-за неприятного мерцания (чем больше яркость фона, тем больше амплитуда мерцания, и тем оно заметнее — особенно при невысокой частоте кадров).

Как изменить эти цвета? Тут помогает наличие в "Опере" двух режимов отображения web-страниц: режима документа и режима пользователя. Каждый из этих режимов можно настроить по своему вкусу (Настройки\Документы\Режимы отображения — **рис.1**). Также можно выбрать, в каком из этих режимов будут открываться новые окна.

Так вот, для режима пользователя можно отключить шрифты и цвета документа, оставив включенными шрифты и цвета пользователя (как на рис.1). На этом же экране настроек можно выбрать, каким именно будет цвет текста и фона в режиме пользователя, причем цвет текста можно указывать отдельно для различных элементов HTML (для обычного текста, для заголовков первого уровня и т.д.). Я оставил предлагаемый по умолчанию черный цвет текста, а в качестве цвета фона выбрал светло-серый. И как только мне не нравится сочетание цветов на какой-либо web-странице, я одним щелчком мыши (**рис.2**) переключаюсь с режима документа (установленного у меня по умолчанию для новых окон) на режим пользователя, и страница отображается с выбранными мной цветами текста и фона.

Рис. 1



Посещенные и непосещенные ссылки

Не знаю, почему, но некоторые web-дизайнеры устанавливают в HTML-документах один и тот же внешний вид посещенных и непосещенных ссылок. В результате посетителю непонятно, какие ссылки он уже посещал, а какие нет. И тут тоже поможет наличие в "Опере" двух режимов отображения. Достаточно настроить режим пользователя так, чтобы шрифты и цвета документа в нем были отключены, а ссылки пользователя — включены (как на рис.1). Чтобы определить желаемый внешний вид ссылок, выберите пункт "Представление ссылки" на том же экране настроек. Тогда в дальнейшем при переключении в режим пользователя посещенные и непосещенные ссылки будут выглядеть именно так, как вам нужно.

Рис. 2

Закладки и сайты с фреймами

Допустим, вы часто заходите на какой-либо сайт (создав закладку с его адресом), и на этом сайте используются фреймы. Соответственно, при посещении сайта сначала загружается описание фреймов, а потом — содержимое каждого фрейма.

Но если вы заходите на сайт только для того, чтобы увидеть содержимое одного конкретного фрейма (например, "Новости"), то выгоднее сразу создать закладку с адресом документа, который загружается в этом фрейме. Для этого достаточно нажать правую кнопку мыши, когда курсор находится внутри фрейма, и в появившемся меню выбрать "Фрейм", "Добавить в закладки". Тогда только этот один документ и будет загружаться, что сэкономит ваше время. А для отображения этого документа будет использоваться вся площадь окна, что повысит удобство чтения.

Надо только учесть вот что — иногда web-дизайнеры вставляют в такие документы Java-скрипт, который определяет, загружен ли документ во фрейме, и если нет, то восстанавливает структуру фреймов. Понятно, что в этом случае экономии не будет. Впрочем, выполнение Java-скриптов в "Опере" можно отключить (Настройки\Плагины\JavaScript).

Еще о закладках

В [1] я уже упоминал, что в закладках можно хранить ссылки на HTML-файлы, находящиеся на вашем компьютере. В связи с этим еще один совет — если у вас в закладках имеются ссылки на большое количество файлов, находящихся в каком-то каталоге, и вы переместили или переименовали этот каталог, то, чтобы адреса файлов в закладках соответствовали новому местоположению файлов, лучше не исправлять адрес в каждой закладке по отдельности, а в текстовом редакторе (например, в Dos Navigator'е) открыть файл Opera\Opera5.adr, в котором содержится информация о закладках (это обычный текстовый файл), и с помощью "Поиск/Замена" исправить нужные адреса автоматически.

Только учтите, что нельзя редактировать файл с информацией о закладках, когда "Опера" запущена. Дело в том, что при запуске она считывает этот файл, а при окончании работы — записывает, так что отредактированный файл окажется перезаписан.

Повышение скорости отображения документов

Здесь я расскажу о некоторых на-

стройках, которые использую для увеличения скорости отображения документов (имеется в виду не повышение скорости загрузки, а повышение скорости прорисовки их на экране).

Во-первых, скорость прорисовки зависит от используемого графического режима. По моим наблюдениям (на 486 DX2/66), чем выше разрешение экрана и больше количество цветов, тем медленнее выполняется скроллинг документов (впрочем, на более мощных компьютерах это замедление может быть практически незаметным). Учитывая это, я работаю в режиме 640x480, 256 цветов.

Во-вторых, я отключил двойную буферизацию при выводе на экран (Настройки\Окна\Предотвращать мерцание). Помимо некоторого ускорения, при этом уменьшается и расход памяти, что, в свою очередь, также влияет на скорость, если размер ОЗУ недостаточен (у меня всего 8 Мб), и используется медленная виртуальная память.

В-третьих, в "Опере" имеются настройки, управляющие выводом рисунков в форматах GIF и JPEG (Настройки\Мультимедиа). Для GIF-рисунков я отключил анимацию (впрочем, это не действует на рекламный баннер, который показывается в незарегистрированной копии "Оперы"). Для JPEG-рисунков имеется целый набор опций, позволяющих выбрать оптимум между качеством рисунков и временем, уходящим на их обработку. Подробнее об этих опциях вы можете прочитать в описании "Оперы", а здесь я приведу свой вариант настроек (**рис.3**).

Из этих настроек видно, что я предпочитаю увеличение скорости за счет снижения качества рисунков. Вы, разумеется, можете предпочесть обратное, и настройки будут иными.

Форматируем абзацы

При отображении HTML-документов абзацы по умолчанию отделяются друг от друга пустой строкой, а абзацный отступ отсутствует. Именно так оформлено подавляющее большинство документов в Интернете. А хотелось бы видеть документы с привычным форматированием абзацев, когда они отделяются друг от друга не пустой строкой, а с помощью абзацного отступа (как, например, в этом тексте). При этом еще и больше текста на экране поместится :-).

И "Опера" позволяет это сделать! Для этого надо использовать пользовательскую таблицу стилей (CSS пользователя).

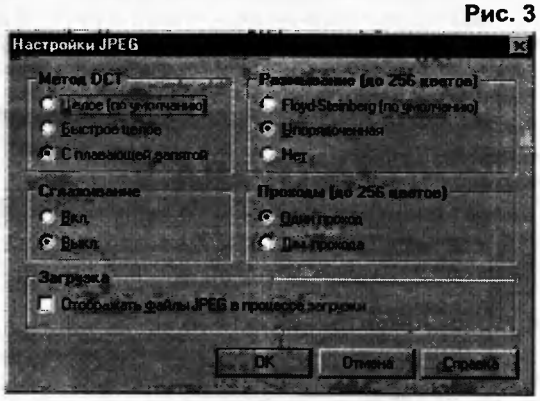
Наберите в текстовом редакторе следующее:

```
p {text-indent: 2em; margin-top: 0; margin-bottom: 0}
p+table,table+p,p+div,div+p {margin-top: 1em}
```

Это и есть CSS пользователя. Небольшой комментарий: в первой строке этой таблицы стилей для абзацев ("p") определяется абзацный отступ ("text-indent") величиной 2em (т.е. две относительных единицы — при этом величина отступа будет пропорциональна размеру шрифта) и определяется нулевой отступ абзаца от предыдущих ("margin-top") и последующих ("margin-bottom") блоков текста. Вторая строка, в принципе, не обязательна, и нужна для того, чтобы таблицы и блоки, ограниченные тегами и , отделялись от абзацев пустой строкой.

Сохраните набранный текст в файле, например, с именем "my.css". Затем в настройках "Оперы" выберите "Документы" (рис.1) и в пункте "CSS пользователя" укажите имя этого файла с полным путем к нему. После этого в пункте "Режимы отображения" в строке "CSS пользователя" укажите, будет ли использоваться эта таблица стилей в режиме документа и в режиме пользователя. Эти настройки, кстати, в любое время можно изменить.

Рис. 3



И еще один пример использования таблицы стилей — если хотите, чтобы текст в абзацах выравнивался по левому и правому краю за счет изменения промежутков между словами, добавьте в файл такую строку:

```
p {text-align: justify}
```

А если хотите такого же выравнивания текста и в списках, то замените в начале этой строки "p" на "p,ul,ol,li".

Вообще, таблицы стилей — очень мощное средство для управления внешним видом документа, но чтобы использовать это средство в полную силу, его надо как следует изучить. Рекомендую начать с прочтения перевода официальной спецификации CSS2, который можно найти на сайте http://pyramidin.narod.ru.

Распаковка сжатых сохраненных документов

Сохраняя некоторые HTML-документы, я обнаружил, что впоследствии открыть их почему-то не удается. При попытке их просмотра с помощью Dos Navigator'а выяснилось, что они упакованы GZIP'ом (несмотря на расширение "html"). Позже я узнал, что некоторые сайты, чтобы сократить объем передаваемой информации, сжимают передаваемые документы. Opera 5.12 распаковывает такие документы при отображении на экране, но почему-то не делает этого при их сохранении.

Как же распаковать такие файлы? Я использую для этого freeware-программу UNTGZ 0.95 by Tillmann Steinbrecher. Скачать ее можно, например, по ссылке ftp://ftp.simtel.net/pub/simtelnet/msdos/arcers/untgz095.zip.

Для распаковки надо указать в командной строке следующие параметры: -d имя_1 имя_2. Здесь: имя_1 — имя упакованного файла, а имя_2 — имя, под которым будет записан распакованный файл.

Еще о сохраненных документах

Не во всех сохраненных HTML-документах содержатся сведения об их кодировке. При открытии в "Опере" документов с неуказанной кодировкой приходится самому выбирать нужную кодировку в меню (вместо обычно выбранного пункта "Автоопределение"). При этом еще и не всегда получается угадать кодировку с первого раза. К тому же, выбранная кодировка действует не только на текущий документ, но и на все вновь открываемые, даже если они на самом деле в другой кодировке. Налицо неудобство. А избежать его можно, предварительно обработав сохраненные HTML-документы с помощью специальной программы, добавляющей в них сведения о кодировке. Эта программа подробно описана в [2], а скачать ее можно с сайта журнала или с моей web-страницы.

Закрытие текущего окна документа

Казалось бы, простейшая операция, но и тут есть на что обратить внимание.

Если текущее окно документа максимизировано, то, как видно на **рис.4**, кнопка закрытия этого окна расположена прямо

Рис. 4

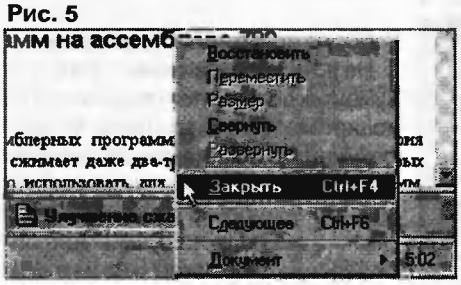
под кнопкой закрытия самой "Оперы". В результате, случайно нажав не ту кнопку, можно закрыть "Оперу", и придется заново запускать ее. Это особенно неприятно, если вы подключены к Интернету с повременной оплатой: приходится ждать, пока "Опера" запустится заново, а время идет...

Для предотвращения случайного закрытия "Оперы" надо проследить, чтобы была включенной опция "Подтверждать выход" (Настройки\Запуск и выход).

Кстати, для закрытия текущего окна можно использовать еще и специальный "мышиный жест" — нажать правую кнопку мыши, сдвинуть мышь вправо, влево, вправо и отпустить правую кнопку. Этот способ удобнее тем, что не надо никуда целиться курсором мыши.

Замечу, что я не использую описанные выше способы закрытия текущего окна, если имеется еще хотя бы одно окно, в котором документ еще не начал загружаться. Дело в том, что, по моим наблюдениям (в Opera 5.12), если в другом окне начинает загружаться документ с фреймами, то его окно аатоматически становится активным. В результате закрыться может именно ставшее активным окно, а не то, которое надо было закрыть.

Рис. 5



В такой ситуации для закрытия окна я предпочитаю щелкнуть правой кнопкой мыши на соответствующей "кнопке" в панели окон "Оперы" и выбрать команду "Закрыть" в появившемся меню (**рис.5**).

Поиск документов в кеше

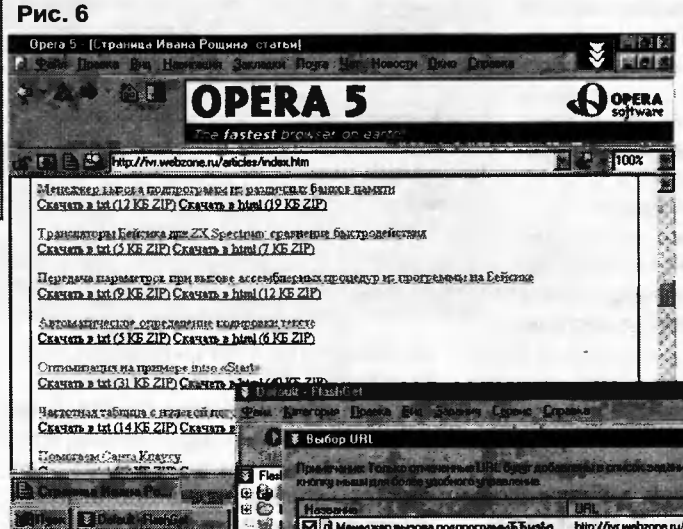
Допустим, вы недавно посетили какую-то web-страницу, она сохранилась в кеше "Оперы", а потом вы решили еще раз ее посмотреть, но не помните ни ее точный адрес, ни заголовок, по которому ее можно было бы быстро найти в истории (Окно\Специальное\История). Просматривать же все содержимое кеша (Окно\Специальное\Кеш) — слишком долго. В таком случае, если только вы помните какую-нибудь содержащуюся на этой странице строку текста, для ее поиска в кеше можно использовать Dos Navigator (я пользуюсь DN OSP 2.7.0 — http://www.dnosp.ru). Он позволяет искать заданную строку в файлах с указанным расширением, причем файлы могут быть в любой из кодировок DOS, WIN, KOI. Искать надо в каталоге Opera\Cache4, в файлах с расширением htm. Чтобы затем узнать адрес найденного документа, надо запомнить имя файла, а потом в "Опере" открыть окно "Кеш" и посмотреть, какой адрес соответствует документу с этим именем файла.

К сожалению, при таком поиске не будут найдены документы, сжатые GZIP'ом (в кеше они тоже хранятся в сжатом виде). Впрочем, сжатые документы попадаются довольно редко.

Скачивание файлов

Иногда необходимо скачать большое количество файлов по ссылкам с web-страницы (**рис.6**). Давать задание на скачивание каждого файла в отдельности (с помощью самой "Оперы" или с помощью FlashGet) очень утомительно. Удобнее сделать так — сначала сохраняем просматриваемую страницу на диск и копируем адрес этой страницы в буфер обмена (если страница содержит фреймы, то сохранять на диск надо именно тот фрейм, в котором находятся ссылки, и копировать в буфер обмена надо тоже адрес именно этого фрейма). Затем в главном меню FlashGet выбираем "Файл", "Импортировать из

Рис. 6



HTML-файла" и открываем файл с сохраненной страницей. В появившемся окне "Ввод URL" вставляем из буфера обмена адрес страницы. После этого в появившемся списке (**рис.7**) оставляем отмеченными только ссылки на те файлы, которые надо скачать. Остается нажать "OK" и указать параметры их скачивания.

Кстати, в списке содержатся адреса не только ссылок, но и рисунков, находящихся на web-странице. Так что при желании можно загружать таким способом и рисунки.

Единственное неудобство — если названия ссылок указаны по-русски, то они будут правильно отображаться во FlashGet только в том случае, если web-страница была в кодировке Windows-1251.

И еще один прием. В [1] я уже упоминал, что в "Опере" можно так настроить внешний вид ссылок, чтобы при наведении курсора на ссылку рядом показывался ее адрес (Настройки\Панели инструментов\Подсказка\Адреса ссылок). Тогда, наводя курсор на ссылки, можно узнать адреса скачиваемых файлов. И если имена файлов или пути к ним идут по порядку (например, так: file_099, file_100, file_101..., или так: file_F, file_G, file_H..., или даже так: dir05\index.html, dir06\index.html, dir07\index.html...), то удобно воспользоваться функцией FlashGet "Добавить пакетное задание", скопировав перед этим в буфер обмена адрес первого скачиваемого файла.

В появившемся окне вставляем из буфера обмена в строку "URL" ранее скопированный адрес, заменяем в нем изменяемую часть на символы "(*)" и указываем границы диапазона изменения (**рис.8**).

Параметр, скрывающийся

Рис. 7

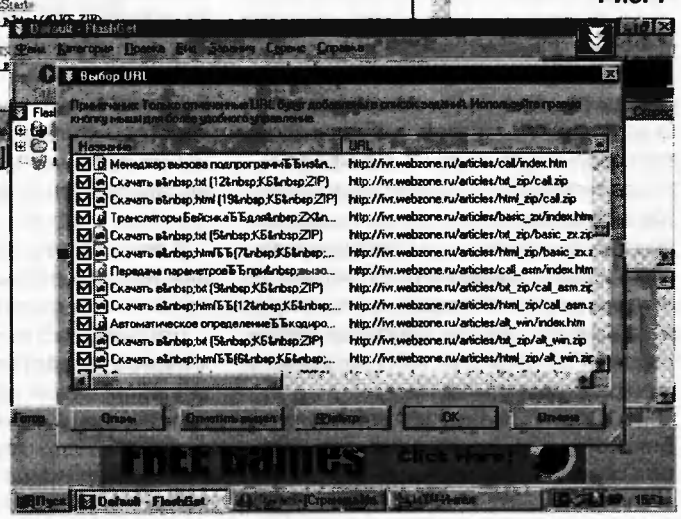
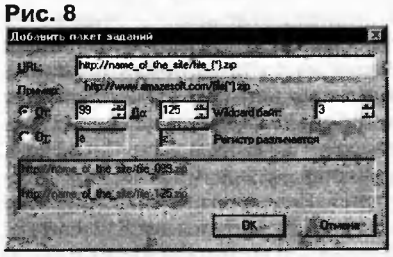


Рис. 8



под малопонятным именем "Wildcard байт" — это просто минимальное количество цифр в записи числа, подставляемого вместо "(*)". Если число записывается меньшим количеством цифр, оно будет дополнено слева нулями.

В нижней части окна отображаются получающиеся адреса первого и последнего файлов в формируемом пакетном задании. Убедившись, что они совпадают с ожидаемыми, остается нажать "OK" и указать параметры скачивания.

Как отменить "мышиный жест"

Многие часто используемые операции в "Опере" можно выполнять с помощью так называемых "мышиных жестов" (ранее я уже упоминал об одном из них, используемом для закрытия окна документа). "Мышиный жест" представляет собой нажатие правой кнопки мыши, перемещение мыши в определенном направлении (или в определенной последовательности направлений) и отпускание правой кнопки.

Информацию о том, какие операции с помощью каких жестов выполняются, вы можете найти в описании "Оперы" (Справка/Мышь), а здесь я расскажу о том, что в описании не упоминается. Допустим, вы начали делать какой-то "жест", но вдруг по какой-то причине решили его отменить. Например, вы уже почти сделали "жест" для закрытия окна, осталось только отпустить правую кнопку мыши, но вдруг что-то на просматриваемой странице привлекло ваше внимание, и вы передумали закрывать окно. Но если отпустить кнопку мыши, то "жест" сработает, и окно закроется! Что же делать? Чтобы отменить "жест", поместите курсор мыши вне окна "Оперы" (например, на панель задач Windows), и только затем отпустите кнопку мыши.

Настройка панели кнопок

Может быть, вы заметили, что панель кнопок в "Опере", в режиме "простая", по умолчанию выглядит как на **рис.9а**, а на скриншотах моей "Оперы" эта панель выглядит немного по-другому **(рис.9б)**. У меня вместо кнопки "Открыть новое окно документа" (которой я почти не пользуюсь) находится кнопка "Домашняя страница" (которой я пользуюсь чаще).

Рис. 9а

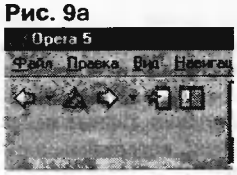
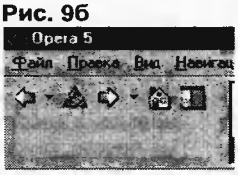


Рис. 9б



Как я это сделал? В файле Opera\Buttons\DefSmall\buttons.ini есть следующие строки:

```
[Main Button Bar]
Version          = 1                    # Version, this must be <> 0
#Button number   = show when full; show when simple; id;
                   spacing in front;
Button 0         = 1;1;10032;0          # Back
Button 1         = 1;1;10056;0          # Reload
Button 2         = 1;1;10031;0          # Forward
Button 3         = 1;0;10072;0          # Home
Button 4         = 1;1;10000;0          # New
Button 5         = 1;0;10004;0          # Print
Button 6         = 1;1;13010;0          # Toggle hotlist
Button 7         = 1;0;10019;0          # Tile
```

В этих строках определяется, какие кнопки будут отображаться на панели в "полном" и "простом" режиме (режим устанавливается в меню "Вид\Панель кнопок" или при щелчке правой кнопкой мыши на панели кнопок).

Как следует из комментария, в описании каждой кнопки (Button 0...Button 7) первый параметр после "=" определяет, будет ли эта кнопка отображаться в "полном" режиме панели (0 — нет, 1 — да), а второй параметр — в "простом" режиме панели.

Таким образом, чтобы в "простом" режиме не отображалась кнопка "New" ("Открыть новое окно документа") и отображалась кнопка "Home" ("Домашняя страница"), я произвел следующие изменения:

```
Button 3     = 1;1;10072;0      # Home
Button 4     = 1;0;10000;0      # New
```

При желании можно изменить и порядок расположения кнопок в панели. Кнопки отображаются так: сначала Button 0, потом Button 1 и т.д. Таким образом, если надо, например, чтобы первой отображалась кнопка "Reload", а второй — "Back", их надо переобозначить: "Reload" — как Button 0, а "Back" — как Button 1:

```
Button 1     = 1;1;10032;0      # Back
Button 0     = 1;1;10056;0      # Reload
```

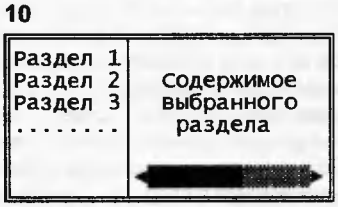
После редактирования файла buttons.ini, если "Опера" уже запущена, ее надо перезапустить, чтобы изменения вступили в силу.

Использование связанного окна

Встречаются сайты, где экран делится по вертикали на два фрейма, в одном из этих фреймов отображается список разделов, а в другом — содержимое выбранного раздела. Если такой сайт рассчитан на большее разрешение экрана, чем установлено у пользователя, то ширина фрейма с содержимым может оказаться недостаточной, и появится полоса горизонтальной прокрутки (это схематично показано на **рис.10**).

Горизонтальная прокрутка делает просмотр очень неудобным. При этом еще бывает и так, что нельзя сдвинуть разделительную линию между фреймами, чтобы выделить под второй фрейм больше места и избавиться от прокрутки. В этом случае как раз и может пригодиться такая возможность "Оперы" как создание связанного окна.

Рис. 10



Для создания окна, связанного с текущим (главным) окном, надо выбрать соответствующий пункт в меню "Окно". После этого все ссылки из главного окна будут открываться в связанном с ним окне. Таким образом, когда вы в главном окне щелкнете по ссылке с названием раздела, содержимое этого раздела будет отображено в связанном окне, занимая всю его ширину, и просматривать его можно будет без горизонтальной прокрутки (если, конечно, ширина содержимого не является уж слишком большой).

Кстати, может оказаться более удобным сделать ширину связанного окна меньше ширины экрана (но так, чтобы необходимость в горизонтальной прокрутке все же не возникла). Дело в том, что короткие строки текста удобнее читать, чем длинные (не зря в газетах и журналах текст располагают в несколько колонок).

Литература

1. И.Рощин. Браузер Opera: несколько советов. — Радиомир. Ваш компьютер, 2003, №3, С.19.

2. И.Рощин. Добавление в HTML-документы информации об их кодировке. — Радиомир. Ваш компьютер, №10, С.13.

ЛИСТАЯ СТРАНИЦЫ

RAID-массивы до недавнего времени применялись в основном только в серверах. Однако не далек тот день, когда они будут использоваться и в обычных ПК. **Что такое RAID-массивы и каково их преимущество**, рассказывает *Д.Патраков* в статье "Дисковое сообщество" **(CHIP, №8/2003, С.54...61)**.

Понятие избыточного массива независимых дисков (RAID) своим появлением обязано работе специалистов из Калифорнийского университета в Беркли. В статье, опубликованной в 1987 г., впервые была предложена идея создания на основе нескольких жестких дисков системы, превосходящей по емкости, производительности и отказоустойчивости самые дорогие винчестеры. Кроме математического обоснования надежности предложенных систем, были также определены различные уровни RAID (с первого по пятый), в той или иной степени обеспечивающие повышенную отказоустойчивость, а также другие преимущества по сравнению с отдельными дисковыми накопителями.

Под дисковым массивом понимают объединенный в одну группу набор жестких дисков. Когда говорят о RAID, нужно отличать физические диски и их массивы от логических. Массивы физических дисков могут быть разделены или сгруппированы для создания одного или нескольких логических массивов. Последние, в свою очередь, могут быть поделены на логические диски, с которыми будет работать операционная система. Логические диски рассматриваются как отдельные жесткив диски, соответственно, они могут быть разбиты на разделы и отформатированы.

Контроллер RAID управляет тем, как будут храниться и передаваться данные между физическими и логическими массивами. Он позволяет операционной системе работать с логическими дисками, не обращая внимания на схему реализации RAID-массива. Функции RAID-контроллера могут быть реализованы аппаратно или программно. Аппаратная реализация больше подходит для уровней RAID, требующих интенсивных вычислений. Программная реализация выглядит более привлекательной, однако центральный процессор может в конечном счете оказаться перегруженным тем объемом данных, вычисления над которыми ему придется осуществлять.

Зеркалирование (mirroring) означает хранение двух копий данных на двух различных жестких дисках или массивах дисков. Этот подход является одним из двух технических приемов использования избыточных данных, которые применяются в RAID-массивах для защиты от их потери. Преимущество очевидно — при выходе из строя одного жесткого диска или дискового массива система сможет продолжить работу за счет использования второй копии данных. Время простоя при использовании зеркалирования минимально, а восстановление данных весьма просто: нужно лишь воссоздать данные из рабочей копии.

Контроль четности — это еще один способ использования избыточности данных в RAID-массивах. Автор поясняет смысл этого понятия на следующем примере. Для X элементов данных создается и хранится еще один — X+1. Если будет потерян любой из X+1 элементов данных, он может быть восстановлен при условии, что известны остальные X элементов.

В этом случае для хранения данных о четности для двух жвстких дисков понадобится лишь один дополнительный диск. Это является очевидным преимуществом использования четности вместо зеркалирования, однако уровень отказоустойчивости в этом случав окажется не столь высоким.

Данные о четности не обязательно будут храниться на одном диске — они могут быть распределены по всему дисковому массиву. Кроме того, как и в случае с зеркалированием, здесь может быть использовано разделение данных. Основным ограничением в данном случае является то, что алгоритм подсчета четности, как правило, требует значительных вычислительных мощностей, что означает необходимость использования аппаратного RAID-контроллера. Процесс автоматического восстановления данных при использовании данных о четности также осуществляется аппаратным контроллером, и даже в этом случае он занимает больше времени, чем при использовании зеркалирования.

Зеркалирование и контроль четности повышают надежность хранения данных. Разделение данных (striping) используется для повышения производительности дискового массива за счет распределения данных по всем его дискам. Предположим, что на одном жестком диске хранится большой файл. Чтобы прочитать этот файл, придется ждать, пока жесткий диск считает его от начала до конца. Но если разбить файл на множество отдельных фрагментов и распределить их между несколькими дисками, все они могут быть считаны одновременно. Ожидание сократится до времени считывания одного фрагмента, поскольку все диски будут работать параллельно. То же будет верно и для записи большого файла на диск.

Разделение данных работает следующим образом. Каждый блок данных, поступающих в RAID-контроллвр, разбивается на более мелкие фрагменты. При этом существует два уровня разбиения данных: разделение на уровне байтов и на уровне блоков. Разделение на уровне байтов (byte level striping) включает в себя разделение данных на отдельные байты и их последовательное сохранение на жестких дисках. К примеру, если данные разбиваются по 16 байтов, а в системе используется четыре жестких диска, то первый байт сохраняется на первом диске, второй — на втором, и т.д. Пятый байт сохраняется на первом диске, и цикл повторяется. Иногда разделение на уровне байтов осуществляется по 512 байтов. Разделение на уровне блоков (block level striping) включает в себя разбивку данных на блоки заданного размера. Размер этих блоков также называется размером страйпа (stripe size). Как правило, в зависимости от реализации RAID-контроллера, имеется возможность использования различных размеров страйпа.

Далее автор кратко характеризует уровни RAID. RAID 0 является наиболее простым вариантом и использует лишь разделение данных. Избыточность данных здесь не используется. Этот уровень обеспечивает высокую производительность, а также наименьшую стоимость реализации за счет отсутствия дополнительной емкости хранения, однако надежность его невелика. Для организации RAID 0 необходимы как минимум два жестких диска, по возможности идентичных, а характеристики его будут определяться главным образом контроллером. Этот уровень RAID приобрел особую популярность на рынке недорогих систем благодаря его низкой стоимости и ощутимого выигрыша в производительности.

Уровень RAID 1 реализуется с использованием зеркалирования. Две идентичные копии данных хранятся на двух дисках. При выходе из строя одного из них система сохранит работоспособность за счет того, что данные будут доступны с другого диска. Восстановление утерянной копии данных является в этом случае очень простой операцией. Такой подход обеспечивает избыточность данных и определенную защиту от сбоев. Этот уровень подходит для тех случаев, где сохранность данных является критичной. Относительная простота и низкая стоимость реализации этого уровня способствовали росту его популярности на рынке недорогих систем. Большинство RAID-контроллеров в наши дни реализуют RAID 1 в том или ином виде.

RAID 2 использует разделение данных на уровне битов с применением для контроля ошибок кода Хэмминга. Используемый здесь подход в чем-то схож с обычным разделением данных с применением четности: данные разделяются на уровне битов и распределяются между жесткими дисками, хранящими данные и коды контроля. При записи данных коды Хэмминга вычисляются и обновляют информацию на отдельных дисках. При чтении данных из дискового массива эти коды используются для проверки искажения данных, возникших с момента их записи. Использование кода Хэмминга позволяет

обнаруживать искажения до двух битов, а искажения а один бит исправлять "на лету". Однако сложность и высокая стоимость RAID-контроллера, а также необходимость использования большого числа дисков привели к тому, что в наши дни этот уровень RAID практически нигде не используется.

Уровень RAID 3 использует разделение данных на уровне байтов с выделенным диском для хранения четности. Иными словами, данные распределяются по дисковому массиву на уровне байтов, при этом один из дисков выделяется для хранения избыточной информации. Идея состоит в том, чтобы обеспечить высокий уровень производительности за счет разделения данных, а отказоустойчивость — за счет хранения информации о четности. Для реализации этого уровня необходимы по меньшей мере три жестких диска. И хотя разделение данных благотворно сказывается на производительности, использование данных о четности ощутимо замедляет запись. Кроме того, интенсивные вычисления, связанные с нвобходимость обновления информации о четности при каждой операции записи, вынуждают использовать аппаратный контроллер. RAID 3 хорошо подходит для работы с большими файлами благодаря малому размеру страйпа.

RAID 4 во многом схож с предыдущим уровнем, единственное различие заключается в использовании разделения данных на уровне блоков вместо разделения на уровне байтов. Преимущество такого подхода состоит в возможности изменения размера страйпа в соответствии с той или иной областью применения. Этот уровень часто предстаалается как смесь RAID 3 и RAID 5, так как обладает выделенным диском для хранения четности, как в RAID 3, и разделением данных на уровне блоков, как в RAID 5. Вместе с тем, он обладает недостатками RAID 3 — необходимостью использования аппаратного контроллера и замедлением записи данных из-за необходимости пересчета информации о четности.

RAID 5 использует разделение данных на уровне блоков и распределенное хранение информации о четности. При этом данные распределяются вместе с хранимой информацией по всему дисковому массиву. Такой подход устраняет ограничение производительности, связанное с использованием одного жесткого диска для хранения данных о четности, однако это не избавляет от необходимости вычислять и записывать дополнительную информацию при каждой операции записи, что приводит к замедлению процесса. Отказоустойчивость обеспечивается за счет отделения информации о четности от данных самого блока. При этом в случае выхода из строя одного из жестких дисков вся информация на нем может быть восстановлена за счет данных на оставшихся дисках, хотя этот процесс несколько сложнее обычного ваиду распределенности данных о четности. Размер страйпа, как и при использовании RAID 4, может варьироваться в зависимости от области применения. RAID 5 — наиболее широко используемый уроаень RAID в наши дни. Для многих он представляется оптимальным сочетанием производительности, отказоустойчиаости и эффектианости использования дискового пространства.

RAID 6 фактически представляет собой расширение RAID 5, обеспечивающее дополнительную отказоустойчивость за счет использования второй независимой схемы распределенного хранения данных о четности. Данные разделяются на уровне блоков, в точности как в RAID 5, а дополнительная информация о четности вычисляется и распределяется по всему дисковому массиву. RAID 6 обеспечивает очень высокую отказоустойчивость, будучи способным противостоять множественным одновременным отказам дисковых накопителей. Оборотной стороной медали является весьма низкая производительность при записи данных, сложность и высокая стоимость контроллеров, а также менее эффективное использование дискового пространства.

Система RAID 7, за счет использовения специально созданного контроллера, работающего под управлением операционной системы реального времени, а также асинхронной обработки и кэширования запросов на операции ввода-вывода системы, является лидером в плане производительности. Она оказывается на 25...90% производительнее систем на базе одного дискового накопителя и в 1,5...6 раз более производительной, чем системы на базе любого другого уровня RAID. Однако платой за это является наибольшая стоимость такой системы в пересчете на 1 Мб, необходимость использования источника бесперебойного питания для предотвращения потерь информации, меньшие сроки наработки на отказ.

Привычная башенная конструкция корпусов ПК стала уж очень привычной. **Нельзя ли сделать корпус более компактным, учитывая возросшие возможности интегрированных в современные чипсеты звуковой и графической подсистем?** Оказывается, можно. Об этом рассказывает *Джон Л.Якоби* в заметке "Дело о невероятно уменьшившихся ПК" **(Мир ПК, №5/2003, С.26...27)**.

Автор рассказывает о двух компьютерах. Модель Navigator Pro компании Alienware стала пераым из компактных ПК, работающим под управлением ОС Windows XP Media Center Edition. Размеры корпуса — 30,5x17,8x20,3 см. Переднюю панель украшают размещенные на ней для удобства доступа разъемы CompactFlash, Memory Stick, Secure Digital и SmartMedia, а также дисковод DVD-RW/-R DVR-A04 компании Pioneer, два порта USB 2.0 и один порт FireWire, разъем для головной гарнитуры, порт для микрофона и один оптический аудиовыход.

Этот ПК является достаточно мощным благодаря 2,66 ГГц процессору Pentium 4, 512 Мб ОЗУ типа DDR SDRAM с частотой 333 МГц, и 120 Гб жесткому диску Western Digital со скоростью вращения 7200 об/мин. В стоимость комплекта не входят монитор и колонки, но их можно приобрести у той же компании.

На задней панели имеются два порта USB, разъем FireWire, обычный последовательный порт, аудиовход и выход, гнездо для подключения наушников и один порт Ethernet. Предусмотрены и два разъема для плат расширения: первый, стандарта AGP, занят 64 Мб видеоплатой GeForce4 Ti 4200, во второй, стандарта PCI, установлена плата телевизионного тюнера Emuzed Maui PCI PVR. Стоимость — 1999 USD.

Второй компьютер выпускается фирмой Sony. Модель VAIO PCV-W10 — это изящный серебристо-серый ПК типа "все в одном". Размеры этой системы, включая встроенный дисплей и складывающуюся клавиатуру, составляют всего лишь 33,4x26,1x50,0 см.

Наиболее примечательной особенностью серийной модели, о которой рассказывает автор, был ее широкоформатный ЖК-экран с пропорциями 1,66:1 и разрешением 1280x768 точек, который идеально подходит для просмотра широкоэкранных DVD-фильмов, воспроизводимых на комбинированном дисководе DVD-ROM/CD-RW. Завершают все это великолепие вмонтированные на боковых сторонах дисплея динамики с хорошим звуком.

Для решения большинства задач с лихвой хватает установленных в системе 1,6 ГГц процессора Celeron, 512 Мб оперативной памяти типа DDR SDRAM с тактовой частотой 266 МГц и 60 Гб жесткого диска, однако интегрированная графическая система, потребляющая для обработки видео 32 Мб оперативной памяти, может не удовлетворить запросы любителей игр.

В модели PCV-W10 нет ни параллельных, ни последовательных портов. Здесь установлены три порта USB 2.0 и два трехштырьковых порта I.Link (FireWire без проводов питания), рвзъем для карт Memory Stick, порты для модема и Ethernet, обычные аудиовход и выход, а также гнездо для наушников. Кроме того, покупатель получает предустановленную Windows XP и два разъема для PC-карт типа II вместо стандартных разъемов для плат расширения. Его стоимость — 1600 USD.

О перспективах еще одной дисплейной технологии рассказывает *С.Асмаков* в статье "Чернила цифровой эпохи" (**КомпьютерПресс, №8/2003, С.135...138**).

Широкое распространение персональных компьютеров, по мнению некоторых специалистов, должно было привести к повсеместному отказу от использования бумажных документов. Появился даже термин "безбумажная технология". Почему же этого не произошло? Это, по мнению автора, объясняется тем, что все современные дисплеи — ЭЛТ, ЖК и плазменные — являются излучающими приборами. Иными словами, картинка на экранах таких дисплеев светится, в то время как напечатанное на пережившей уже не одно столетие бумаге изображение отражает падающий на него свет. Недостаток любого излучающего дисплея — сильная зависимость качества экранного изображения от внешнего освещения. При ярком солнечном свете рассмотреть что-либо на излучающих дисплеях становится крайне затруднительно, не говоря уже о таких побочных эффектах как появление бликов на поверхности экрана и искаженная цветопередача. К тому же, и в полной темноте, и в условиях сильного затемнения работа с излучающим дисплеем вызывает повышенную утомляемость. Кроме того, каждый из типов ныне распространенных излучающих дисплеев имеет собственные недостатки: у ЭЛТ — это мерцание кадровой развертки, а у ЖК — ограниченный угол обзора и неравномерность подсветки экрана.

Конечно, было бы неверно утверждать, что все современные дисплеи никуда не годятся. Однако, для одних задач (например, для воспроизведения видео) они подходят, а для других (в частности, для чтения объемных текстовых документов) — не очень. Значительно повысить комфортность при работе с электронными текстовыми документами могло бы создание отражающих дисплеев. Однако воплощение столь простой и изящной идеи натолкнулось на отсутствие соответствующей технологии.

Выход из этой ситуации был предложен специалистами основанной в 1997 г. американской компании E Ink Corporation, которые разработали оригинальную дисплейную технологию, получившую название electronic ink (электронные чернила). Базовыми элементами таких чернил являются микрокапсулы, диаметр которых не превышает толщины чаловеческого волоса. Внутри каждой микрокапсулы находится большое количество пигментных частиц двух цветов: положительно заряженных белых и отрицательно заряженных черных, а все внутреннее пространство микрокапсулы заполнено прозрачной жидкостью.

Слой микрокапсул расположен между двумя рядами взаимно перпендикулярных гибких электродов (сверху — прозрвчных, снизу — непрозрачных), образующих адресную сетку. При подаче напряжения на два взаимно перпендикулярных электрода а точке их пересечения возникает электрическое поле, под действием которого в расположенной между ними микрокапсуле перераспределяются пигментные частицы. Частицы с одним зарядом собираются в верхней части микрокапсулы, а с противоположным — в нижней. Для того чтобы поменять цвет точки экрана с белого на черный или наоборот, достаточно изменить полярность напряжения, поданного на соответствующую пару электродов. Таким образом, пиксел экрана, соответствующий данной микрокапсуле, окрасится в черный либо в белый цвет; при этом пигментные частицы, сгруппировавшиеся в верхней части микрокапсулы, скроют от взгляда наблюдателя все частицы, сосредоточенные в ее нижней части.

Вышеописанная модель позволяет создавать монохромные дисплеи с однобитной разрядностью, то есть каждый из пикселов экрана может быть либо белым, либо черным. Если же один управляющий состоянием микрокапсулы электрод заменить двумя, то станет возможным формирование полутонов за счет закрашивания одной половины микрокапсулы в белый цвет, а другой — в черный.

В качестве подложки для создания дисплея на основе электронных чернил можно использовать практически любой материал: стекло, пластик, ткань и даже бумагу. А это, в свою очередь, открывает перспективы создания ультратонких гибких дисплеев, максимально близких по своим механическим и оптическим характеристикам к обычной бумаге. Каковы достоинства и недостатки новой технологии?

Дисплей на основе электронных чернил является отражающим прибором, поэтому изображение на нем будет хорошо читаться как в сумерках, так и в ярких лучах полуденного солнца. Аналогично изображению, отпечатанному на бумаге, картинка на экране такого дисплея будет отлично просматриваться под любым углом, причем без потери контрастности.

Еще одно важное преимущество электронных чернил — отсутствие какого бы то ни было мерцания. Элементы экрана остаются абсолютно неподвижными до того момента, когда происходит смена изображения.

Как известно, все виды излучающих дисплеев требуют постоянного питания для сохранения экранного изображения. В отличие от них, дисплеи на основе электронных чернил нуждаются в подаче питания лишь тогда, когда меняется содержимое экрана. Даже при полном отключении питания изображение на экране такого дисплея сохранится. А если добавить к этому отсутствие какой бы то ни было подсветки, то вполне очевидно, что уроаень энергопотребления дисплеев на основе электронных чернил как минимум на порядок ниже даже по сравнению с самыми экономичными ЖК-моделями, не говоря уже о столь "прожорливых" твхнологиях как ЭЛТ и ПДП.

В числе неоспоримых достоинста дисплеев на базе электронных чернил — чрезвычайно малые толщина и вес. Уже сегодня возможно создание дисплейных модулей, имеющих вдвое меньшую толщину и значительно меньшую массу, чем аналогичные изделия на ЖК-матрицах. А по мере совершенствования технологии электронных чернил толщину построенных на ее основе дисплейных модулей можно будет уменьшить еще в несколько раз.

Благодаря максимальной простоте конструкции и использованию стандартных производственных процессов, себестоимость изготовления дисплеев на основе электронных чернил будет значительно ниже по сравнению с аналогичными изделиями, построенными на базе ЖК- и OLED-матриц.

Разумеется, как и любое другое решение, дисплеи на базе электронных чернил не лишены определенных недоствтков. Судя по отсутствию информации о времени отклика, эти устройства (по крайней мере, на данном этапе своего развития) обладают довольно большой инерционностью и по этой причине мало пригодны для отображения видео и анимации. Также не совсем ясно, насколько устойчивы содержащиеся в микроквпсулах пигментные частицы к выцветанию под действием солнечных лучей. Учитывая принцип действия электронных чернил, весьма актуальной проблемой можно назвать разработку эффективной защиты таких дисплеев от разрядов статического электричества, а также от внешних электрических полей.

Однако руководители E Ink утверждают, что уже в ближайшее время дисплеи на базе электронных чернил появятся в самых разнообразных электронных устройствах: КПК, пейджерах, мобильных телефонах, наручных часах и т.п. Однако наиболее перспективной областью применения указанных дисплеев являются устройства для чтения электронных книг. Согласитесь, что читать большие объемы текста с неизлучающей поверхности, изображение на которой одинаково хорошо просматривается под любым углом и практически при любом освещении, гораздо комфортнее, чем с экрана даже самого ультрасовременного ЖК-дисплея. При этом, благодаря значительно более низкому уровню энергопотребления, время работы такого устройства на одном комплекте элементов питания будет в сотни раз больше по сравнению с аналогичным изделием, оснащенным ЖК-дисплеем такого же размера.

В будущем вполне возможно появление принципиально новых классоа информационных устройств, например, интерактивных электронных газет, снабженных экраном на базе электронных чернил и

беспроводным интерфейсом, через который будет происходить регулярное обновление содержимого страниц. Кстати говоря, электронная газета — это уже вполне реальная вещь. В октябре 1999 года E Ink в сотрудничестве со всемирно известной компанией Lucent Technologies начала работы по созданию прототипа подобного устройства, а уже год спустя состоялась его первая публичная демонстрация. Прототип представлял собой активно-матричный монохромный дисплей на базе электронных чернил площадью 25 кв.дюймов, изготовленный на гибкой пластиковой подложке. Причем для его изготовления была разработана недорогая технология нанесения на подложку пластиковых полупроводниковых элементов, формируемых методом высокоточной штампоаки. В отличие от наиболее распространенного в настоящее время литографического процесса, применение штамповки позволяет значительно удешевить себестоимость продукции и сократить производственный цикл. Разработаны пластиковые транзисторы, обладающие такими же электрическими характеристиками, как и изготавливаемые по традиционной технологии тонкопленочные транзисторы на основе кремния. Однако при этом пластиковые транзисторы обладают гибкостью, более высокой механической прочностью и значительно меньшей массой.

Еще одной областью применения таких дисплеев могут стать информационные табло в магазинах, на станциях пассажирского транспорта, на вокзалах и т.п. Благодаря низкой себестоимости производства, изготовление "электронно-чернильных" дисплеев большой площади обойдется значительно дешевле по сравнению с ЖК- или плазменными панелями, не говоря уже о таких немаловажных достоинствах как хорошая читаемость при ярком солнечном свете и чрезвычайно низкий уровень энергопотребления.

В феврале 2001 г. было заключено долгосрочное соглашение о сотрудничестве с одним из мировых лидеров в производстве полупроводниковых приборов — компанией Royal Philips Electronics. В рамках этого соглашения компания Philips уже на протяжении даух лет занимается разработкой интегрированных электронных компонентов, а также управляющих модулей и прочих необходимых подсистем для дисплеев на базе электронных чернил. В результате на двух весенних выставках этого года компания Philips продемонстрировала работающие прототипы устройств, использующих активно-матричные монохромные дисплеи на основе электронных чернил. Примечательно, что один из этих прототипов был оформлен в "книжном" формате — с внутренней стороны каждой из створок складного корпуса располагался отдельный экран.

Согласно заявлению руководителей Philips, полномасштабное коммерческое производство подобных монохромных дисплейных модулей с высоким разрешением экрана планируется начать в следующем году. Philips уже обнародовала предварительные технические характеристики изделий первого поколения: размер экрана — от 3 до 14 дюймов по диагонали; разрешающая способность — порядка 125...150 dpi, угол обзора — 180°, контрастность — 8:1. Для управления дисплейным модулем будет использоваться 8-разрядный параллельный интерфейс; кроме того, предусмотрена возможность оснащения дисплейного модуля сенсорной панелью.

Пока дисплеи на электронных чернилах обладают весьма ограниченной возможностью передачи полутонов. Так, у устройств первого поколения разрядность составляет 2 бита на один пиксел (то есть 4 градации серого), однако ожидается, что в следующем поколении таких дисплеев каждый пиксел экрана будет отображать уже 16 градаций серого, а в дальнейшем вполне вероятно появление и цветных моделей.

Говоря о цветных дисплеях на основе электронных чернил, автор отмечает еще одного технологического партнера E Ink — японскую компанию TOPPAN Printing, специализирующуюся на разработке и производстве светофильтров для самых различных применений. В прошлом году совместными усилиями разработчиков E Ink, TOPPAN и Philips был изготовлен прототип цветного активно-матричного дисплея на базе электронных чернил. Экран этого дисплея, позволяющего отображать 4096 цветов, имел размер 5 дюймов по диагонали и разрешение 320x234 пиксела. Примечательно, что формовка подложки, а также нанесение тонкопленочных компонентов, электронных схем и светофильтров этого дисплея было осуществлено с использованием стандартных производственных процессов, применяемых при изготовлении ЖК-панелей. Первая публичная демонстрация этого прототипа состоялась в мае прошлого года на выставке SID Expo 2002.

В заключение автор отмечает, что в данный момент электронные чернила компании E Ink являются поистине уникальной разработкой, имеющей огромный потенциал. Начало коммерческого производства отражающих дисплеев нового поколения, намеченное на будущий год, может оказать огромное влияние на дальнейшее развитие рынка портативных электронных устройств и мобильных ПК, не говоря уже о множестве других отраслей, где применяются дисплейные панели.

С одной стороны, возможность внедрения дисплеев на базе электронных чернил в мобильные телефоны и КПК представляется весьма спорной. В эпоху расцвета мультимедийных функций портативной электронике необходимы цветные экраны с малым временем отклика, и электронные чернила здесь вряд ли смогут на равных конкурировать с цветными ЖК- и OLED-дисплеями. С другой же стороны, внедрение новых отражающих дисплеев в устройства для чтения электронных книг действительно может серьезно изменить облик и возможности этих аппаратов, а также сделать их более доступными за счет существенного снижения цены. Разумеется, во многом это будет зависеть от надежности, долговечности и прочности новых дисплеев, ведь при запуске в серийное производство даже незначительная на первый взгляд недоработка может оказать крайне негативное влияние на отношение потенциальных покупателей к новой технологии как таковой.

Свершилось!!! **В продажу поступил первый персональный компьютер на 64-разрядном микропроцессоре.** Об этом сообщается в заметке "Пятое поколение "яблок" **(CHIP, №8/2003, С.14)**. Но вы не спешите бежать в магазин, дождитесь 128-разрядного или 256-разрядного ПК.

Это компьютер Power Mac G5 компании Apple. В нем использован RISC-процессор IBM PowerPC 970, тактовая частота которого может достигать 2 ГГц. Ядро процессора является 64-разрядным и позволяет работать с любыми приложениями, в том числе и с имеющимися 32-разрядными. Такое свойство позволяет использовать новые компьютеры со старым программным обеспечением, что на начальном этапе их использования будет немаловажным.

Частота системной шины — 800 МГц, объем кэш-памяти второго уровня — 512 Кб, а для улучшения работы с мультимедийными приложениями используется собственная технология, носящая название Altivec. Процессор изготавливается по 0,13-мкм техпроцессу с использованием медных соединений.

В Power Mac G5 используется оперативная память DDR SDRAM PC2700 или PC3200 (в зависимости от конкретной модели), жесткий диск Serial ATA емкостью 80 или 160 Гб и высокоскоростные видеокарты NVIDIA GeForce FX5200, ATI Radeon 9600 Pro, ATI Radeon 9800 Pro. Для подключения к сети, кроме обычного модема, предлагаются два сетевых контроллера — Gigabit Ethernet и AirPort. Кроме того, Power Mac G5 оборудован еще и контроллером FireWire (работающим со скоростью как 400, так и 800 Мбит/с) и портом USB 2.0.

Компьютер имеет широкие возможности расширения, поскольку здесь присутствуют шесть слотов PCI, три из которых являются 64-разрядными на 33 МГц, два — 64-разрядными на 100 МГц и один — PCI-X на133 МГц.

Младшая модель, основанная на процессоре с тактовой частотой 1600 МГц, стоит 2000 USD, а топовая модель с двумя процессорами 2 ГГц — 3000 USD.

А.КОРБИТ, А.ЩЕРБАКОВ,
г.Минск, БГУИР.

Программирование в среде Visual C++ 6.0

Работа с файлами

Цель этой статьи — изучить возможности классов библиотеки MFC для создания и работы с файловой системой.

Описание классов для работы с файлами

CFile (класс файлов) — это базовый класс библиотеки MFC для работы с файлами, обеспечивающий небуферизованный двоичный доступ к файлу. Этот класс используется совместно с классом CArchive для сохранения данных в приложениях, поддерживающих архитектуру "Документ/Вид". Класс CArchive (класс архивов) поддерживает сериализацию объектов MFC, т.е. позволяет сохранять на диске достаточно сложные объекты, а также загружать их в память по мере необходимости. Объекты хранятся в файлах с последовательным доступом.

Прежде чем использовать объект класса архива, следует создать соответствующий ему объект класса CFile. При этом необходимо удостовериться, что атрибуты доступа к данным в этих объектах соответствуют друг другу. Одному объекту класса CFile может соответствовать не более одного активного объекта класса CArchive.

Класс CFile имеет три формы конструкторов.

Конструктор по умолчанию — **CFile()** — не открывает файл, а только создает его. В этом случае файл открыть нужно явно, вызвав для этого метод **Open()**.

Вторая форма конструктора — **CFile(int hFile)** — создает объект для существующего файла с указанным дескриптором. После разрушения объекта файл необходимо закрыть самостоятельно.

Третий конструктор класса CFile имеет следующий вид:

```
CFile( LPCTSTR lpszFileName, UINT nOpenFlags );
```

Первый параметр— *lpszFileName* — строка, содержащая путь к открываемому файлу.

Второй параметр — *nOpenFlags* — флаг доступа к файлу.

В таблице приведены основные флаги, они могут комбинироваться оператором побитового ИЛИ (|).

Флаг	Действие
CFile::modeCreate	Создает новый файл
CFile::modeRead	Разрешается только чтение
CFile::modeReadWrite	Разрешается и чтение, и запись
CFile::modeWrite	Разрешается только запись
CFile::typeBinary	Устанавливает двоичный режим доступа
CFile::typeText	Устанавливает текстовый режим доступа

Имея объект класса **CFile**, можно создать и объект класса **CArchive.** Конструктор этого класса имеет вид:

```
CArchive( CFile* pFile, UINT nMode, int nBufSize = 4096, void* lpBuf = NULL );
```

Как можно увидеть, при создании объекта архива обязательно нужен объект **CFile**, указатель на который передается через первый параметр (* *pFile*). Второй параметр — *nMode* — это режим объекта. Наиболее часто используются два основных режима:

- CArchive::load — чтение данных;
- CArchive::store — запись данных.

После создания объекта класса **CArchive** запись в архив можно производить стандартными потоками ввода-вывода cin и cout, используя операторы << и >>.

В конструкторе класса **CFile** требуется передать имя файла. Чтобы избавить пользователя от необходимости вводить имя файла вручную, в MFC существует класс **CFileDialog**, который предоставляет стандартный диалог выбора файлов. Конструктор класса CfileDialog имеет вид:

```
CFileDialog( BOOL bOpenFileDialog, LPCTSTR lpszDefExt = NULL, LPCTSTR
lpszFileName = NULL, DWORD dwFlags = OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT,
LPCTSTR lpszFilter = NULL, CWnd* pParentWnd = NULL );
```

Объекты класса **CFileDialog** представляют диалоговые панели "Открыть" или "Сохранить как" в зависимости от параметра bOpenFileDialog. Если параметр bOpenFileDialog содержит значение TRUE, то создается объект, управляющий диалоговой панелью "Открыть", а если FALSE — диалоговой панелью "Сохранить как".

Параметр bOpenFileDialog является единственным обязательным параметром, который необходимо указать. Остальные параметры конструктора класса **CFileDialog** задают различные режимы работы панели и могут не указываться.

Во многих случаях имена файлов, которые нужно открыть или закрыть, имеют определенное расширение. Параметр lpszDefExt позволяет задать расширение файлов, используемое по умолчанию. Т.е. если пользователь при определении имени файла не укажет расширение, имени файла автоматически присваивается расширение, принятое по умолчанию. Если при определении свойств диалоговой панели программист присвоит параметру lpszDefExt значение NULL, то расширение файлов должно задаваться пользователем явно.

В некоторых случаях требуется, чтобы диалоговые панели отображались с уже выбранным именем файла. Чтобы указать имя файла, используемое по умолчанию, применяется параметр lpszFileName. Если параметр lpszFileName имеет значение NULL, данная возможность не реализуется.

Диалоговые панели выбора файлов обычно имеют список так называемых фильтров, включающих названия типов файлов и расширения имен файлов данного типа. Выбрав фильтр, пользователь указывает, что он желает работать только с файлами определенного типа, имеющими соответствующее расширение. Файлы с другими расширениями в диалоговых панелях не отображаются.

Список фильтров можно указать через параметр lpszFilter. Одновременно можно указать несколько фильтров. Каждый фильтр задается двумя строками — строкой, содержащей имя фильтра, и строкой, в которой перечислены соответствующие данному фильтру расширения имен файлов. Если одному типу соответствует несколько расширений, они разделяются символом ";". Строка, содержащая имя фильтра, отделяется от строки с расширениями файлов символом "|". Если используется несколько фильтров, то они также отделяются друг от друга символом "|". Например, в качестве строки, задающей фильтры, можно использовать строку вида:

```
char szFilters[] =" Data (*.dat) |*.dat| Packed Data
(*.pac;*.wav) | *.pac; *.wav| All Files (*.*)| *.* ||";
```

Пример написания программы

Необходимо написать программу, вводящую в файл или читающую из файла ведомость абитуриентов, сдавших вступительные экзамены. Каждая запись должна содержать фамилию, а также оценки по физике, математике и сочинению. Список абитуриентов требуется вывести отсортированным в порядке уменьшения их среднего балла.

Для решения задачи создадим проект на основе диалогового окна, отключив на втором шаге опцию AboutBox. Используя панель Controls, нанесите на заготовку формы 4 элемента типа Editbox, 4 — типа Statictext, 4 — типа Button, 1— типа

Рис.1

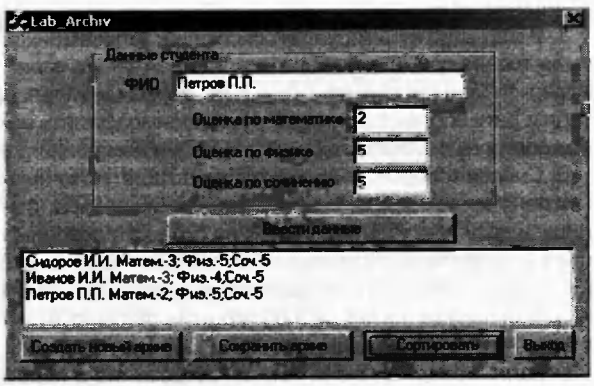


Рис.2

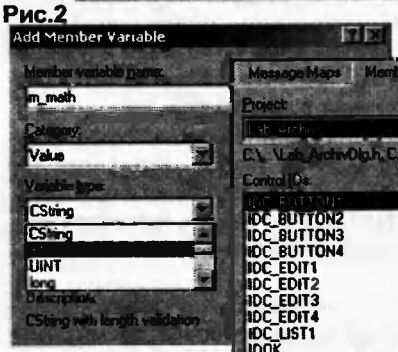
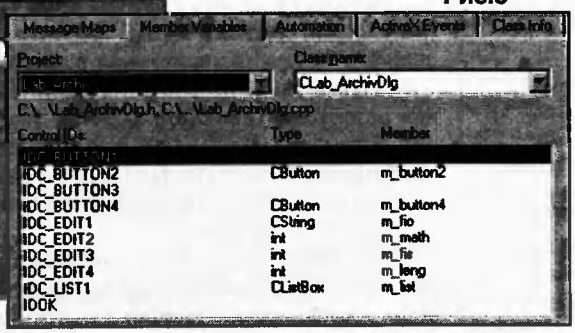


Рис.3



GroupBox и один элемент типа ListBox. Название кнопки "OK" необходимо переименовать в "Выход" (**рис.1**).

В свойствах элемента ListBox на закладке Styles отключите опцию Sort.

Затем с помощью ClassWizard выберите подходящий тип переменной (**рис.2**) и свяжите элементы управления с переменными, как показано на **рис.3**.

Текст программы:

```
#include "stdafx.h"
#include "Lab_Archiv.h"
#include "Lab_ArchivDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
struct CStudent     //данные о студенте.
{
        CString fio;
        int math,fis,leng;
};
CStudent Stud[100];
int n_zap;  //число записей.
/////////////////////////////////////////////////////////////////////////
// CLab_ArchivDlg dialog

CLab_ArchivDlg::CLab_ArchivDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CLab_ArchivDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CLab_ArchivDlg)
        m_fio = _T("Иванов И.И.");
        m_math = 3;
        m_fis = 4;
        m_leng = 5;
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require
        // a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CLab_ArchivDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CLab_ArchivDlg)
        DDX_Control(pDX, IDC_BUTTON4, m_button4);
        DDX_Control(pDX, IDC_BUTTON2, m_button2);
        DDX_Control(pDX, IDC_LIST1, m_list);
        DDX_Text(pDX, IDC_EDIT1, m_fio);
        DDX_Text(pDX, IDC_EDIT2, m_math);
        DDX_Text(pDX, IDC_EDIT3, m_fis);
        DDX_Text(pDX, IDC_EDIT4, m_leng);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CLab_ArchivDlg, CDialog)
        //{{AFX_MSG_MAP(CLab_ArchivDlg)
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
        ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
        ON_BN_CLICKED(IDC_BUTTON3, OnButton3)
        ON_BN_CLICKED(IDC_BUTTON4, OnButton4)
        //}}AFX_MSG_MAP
        END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////
// CLab_ArchivDlg message handlers

BOOL CLab_ArchivDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // Set the icon for this dialog.
        // The framework does this automatically
        // when the application's main window is not
        // a dialog
        SetIcon(m_hIcon, TRUE);     // Set big icon
        SetIcon(m_hIcon, FALSE);    // Set small icon
        m_button2.EnableWindow(FALSE);
    //неактивна кнопка "Создать новый архив"
        m_button4.EnableWindow(FALSE);
    //неактивна кнопка "Сортировать"
    // TODO: Add extra initialization here
    return TRUE; // return TRUE  unless you set the focus to a control
}
// If you add a minimize button to your dialog, you will need
// the code below  to draw the icon.  For MFC applications using
// the document/view model, this is automatically done for you
// by the framework.

void CLab_ArchivDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting

                SendMessage(WM_ICONERASEBKGND, (WPARAM)
                        dc.GetSafeHdc(), 0);

                // Center icon in client rectangle
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // Draw the icon
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}
// The system calls this to obtain the cursor to display while
// the user drags the minimized window.
HCURSOR CLab_ArchivDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

void PrintList(CListBox &list) //функция печати результата
{
        list.ResetContent();
        for(int i=0;i<n_zap;i++)
```

```
        {
        CString s;
        s.Format(" математ.-%d; физ.-%d;
        соч..%d",Stud[i].math,Stud[i].fis,Stud[i].leng);
        list.AddString(Stud[i].fio+s);
        }
}
void CLab_ArchivDlg::OnButton1()
{       // обработчик кнопки "Ввести данные"
        // TODO: Add your control notification handler code here
        UpdateData();
        Stud[n_zap].fio=m_fio;
        Stud[n_zap].math=m_math;
        Stud[n_zap].fis=m_fis;
        Stud[n_zap++].leng=m_leng;
        PrintList(m_list);
        m_button2.EnableWindow();
        //активизация кнопки "Создать новый архив"
        if (n_zap>1) m_button4.EnableWindow();
//активизация кнопка "Сортировать"
}

void CLab_ArchivDlg::OnButton2()
{       // обработчик кнопки "Создать новый архив"
        char szFilters[] ="Data Files (*.dat)|*.dat||";
        //фильтр для диалога выбора файлов
    CFileDialog mFileOpen(TRUE, "dat", "Archiv1.dat", NULL, szFilters);
        if(mFileOpen.DoModal()!=IDOK) return;
        CFile file(mFileOpen.GetFileName(), CFile::modeWrite |
        CFile::modeCreate);
         CArchive ar(&file, CArchive::store);

        /ar<<n_zap;           //запись в файл числа студентов
        for(int i=0; i<n_zap;i++)
        {
           ar << Stud[i].fio; //запись фамилии
           ar << Stud[i].math;
           ar << Stud[i].fis;
           ar << Stud[i].leng;
        }
        ar.Close();
        file.Close();
}
void CLab_ArchivDlg::OnButton3()
{       // обработчик кнопки "Сохранить архив"
        char szFilters[] ="Data Files (*.dat)|*.dat||";
  CFileDialog mFileOpen(FALSE, "dat", "archiv1.dat", NULL, szFilters);
        if(mFileOpen.DoModal()!=IDOK) return;
        CFile file(mFileOpen.GetFileName(), CFile::modeRead);
        CArchive ar(&file, CArchive::load);
        ar>>n_zap;
        for(int i=0; i<n_zap;i++)
        {
           ar >> Stud[i].fio;
           ar >> Stud[i].math;
           ar >> Stud[i].fis;
           ar >> Stud[i].leng;
        }
        ar.Close();
        file.Close();
        PrintList(m_list);
    m_button2.EnableWindow();
        if (n_zap>1) m_button4.EnableWindow();
}
void CLab_ArchivDlg::OnButton4()
{       // обработчик кнопки "Сортировать"
        for(int i=0;i<n_zap-1;i++)
        for(int j=i+1;j<n_zap;j++)

if(Stud[i].math+Stud[i].fis+Stud[i].leng<Stud[j].math+Stud[j].fis+Stud[j].leng)
                {
                        CStudent t;
                        t=Stud[i];
                        Stud[i]=Stud[j];
                        Stud[j]=t;
                }
        PrintList(m_list);
}
```

Индивидуальные задания

Программа должна создавать файл, читать данные из файла и обрабатывать их в соответствии с индивидуальным заданием, выводя результат в ListBox.

1. В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., домашний адрес покупателя и дату постановки на учет. Необходимо удалить из списка все повторные записи, проверяя Ф.И.О. и домашний адрес.

2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Необходимо вывести в алфавитном порядке список товаров, хранящихся больше месяца, стоимость которых превышает 1000000 руб.

3. Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Требуется вывести список очередности предоставления мест в общежитии.

4. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия. Необходимо вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

5. На междугородной АТС информация о разговорах содержит: дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Вывести по каждому городу общее время разговоров с ним и сумму.

6. Информация о сотрудниках фирмы включает: Ф.И.О., табельный номер, количество проработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12% от суммы заработка.

7. Информация об участниках спортивных соревнований содержит: наименование страны, название команды, Ф.И.О. игрока, игровой номер, возраст, рост, вес. Вывести информацию о самой молодой, рослой и легкой команде.

8. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

9. Различные цехи завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха. Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию в порядке убывания количества.

10. Информация о сотрудниках предприятия содержит Ф.И.О., номер отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

11. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О., адрес, оценки. Определить количество абитуриентов, проживающих в г.Минске и сдавших экзамены со средним баллом не ниже 4,5. Вывести их фамилии в алфавитном порядке.

12. В справочной аэропорта хранится расписание вылетов самолетов на следующие сутки. Для каждого рейса указаны: номер рейса, тип самолета, пункт назначения, время вылета. Вывести все номера рейсов, типы самолетов и времена вылета для заданного пункта назначения в порядке возрастания времени вылета.

Литература

1. С.Холзнер. Visual C++ 6: учебный курс. — СПб.: Питер, 1999. — 576 с.

Решение задач на стратегические игры

М.ДОЛИНСКИЙ,
г.Гомель.

Общий принцип решения таких задач заключается в следующем. Вводится множество состояний игры. Естественно, помечаются одно или несколько начальных выигрышных состояний. Далее работает цикл типа "Пока добавилось хоть одно выигрышное состояние". Просматриваются неопределенные еще состояния и предпринимаются попытки превратить их в выигрышные — разумеется, на основании правил игры.

1. Задача "Алиса и Боб"

(Четвертьфинал NEERC'98, Западный регион, Минск, октябрь 1998 г.)

Дан ориентированный граф G=(V,E), вершины которого являются позициями в игре. В игре участвуют два игрока — Алиса и Боб. Они двигают фишку из позиции в позицию по дугам графа. Множество позиций V разделено на два подмножества A и B (A*B=0, A+B=V). В позициях A ход делает Алиса, а в позициях B — Боб. Если игра находится в позиции v из множества V, владелец этой позиции выбирает выходящую из нее дугу (v,w) из множества E и двигает фишку из v в w. Игра начинается в некоторой позиции. Алиса выигрывает, если фишка оказалась в позиции t из множества V. Если за любое количество ходов фишка не попадает в эту позицию — выигрывает Боб.

Требуется определить, какой игрок выигрывает, в зависимости от стартовой позиции при оптимальной стратегии обеих сторон.

Считаем, что V = {1,K,n} (1 ≤ n ≤ 1000). Исходные данные представлены в текстовом файле D.IN, имеющем следующую структуру.

Первая строка:

n m t

Здесь: n — число вершин; m — число дуг; t — заключительная позиция.

Со второй строки записаны строки:

b1 b2 K bn v1 w1 v2 w2 K vm wm

Здесь: bi=1, если i принадлежит A; bi=0, если i не принадлежит A; vi wi — список дуг графа; vi — начальная вершина, wi — конечная вершина соответствующей дуги (1 ≤ m ≤10000). Все параметры — целые числа, разделенные пробелами.

Пример входного файла

```
7 12 7
0 1 0 0 0 1 0
1 2 1 4 1 3
2 4 2 5
3 1 3 6 3 7
4 3 4 6
5 6
6 7
```

Результат требуется представить в текстовом файле D.OUT, в котором следует записать n целых чисел — x1 x2 ... xn, где xi=1, если позиция i выигрышна для Алисы; иначе xi=0.

Для приведенного примера результат представляется строкой

```
0 1 0 0 1 1 1
```

Идея решения подобной задачи коротко может быть сформулирована следующим образом. Все позиции, в принципе, разбиваются на два подмножества — победные для Алисы (назовем их выигрышными) и те, в которых выигрывает не Алиса, а Боб (назовем их проигрышными).

Из всех выигрышных позиций ход игры, при любых ходах Боба и правильных ходах Алисы, приводит к позиции t.

Из всех проигрышных позиций, при любых ходах Алисы и правильных ходах Боба, ход игры НЕ ПРИВОДИТ в позицию t.

Очевидно, что заведомо выигрышными являются те позиции, в которых — ход Алисы, и из которых есть НЕПОСРЕДСТВЕННАЯ дуга в состояние t.

Отталкиваясь от тех позиций, которые являются заведомо выигрышными, итеративно строим ВСЕ остальные выигрышные позиции следующим образом. Если в текущей позиции — ход Алисы, и есть хоть одна дуга в любую из позиций, уже объявленных выигрышными, то и текущая позиция объявляется выигрышной. Аналогично, если в текущей позиции ход Боба, и ВСЕ дуги ведут в позиции, возможно различные, но уже ВСЕ объявленные выигрышными, то и текущая позиция объявляется выигрышной. Процесс пересмотра "невыигрышных" позиций продолжается до тех пор, пока добавляется хоть одна выигрышная позиция.

Более формально изложенный выше алгоритм может быть записан следующим образом:

```
Помечаем все состояния как проигрышные.
    Помечаем начальные выигрышные позиции
      (ход А и есть дуга в конечную позицию t).
    ПОКА изменилось на выигрышное состояние хоть одной позиции,
      ЦИКЛ по всем невыигрышным позициям.
        ЕСЛИ ход А, и есть хоть одна дуга в выигрышную позицию
             ИЛИ
             ход Б, и ВСЕ дуги ведут в выигрышную позицию
          ТО меняем состояние этой позиции на "выигрышная".
```

Рассмотрим одну из возможных реализаций решения данной задачи. По условиям задачи нам дан ориентированный граф из N вершин, и для каждой из этих вершин определяется, кто должен ходить из этой вершины — Алиса или Боб (если B[i]=1, то Алиса, если B[i]=0, то Боб).

Вводим также массив S, который будет обозначать состояние позиции (если S[i]=1, то выигрывает Алиса, иначе — Боб).

Главная программа решения данной задачи может выглядеть следующим образом:

```
begin
  InputData;
  Changed:=true;
  while Changed do
    begin
      Changed:=false;
      for i:=1 to N do
        if (S[i]=0) and
           (((B[i]=1) and ExistArc(i)) or
            ((B[i]=0) and  AllArcs(i)))
           then begin
                  S[i]:=1; Changed:=true;
                end;
```

2. Н.Ю.Секунов. Самоучитель Visual C++ 6 (+ дискета). — СПб.: BHV-Санкт-Петербург, 1999. — 960 с.

3. К.Грегори. Использование Visual C++ 6. Спец.издание. — СПб.: Вильямс, 1999. — 864 с.

4. Ю.В.Тихомиров. Visual C++ 6 (+ дискета). — СПб.: BHV-Санкт-Петербург, 1999. — 496 с.

5. Майкл Дж. Янг. Visual C++ 6. Полное руководство (+ CD-ROM). — BHV-Киев, 2000. — 1056 с.

6. С.Гилберт, Б.Маккарти. Самоучитель Visual C++ 6 (+ CD-ROM). — М.: Диа-Софт, 2000. — 496 с.

7. К.Паппас, У.Мюррей. Visual C++ 6 для пользователя. — BHV-Киев, 2000. — 336 с.

8. К.Паппас, У.Мюррей Visual C++ 6: руководство разработчика. — BHV-Киев, 2000. — 624 с.

9. И.Баженова. Visual C++6. 0. VISUAL STUDIO 98: Уроки программирования. — М.: Диалог-МИФИ, 2001. — 416 с.

10. Лэйси Дж.Visual C++ 6 Distributed. Сертификационный экзамен — экстерном. — СПб.: Питер, 2001. — 624 с.

11. А.Черносвитов Visual C++ 7: учебный курс (с дискетой). — СПб.: Питер, 2001. — 528 с.


```
  end;
 OutputData;
end.
```

В главной программе вызываются следующие процедуры и функции:

- InputData — для ввода и инициализации исходных данных;
- OutputData — для вывода ответа;
- ExistArc(i) — для выяснения, существует ли дуга из текущей вершины i в выигрышную позицию;
- AllArcs(i) — для выяснения, ВСЕ ли дуги из текущей вершины i ведут в выигрышные позиции.

Рассмотрим их более подробно.

```
procedure InputData;
  begin
    assign (input,'d.in'); reset(input);
    read (N,M,T);
    for i:=1 to N do read (B[i]);
    for i:=1 to M do
      begin
        read (from,tov); inc(ka[from]);
        a[from,ka[from]]:=tov;
      end;
    close(input);
    for i:=1 to N do S[i]:=0;  S[T]:=1;
  end;
```

Как видно из текста, исходный ориентированный граф представляется набором дуг из вершины. ka[i] содержит информацию, СКОЛЬКО дуг выходит из вершины i. a[i,j] (для j от 1 до ka[i]) содержит номер вершины, в которую ведет из вершины i дуга с номером j. Вершина T помечается как выигрышная.

```
procedure OutputData;
  begin
    assign (output,'d.out'); rewrite(output);
    write (S[1]);
    for i:=2 to N do write(' ',S[i]);
    close(output);
  end;
```

Состояния выводятся в соответствии с форматом вывода в одну строку (без пробела за последним состоянием).

```
function ExistArc(i:integer):boolean;
  var j : integer;
  begin
    ExistArc:=false;
    for j:=1 to ka[i] do
      if S[a[i,j]]=1
        then begin
               ExistArc:=true;
               exit;
             end;
  end;
```

Вначале переменной ExistArc присваивается значение false. Затем просматриваются все дуги из вершины i. Если хоть одна из них ведет в вершину, которая уже объявлена выигрышной, то переменная ExistArc получает значение true, и работа функции ExistArc завершается с возвращением значения true в качестве результата. Если же ни одна из дуг не ведет в выигрышную вершину, то значение переменной ExistArc так и останется false, а функция ExistArc завершит свою работу с возвращением значения false в качестве результата.

```
function AllArcs(i:integer):boolean;
  var j : integer;
  begin
    AllArcs:=true;
    for j:=1 to ka[i] do
      if S[a[i,j]]=0
        then begin
               AllArcs:=false;
               exit;
             end;
  end;
```

Вначале переменной AllArcs присваивается значение true. Затем просматриваются все дуги из вершины i. Если хоть одна из них ведет в проигрышную вершину, то переменная AllArcs получает значение false, а функция AllArcs завершает свою работу с возвращением false в качестве результата. Если же ни одна из дуг из вершины i не ведет в проигрышную вершину, то переменная AllArcs сохраняет свое первоначальное значение true, и фунция AllArcs завершает свою работу, возвращая значение true в качестве результата.

Ниже приводится полный текст решения задачи "Алиса и Боб":

```
program nee4w98d;
const
  MaxN = 1000;
  MaxM = 10000;
  MaxK = 20;
var
  i,N,M,T, from, tov  :  integer;
  Changed             :  boolean;
  S,Ka      :  array [1..MaxN] of integer;
  B         :  array [1..MaxN] of byte;
  a         :  array [1..MaxN,1..MaxK] of integer;

  function ExistArc(i:integer):boolean;
    var j : integer;
    begin
      ExistArc:=false;
      for j:=1 to ka[i] do
        if S[a[i,j]]=1
          then begin
                 ExistArc:=true;
                 exit;
               end;
    end;

  function AllArcs(i:integer):boolean;
    var j : integer;
    begin
      AllArcs:=true;
      for j:=1 to ka[i] do
        if S[a[i,j]]=0
          then begin
                 AllArcs:=false;
                 exit;
               end;
    end;

  procedure InputData;
    begin
      assign (input,'d.in'); reset(input);
      read (N,M,T);
      for i:=1 to N do read (B[i]);
      for i:=1 to M do
        begin
          read (from,tov); inc(ka[from]);
          a[from,ka[from]]:=tov;
        end;
      close(input);
      for i:=1 to N do S[i]:=0;  S[T]:=1;
    end;

  procedure OutputData;
    begin
      assign (output,'d.out'); rewrite(output);
      write (S[1]);
      for i:=2 to N do write(' ',S[i]);
      close(output);
    end;

begin
  InputData;
  Changed:=true;
  while Changed do
    begin
      Changed:=false;
      for i:=1 to N do
        if (S[i]=0) and
          (((B[i]=1) and ExistArc(i)) or
           ((B[i]=0) and  AllArcs(i)))
          then begin
                 S[i]:=1; Changed:=true;
               end;
    end;
  OutputData;
end.
```

2. Задача "Ладья и конь"

(Гомельская областная олимпиада по информатике, январь 2000 г.)

Игра ведется на шахматной доске размером N*M (N — количество горизонталей, M — количество вертикалей). В игре участвуют ладья и конь. Первой ходит ладья. Ладья выигрывает, если ей удается сбить коня.

Ограничения

2≤N≤8

2≤M≤8

Конь ставится в такую начальную позицию, из которой он может сделать по крайней мере один ход (например, на доске 3*3 конь не может оказаться на поле b2, т.к. любой ход из этой позиции выводит за пределы доски).

Формат ввода

Первая строка — N и M.

Вторая строка — позиции ладьи и коня.

Например, b1 c2 означает, что ладья стоит на второй вертикали и первой горизонтали, конь стоит на третьей вертикали и второй горизонтали. Вертикали обозначаются латинскими буквами от a до h, горизонтали обозначаются цифрами от 1 до 8.

Формат вывода

Минимальное число ходов, которые должна сделать ладья, чтобы сбить коня, если это возможно; иначе — должен быть напечатан 0.

Пример ввода

```
3 3
b1 c2
```

Пример вывода

```
3
```

Пояснения к примеру

Конь ловится за три хода:

1. Лb1-b3 Кc2-a1 (1. ... Кc2-a3 2. Лb3:a3X)
2. Лb3-b2 Кa1-b3 (2. ... Кa1-c2 3. Лb2:c2X)
3. Лb2:b3X

Общая идея решения задачи такова. Строим все состояния, в которых ладья выигрывает за один ход. Это состояния, в которых ладья и конь находятся на одной горизонтали или на одной вертикали.

Затем по ним строим все состояния, в которых ладья выигрывает в два хода. И так далее. По состояниям, в которых ладья выигрывает за Number ходов, строим состояния, в которых ладья выигрывает за Number+1 ход. Если новые победные состояния построить не удается, процесс завершается.

Правило построения новых победных состояний по старым таково. Пусть ладья стоит в позиции (il,jl), а конь стоит в позиции (ik,jk), и в такой позиция ладья сбивает коня не более чем за Number ходов при наилучшей игре коня.

Для построения ВСЕХ позиций, порождаемых данной, и в которых ладья сбивает коня не более чем за Number+1 ход, строим все возможные поля, откуда сюда мог попасть конь за один ход. Для каждой такой позиции коня проверяем, что ВСЕ ходы коня из этой позиции ведут в позицию, в которой победа достигается не более чем за Number ходов.

Если такая позиция коня (x,y) найдена, то искомыми позициями (победными за не более чем Number+1 ход) будут ВСЕ позиции вида

```
(MovX,MovY,x,y),
```

где MovX и MovY такие, что ладья переходит из них в исходную позицию (il,jl) ровно за 1 ход — т.е. они находятся с позицией (il,jl) на одной горизонтали или на одной вертикали.

Ниже описывается одна из возможных реализаций предложенного алгоритма. Тело главной программы, решающей задачу, выглядит следующим образом:

```
begin
  InputData;
  Changed:=true; Number:=0;
  while Changed do
    begin
      Changed:=false;
      Inc(Number);
      for il:=1 to M do                    {По всем состояниям}
        for jl:=1 to N do
          for ik:=1 to M do
            for jk:=1 to N do
              if S[il,jl,ik,jk]=Number
                then PutAllNewWinStates(il,jl,ik,jk);
    end;
  OutputData;
end.
```

Рассмотрим операторы более подробно.

```
procedure InputData;
  var c : char;
  begin
    assign (input,'input.txt'); reset(input);
    readln (N,M);
    read(c); li:=Ord(c)-Ord('a')+1; read(lj); read(c);
    read(c); ki:=Ord(c)-Ord('a')+1; read(kj);
    close(input);

    for il:=1 to M do                  {Все состояния проигрышны}
      for jl:=1 to N do
        for ik:=1 to M do
          for jk:=1 to N do
            S[il,jl,ik,jk] :=0;

for il:=1 to M do                  {Выигрышные состояния за 1 ход}
      for jl:=1 to N do                          {если ход ладьи}
        for ik:=1 to M do                {и конь на одной вертикали}
          for jk:=1 to N do               {или на одной горизонтали}
            if ((il=ik) or (jl=jk)) and                  {с ладьей}
               not ((il=ik) and (jl=jk))
                                              {но не в одной позиции}
              then S[il,jl,ik,jk] :=1;

  end;
```

В этой процедуре выполняется следующая работа.

Вводим исходные данные, преобразовываем их к числовому виду:

(li,lj,ki,kj).

Строим 4-мерный массив состояний:

S[1..MaxM,1..MaxN,1..MaxM,1..MaxN]

Здесь: первые две координаты — позиция ладьи; вторые две координаты — позиция коня; S[il,jl,ik,jk] — минимальное количество ходов для победы ладьи над конем из позиции ладья(il,jl), конь(ik,jk).

Все состояния обнуляем (0 — победа ладьи недостижима).

Заносим 1 в состояния, в которых ладья непосредственно может сбить коня — т.е. ладья и конь стоят на одной горизонтали или на одной вертикали.

```
  ...
  Changed:=true; Number:=0;
  while Changed do
    begin
      Changed:=false;
      Inc(Number);
  ...
```

Устанавливаем признак "состояния изменялись". Количество ходов до победы ладьи — Number = 0. Пока состояния изменяются, сбрасываем признак "состояния изменялись".

Инкрементируем Number — количество ходов до победы ладьи.

```
    ...
      for il:=1 to M do                    {По всем состояниям}
        for jl:=1 to N do
          for ik:=1 to M do
            for jk:=1 to N do
              if S[il,jl,ik,jk]=Number
                then PutAllNewWinStates(il,jl,ik,jk);
    ...
```

По всем состояниям, если текущее состояние — победное за Number ходов, с помощью процедуры

```
PutAllNewWinStates(il,jl,ik,jk)
```

строим все состояния, победные за Number+1 ход. Если нашли такое состояние, то устанавливаем признак Changed.

```
...
   end;
OutputData;
...

procedure OutputData;
  begin
    assign (output,'output.txt'); rewrite(output);
    writeln(S[li,lj,ki,kj]);
    close(output);
  end;
...
```

После завершения цикла While выводим результат S[li,lj,ki,kj]. Т.е. по построению, это минимальное количество ходов, требуемое ладье, находящейся в позиции (li,lj), чтобы поймать коня, начинающего в позиции (ki,kj). Если ладья не может поймать коня, то ответ — 0.

Рассмотрим более подробно процедуру PutAllNewWinStates(il,jl,ik,jk):

```
procedure PutAllNewWinStates(il,jl,ik,jk:integer);
  begin
     for i:=1 to 8 do
       begin
         CurrentX := ik+steps[i,1] ;
         CurrentY := jk+steps[i,2] ;
         if (CurrentX>0) and (CurrentX<=M) and
            (CurrentY>0) and (CurrentY<=N)
          then PutIfAll(CurrentX,CurrentY);
       end;
  end;
```

Для всех возможных позиций (CurrentX,CurrentY), из которых конь мог попасть в позицию (ik,jk), вызываем процедуру PutIfAll(CurrentX,CurrentY) для нахождения состояний, которые победны за Number+1 ход.

Рассмотрим более подробно процедуру PutIfAll(CurrentX, CurrentY):

```
procedure PutIfAll(x,y:integer);
  var All, Was : boolean;
  begin
    j:=1; Was:=false; All:=true;
    while All and (j<=8) do
      begin
        CurrentX := x+steps[j,1] ;
        CurrentY := y+steps[j,2] ;
        if (CurrentX>0) and (CurrentX<=M) and
           (CurrentY>0) and (CurrentY<=N)
          then begin
                 if (S[il,jl,CurrentX,CurrentY]<=Number)
                    and
                    (S[il,jl,CurrentX,CurrentY]<>0) and
                    not ((il=CurrentX)and(jl=CurrentY))
                    then Was := true
                    else All := false;
               end;
        inc(j)
      end;
    if Was and All
      then
        begin
          for ilt:=1 to M do
            if (S[ilt,jl,x,y]=0) and (ilt<>il)
              then begin
                     S[ilt,jl,x,y]:=Number+1;
                     Changed:=true;
                   end;
          for jlt:=1 to N do
            if (S[il,jlt,x,y]=0) and (jlt<>jl)
              then  begin
                      S[il,jlt,x,y]:=Number+1;
                      Changed:=true;
                    end;
        end;
  end;
```

Для всех позиций коня, из которых можно было попасть в позицию (x,y), проверяем, что был хоть один реальный ход, и ВСЕ ходы были в состояния, в которых победа одерживалась не более чем за N ходов.

Если это так, то все состояния, которые возникают из текущего одним ходом коня в (x,y) и одним ходом ладьи (по горизонтали или вертикали), объявляются победными за Number+1 ход.

Полный текст решения задачи:

```
program go00d1t2;
const
  MaxN = 8;
  MaxM = 8;
var
  Changed             :  boolean;
  i,j,N,M, CurrentX, CurrentY,ilt,jlt,
  li,lj,ki,kj, il,jl,ik,jk, Number     : integer;

  S    : array [1..MaxM,1..MaxN,1..MaxM,1..MaxN] of
integer;

  P    : array [0..9] of longint;
  AllP : longint;

  procedure InputData;
    var c : char;
    begin
      assign (input,'input.txt'); reset(input);
      readln (N,M);
      read(c); li:=Ord(c)-Ord('a')+1; read(lj); read(c);
      read(c); ki:=Ord(c)-Ord('a')+1; read(kj);
      close(input);

      for il:=1 to M do          (Все состояния проигрышны)
        for jl:=1 to N do
          for ik:=1 to M do
            for jk:=1 to N do
              S[il,jl,ik,jk] :=0;

      for il:=1 to M do    {Выигрышные состояния за 1 ход}
        for jl:=1 to N do                  {если ход ладьи}
          for ik:=1 to M do    {и конь на одной вертикали}
            for jk:=1 to N do   {или на одной горизонтали}
              if ((il=ik) or (jl=jk)) and       {с ладьей}
                 not ((il=ik) and (jl=jk))
                then S[il,jl,ik,jk] :=1;

    end;

  procedure OutputData;
    begin
      assign (output,'output.txt'); rewrite(output);
      writeln(S[li,lj,ki,kj]);
      close(output);
    end;

    type                        {Все текущие возможные ходы}
      Knight = array [1..8,1..2] of integer;
                                        {Возможные ходы коня}
    const
      Steps : Knight = (( 1,-2),( 1, 2), {Массив констант}
                        (-1,-2),(-1, 2),
                        ( 2,-1),( 2, 1),

  procedure PutIfAll(x,y:integer);
    var All, Was : boolean;
    begin
      j:=1; Was:=false; All:=true;
      while All and (j<=8) do
        begin
          CurrentX := x+steps[j,1] ;
          CurrentY := y+steps[j,2] ;
          if (CurrentX>0) and (CurrentX<=M) and
             (CurrentY>0) and (CurrentY<=N)
            then begin
                   if (S[il,jl,CurrentX,CurrentY]<=Number)
                      and
                      (S[il,jl,CurrentX,CurrentY]<>0) and
                      not ((il=CurrentX)and(jl=CurrentY))
                      then Was := true
                      else All := false;
                 end;
          inc(j)
        end;
```

ИССЛЕДОВАНИЕ ПРОГРАММ

CyberManiac /HI-TECH,
www: http://wasm.ru/

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КРЭКИНГА

(Продолжение. Начало в №№8-11/2003)

Поиск констант с плавающей точкой — занятие с одной стороны более сложное, чем поиск целочисленной константы, но с другой — куда более простое. В чем сложность и в чем простота этого занятия?

По традиции начнем с плохого. Во-первых, формат представления чисел с плавающей точкой весьма нетривиален, и вы вряд ли сможете в уме привести шестнадцатеричный дамп такого числа в "человеческий" вид (возьмите документацию по процессорам Intel и попробуйте перевести число 1.23 в машинное представление, а затем проделать обратную операцию — вы сами убедитесь, насколько сложна эта задача). Более того, даже целые числа в представлении с плавающей точкой выглядят весьма неординарно — к примеру, дамп самого что ни на есть обычного числа 123, приведенного к типу Double, выглядит как 00 00 00 00 00 C0 5E 40. Если вы способны с первого взгляда отличить число с плавающей точкой от кода программы или каких-либо иных данных и оценить величину этого числа — я рад за вас, но большинство людей, к сожалению, такими способностями не обладают.

Во-вторых, при работе с дробными числами нередко возникают проблемы, связанные с машинным округлением и потерей точности. Самым ярким примером, наверное, может служить особенность математических программ ПЗУ некоторых моделей Spectrum — с точки зрения такого Спектрума выражение 1/2=0.5 было ложным. Это, конечно, было давно, но не следует считать, что современные компьютеры полностью свободны от этой проблемы. И вот практическое тому подтверждение.

Откомпилируйте под Delphi следующий код — i:=sin(1); i:=arcsin(i) и посмотрите, как будет меняться результат при изменении типа переменной I от Single до Extended. Например, если I имеет тип single, в результате вычислений получим, что arcsin(sin(1))= 0.999999940395355. Такие "спецэффекты" — следствие все той же потери точности в процессе вычислений.

В-третьих, округлением чисел процессор может заниматься не только по собственному желанию, но и по велению программы. К примеру, в большинстве бухгалтерских программ всевозможные ставки налогов выводятся с точностью до копеек. Однако из того, что вы видите на экране число 10.26, совершенно не следует, что результат расчетов представлен в памяти ЭВМ именно как 10.26. Реальное значение соответствующей переменной может быть равно 10.258 или 10.26167, которое и участвует в реальных расчетах, и лишь при выводе на экран для удобства пользователя было произведено округление до двух знаков после запятой.

Я не случайно столько места уделил округлению и точности представления чисел — именно эти особенности чисел с плавающей точкой в наибольшей мере затрудняют поиск нужных значений в памяти программы. Программисты знают, что при работе с действительными числами для проверки условия равенства некоторой вычисляемой величины другой величине не рекомендуется использовать сравнения вида f(a)=b. Причина этого лежит все в той же проблеме округления и потери точности при расчетах — вспомните вышеприведенные примеры со Спектрумом или арксинусом и синусом единицы. Вместо простой проверки равенства обычно используется условие "значения считаются равными, если абсолютная величина разности между ними не превышает некоторой величины": abs(f(a)-b)≤delta, где delta — максимально допустимая величина разности, после которой числа не считаются равными. Поэтому если вы хотите найти в памяти некоторое число F с плавающей точкой, вы в действительности должны искать все числа из промежутка [F-delta; F+delta], причем определить значение delta чаще всего можно лишь опытным путем. Это утверждение распространяется и на тот случай, когда вы знаете округленное значение переменной, но в этом случае величина delta будет зависеть от того, до скольки знаков округлено значение переменной. Так, если число округлено до сотых, нетрудно догадаться, что delta=0.005.

Вот тут-то вы и столкнетесь с чисто практической проблемой неприспособленности существующих отладчиков к поиску чисел с плавающей точкой. Сама по себе функция поиска действительных чисел в адресном пространстве программы в отладчиках встречается редко, а уж отладчиков, поддерживающих поиск чисел из заданного промежутка, я вообще не встречал. Поэтому вам, вероятно, придется для этой цели написать собственный инструмент либо искать способ получить нужную информацию каким-то другим путем.

```
    if Was and All
      then
        begin
          for ilt:=1 to M do
            if (S[ilt,jl,x,y]=0) and (ilt<>il)
              then begin
                     S[ilt,jl,x,y]:=Number+1;
                     Changed:=true;
                   end;
          for jlt:=1 to N do
            if (S[il,jlt,x,y]=0) and (jlt<>jl)
              then  begin
                      S[il,jlt,x,y]:=Number+1;
                      Changed:=true;
                    end;
        end;
  end;

procedure PutAllNewWinStates(il,jl,ik,jk:integer);
  begin
    for i:=1 to 8 do
      begin
        CurrentX := ik+steps[i,1] ;
        CurrentY := jk+steps[i,2] ;
        if (CurrentX>0) and (CurrentX<=M) and
           (CurrentY>0) and (CurrentY<=N)
         then PutIfAll(CurrentX,CurrentY);
      end;
  end;

begin
  InputData;
  Changed:=true; Number:=0;
  while Changed do
    begin
      Changed:=false;
      Inc(Number);
      for il:=1 to M do                {По всем состояниям}
        for jl:=1 to N do
          for ik:=1 to M do
            for jk:=1 to N do
              if S[il,jl,ik,jk]=Number
                then PutAllNewWinStates(il,jl,ik,jk);
  end;
  OutputData;
end.
```

(Продолжение следует)

И, наконец, нельзя забывать, что кроме стандартных для платформы x86 типов Single, Double и Extended (32-, 64- и 80-битных соответственно) существует еще несколько довольно экзотических, но все еще используемых форматов. Это, к примеру, Currency (64-битные, с фиксированным положением десятичной точки) или 48-битные паскалевские Real. Возможно также использование "самодельных" форматов. Особенно часто встречаются числа с фиксированным положением десятичной точки (обычно такое делается для повышения скорости работы программы и применяется, в основном, в процедурах кодирования/декодирования аудио- и видеоинформации). Знать о таких вещах совсем не лишне, хотя, конечно, вероятность столкнуться с такими числами в современных программах довольно низка.

Теперь немного поговорим о том хорошем, что есть в числах с плавающей точкой. Как известно, изначально в процессорах x86, встроенных аппаратных и программных средств для обработки чисел с плавающей точкой не предусматривалось. Низкая скорость расчетов, в которых использовались действительные числа, вызвала к жизни математические сопроцессоры, как традиционные x87, так и весьма экзотические девайсы Weitek. Победившая линейка сопроцессоров x87 (они с некоторых пор стали интегрироваться в ядро процессора и потому перестали существовать как отдельные устройства) имела следующую особенность — новые "математические" команды активно использовали для обмена информацией оперативную память. Посмотрите, к примеру, на важнейшие команды сопроцессора fst и fld — в качестве параметра этих команд могут выступать указатели на области памяти, которые предполагается использовать для чтения/записи данных. Более того, использование указателей в качестве одного из параметров характерно и для многих других команд сопроцессора. Поэтому ищите ссылки, используемые командами сопроцессора в качестве параметров — и вы легко доберетесь до данных, на которые эти ссылки указывают.

Из этого следует вывод — хороший дизассемблер или отладчик способен "догадаться", что по адресу, указанному в аргументах этих команд, находится число с плавающей точкой, и отобразить это число. Если же ваш дизассемблер/отладчик об этом не догадывается — вам придется вручную (точнее говоря, при помощи соответствующих программ) вычислить значение, которое находится по этому адресу. И пока вы будете копировать байтики из одной программы в другую, у вас будет достаточно времени подумать об обновлении инструментария.

Но и здесь не обошлось без ложки дегтя — компиляторы фирмы Borland, видимо, ради особой оригинальности, для загрузки констант в стек сопроцессора могут воспользоваться комбинациями вроде:

```
mov [i],$9999999a
mov [i+$4],$c1999999
mov mov [i+$8],$4002
fld tbyte ptr [i]
```

Хотя, казалось бы, ничто не мешало положить несчастное число в секцию инициализированных данных... Тут уж не до "умного" поиска — разобраться бы, что и куда вообще загружается. Хотя, при желании и умении обращаться с регулярными выражениями (или умении программировать) можно искать даже в таком коде.

Другим свойством действительных чисел, облегчающим автоматический поиск известной величины, является само их внутреннее устройство. Достаточно большая длина этих чисел (32 бита, а чаще всего — 64 или 80) и сложный формат хранения позволяют искать числа с плавающей точкой в любых файлах, в том числе и в исполняемых файлах программ, непосредственно в двоичном виде, причем вероятность ложного срабатывания будет незначительной. Даже существование нескольких различных форматов представления действительных чисел не представляет серьезного препятствия — соответствующая программа очень проста и пишется за считанные минуты. Народная мудрость гласит: "лучше один раз увидеть, чем сто раз услышать", поэтому в качестве практики я рекомендую вам самим написать и отладить такую программу — это не только усовершенствует ваши навыки в программировании, но и позволит поближе познакомиться с миром действительных чисел. Затем, если захотите, вы сможете доработать эту программу таким образом, чтобы она могла осуществлять нечеткий поиск в файле, о котором я говорил выше, т.е. поиск всех значений, подходящих под заданный пользователем интервал.

И, наконец, рассмотрим третий из наиболее распространенных в программах простых типов данных — текстовые данные. Вообще, методы представления текстовых строк и массивов в коде программ имеют давние и богатые традиции. Наиболее старым способом является выделение под строку участка фиксированного размера, причем неиспользуемая часть блока заполняется "нулевыми" символами. В чистом виде этот прием уже давно не встречается (из примеров вспоминаются разве что старые реализации классического Паскаля), но нечто подобное иногда используется в программах на Си для хранения массивов строк — под хранение каждого из элементов массива отводится блок фиксированного размера, хотя сами элементы, по сути, являются ASCIIZ-строками.

Хранение строк в блоках фиксированного размера имело два принципиальных недостатка: неэффективное расходование памяти при хранении большого числа строк различной длины и жесткое ограничение на максимальную длину строки. Всех этих недостатков были лишены строки с завершающим символом. Идея была проста — выбирается какой-либо малоиспользуемый символ, который интерпретируется программой как признак конца строки. В языке Си таким символом стал символ с кодом 0 (а строки, оканчивающиеся нулем, окрестили ASCIIZ-строками); некоторые системные функции MS-DOS в качестве завершающего символа использовали символ "$". Несмотря на ряд недостатков, строки с завершающим символом претерпели ряд усовершенствований и используются до сих пор. С началом активного использования UNICODE появилась модификация строк с завершающим символом и для этой кодировки. Зная образ мышления программистов на Си, нетрудно догадаться, что в качестве завершающего символа была использована пара нулевых байтов — (0,0). Нужно отметить, что если возникает необходимость укоротить такую строку в тексте программы на несколько символов, то обычно для этого достаточно всего лишь вписать в нужную позицию завершающий символ. Т.е., если у вас есть программа, написанная на C/C++, в заголовке окна которой написано что-то вроде "Cool Program - Unregistered", и вы не хотите видеть напоминание о том, что она "Unregistered", просто замените в файле программы пробел после слова Program на символ с кодом 0. После этого слово "Unregistered" вы почти наверняка больше не увидите. Этим же способом ненужную строку можно вообще превратить в пустую, просто поставив в ее начало завершающий символ!

Описанная техника укорачивания и "обнуления" строк пригодна не только для того, чтобы убирать неэстетичные надписи в заголовках программ, в действительности ее возможности гораздо шире. Приведу пару примеров из собственной практики. Один раз мне в руки попалась некая программа, которая очень любила при печати документов в заголовок вставлять надпись "This report created by demo version ..." шрифтом аршинного размера. Разумеется, мне это не понравилось, и при помощи нехитрых манипуляций в шестнадцатеричном редакторе я "обнулил" строку с надписью, ос-

корблявшей мои эстетические чувства. В другом случае я подобным же образом расправился с одной программой-генератором справок, которая считала, что незарегистрированность — это повод вставлять рекламный текст в каждую статью справки. Небольшой memory patch, исправлявший в программе "на лету" несколько байтов, смог убедить капризную программу в ее принципиальной неправоте.

Более эффективными по сравнению с ASCIIZ-строками являются строки с указанием длины. Такие строки позволяют использовать в тексте все 256 ASCII-символов, хранить не только текстовые, но и любые другие двоичные данные, а также применять по отношению к этим данным строковые функции. Кроме того, вычисление длины строки требует лишь одной операции чтения данных по ссылке, в отличие от ASCIIZ-строк, где для определения длины необходимо последовательно сканировать все символы строки до тех пор, пока не встретится завершающий символ. Как такового, стандарта на строки с указанием длины не существует — можно лишь говорить о конкретных реализациях таких строк в различных компиляторах и библиотеках. В частности, в коде программ на Delphi 7 строковые константы хранятся следующим образом:

- 4 байта — длина строки **в байтах** (для UNICODE-текстов это значение в два раза больше длины строки в символах);
- содержимое строки;
- завершающий символ (#0 для ANSI-строк, #0#0 для UNICODE-строк). Завершающий символ никак не используется в "родных" функциях и процедурах Delphi, но значительно упрощает вызов функций WinAPI (которые используют строки с завершающим символом) и использование сторонних библиотек.

Зная все это, нетрудно разработать способ укорачивания Delphi'йских строк — для этого требуется изменить длину строки в первых четырех байтах и поставить еще один завершающий символ в нужную позицию. Надо отметить, что тексты-свойства компонентов в ресурсах программ на Delphi хранятся в несколько ином формате, поэтому прежде чем пытаться вмешиваться в код программы, стоит побольше узнать об особенностях реализации встроенных типов в компиляторе, при помощи которого создана исследуемая программа.

Поиск текстовых констант в программе — совсем не такое простое дело, как это могло бы показаться с первого взгляда. Прежде всего, не следует полагаться на "интеллектуальность" дизассемблера — поиск текстовых строк при дизассемблировании обычно основан на анализе ссылок, встречающихся в коде программы. Поэтому, если в коде программы нет прямой ссылки на строку, дизассемблер эту строку может просто "не увидеть". Чтобы убедиться в этом, рассмотрим несложный пример:

```
.data
line1 db "Line 1",0
line2 db "Line 2",0
line3 db "Line 3",0
LineArr dd OFFSET line1, OFFSET line2, OFFSET line3

.code
...
GetMsgAddr proc MessageIndex:DWORD
mov ebx,MessageIndex
mov eax, OFFSET LineArr
mov eax,[eax+ebx*4]
ret
GetMsgAddr endp
...
```

Этот код представляет собой максимально упрощенную реализацию списка сообщений и функции, получающей адрес текстовой строки по номеру сообщения. Откомпилировав этот пример, загрузим его в W32Dasm и посмотрим, что получится. Получилось следующее — дизассемблер успешно распознал строку "Line 1", но строки "Line 2" и "Line 3" не обнаружил. А вот IDA успешно распознал все три строки, и создал для них именованные метки. Впрочем, и IDA при большом желании можно обмануть — достаточно лишь вписать перед текстом самой строки ее длину в байтах (именно так хранит строки Delphi). После этого IDA хотя и обнаруживает сам факт наличия текстовых строк в программе (в окне Strings эти строки видны), но в дизассемблированном тексте программы эти строки выглядят как последовательность db... , которые нужно приводить в желаемый вид вручную. Кстати, W32Dasm после этой модификации не увидит вообще ни одной строки. Если же вам и этого мало, вместо "Line 1" напишите "Строка 1" — все тексты на русском языке знаменитый дизассемблер гордо проигнорирует. И это только начало. А ведь текстовые строки могут находиться не только в сегменте кода/инициализированных данных, но и в секции ресурсов программы...

Здесь могут помочь специализированные программы, сканирующие указанный файл и вычленяющие из него все текстовые строки (или то, что похоже на текстовые строки). Кроме того, вам потребуются смещения этих строк от начала файла, поэтому ваш инструмент должен предоставлять и такой сервис. Однако использование таких программ (и самостоятельное их написание) осложняется двумя факторами — разнообразием существующих кодировок текста и существованием национальных символов в некоторых языках (классический strings.exe и многие другие подобные программы "не понимают" русскую секцию UNICODE). Те же проблемы с UNICODE и национальными кодировками характерны и для программного обеспечения, осуществляющего поиск в текстовых файлах. К тому же, русские тексты в UNICODE совершенно нечитабельны в шестнадцатеричных редакторах и просмотрщиках. Все это необходимо учитывать при выборе инструментов поиска текстовых строк, а выбранный инструмент перед использованием желательно проверить на подходящем "пробном камне".

Напоследок расскажу про весьма простой, но весьма эффективный в некоторых случаях способ поиска численных переменных в работающей программе. Этот способ основан на многократном сканировании адресного пространства программы, отслеживании и анализе всех изменений в этом пространстве. Лучше всего этот прием работает на программах, в которых установлено ограничение на количество тех или иных действий, вроде ограничения на число записей, добавляемых в документ. И используется для этого совсем не крэкерский инструментарий. Вы, наверное, знакомы с программами типа Game Wizard или ArtMoney, которые позволяют искать в работающей компьютерной игре количество денег или оставшихся жизней. Для тех, кто не сталкивался с такими программами, вкратце опишу алгоритм их работы:

1. Пользователь выбирает из списка работающих в данный момент программ подопытную игру.
2. Пользователь вводит в программу поиска начальное количество денег (хитов и т.п.), которое в данный момент существует в игре.
3. Программа сканирует адресное пространство и строит список всех значений (точнее, адресов, по которым расположены эти значения), совпадающих с введенными пользователем.
4. Пользователь выполняет в игре какое-либо действие, в результате которого количество денег изменяется.
5. В программу поиска вводится новое количество денег.
6. Программа проверяет все значения из построенного списка и исключает из него те значения, которые не соответствуют введенному пользователем.
7. Пункты 4...6 повторяются до тех пор, пока список адресов не укоротится настолько, чтобы можно было прове-

рить назначение каждого элемента списка вручную.

8. Пользователь проверяет каждый элемент списка, записывая по найденным адресам новые значения и наблюдая, как это повлияет на количество денег в игре.

Проницательный читатель наверняка уже догадался, что защищенная программа ничем в принципе не отличается от компьютерной игры, а число добавленных в документ записей — это то же самое, что количество виртуальных "золотых монет". И потому, воспользовавшись соответствующей программой (хотя бы все той же ArtMoney), можно определить адреса всех переменных, которые могут хранить счетчик добавленных в документ записей. Дальнейшие действия зависят исключительно от вашего желания — можно поставить аппаратную точку останова на чтение из этой переменной и попытаться добраться до команды сравнения существующего числа записей с максимальным. Можно погрузиться в изучение дизассемблерного листинга в поисках команды увеличения переменной и сделать так, чтобы значение счетчика не увеличивалось. Можно даже попробовать модифицировать счетчик из ArtMoney, посмотреть, что из этого получится, и если из этого получится что-то хорошее — написать memory patch, каждые 100 миллисекунд обнуляющий счетчик.

На этом мой рассказ о простых типах данных в основном закончен, а информацию о методах хранения составных типов, таких как структуры и массивы, о том, где обычно в программах лежат константы, а также об особой роли указателей вы найдете в следующей главе.

(Продолжение следует)

Автоматическая рекомпиляция и ее применение в целях защиты программных продуктов от методов обратной инженерии

А.ТЕРЁШКИН /HiTech.

1. Общие сведения о формате PE

Формат PE (Portable Executable) — формат исполняемых файлов (DLL и EXE) для Win32 и Win32s, а также драйверов WinNT. Этот формат был разработан Microsoft на базе COFF (Common Object File Format), используемом как формат объектных и исполняемых файлов в некоторых операционных системах семейства UNIX и VMS.

В начале каждого PE-файла находится MS-DOS EXE (формат MZ), называемый "stub", что делает каждый PE-файл исполнимым и в MS-DOS. При запуске файла в win32 система пропускает 16-битный код, находящийся в этом MZ-EXE.

После "stub" следует 32-битая сигнатура PE-файла, а за ней — файловый заголовок, описываемый структурой IMAGE_FILE_HEADER. В этом заголовке указывается тип машины, на которой будет исполняться этот файл; сколько секций он содержит; время, когда была осуществлена сборка файла; флаги (например, является ли файл DLL или нет) и т.д.

За файловым заголовком находится опциональный заголовок — IMAGE_OPTIONAL_HEADER. Этот заголовок присутствует в PE всегда, хотя и называется "опциональным". Название наследуется от COFF, в котором опциональный заголовок присутствовал в библиотеках, но не в объектных файлах. В этом заголовке указываются параметры загрузки образа — размер кода, предпочитаемый адрес загрузки, подсистема, резервируемый объем стека, и др. Опциональный заголовок заканчивается массивом структур IMAGE_DATA_DIRECTORY, указывающих на адреса соответствующих директорий в секциях.

За заголовками следует декларация секций, представленная массивом заголовков IMAGE_SECTION_HEADER. Каждый такой заголовок описывает одну секцию — ее имя, виртуальный и физический адреса, флаги и т.п. В секциях могут находиться исполняемый код, данные, ресурсы — то есть главное содержимое исполняемого файла. После массива структур IMAGE_SECTION_HEADER начинается непосредственно "память" секций.

Табл. 1

DOS stub
Сигнатура PE
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
Массив IMAGE_DATA_DIRECTORY
Массив IMAGE_SECTION_HEADER
Секция 1
...
Секция n
Оверлей

В целом, любой PE файл можно представить так, как показано в **табл.1.**

1.1. Относительные виртуальные адреса (RVA)

Для представления адреса объекта в памяти формат PE использует относительную виртуальную адресацию. Относительный виртуальный адрес (Relative Virtual Address, RVA) используется, когда неизвестен базовый адрес. RVA —это число, которое надо прибавить к базовому адресу для получения линейного адреса.

Базовый адрес — это адрес, по которому загружен PE-модуль. Он может варьироваться из-за коллизии регионов модуля, загруженного в память, и загружаемого модуля — в этом случае, при наличии у загружаемого модуля таблицы релокаций, происходит его перебазирование; базовый адрес не будет совпадать с предпочитаемым.

Относительный виртуальный адрес может не совпадать с относительным физическим адресом (Relative Physical Address, RPA), считающимся относительно начала файла. Это происходит, когда содержимое секций в файле выровнено не так, как в памяти. Очень часто секции в файле выравниваются на границу 512-байтного блока, а в памяти — на границу страницы (4096 байтов).

Перевод RVA в RPA производится следующим образом — ищется секция, региону которой принадлежит данный RVA (т.е. данный RVA должен быть не меньше RVA начала секции и меньше RVA начала секции плюс ее физический размер), затем считается смещение RVA относительно начала найденной секции, и полученное смещение прибавляется к относительному физическому смещению начала этой секции.

1.2. Методы обратной инженерии

Обратная инженерия — частичное или абсолютное восстановление программного кода в исходных текстах — применяется с разными целями. Одной из самых оправданных целей является самообучение — реконструирование каких-либо know-how программного продукта, если его исходные тексты недоступны. Другой часто встречающейся целью является анализ защиты shareware-продуктов для ее последующей нейтрализации.

Разумеется, разработчики нередко прибегают к хитростям, чтобы усложнить работу "крэкеров". Они направлены против следующих методов анализа:

- дизассемблирование. Дизассемблированию могут быть подвергнуты как файл, так и память;
- анализ памяти работающей программы с целью поиска каких-либо структур, используемых активной защитой;
- трассировка исполнения работающей программы;
- эмуляция исполнения.

Подробнее рассмотрим эти методы.

Дизассемблирование может быть полным или частичным — только алгоритма защиты. Для частичного дизассемблирования самым сложным этапом является локализация алгоритма защиты. Методы противодействия этому дизассемблированию весьма разнообразны, они (впрочем, как и защиты) варьируются от тривиальных до сверхсложных. Условно их можно разделить так:

- против дизассемблирования файла — шифрование отдельных участков до старта программы и расшифровка при старте. Таким образом, в файле не содержится открытого для дизассемблирования кода;
- модифицирующийся код — опкоды изменяются до или даже во время их исполнения, что влечет за собой изменение алгоритма и сбивает с толку исследователя;
- разброс алгоритма защиты по всей программе, совмещение ее с другими блоками программы, что усложняет процесс локализации защиты;
- искусственное "нагромождение", запутывание кода в алгоритме защиты, маскировка этого алгоритма;
- switch и им подобные методы (использование в защите эмуляции работы конечных автоматов).

Анализ памяти работающей программы. Метод основан на сопоставлении снимков памяти программы в разные моменты ее работы либо при неоднократных запусках. Главным моментом здесь является чтение памяти чужого процесса. Средствами Win32 можно написать эту процедуру несколькими способами, причем программе практически нельзя активно противодействовать грамотно написанному продвинутому дамперу памяти — память все равно будет сохранена.

Размещение критичных участков в памяти, которая не будет сохранена при дампе процесса: например, в памяти другого процесса либо по адресу выше 2 Гб (для NT необходимы администраторские привилегии).

Маскирование вызовов API-функций. Метод, примененный в криптере PE "Armadillo" — технология CopyMem II. Защищенный EXE сам себя запускает в режиме отладки. Перед стартом главного потока в контекст процесса копируется только страница, содержащая точку входа. По мере поступления глобальных исключений защиты (GPF) от отлаживаемой программы (из-за обращения к незагруженным страницам), отладчик расшифровывает и копирует требуемую страницу в контекст отлаживаемой программы и отпускает поток, вызвавший исключение. Попутно отладчик убирает из контекста неиспользуемые страницы. Слабость этого метода — в процедуре определения адреса страницы, обращение к которой вызвало исключение. Достаточно искусственно зациклить область кода "определение адреса-> адрес-> скопировать страницу" для копирования *всех* изначально присутствовавших в программе страниц, и запретить их выгрузку, как мы получим почти готовую для исполнения программу без Armadillo.

Трассировка исполнения работающей программы. Данный метод позволяет упростить локализацию алгоритма защиты, а также способствует процедуре его реверсирования, так как предоставляет практически безграничные возможности по изменению алгоритма в процессе его исполнения. Средствами Win32 трассировку можно вести как на уровне пользователя, так и в режиме ядра. Метод трассировки является самым популярным, и при анализе защит используется практически всегда, поэтому имеются самые разнообразные приемы противодействия ему:

- противодействие отладчикам 3-го кольца — отладчик чаще всего имеет не больше привилегий, чем отлаживаемая программа, поэтому определить присутствие отладчика не составляет труда. Дальнейшие действия программы могут варьироваться от запуска отладчика по ложному следу до стирания копий FAT (именно этим опасно трассирование без анализа трассируемого кода);
- противодействие отладчикам 0-го кольца — отладчик работает в режиме ядра, что не позволяет трассируемому коду нарушить его работу без выхода на такой же уровень привилегий (в системах WinNT). Поэтому чаще всего противодействие сводится к приемам, направленным лишь против определенных (популярных) отладчиков, в большинстве случаев имеющих интерфейс 0-е кольцо <-> 3-е кольцо (сброс аппаратных точек останова, выполнение backdoor-команд отладчика). Практически все отладчики 0-го кольца имеют "следы" в 3-м, что позволяет также определить их присутствие;
- сброс программных точек останова отладчиков 0-го и 3-го колец — точки останова (опкоды "INT 3" — 0xCC) обычно расставляются взломщиком в начало функций API, вызываемых защитой, и в процедуры самой защиты после анализа дизассемблированного кода программы. В большинстве случаев точки останова для предварительного анализа кода расставляются в начала интересующих процедур. Определить такую ситуацию несложно — достаточно проверить такие "слабые" места на изменение кода. Сложнее — корректно восстановить замещенный байт. Решение очевидно, если точка останова была поставлена в начало процедуры, оформленное обычным образом — организацией стекового фрейма (PUSH EBP / MOV EBP, ESP). Определение факта изменения кода в процессе исполнения можно производить подсчетом контрольной суммы "опасных" участков.

Эмуляция исполнения. Если отладчики, описанные выше, проводят "физическую" трассировку, пользуясь отладочными средствами ОС, то трассировка этого типа отладчиков — "логическая", т.е. код программы не исполняется. Последним таким отладчиком с пользовательским интерфейсом является TR 2.52 (DOS), написанный Liu TaoTao. Написание полнофункционального эмулятора кода для Win32 — задача почти невыполнимая из-за огромного числа объектов, эмулирование работы с которыми необходимо. Несмотря на это, существуют узкоспециализированные эмуляторы — это эвристические анализаторы антивирусов. Противодействие эмуляторам делится на два вида:

- активное использование кодом объектов, сложных для эмулирования, с целью поиска расхождений результатов с результатами, полученными в "живой" системе;
- использование известных уязвимостей популярных эмуляторов.

2. Декомпиляция

Здесь и далее под декомпиляцией региона исполняемого файла подразумевается процесс представления указанного региона памяти этого файла в виде последовательности блоков, удовлетворяющих следующим требованиям:

- при выстраивании блоков в порядке возрастания их виртуальных адресов, участки памяти, на которые ссылаются блоки, представляют собой декомпилируемый регион исходного файла;
- блоки делятся на "представляющие код", "представляющие данные" и "метки". Разделение проводится с точностью,

Табл. 2

Флаг	Описание
FL_RVA	Блок содержит значение относительного виртуального смещения
FL_DELTA	Содержимое блока — расстояние между двумя метками
FL_FIXUP	Элемент релокации (см. формат PE)
FL_LABEL	Метка
FL_OPCODE	Начало опкода (первый байт)
FL_CODE	Какой-либо байт опкода
FL_HAVEREL	Опкод имеет относительный аргумент
FL_CREF	Метка кода
FL_DREF	Метка данных
FL_NEXT	Временный флаг декомпиляции
FL_PHYS	Аргумент блока — физическое относительное смещение (в отличие от виртуального)
FL_PRESENT	Блок присутствует физически
FL_VPRESENT	Блок присутствует виртуально (только в памяти)
FL_FORCEOBJALIGN	Начало блока должно быть выровнено виртуально, т.е. секционное выравнивание
FL_FORCEFILEALIGN	Начало блока должно быть выровнено физически, т.е. файловое выравнивание
FL_DATA	Содержимое блока — данные
FL_STOP	Маркер конца ветви кода
FL_JUMPTABLE	Маркер таблицы переходов
FL_BLOCKSTART	Начало замкнутого блока
FL_BLOCKEND	Конец замкнутого блока

позволяющей выделить в алгоритме декомпилируемого модуля все непрерывные участки кода и данных (когда это возможно), а также относительные указатели (кода на код и кода на данные), и отделить абсолютные ссылки от констант;

- блок может объединять последовательно идущие байты, но с условием, что каждый байт такой совокупности будет иметь один и тот же набор флагов;
- в случае представления кода, в блоке не может находиться более одной инструкции.

Возможные значения флагов блока приведены в **табл.2.** Блок может иметь одновременно несколько флагов.

2.1. Этапы декомпиляции

Для успешного завершения процесса декомпиляции необходимо иметь корректную информацию о длине блоков. По определению, данные одного блока имеют один и тот же флаг, и все участки памяти декомпилируемого региона, имеющие на себя ссылки, помечены как "метки". Следовательно, в результате декомпиляции в наборе блоков будут присутствовать блоки нулевой длины с флагом "метка" (FL_LABEL), а также блоки длиной 4 байта с флагом "ссылка" (FL_RVA, FL_FIXUP, FL_HAVEREL или FL_DELTA).

Таким образом, необходимо произвести предварительный обход модуля для пометки участков его структуры как "метки", "ссылки", "элементы релокации".

В случае, когда производится декомпиляция модуля целиком, также необходимо пометить соответствующие элементы структуры как "присутствующие физически", "присутствующие виртуально", "секционное выравнивание", "файловое выравнивание", "физическое смещение", "ресурсы".

2.2. Анализ структуры PE

На этом этапе происходит лишь пометка участков памяти региона соответствующими флагами. Будем рассматривать наиболее общий случай декомпиляции — декомпиляцию модуля целиком, т.е. включая всю его структуру.

Рассмотрим операции, с помощью которых производится пометка памяти флагами.

1. "Пометить RVA" (MarkRva)

```
void MarkRva(DWORD rva)
{
flag[rva]|=FL_RVA|FL_DATA;
flag[ibaseDD[rva]]|=FL_LABEL|FL_DREF;
arg1[rva]=ibaseDD[rva];
}
```

Служит для описания ссылок и меток. Адрес памяти, относительно ссылающийся на какой-либо адрес памяти, помечается как RVA и "данные". Адрес ссылки помечается как "метка, имеющая ссылку из данных".

2. "Пометить RVA, ссылающуюся на данные" (MarkRvaData)

```
void MarkRvaData(DWORD rva)
{
flag[rva]|=FL_RVA|FL_DATA;
flag[ibaseDD[rva]]|=FL_LABEL|FL_DREF|FL_DATA;
arg1[rva]=ibaseDD[rva];
}
```

Служит для описания меток и ссылок на данные.

Адреса памяти, относительно ссылающиеся на какой-либо адрес памяти, помечаются как RVA и "данные". Адреса ссылки помечаются как "метка на данные, имеющая ссылку из данных".

3. "Пометить данные, являющиеся размером какого-либо объекта структуры PE-модуля, как дельта-смещение одного RVA относительно другого" (MarkDelta)

```
void MarkDelta(DWORD rva,DWORD destrva,DWORD blocksize)
{
flag[rva]|=FL_DELTA;
flag[destrva]|=FL_LABEL|FL_DREF;
flag[destrva+blocksize]|=FL_LABEL|FL_DREF;
arg1[rva]=destrva;
arg2[rva]=destrva+blocksize;
}
```

Служит для пометки дельта-смещений. Этот тип данных необходимо обрабатывать отдельно, так как дельта-смещения объектов PE-файла могут измениться после обработки содержимого блоков, полученных после декомпиляции, поэтому для рекомпиляции модуля необходимо скорректировать все помеченные дельта-смещения.

Переданный функции адрес памяти помечается как дельта-смещение. Адреса ссылок начала и конца участков помечаются как "метки, имеющие ссылки из данных".

4. "Пометить элемент релокации" (MarkFixup)

```
void MarkFixup(DWORD rva,DWORD preferredibase)
{
flag[rva]|=FL_FIXUP;
arg1[rva]=ibaseDD[rva]-preferredibase;
if(arg1[rva]<(poh->SizeOfImage))
        flag[arg1[rva]]|=FL_LABEL|FL_DREF;
#ifdef DEBUG_RELEASE
else
printf("\tWarning! Relative array addressing at .%X\n",rva);
#endif
}
```

Служит для пометки элементов релокации модуля. Адрес элемента релокации помечается флагом FL_FIXUP. Отличие релокаций от RVA состоит в том, что число, являющееся элементом релокации, равно виртуальному адресу ссылки, т.е. от адреса ссылки не отнимается база.

Если адрес ссылки присутствует в регионе модуля, то он помечается как "метка, имеющая ссылку из данных".

Анализ структуры PE начинается с выделения его заголовков — файлового, опционального и первого секционного.

Затем с помощью MarkRva помечаются:
- адрес точки входа (AddressOfEntryPoint);
- начало исполняемого кода (BaseOfCode);
- начало инициализированных данных (BaseOfData).

С помощью MarkDelta помечаются:
- размер образа (SizeOfImage);

- размер кода (SizeOfCode);
- размер заголовков (SizeOfHeaders).

Для каждой секции в модуле:

- физически присутствующие байты помечаются флагами FL_PRESENT, виртуально — как FL_VPRESENT;
- как дельта-смещения помечаются виртуальный и физический размеры секции;
- помечается виртуальный адрес начала секции.

По разнице между размером файла и адресом физического конца последней секции определяется наличие оверлея.

Для всех директорий данных в модуле помечаются RVA их начала и дельта-смещение их размера. Каждый байт директорий помечается как "содержащий данные" — FL_DATA.

Отдельно обрабатываются директории импортов, экспортов, релокаций и ресурсов.

В директории импорта как указатели на данные помечаются имена всех переходников (thunk), и для обеих ветвей (FirstThunk и OriginalFirstThunk) как данные помечаются адреса импортируемых функций API, которые будут заполнены загрузчиком, и имена этих функций.

В директории экспорта как данные помечаются:

- адрес имени модуля;
- адрес указателя на массив имен экспортируемых функций;
- адрес указателя на массив адресов экспортируемых функций;
- адрес указателя на массив числовых имен (AddressOfNameOrdinals).

Для каждой экспортируемой функции ее имя и адрес помечаются как ссылка на RVA.

В директории релокаций перебираются все элементы и, исходя из типа элемента, вычисляется его виртуальный адрес. Затем этот адрес помечается как указатель на элемент релокации (MarkFixup).

Дерево ресурсов обходится рекурсивно; для каждого элемента помечается его размер. Если элемент представляет собой данные, ссылки на них маркируются с помощью MarkRvaData. Входы веток помечаются как "ресурсы" (FL_RES8).

(Окончание следует)

Пишем сортировщик E-mail-адресов на Delphi

ViNCE.

Предисловие

В настоящее время в WWW размещено огромное количество электронных почтовых ящиков, или e-mail. На многих серверах совершенно разной информационной направленности (развлечения, отдых, кино, музыка...) хранятся (существуют) базы таких адресов. Самым элементарным примером хранения базы e-mail может служить обыкновенный форум. Когда новый пользователь заходит на форум и регистрируется, в базу заносится его e-mail. Безусловно, базами e-mail'ов пользуются не только форумы, но и люди, занимающиеся рассылкой рекламы или мусора (спамеры). Пожалуй, все "интернет-жители" сталкивались с "трудом" спамеров — рассылками с бесполезным содержанием, например, письмами с подобного рода текстом: "...и вы заработаете за неделю $100000, и это реальность". Я крайне отрицательно отношусь к подобной деятельности, но речь в статье пойдет о другом.

Что? Зачем? И как?

В нашем случае мы напишем простенькую программку, которая займется сортировкой e-mail-адресов по заданной маске. Зачем нужна подобная программа?

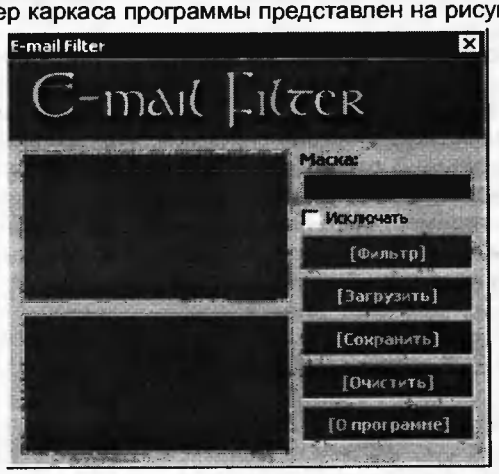
Во-первых, возможно, по роду своей профессиональной деятельности вам необходимо рекламировать определенный продукт, или вы — владелец какой-нибудь компании и рассылаете своим определенным сотрудникам определенные сообщения. Для более простого понимания ситуации рассмотрим пример. Допустим, в вашей фирме одно отделение — отдел продаж, а второе — бухгалтерия. Адреса отдела продаж выглядят следующим образом — sotrudnik@buy.company.ru, а адреса бухгалтеров — sotrudnik@buh.company.ru. Заметна разница! При необходимости посылки сообщения всем бухгалтерам вам потребуются все адреса e-mail бухгалтеров. Если слать все и всем подряд, то из этого получится спам (spam). Наша программа и займется сортировкой адресов по заданной маске.

Во-вторых, вы можете быть спамером (я надеюсь что это не так), и у вас есть большая база e-mail'ов, например, на 150000 строк. Отбирать нужные адреса из такого списка очень неудобно и долго, а наша программа с соответствующими настройками вам поможет.

В-третьих, вы можете быть неважно кем, но у вас есть большая база e-mail'ов, в которой есть "мертвые адреса", а вручную их удалять слишком трудоемко. Используя нашу программу в альтернативном режиме, можно выявить нужные адреса, которые будут исключены из базы.

Собственно кодинг

Итак, вступительная часть завершена, и мы приступим к написанию нашей программы. Для начала создадим форму и расположим на ней все необходимые компоненты. Пример каркаса программы представлен на рисунке.



Для создания приятного дизайна программы я использовал Photoshop, но это — другая история. Каркас получился из 13 компонентов: Image1 — 1; ListBox1 — 1; Memo1 — 1; Label1 — 1; CheckBox1 — 1; Panel — 5; Edit1 — 1; OpenDialog1 — 1; SaveDialog1 — 1. Как видите, компонентов не так уж и много, теперь осталось "придать им жизнь". Кликаем по кнопке "[Фильтр]", и открывается процедура нажатия на кнопку. Чуть выше я уже говорил, что программа будет работать в двух режимах сортировки: первый — выборка отдельных

адресов e-mail по маске, второй — исключение адресов из списка e-mail в соответствии с маской.

При работе программы нам потребуются некоторые переменные: c_email — для хранения текущего e-mail-адреса; i — для организации циклов (for).

После "разборки" с переменными напишем в процедуре код (для работы в первом режиме):

```
Memo1.Clear; //Подготавливаем поле для ввода e-mail'ов
if CheckBox1.Checked=false then
begin
 if ListBox1.Items.Count=0 then
 ShowMessage('Список фильтруемых адресов e-mail пуст!');
 for i:=0 to ListBox1.Items.Count-1 do
```

Рассмотрим этот кусок кода более подробно. Сначала идет проверка режима работы программы (если нет галочки, то приступаем к выполнению следующего кода...). Далее идет проверка, есть ли в e-mail-списке адреса. Если нет, то пользователь получает оповещение. После этого запускается цикл for, который работает с каждым отдельным e-mail-адресом:

```
//Получаем i-адрес
 c_email:=ListBox1.Items[i];
   if pos(Edit1.Text,c_email)>0 then
   Memo1.Lines.Add(c_email);
```

В переменную c_email записывается текущий обрабатываемый адрес. Далее следует проверка адреса на наличие в нем символьного сочетания маски. Например, адрес — "support@company.ru", а маска — comp. Так как в этом адресе есть сочетание "comp", то адрес подойдет под фильтр и будет выведен в поле Memo.

Код для выборки отдельных адресов из списка e-mail готов, теперь напишем код для исключения из списка отдельных адресов и вывода новой базы. Тут все просто — как и в первом случае, копируем код свеженаписанной процедуры и немного его модифицируем:

```
if CheckBox1.Checked=true then
 begin
  if ListBox1.Items.Count=0 then
  ShowMessage('Список фильтруемых адресов e-mail пуст!');
  for i:=0 to ListBox1.Items.Count-1 do
   begin
   //Получаем i-адрес
   c_email:=ListBox1.Items[i];
   if not (pos(Edit1.Text,c_email)>0) then
   Memo1.Lines.Add(c_email);
  end;
 end;
```

В чем отличие? Мы добавили к условию проверки наличия символов в адресе e-mail команду not (отрицание) и тем самым изменили условие на противоположное. Теперь оно "звучит" так: "Если в текущем адресе e-mail не найдено символов маски, то выводим этот адрес в поле Memo".

Для удобства работы с программой можно добавить (как показано на рисунке) кнопки "[Сохранить]" и "[Загрузить]" для импортирования и экспортирования базы адресов. Давайте напишем код для этих кнопок:

```
//Кнопка "[Загрузить]"
procedure TForm1.Panel3Click(Sender: TObject);
begin
if OpenDialog1.Execute then
ListBox1.Items.LoadFromFile(OpenDialog1.FileName);
end;
//Кнопка "[Сохранить]"
procedure TForm1.Panel5Click(Sender: TObject);
begin
if SaveDialog1.Execute then
Memo1.Lines.SaveToFile(SaveDialog1.FileName);
end;
```

Также необходимо установить фильтр для компонентов OpenDialog1 и SaveDialog1. Для этого впишите в поле "Filter" обоих компонентов следующий текст: "All Files (*.*)|*.*".

Дополнительно можно добавить кнопку "[Очистить]", которая будет очищать поля ListBox1 и Memo1.

Все! Программа готова, ее осталось только откомпилировать и проверить на любой e-mail-базе.

Еще раз о System Tray

А.СНИТКО,
г.Мосты.
E-mail: zx4ever@torba.com
WWW: http://www.zx4ever.narod.ru

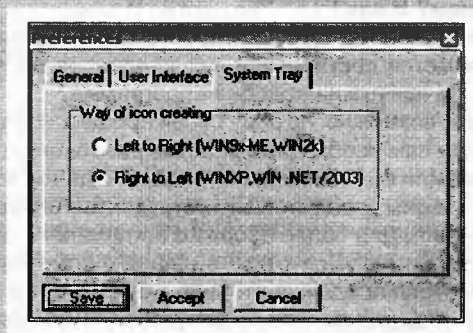
Как-то раз, исследуя возможности операционной системы Windows .NET Enterprise Server 2003, я натолкнулся на интересную особенность работы System Tray в этой системе. Состоит она в том, что создающиеся иконки помещаются в область состояния в порядке, обратном, чем в других системах Windows. В результате, например, если намечается, что System Tray будет выглядеть как показано на **рис.1**, вы получите совсем другое (**рис.2**).

Рис. 1

Рис. 2

Решение этой проблемы приходит само собой — создавать иконки в обратном порядке. При этом для совместимости с различными версиями ОС необходимо либо автоматически определять версию Windows, либо предоставить пользователю возможность вручную выбирать направление создания иконок, что, по моему мнению, более практично и надежно. Так я и сделал в своем ThunderPlayer (**рис.3**).

Рис. 3



Конечно, Windows .NET Server 2003 на момент написания статьи является еще достаточно малораспространенной и незнакомой ОС для большинства пользователей. Но не пугайтесь, так как все описанное будет справедливо и для Windows XP — обе системы основаны на одном и том же Explorer.

Обратная сторона Луны, или дисковые перевороты

А. ГОРЯЧКИН,
г. Кыштым, Челябинской области,
E-mail: anatoliy_goryach@mail.ru

Наверное, у каждого пользователя возникала ситуация, когда при форматировании трехдюймовой дискеты операционная система выдавала сообщение типа "у дискеты повреждена нулевая дорожка, и ее невозможно отформатировать". Такие дискеты правильнее всего сразу выбросить в мусорную корзину и навсегда забыть о них. Так поступил бы любой иностранец, но российский менталитет нам этого делать не позволяет. Наши отечественные "очумелые ручки" не желают с этим мириться и пытаются продлить жизнь дискет, причем небезуспешно. Так что не спешите выбрасывать дискеты с поврежденной нулевой дорожкой. У вас есть реальный шанс восстановить их.

Для того чтобы понять, что такое нулевая дорожка трехдюймовой дискеты, будет не лишним освежить в памяти некоторые теоретические знания. На нулевой дорожке находится системная область диска, в которой располагаются: загрузочный сектор (boot-sector), две копии 12-разрядной FAT (таблицы размещения файлов) и корневой каталог. Нулевая дорожка расположена по внешнему диаметру дискеты. Таким образом, при стандартном форматировании трехдюймовая дискета 1,44 Мб размечается на 80 дорожек и 18 секторов. Дорожки представляют собой сплошные концентрические кольца, а сектора можно наглядно представить в виде долек круглого торта. Есть два простых способа восстановления дискет с поврежденной нулевой дорожкой. Первый способ — программный. Существуют программы-утилиты, которые позволяют в случае повреждения нулевой дорожки восстанавливать ее. Причем после восстановления для доступа к такой дискете не нужно подгружать какие-либо резидентные драйверы. Одна из таких программ — утилита для форматирования гибких дисков Fformat v2.70. Эта DOS-программа довольно "древняя" (год выпуска — 1993), но продолжает свою жизнь в силу того, что обладает способностью восстанавливать "убитые" дискеты. Причем делает это довольно успешно — в 80% случаев утилита "возвращает" дискету в исправном рабочем состоянии. К достоинствам программы Fformat v2.70 относится и то, что она форматирует дискеты таким образом, что скорость обращения к дискете увеличивается в 1,5...1,7 раза. Это достигается за счет применения оптимизации размещения секторов при форматировании дискеты.

Все отформатированные утилитой Fformat дискеты нормально работают в операционной системе Windows 9x/Me. Интересен тот факт, что если попытаться загрузиться с дискеты, отформатированной Fformat, то вместо стандартного сообщения типа "Non-System disk or disk error" на экране монитора появится окно с сообщением, что диск не является системным. Вся процедура рисования окна, в числе прочих прибамбасов, умещается всего в 512 байтах.

Разработчик утилиты Fformat v2.70 — Алексей Шамароков. Меню программы составлено на английском и русском языках, поэтому освоить работу с программой будет несложно. Пользователи Windows 9x могут работать с утилитой либо в режиме эмуляции MS-DOS, либо загрузившись с загрузочной дискеты. По этой причине, к сожалению, не могу показать скриншот программы Fformat. К программе прикладывается весьма подробное описание, которое заинтересует не только пользователей, но и программистов. Утилиту Fformat v2.70 можно скачать по адресу http://www.anatoliygor.nm.ru/download.htm

Второй способ — "хирургический". Его применяют в тех случаях, когда утилита Fformat v2.70 не смогла восстановить "убитую" дискету. Суть этого метода состоит в том, что надо перевернуть находящийся внутри корпуса дискеты магнитный диск. Для этого с помощью перочинного ножа разбирают корпус дискеты и извлекают оттуда магнитный диск. Полностью разбирать корпус дискеты нет необходимости, тем более, что это затруднит последующую сборку. Достаточно "раскупорить" один из углов корпуса (с той стороны, где нет шторки) — и это позволит без особого труда извлечь магнитный диск.

После того как магнитный диск будет извлечен, его отделяют от металлического "пятачка" (кольца сцепления), находящегося в центре диска, переворачивают и снова плотно прижимают к "пятачку". Потом магнитный диск вставляют в корпус дискеты. Корпус дискеты аккуратно заклеивают, после чего дискету форматируют как обычно, штатными средствами.

Для чего нужно переворачивать магнитный диск? Дело в том, что нулевая дорожка расположена только на одной стороне магнитного диска. После "переворота" нулевая дорожка окажется на другой стороне диска, и будет теперь иметь N1. Операцию по переворачиванию магнитного диска надо проводить очень аккуратно. Ни в коем случае нельзя прикасаться пальцами к рабочей поверхности магнитного диска. Магнитный диск желательно извлекать в медицинских резиновых перчатках (только не в тех, в которых работают электрики, а в тех, которые применяются в хирургии и гинекологии) во избежание попадания грязи и жира на магнитный слой. Не стоит допускать перегибов, заломов, а также появления царапин на поверхности магнитного диска.

В обоих случаях после форматирования произведите проверку физического состояния поверхности дискет с помощью программы ScanDisk, включив для этого опцию "Полная проверка".

Если же дисковая "реанимация" не поможет восстановлению, и "пациент" (то бишь дискета) будет скорее мертв, чем жив, то отчаиваться не стоит. Корпус трехдюймовой дискеты можно с успехом использовать в качестве чехла для "продвинутых" шпаргалок. "Фишка" заключается в замене вконец испорченного магнитного диска на картонный. Для этого гибкий магнитный диск, извлеченный из корпуса дискеты, отделяют от металлического "пятачка" и используют в качестве трафарета, по которому вырезают из плотной бумаги или тонкого картона соответствующего диаметра кружок и отверстие в центре круга. Далее, на картонном кружке пишут с обеих сторон шпаргалку и приклеивают к металлическому "пятачку". Затем картонку вставляют в корпус дискеты. Заклеивать в этом случае корпус дискеты не нужно, достаточно скрепить обе половинки корпуса прозрачным скотчем. Это позволит быстрее и удобнее менять шпаргалку для следующего экзамена.

На корпусе дискеты имеется шторка, отодвигая которую, вы увидите то, что написано на картонном диске. Для обзора всей поверхности картонного диска вращают находящийся в центре металлический "пятачок". Если преподаватель заметит дискету и попросит убрать со стола посторонние предметы, скажите ему, что забыли дома линейку и используете дискету для построения графиков, прямых углов, линий и т.д.

Совет по теме

При покупке новых дискет лучше приобретать те, у которых на нижней части корпуса указан номер (обычно десятизначный). Пронумерованные дискеты прослужат дольше дискет, у которых эти номера отсутствуют, так как они более качественны.

Вместо заключения. Даешь вечную жизнь дискетам! Да здравствует наши современные Левши и Кулибины!

Устройство дистанционного отключения модемов

В.ВАСИЛЕНКО,
г.Свердловск,
Луганской обл.

Современный модем — это сложное устройство, обладающее широкими возможностями. Он может автоматически устанавливать связь с другим модемом, принимать и передавать факсимильные сообщения, использовать сложные протоколы сжатия данных и коррекции ошибок, автоматически изменять скорость обмена сигналами в зависимости от качества телефонной линии и т.д. Многие модемы позволяют производить подстройку приемной части в зависимости от уровня принимаемого сигнала, изменять уровень передаваемого сигнала и параметры набора номера. Некоторые модели (например, IDC 2814) могут, пропустив тестирующие сигналы, оценить частотную характеристику канала, измерить отношение сигнал/шум и некоторые другие параметры. Но даже самые "навороченные" модемы, специально адаптированные под отечественные линии, иногда "зависают", и вывести их из этого состояния можно только выключением питания. Дело может усложняться в случае удаленного доступа, особенно если компьютер (или почтовый сервер) работает с несколькими модемами. Необходимо добраться до места физического расположения модема и вручную выключить/включить питание. Если компьютер имеет несколько модемов, можно использовать устройство дистанционного отключения модемов (УДО). Блок-схема такого устройства изображена на **рис.1**.

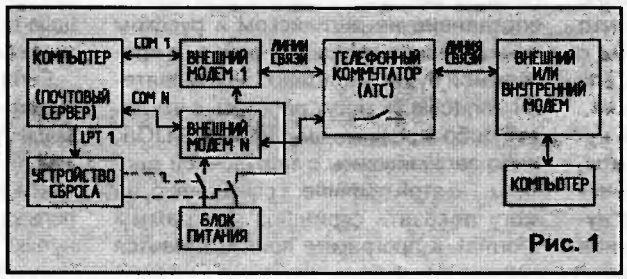
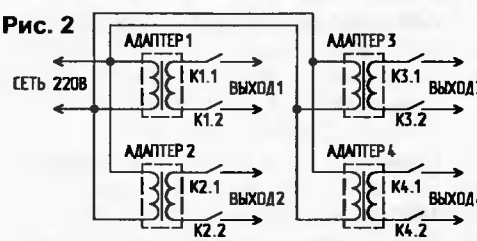
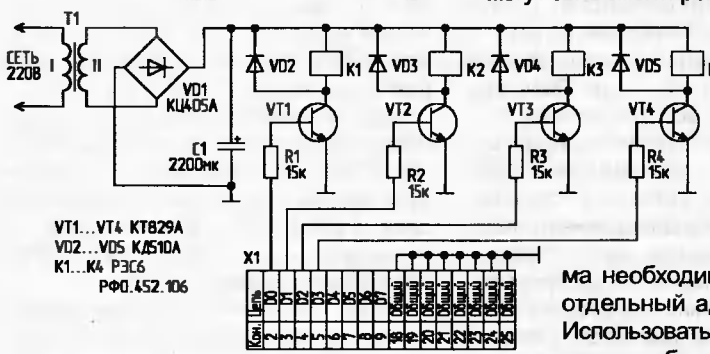


Рис. 1

Компьютер (или почтовый сервер) имеет несколько внешних модемов, работающих через его последовательные (COM) порты. Каждый модем имеет свой блок питания (адаптер), представляющий, по существу, понижающий трансформатор. Каждый модем имеет (в идеале) свой присвоенный телефонный номер, т.е. отдельный выход на автоматическую телефонную линию (АТС). Дистанционно модемы можно отключать посредством другого компьютера с модемом. Необходимо установить связь с каким-либо модемом почтового сервера (маловероятно зависание всех модемов сервера одновременно) и активизировать каким-либо образом управляющую программу, которая на некоторое время (несколько секунд) выставляет уровень логической "1" в разрядах параллельного (LPT) порта сервера. Уровни напряжения в разрядах LPT-порта служат входными сигналами для УДО, представляющего собой отдельную конструктивную единицу. Исполнительные элементы УДО отключают питание от того или иного модема.

Принципиальная схема УДО изображена на **рис.2**, она достаточно тривиальна. По существу, УДО представляет собой набор транзисторных ключей, в коллекторные цепи которых включены электромагнитные реле, контактные группы которых коммутируют выходы

Рис. 2




адаптеров модемов и местного источника питания, который состоит из понижающего трансформатора T1, диодного моста VD1 и фильтрующего конденсатора C1. В базовые цепи транзисторов VT1...VT4 включены токоограничивающие резисторы R1...R4. На схеме изображены четыре ключа (используются разряды D0...D3 LPT-порта), но без особых проблем их количество можно увеличить до восьми, управляя таким образом восемью модемами. Коллекторные цепи транзисторов защищены от выбросов напряжения, возникающих в моменты срабатывания реле, защитными диодами VD2...VD5. В качестве ключей используются составные транзисторы КТ829А, в качестве исполнительных устройств — реле РЭС6 (паспорт РФ0.452.106 или РФ0.452.116). Понижающий трансформатор должен обеспечивать на вторичной обмотке напряжение 10...12 В при токе не менее 0,5 А (в авторском варианте использовался накальный трансформатор с напряжениями на вторичных обмотках 5 и 6,3 В). Можно использовать трансформаторы с другими характеристиками и, соответственно, другие реле. Необходимо только проследить (в случае использования реле с более высоким значением рабочего тока), чтобы транзисторы обладали достаточным коэффициентом усиления по току. К подбору конкретных экземпляров транзисторов необходимо отнестись внимательно. Мне встречались транзисторы серии КТ829А последних лет выпуска с пониженным сопротивлением коллектор-эмиттер, повышенным обратным током коллектора, причем это могло проявляться только в процессе работы.

Конструктивно УДО собрано в отдельном корпусе. Разъемы для подключения адаптеров питания модемов могут быть смонтированы на общей печатной плате (вместе с ключами) или выведены наружу посредством гибких проводников. При этом для каждого модема необходимо использовать отдельный адаптер питания. Использовать для питания модемов общий понижающий трансформатор не следует. Схемы питания и отключения питания различных модемов имеют свои особенности, к тому же, модемы гальванически соединяются через общие провода разъема LPT. В случае использования общего понижающего трансформатора могут быть различные неприятности, вплоть до выгорания выпрямительных диодов и других элементов модемов (в чем автор и убедился на собственном опыте :).

Литература

1. Игловский И.Г., Владимиров Г.В. Справочник по слаботочным электрическим реле. 3-е изд., перераб. и доп. — Л.: Энергоатомиздат, 1990.

iS-DOS
Frequently Asked Questions

Е.ИЛЯСОВ,
iS-DOS support team (ist), 2002-2003.
412302, г. Балашов, ул. Красина, 82.

За несколько лет жизни в iS-DOS'е через наши органы чувств, причем в обоих направлениях, прошло множество вопросов и ответов по iS-DOS'ной теме. Самые частые, интересные или важные из них стали основой для "разбора полетов", приведенного ниже.

В качестве проблем, послуживших источником вдохновения для комментирования по тематике iS-DOS и обозначения своей точки зрения, были выбраны реальные вопросы, встречавшиеся в ходе контактов с другими пользователями (выдержки из писем приведены дословно (досимвольно!), с сохранением орфографии и пунктуации), или взятые из других источников — синклеровских электронных и полиграфических изданий. К ним были также добавлены и некоторые "виртуальные" вопросы, которые мы нередко задавали сами себе в ходе выяснения нюансов взаимодействия с системой iS-DOS.

Оболочка

User> *... Кроме того сама система IS Dos тотально убивает диски, особенно 5,25. После двух-трех месяцев юзания этой "дискоубийцы", отформатировать дискету не удается никакими способами.*

ist> Действительно, система активно использует обращения к диску. Точнее, к системному устройству. Только не стоит забывать, что современный парк реальных Sinclair-совместимых компьютеров примерно на 90% состоит из машин с памятью не менее 128 Кб — это экспертные оценки производителей реальной техники. Другими словами, практически всегда есть возможность организовать в верхней или расширенной памяти RAM-диск с системными файлами, который значительно уменьшит, а при достаточно большом объеме доступной памяти и вовсе может исключить обращения к флоппи-дисководам (можно посоветовать почитать на эту тему, к примеру, статью Вадима Елисеева о работе с электронным диском в среде iS-DOS из 4-го номера петербургского электронного журнала ZX-Format). Кроме того, современная работа под iS-DOS — это еще и работа с винчестером, а там и вовсе дисковод нужен только от случая к случаю, уже абсолютно независимо от объема оперативной памяти.

"Дискоубийцей" же iS-DOS может стать, пожалуй, в основном для 48-х машин без винчестера (согласитесь, довольно экзотическая конфигурация). Но если человек "вошел во вкус" iS-DOS, и при этом ему явно не хватает аппаратных ресурсов его компьютера, то не разумнее ли приобрести новый комп, с которым все подобные проблемы будут казаться лишь дурным сном? Впрочем, есть еще вариант — сэкономить на поддержке и развитии платформы, и остаться блаженствовать у своего разбитого корыта.

User> *Я пользуюсь IS-DOSом с 96-го года. Для того чтобы увеличить размер пользовательской памяти, поснимал всю резидентуру (gmen, там, mkdir и др.). Вместо gmen.res хотел вызывать gmen.com. Для этого забил в extkey строку*

```
9 Q:SHELL\gmen.com
```

А не фиг бы вам? Жмешь "9", а gmen.com не вызывается, хоть ты сколько жми! Только боксик "menu" в информационной строке издевательски так подмигивает. Может, просто версия IS-DOS жутко устарела , пора на новую перейти — уж там-то все путем будет...

ist> Согласно фирменному описанию iS-DOS, клавиша <9> жестко зарезервирована в системе для вызова именно резидентной задачи "gmen". Если есть желание или необходимость использовать "com"-овский аналог резидентной задачи — программу "gmen.com", то можно "повесить" ее на какую-нибудь другую, более или менее удобную клавишу или сочетание клавиш, вроде <Symbol_Shift>/<9>. Для этого в системном файле "extkey.txt" должна быть, к примеру, такая строка:

```
`41 [путь]gmen
```

где "`41" — это десятичный код выбранной клавишной комбинации, в данном случае — <Symbol_Shift>/<9>.

Вместо десятичного кода можно использовать шестнадцатеричный (да хоть двоичный!) код. В этих случаях префиксом кода, вместо апострофа должен быть "#" или "%" соответственно. В нашем примере:

```
#29 [путь]gmen
```

или

```
%00101001 [путь]gmen
```

Если вы не помните наизусть коды всех клавиш и их комбинаций, а лезть в справочник жутко неохота, то можно воспользоваться служебной утилиткой по имени "ktest.com" (в старых, "слотовских" поставках 1994...1996 годов) или "ktest_2.com" (в фирменной ИскраСофтовской поставке обычно находится в каталоге SERVICE\ANALECTA\), которая выводит на экран раскладку портов, символ и код нажатой клавиши или комбинации нажатых клавиш в шестнадцатеричной и десятичной системах.

Есть еще один способ, — использовать вместо кодов соответствующий данному сочетанию клавиш символ из кодовой таблицы ASCII:

```
) [путь]gmen
```

Этот вариант — самый экономный (на обозначение управляющей клавиши расходуется всего лишь один байт), но файл "extkey.txt" будет изнутри выглядеть несколько необычно, и эта необычность будет сильно зависеть от установленного в системе драйвера экрана.

Что же касается перехода на новую версию iS-DOS, то решение само по себе разумное, хотя и несколько запоздалое (самые "ходовые" сейчас версии — 1999 и 2000 годов). Только не стоит ждать, что в новых версиях были изменены системные стандарты — "шараханья" в этом смысле могли пойти только во вред. Главное отличие "новых" систем — гораздо меньшее количество "жучков", опти-

мизация по скорости работы и другой формат каталогов, сменившийся еще в канун 1997 г.

RAM-диск

User> *...У меня "Скорпион", поэтому, естественно, я "на всю катушку" использую RAM-DISK. Autoexec.bat я составил таким образом, чтобы на электронном диске после окончания загрузки уже находились самые нужные и часто употребляемые системой файлы. Обращения к системному диску почти не бывает. И я сделал так, чтобы по окончании загрузки системы на левой панели был каталог диска "A", а на правой — электронного диска "C". Удобно мне так, и я к этому привык. Но вот, после каких-то манипуляций мне понадобилось перезаписать файл "is_dos.sys" и переподключить систему. Неожиданно оказалось, что панели поменялись местами, и теперь слева панель диска "C", а справа — диска "A". Но мне это неудобно и непривычно. Как сделать, чтобы все стало на место?*

Keeper> "Элементарно, Ватсон!" :-). Надо просто вызвать командную строку, переписать еще раз файл "is_dos.sys" и переподключить систему последовательными командами "con is" и "boot". Это если вы работаете в Classic'e, а если в Chick'e, то после записи файла "is_dos.sys" надо просто запустить "ZS_boot.bat". Надеюсь, вы знаете, что записать файл "is_dos.sys" надо командой "sv is" из командной строки.

ist> Можно просто после загрузки перечитать нужные устройства (четыре нажатия клавиш — <Edit>, <A>, <Caps_Lock>, <C>). Можно использовать в "autoexec'e" утилиту "pan.com" с соответствующим ключом. Но есть способ лучше. К сведению пользователей, малоизвестный факт — при сохранении системы в файле "is_dos.sys" с помощью утилиты "sv.com" запоминается, кроме прочих мелочей, типа буфера "mon+.res" и списка "path.txt", еще и активная панель системы, т.е. та, на которой в момент вызова "sv.com" стоял курсор. Именно с текущей панели начинают работу вызываемые различными стандартными для системы iS-DOS способами прикладные программы и утилиты. Отсюда простой совет — вызывайте "sv.com" для сохранения системы в файле, когда курсор находится на правой панели. В этом случае после загрузки системы создаваться и открываться электронный RAM-диск будет именно на активной, т.е. в данном случае, на правой панели.

Ну а самое главное, что хотелось бы заметить по вопросу "где какая панель открыта", обычно "проходится" в начальных классах средней школы — от перемены мест слагаемых сумма не изменяется.

User> *У меня SCORPION и RAM-ДИСК в IS-DOS на 512 блоков, т.е. на 128 Кб. Только вот какой глюк там случается. Если RAM-ДИСК заполнен до конца или почти до конца, то файлы, которые в хвосте RAM-ДИСКА, оказываются попорченными. А у знакомого синклериста PROFI тоже на 256 Кб и тоже с RAM-ДИСКОМ на 512 блоков, и ничего такого никогда не было. Наверное, драйвер RAM-ДИСКА у SCORPIONA глючный?*

ist> За долгое время развития и совершенствования системы iS-DOS (т.е. с 1992 г.) все основные прикладные программы и драйвера были достаточно хорошо "вылизаны" фирмой IskraSoft для надежной и беспроблемной работы. Вряд ли стоит "грешить" на скорпионовский драйвер RAM-диска, если, конечно, не пользоваться каким-нибудь "левым" творением безвестного "гениального" кодера. Возможно, здесь "собака зарыта" в программно-аппаратных особенностях компьютеров серии "Scorpion".

Отличительная "фишка" этих машин — встроенный ("прошитый" в ПЗУ) теневой сервис-монитор. Для собственных нужд он "оккупирует" восьмую и девятую "банки" дополнительной памяти. Из-за этого целый "букет" сюрпризов в iS-DOS'e — искажение информации в буфере копирования, неполная системная совместимость, "глюки" в Chick'e, не работают некоторые исдосные программы и утилиты. Версия iS-DOS Millennium появилась на свет в том числе и по той причине, что IskraSoft была вынуждена "бороться" именно со скорпионовским "теневиком", и в ней эти недостатки "Скорпионов" частично обойдены. Вполне вероятно, что в описанном случае проблема кроется именно в "теневике".

Чтобы "не мудрствовать лукаво", можно посоветовать "шаманский" способ, совершенно ненаучный и далеко не безупречный с точки зрения системного подхода, но вполне действенный и провереный на двух машинах (драйвер RAM-диска "edsco-b.blk" от 02.08.96, длиной 219 байтов) — не делать на "Скорпионе" RAM-диск максимального размера, а укоротить его на одну "банку", то есть на 64 блока. Другими словами, RAM-диск должен быть размером 448 блоков, и проблемы с искажением информации в скорпионовском "хвосте" должны исчезнуть.

Винчестер

User> *Вы только не подумайте, что я что-то имею против винта по немовской схеме, но вот в скорповском Смуке можно эмулировать работу с дискетами TR-DOS прямо на винте, и это, говорят, очень удобно. В общем-то, TR-DOS мне постольку поскольку, но если бы можно было так же сделать и на немовском винте, то хуже не было бы, верно? Хотя бы с IS-DOS'ными дискетами.*

ist> Против винчестера обычно что-то имеют те, кому он просто не нужен, для кого компьютер ближе к игрушке, чем к рабочему инструменту. Те же, кто прочувствовал наяву, а не на словах разницу между TR-DOS и iS-DOS для деловых применений, по достоинству оценивают прирост производительности на порядок, который дает винчестер, работающий под управлением iS-DOS с контроллером IDE-DRIVE разработки фирмы (c)Nemo.

Что же касается сравнения с винчестером, работающим под "чутким руководством" контроллера SMUC разработки фирмы (c)Scorpion, то по этому поводу следует заметить, что "можно эмулировать работу с дискетами TR-DOS прямо на винте" — это сказано не совсем точно. Вернее будет сказать, что это единственно возможный для SMUC'а режим работы винчестера совместно с системой TR-DOS. В контроллере IDE-DRIVE реализована совершенно иная концепция взаимодействия с TR-DOS, наиболее полноценно воплощенная "в железе" на компьютерах KAY последней модели — KAY-1024 turbo (v.1.5). Здесь диски формата TR-DOS хранятся на винчестере в виде стандартных для системы iS-DOS файлов, содержащих побайтовую копию обычных

TR-DOS'ных дисков (расширение имени файлов — ".trd"). В этом случае стыковка с TR-DOS достигается предельно просто — через электронный квазидисковод, имитирующий дисковод реальный, но примерно с десятикратным ускорением. Для ранних моделей KAY'ев и других Sinclair-совместимых клонов с контроллером IDE-DRIVE стыковка iS-DOS/TR-DOS достигается чуть сложнее. Там используется другой подход, дающий возможность просто хранить на винчестере кучу виртуальных TR-DOS'ных дискет и обходиться при этом всего десятком реальных, перенося на них по мере необходимости нужные "trd"-болванки. В то же время, использование контроллера SMUC с другими Sinclair-совместимыми машинами, помимо "Скорпионов", невозможно. Подробнее об этом и о многих других вопросах, связанных с винчестером под iS-DOS, можно узнать в информационном сборнике "IS-files" и в газете "Абзац".

Что касается "эмуляции" iS-DOS'ных дискет на iS-DOS'ном же "винте", то здесь тоже все предельно просто. Сначала создайте на своем винчестере устройство iS-DOS (или несколько, если в этом есть необходимость, и позволяет объем накопителя) с размером, равным или чуть большим, чем объем обычной дискеты iS-DOS (с запасом на часто встречающиеся "сверхнормативные" дорожки) — 3320...3400 блоков. Затем скопируйте на это устройство нужную iS-DOS'ную дискету с помощью утилиты "abc.com" или ("abc+.com"). И теперь можете делать с этой квазидискетой все, что угодно, теперь она "сэмулирована" на "винте", и все операции с ней будут происходить намного шустрее. Копирование же такой квазидискеты обратно на реальную дискету занимает по времени менее минуты.

Архивы

User> *Ситуация такова. Создаю на своей 128-й машине RAM-диск в 320 блоков (80 Кб), все как положено. Затем копирую туда zip-ный архив с текстами и разворачиваю его на этом самом RAM-диске, чтобы все было пошустрее. Естественно, рассчитываю так, чтобы все там свободно помещалось. И вот тут начинается самое интересное: если архив маленький, то все о'кей, если уже посолиднее, то либо комп зависает, либо isunzjr выкидывает сообщение об ошибке с номером — каким, не помню. Пробовал разворачивать зипы из нескольких файлов и из одного, особой разницы при этом не заметил. Пробовал читать зип с реального диска, а раскрытые файлы складывать на RAM-диск — иногда работает уже лучше, без сбоев, а иногда и нет. Сами зипы знаю, что нормальные — на обычной дискете разархивируются нормально. Самое противное, что у приятеля на Скорпионе такой ерунды не наблюдается. Что за чертовщина?*

ist> Никаких "барабашек" здесь нет. Распаковщик "isunzjr.com" (автор — Михаил Кондратьев, г.Санкт-Петербург, 1997 г.) — это программа довольно сложная и требовательная к аппаратным ресурсам. Для своей работы она использует некоторые банки дополнительной памяти в 128-килобайтных компьютерах. Как раз те, которые входят в состав 80-килобайтного RAM-диска. Если RAM-диск пуст или заполнен не весь, то существует вероятность, что такой фокус пройдет безнаказанно. Но, может быть, и нет — этакая разновидность русской рулетки. Вовсе самоубийственный случай — если RAM-диск битком набит системными файлами для ускорения работы в iS-DOS. Тогда "isunzjr.com" часть из них обязательно угробит. Так же, кстати, поступают и некоторые другие программы. Например, хранитель ресурса экрана "starnew" или графический редактор "iS-DOS Art Studio".

Что же касается компьютеров "Scorpion" и других машин с объемом оперативной памяти более 128 Кб, то правилами хорошего тона среди опытных пользователей iS-DOS подразумевается, что RAM-диск для них организуется без использования "банок" памяти в пределах первых 128 Кб. Эти самые банки обычно отдаются "на откуп" копировщикам и таким вот программам с большим аппетитом на аппаратные ресурсы. Поэтому подобные программно-аппаратные конфликты на таких синклерах, как правило, не возникают.

User> *Еще хотел узнать, как лучше делать зипные архивы для хранения, например, текстов на Спектруме. Я принес файлы в печатное агентство, а там мужик шибко умный попался, спрашивает, каким методом зиповать? Я сказал — стандартным, и получил что хотел. Вроде, все работает. Но, может быть, какой-то другой метод лучше сжимает или быстрее раззиповывает?*

ist> В 95 случаях из 100 писюшные операторы на рабочих местах предпочитают пользоваться для архивирования двухрежимным архиватором "WinRAR", поскольку он входит в состав практически всех дистрибутивов Windows. В подрежиме создания ZIP-архивов там обычно пользуются методом, включенным по умолчанию, а именно — "normal". Кроме того, там есть еще "fastest" ("быстрейший"), "fast" ("быстрый"), "good" ("хороший"), "best" ("лучший") и "store" ("сохраняющий"). За точность перевода не ручаемся, но архивы, созданные с помощью любого из этих методов, прекрасно "понимаются" исдосным кондратьевским распаковщиком "isunzjr.com". Но "store" использовать смысла нет, поскольку он ничегошеньки не архивирует, а только создает архив со стандартным zip'ным заголовком, в котором содержится неизмененный исходный файл — т.е. результирующий файл получается даже длиннее исходного. Названия остальных методов подразумевают некоторые различия по скорости работы и по степени сжатия. Но на практике эти самые различия между ними слишком незначительны, для того чтобы можно было уверенно выделить какой-либо один-единственный метод и предпочесть его всем остальным.

Для сравнения методов по эффективности сжатия и по времени, затрачиваемому на распаковку полученных архивов, был проведен короткий экспресс-тест. Архивировался русскоязычный текстовый файл средней рыхлости, в альтернативной кодировке, содержащий статью, перепечатанную из периодического издания. Объем исходного файла составлял 12869 байтов. Последующая распаковка производилась на винчестере, работающем через драйвер "ide+5.blk" в составе компьютера KAY-1024 turbo, с выключенным турборежимом. Числовые данные, приведенные в графе "время" — это результаты арифметического усреднения, полученные по итогам трех замеров.

Метод	Размер полученного архива, байтов	Время распаковки, с
fastest	7143	7,64
fast	6991	7,56
normal	6735	7,37
good	6734	7,45
best	6734	7,46
store	12997	4,33

Из таблицы можно видеть, что режимы "store", "fastest" и "fast" применять для архивного хранения файлов на Sinclair нецелесообразно. А отличия в эффективности упаковки методов "good" и "best", по сравнению со стандартным "normal", установленным обычно по умолчанию, настолько невелики, что ими можно элементарно пренебречь.

Теоретически, эти "лучшие" методы могут дать некоторую экономию на очень больших файлах, типа "trd"-болванок. А в нашей практике, на текстовых файлах длиной около 150 Кб, экономия составляла всего лишь несколько десятков байтов. Поэтому подумайте хорошенько, прежде чем начинать морочить голову бедным писишникам, заставляя их использовать свое серое вещество по назначению. По нашему мнению, все-таки проще, быстрее и надежнее сделать все "на автопилоте". В конце концов, далеко не во всех печатных агентствах попадаются "шибко умные мужики"...

На самом же деле проблема здесь заключается совсем в другом. Нельзя считать нормальным само положение, при котором архивы для использования в iS-DOS на Sinclair могут "заготавливаться" только лишь с привлечением машинных ресурсов другой компьютерной платформы. Более естественным было бы использовать соответствующую прикладную утилиту, работающую в среде iS-DOS. Но сам тот факт, что за шесть лет, прошедшие со времени выхода в пользовательский свет утилиты "isunzjr.com", так и не было создано ZIP-архиватора под iS-DOS, свидетельствует о том, что пользователи и программисты не замечают криминала в допущении платформы IBM PC в технологические цепочки информационных технологий, реализуемых на Sinclair. В то же время, каких-то особых технических проблем для реализации подобных программ на Sinclair здесь нет.

Просмотр текстов

User> *... ис-досный вьювер выводит на экран целые групы строк, повтореные по два раза. Сначала думал, что программа рухнутая. Скинул с дистрибутива, работает так же. Пробовал разные версии вьювера разных годов выпуска — без разницы. Сам файл тоже проверял — загружал в редактор, а там все выглядит нормально. Может, в самой системе что-то рухнуло?*

ist> Такой эффект может возникнуть, как правило, в iS-DOS Classic, если текстовому просмотрщику "tv.com" уже тесновато в памяти, но еще не настолько, чтобы ругнуться ошибкой "Error 130".

Лечится эта ситуация обычно снятием какого-нибудь наименее нужного резидента или драйвера (или нескольких, если потребуется).

(Окончание следует)

Кое-что о Load и Save

Я.ОЧАКОВСКИЙ,
р-ц Петропавловское, Алтайского кр.

Наверняка любой пользователь Спектрума, даже имеющий все блага "железа", хотя бы из интереса, работал с кассетным накопителем, проще говоря — с обыкновенной магнитофонной компакт-кассетой. Но, скорее всего, далеко не каждый знает о всех возможностях таких скромных, на первый взгляд, операторов Бейсика как Load и Save.

Так что же еще можно с ними творить?

**Load "a$"** — загружает программу в память компьютера. Имя программы задается значением a$. Если имя программы не указывается в кавычках, то загружается первая встретившаяся программа.

Load "a$" CODE [X], [Y] — загружает в память компьютера кодовый блок (последовательность байтов). Имя файла, в котором был сохранен блок на магнитной ленте, задается символьным значением a$. За ключевым словом CODE указывается адрес размещения блока кодов в памяти компьютера — X и количество загружаемых файлов — Y.

Load "a$" Data b [$] () — загружает с магнитной ленты в память компьютера массив. Имя файла, содержащего массив, задается символьным значением a$. Следом за Data указывается имя массива — b (буква или буква со знаком $) с пустыми скобками.

Load "a$" Screen$ — загружает с магнитной ленты блок кодов непосредственно в экранную область памяти, и на экране воспроизводится ранее записанная картинка. Имя файла, в котором она сохранена на ленте — a$.

Save "a$" [Line x] — записывает Бейсик-программу на ленту. Программе присваивается имя, задаваемое символьным значение a$, которое должно содержать не более десяти символов и не может быть "пустым" (это условие должно соблюдаться в любых случаях использования оператора Save, в т.ч. и во всех нижеперечисленных). Также может быть использовано ключевое слово Line, указывающее, что программа, после ее загрузки оператором Load, будет сразу запущена со строки с номером X.

Save "a$" CODE X, Y — сохраняет на магнитной ленте блок кодов (последовательность байтов). Имя файла, в который записывается блок, задается символьным значением a$. За ключевым словом CODE указывается адрес размещения блока кодов в памяти компьютера — X и количество загружаемых байтов — Y.

**Save "a$"** — сохраняет на магнитной ленте массив. Имя файла, в котором сохраняется массив, задается символьным значением a$. Следом за DATA указывается имя массива b (буква или буква со знаком $) с пустыми скобками.

Save "a$" Screen$ — записывает на магнитную ленту экранное изображение. Имя файла, в котором сохраняется экранное изображение, задается символьным значением a$.

Will Rock

А.ВЕНДИЛОВСКИЙ,
г.Минск.

... Древние времена, когда боги восседали на Олимпе, когда слагались мифы и легенды... Но ничто не вечно, и армии честолюбивых римлян ворвались в процветающую Грецию. Скоро уже их марширующие с золотыми орлами легионы подступали к Афинам. А боги бездействовали, предпочитая наблюдать за страданиями смертных.

— Вы не можете так поступить! Ведь они — ваш народ, который верит в вас! Громовержец, они приносили вам жертвы и возносили молитвы, а теперь ждут от всех вас помощи! А что собираетесь сделать вы? Закрыть Олимп от доступа к нему людей? Что за секреты вы хотите сохранить? А если вера людей уйдет от вас, и вы станете всего лишь тенями, мертвым пантеоном глупых божков, чьи призраки будут бессильно взирать с небес!? Вы...

— Заткнись, Прометей! Не испытывай мое терпение! — Зевс-Громовержец поднялся со своего трона, его глаза пылали. Ты порядком мне надоел со своей авантюрой с огнем, или тебе было мало того наказания, и тебя вновь стоит приковать к скале, а печень скормить орлу? Они всего лишь людишки, а мы БОГИ! Вера не исчезнет, пусть под другими именами, но мы вернемся, и покорители этих земель склонятся перед нашим могуществом! Одни смертные ничем не отличаются от других, а мы будем карать и миловать! И что значит время для бессмертных?

— Ты предаешь всех, кто в тебя верил! Ты... — в это время волна огня ударила в Прометея. В последний миг падения с Олимпа он увидел печальные глаза Афины, и отчаянный план родился в его голове. Земля содрогнулась, молнии распороли небо, очерчивая гигантские звезды и пылающие колеса. Гул ударил в землю, как тяжелый молот, и море отхлынуло от берегов. С трудом поднявшись на ноги, Прометей увидел, как бледнеет жилище богов. Здесь, возле входа в Храм Олимпа, он чувствовал гигантские силы, формирующие реальность. С отчаянностью и безрассудством, которые приходят обычно с безнадежностью, он призвал свои силы, всю мощь расы Титанов, всю веру людей в него, и ударил ею в сердце нарождающегося вихря. Под могучим напором двух сил равновесие реальности качнулось, отбрасывая обиталище небожителей еще дальше в небытие. Их силы таяли в этом мире, как таял сейчас перед глазами Прометея сияющий Храм. Чувствуя, как уходит из него сила, превращаясь в безмолвный камень, он успел оставить на воротах предупреждение. Земля содрогнулась еще раз, и лавина засыпала и Врата, и статую Прометея...

... Века проносились над Землей, вздымались и исчезали города, новые легионы шагали по старым дорогам, только покой Врат никто не нарушал...

* * *

Мой босс-профессор в очередной раз засобирался в дорогу. По слухам, на Олимпе откопали какой-то неизвестный Храм. Естественно, все археологи переполошились — как же, ведь это величайшая находка со времен обнаружения Трои! Нельзя сказать, что археология была моим коньком, и я любил бежать на край света, чтобы откопать черепок трехтысячелетней давности. Но если бы вы видели дочь Профа, то побежали бы не только на край света, но и дальше. От ее взгляда я терял волю, и мной можно было командовать как послушным щенком. Такие мелочи как выходки террористической группы "Новые Олимпийцы", естественно, не могли остановить Профа. Представьте, эти отморозки всерьез верили, что можно вернуть древних богов и старые денечки. Свою решимость они подкрепляли то убийствами, то человеческими жертвоприношениями. Как можно в такое верить и ради абсолютно безумных идей идти на такое — просто не имею ни малейшего представления. На самой горе нас ждали очень странные горе-археологи, которые, если не слонялись поблизости от нас, то глушили пиво в соседнем баре. Но нам было не до них — вход в храм был буквально похоронен под толщей земли, которая за тысячелетие превратилась практически в скалу. Неподалеку была найдена статуя, прекрасно сохранившаяся, правда, опознать ее никому не удалось. В тот вечер Проф расчистил надпись на входе, и мы занялись ее переводом. Это было предупреждение, предупреждение никогда не открывать эти Врата. Надпись плясала перед глазами, и за ней чувствовалась власть — я увидел и поверил, несмотря на свой скептицизм и достаточно рациональный подход к жизни. Нам стало страшно, впервые за много лет работы захотелось просто снова завалить эти Врата землей и забыть об их существовании. В этот момент женский крик раздался рядом с нашей палаткой, а в следующую секунду мы были окружены толпой вооруженных "археологов". Один из бандитов держал нож возле горла дочери профессора.

— Давайте, док, без глупостей! — мерзко осклабился один из террористов, вы открываете Врата, а иначе, мы нарисуем вашей дочурке еще одну улыбку под подбородком.

Мы прошли к Вратам. Профессор медленно прочитал надпись на створке, и с небес ударил гром, сквозь щели досок стал пробиваться яркий свет, и створки медленно стали отходить в разные стороны.

— Девчонку принесем в жертву Зевсу, а этих прикончите здесь!

В этот момент Профессор выхватил пистолет и бросился к дочери. Раздалось несколько выстрелов, и профессор упал замертво. Пистолет по чистой случайности подкатился к моим ногам, я не стал ждать, пока меня прикончат, и открыл огонь первым, спрятавшись за статуей. Я видел, как девушку уволокли за ворота, но ничем не мог помочь. Чья

пуля попала в статую, не могу сказать, но ярчайшая вспышка осветила все вокруг. Уже не было статуи, раскопок, только ярчайший свет и голос.

— Значит, врата открыли... проклятье! За это время боги явно сильно разозлились за свое заточение, милости у них не дождешься, как снега летом. Дело просто дрянь.

— Кто, кто ты?

— Прометей, друг мой. Когда древние боги вернулись в этот мир, моя сила тоже вернулась, и сознание пробудилось. Я вижу твое сознание, и в нем вижу мир, который так сильно изменился за время моего сна. Остается надеяться, что ты сможешь остановить древних богов, прежде чем они уничтожат твой мир. И нужно спасти девушку... Я дам тебе силу Титанов, нашу древнюю мощь, а дальше все в твоих руках!

— Но они боги! Как я могу бороться с богами!

— Запомни, боги играют с судьбами смертных, но только смертные решают судьбы богов!

И этот яркий свет стал впитываться каждой частичкой моего тела...

Через несколько часов

Никто бы уже не узнал в тот момент в этой машине смерти молодого и слегка неуверенного в себе археолога. Он метался по древней арене как молния, поливая из пулемета бросающиеся на него толпы минотавров и скелетов, ловко уворачиваясь от летящих в него копий и топоров. Свинцовый град раздирал плоть врагов, выбивая фонтанчики крови, и только грохот очередных упавших доспехов или предсмертный крик говорили, что еще одним его врагом стало меньше. Десятки орлов догорали на арене в собственном пламени, настигнутые свинцовой немезидой в воздухе. Ловко отбросив пулемет, герой выхватил саперную лопатку, сшиб последнего противника с ног, а потом, не теряя ни секунды, той же лопаткой перебил ему глотку. Он был весь в крови тех, кто стал ему поперек дороги. Он стал воплощением справедливости. Не менее ловко, словно проделывал это тысячу раз, он закинул лопатку за спину, посмотрел на Олимп и тихо произнес: "Ну приготовьтесь, божественные заср...ы, я иду!"

* * *

Дело Серьезного Сэма живет и процветает! Иногда, когда очередной стелс-экшен уже порядком надоест, а после общения с начальством адреналин немедленно требует выхода энергии, подобная игра — как манна небесная. Шутер от первого лица, который обычно относят к спинномозговым, с невероятным количеством врагов и оружия (моря крови, груды поверженной плоти и всего, что обычно входит в подобный джентльменский набор). Такой экшен сделать качественным очень сложно — он должен быть достаточно сложным, но не настолько, чтобы от бессилия падали руки, одновременно оставаясь красивым и захватывающим. Забегая вперед, могу сказать — разработчикам это удалось.

Движок игры. Я бы сказал, что для такой игры он стремится к идеальному — особенно впечатляют фрески на стенах, амфоры и прочие предметы обстановки, расписанные картинами из мифологии древней Греции. Противник тоже не выглядит особенно ужасно, модели персонажей прорисованы достаточно качественно. Не ждите, что в подобной игре на каждую модель будут приходиться десятки тысяч полигонов — ведь противников на уровне всегда очень много, и нужно, чтобы не возникали такие ненавистные всем геймерами "тормоза", делая игру эксклюзивом для избранных, имеющих дорогие компьютеры. Конечно, можно найти недочеты, но такие ляпы как выпадающие пикселы (иногда вместе с игроком, иногда прямо в Винду) тут отсутствуют. Владельцы Радеонов будут особенно рады, движок игры оптимизирован под продукцию ATI. Правда, остальным не стоит сильно переживать по этому поводу — игра прекрасно бегает даже под второй GeForce. Можно спокойно включать все дополнительные графические навороты. Единственно, игра очень сильно реагирует на объем оперативной памяти. Если у вас 512 Мб ОЗУ, то вам придется даже думать, как "притормозить" игрушку. Дизайн уровней достаточно хорош — есть где развернуться, побегать от монстров, проявляя чудеса прыти. Здесь не будет новомодных "разбей все" или "дверь можно открыть разными путями". Это всего лишь место, где вам придется приложить все усилия, чтобы выжить. Слава Богу, отсутствуют и заумно-мудреные задачи из серии "мы сами понятия не имеем, как открыть эту дверь". Парочку выключателей всегда легко найти, хотя бы ориентируясь на плотность монстрятника :).

Об AI в таких играх говорят мало — монстры тут берут не умом, а количеством, просто бросаясь на тебя. Этого обычно бывает достаточно, ведь главная фишка таких игр — это адреналиновый драйв, когда пробиваешься через толпы монстров, делая ставку на реакцию, а не на хитроумные комбинации.

О геймплее можно выразиться кратко — игра затягивает, в нее хочется играть. Даже переигрывая по нескольку раз один и тот же эпизод, не теряешь ощущение азарта, что достаточно редко встречается в современных играх.

Вывод — очень хорошая игра, которую рекомендую всем любителям экшена.

Диск предоставлен интернет-магазином "MediaCraft" www.MediaCraft.shop.by

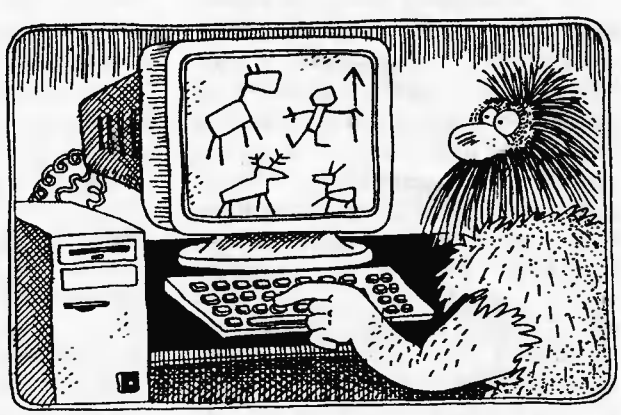


Рисунок О.Попова

КУПЛЮ, ПРОДАМ, ОБМЕНЯЮ

*Для публикации бесплатных объявлений **некоммерческого характера** о покупке и продаже компьютерных комплектующих и программ, их текст можно присылать в письме по адресам:*

119454, г.Москва, а/я 37 или
220095, г.Минск-95, а/я 199,

*передавать по тел. в Минске **(017) 249-41-47**, либо через E-mail:*

rl@radiopage.by
rm@radio-mir.com
*WWW: **http://radio-mir.com***

Продам на кассетах по низкой цене коллекцию программ для ZX-Spectrum.
692446, Приморский край, г.Дальнегорск-6, а/я 34.
Email: alecs-dms@mail.ru

Куплю: схемы телевизора "Альфа 51ТЦ-4301Д", радиоприемника "Меридиан-210"; книгу "Программирование на Бейсике для персональных ЭВМ радиолюбителя" изд. "Патриот".
169312, г.Ухта, ул.Строителей, 19 — 6,
Голивец В.Г.

Продам книги по ZX-Spectrum:
- А.Ларченко, Н.Родионов. ZX Spectrum для пользователей и программистов;
- Н.Родионов. Адаптация программ к системе TR-DOS;
- ZX-Ревю, 1991 (выпуски 1...12);
- Справочник "Системные программы для ZX Spectrum 128 K";
- Справочник по системным программам для ZX Spectrum;
- Описание дисковой системы TR-DOS для компьютеров ZX Spectrum.
А. Горячкин.
E-mail: anatoliygor@nm.ru

Куплю компьютерные комплектующие (б/у); аксессуары для компьютеров (мышки, колонки и т.п.) в нерабочем состоянии.
640023, г.Курган-23, а/я 2055, Анатолий.
E-mail: anatolii25@rambler.ru

Предлагаю программное обеспечение (игры, системные программы, электроника, текстовые файлы утилит на Ассемблере Z80) для компьютеров ZX-Spectrum 128/48 на кассетах или аудио компакт-дисках.
150533, Россия, Ярославская обл.,
Ярославский р-н, с.Курба,
ул.Юбилейная, 11 — 15, Бутов А.Л.
Тел. (0852) 66-31-58.

Ищу схемы, описания сигналов IBM-совместимых ПК, ISA, PCI, AGP, Keyboard, Mouse-интерфейсы и др., документы по проектированию устройств под ПК.
E-mail: tonder@trktvs.ru

Продаю компьютер 486 по частям: материнская плата, процессор Intel 486SX25, память SIMM 8x1 Мб 30pin, мультикарта HDD/LPT/2xCOM, видеокарте SVGA 512 Кб. Вышлю почтой.
354024, г.Сочи, ул.Ручей де Симона, 119,
Пареев М.В.

Ищу компилятор Бейсика версии MC2b.v4. Приложить конверт с обратным адресом.
632383, Новосибирская обл.,
г.Куйбышев, 1 — 20 — 45, Третьяков А.А.

Ищу документацию (инструкции) на матричные принтеры "EPSON" модели FX-1050 и LX-100. Куплю, обменяю на техническую литературу, схемы, детали.
141065, Московская обл., г.Королев, Болшево-5,
а/я 1, Брускин В.Я.
E-mail: bruskin@inbox.ru

Куплю процессор Z80 SIO или его аналог U856.
452403, Башкортостан, г.Давлеканово,
ул.50 лет Башкирии, 86, Дроздов С.

Продаю: С1-55, С1-112, Г3-36А, Ч3-32, М1109, М2051, Щ4313, Ц4353, Р40114, КТ817Г/КТ819Г (б/у), программы для "ZX-Spectrum" и др.
Куплю документацию на Ч3-32, Г3-36А.
Возможен обмен на тюнер ("Пионер", "Техникс" и др).
Тел. (095) 944-53-13, с 19.00 до 21.00, Михаил.

Продаю:
- компьютер "Scorpion 256" (расширенная клавиатура для "Scorpion"; контроллер для подключения IBM-клавиатуры и мыши; 2 дисковода "Robotron");
- монитор "Электроника 32ВТЦ-202;
- два принтера (без головок) "Электроника СМ 6312";
- 160 дискет 5,25S (системные программы, электронные журналы и газеты);
- черно-белый монитор "Электроника МС6105.01";
- два дисковода TEAC на запчасти;
- дисковод "Электроника 5305";
- дисковод "Epson SD-600".
165300, Архангельская обл., г.Котлас,
ул.Мартемьяновская, 44 — 7, Ятченко Н.Ф.

Уважаемые читатели!

Подписаться на наши журналы (индексы: **48996, 72370** (годовая) — "Радиомир", **48924, 71545** (годовая) — "Радиомир. КВ и УКВ", **48925, 45995** (годовая) — "Радиомир. Ваш компьютер") в 2004 г. можно по каталогу Агентства "Роспечать", **стр.392** или по каталогу РО "Белпочта" "Газеты и журналы Республики Беларусь", **стр.171** (раздел "Издания РФ, распространяемые по прямым договорам").

Кроме того, **предприятия** регионов России, а также ближнего и дальнего зарубежья могут оформить подписку через ООО "Корпоративная Почта" по телефонам: (095) 953-92-62, 953-92-02, 953-93-20.

Те, у кого возникли проблемы с подпиской, могут получить их из редакции. Там же можно заказать имеющиеся в наличии отдельные номера журналов за предыдущие годы.

Год	Можно заказать следующие номера журналов			Расценки на 1 экз. (с учетом пересылки)		
	Радиолюбитель	Радиолюбитель. Ваш компьютер	Радиолюбитель. КВ и УКВ	По России и СНГ (рос. руб.)	По Беларуси (бел. руб.)	По Украине (гривны)
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	3, 5, 6	20	700	5
2000	2 — 6, 6 — 12	2 — 6, 10 — 12	4, 6, 7, 9, 11, 12	20	800	5
2001	2, 4 — 6	1, 2, 4 — 6	2 — 6	25	900	5
	Радиомир	Радиомир. Ваш компьютер	Радиомир. КВ и УКВ	По России и СНГ (рос. руб)	По Беларуси (бел. руб.)	По Украине (гривны)
2001	7 — 12	7, 9 — 12	7, 8, 10 — 12	30	1000	6
2002	1 — 12	1 — 12	1 — 3, 5 — 12	35	1500	7
2003	1 — 12	1 — 12	1, 5 — 12	40	1800	8
2004	1 — 12	1 — 12	1 — 12	45	2100	9

Наши платежные реквизиты:
- для жителей России и стран СНГ (кроме Беларуси) —
получатель: ООО "Радиомир Пресс", ИНН 7729404741, КПП 772901001,
р/с 40702810900010000084 в ООО КБ "Агропромкредит", ф-л Центральный, г.Москва,
корр. счет 30101810500000000109, БИК 044525109
Адрес банка: 125315, г.Москва, Ленинградский пр-т, дом 76, корп. 4;
- для жителей Беларуси —
получатель: УП "РЛД", УНН 190218688
р/с 3012000004882 в ф-ле №524 АСБ "Беларусбанк" в г.Минске код 121.
Адрес банка: 220028, г.Минск, ул.Физкультурная, 31.

***При оплате через Сбербанк** в графе "назначение платежа" необходимо написать свой почтовый индекс, полный адрес, фамилию, имя и отчество полностью и точно перечислить, какие конкретно номера какого из журналов Вы заказываете.*

***При оплате почтовым переводом** все сведения о заказе необходимо указать в графе "Для письма".*

Для ускорения процесса получения журналов заказ можно продублировать по *E-mail: **rm-sales@radio-mir.com**. Вся информация — там же или по тел. в г.Минске **(017) 221-01-10, (017) 505-13-65**.*

Приобретение отдельных номеров журналов

В РОССИИ:

В **ЗАО "Интерпресс — Экспресс"**: тел. в Москве (095) 208-85-50, 208-67-55, e-mail: inter@aif.ru

В магазинах радиодеталей **"ЧИП и ДИП"**:
- г.Москва, ул.Гиляровского, д.39 (ст. метро "Проспект Мира" — радиальная);
- г.Москва, ул.Ивана Франко, д.40, к.1, стр. 2 (платф.Рабочий поселок, 15 мин. от Белорусского вокзала);
- г.Москва, ул.Беговая, д.2;
- г.Ярославль, ул.Нахимсона, 12, тел. (0852) 27-57-15.

На радиорынках в Москве: Митинском (места R4, S8, K52, E50, G56), Царицынском (место 121).

В **ООО "ТРЭНТЭКС"**: 111024, г.Москва, 4-я Кабельная, 2А (ст. метро "Авиамоторная"), тел. (095) 258-91-94, 258-91-95. E-mail: abook@mail.ru, abook@inbox.ru
На радиорынках: "Царицыно" (место 13/А), "Митино" (ряд 1, контейнер 17 и место Т-8); в ТК "Савеловский" (ст. метро "Савеловская", места А4, А5); на книжной ярмарке на "Тульской" (ст. метро "Тульская", место 515-19).

На УКРАИНЕ:

В Киеве в фирме **"Торм"**, тел. (044) 227-72-73.

В БЕЛАРУСИ:

В Минске в магазинах **"Книга XXI век"**, пр.Ф.Скорины, д.92, тел. (017) 264-27-97 (ст.метро "Московская") и **"Глобус"**, ул.Володарского, д.16, тел. (017) 227-30-67 (ст.метро "Площадь Незаависимости").

ВАШ КОМПЬЮТЕР — 2003

ВОКРУГ ПК

ПРОГРАММЫ И АЛГОРИТМЫ

ДИАЛОГ ПРОГРАММИСТОВ

РЕЦЕПТЫ

МИР 8 БИТ

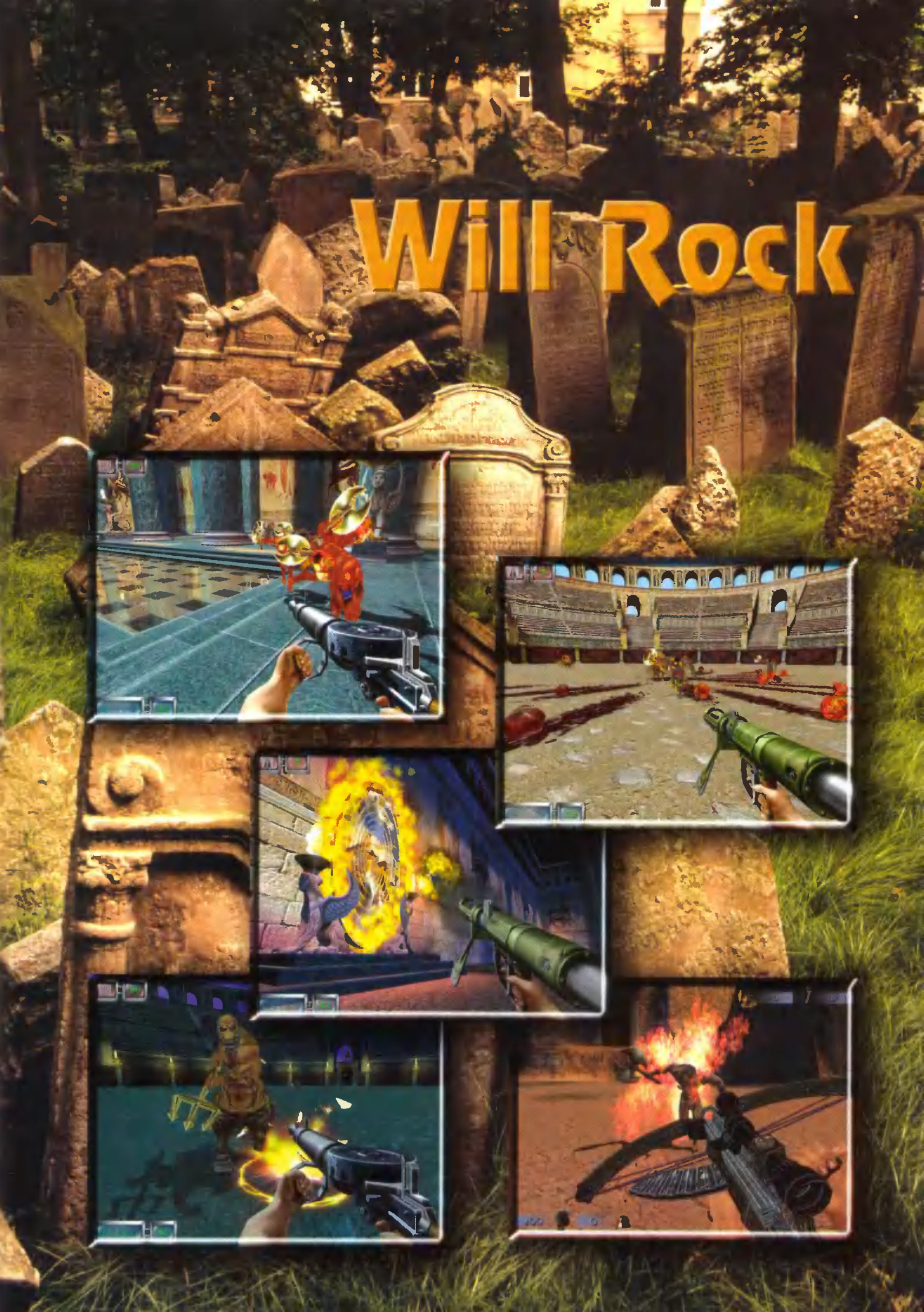
ИГРОТЕКА

ДОСКА ОБЪЯВЛЕНИЙ

ВАШ КОМПЬЮТЕР — 2002

До встречи в 2004 году!

Рисунок О.Попова


Will Rock