http://radio-mir.com

индексы: 48925, 45995

радиомир

Ваш компьютер

11 • 2003

PRO100 SOFT

6100 ПЛАНОВ ДОМОВ

HOME PLAN FINDER 3.0

Вы за несколько минут сможете подобрать дом Вашей мечты.
По полученным вариантам оболочка выдает внешний вид, схемы каждого этажа с пояснениями, размерами и др.

2003

АНЕКДОТЫ, ТЕСТЫ, КОМПЬЮТЕРНЫЕ ПРИКОЛЫ 2

ПОДБОРКА ЛУЧШИХ АНЕКДОТОВ И ПРИКОЛ

Темы для рабочего стола
Курсеры
Иконки

АНГЛИЙСКИЙ ЯЗЫК

УСКОРЕННЫЙ КУРС

ИЛЛЮСТРАТИВНАЯ ЖИВОПИСЬ XX ВЕКА

КЛАССИКИ ФЭНТЭЗИ

fantasy classics

1.

1. классики фэнтэзи
2. живые легенды
3. темная реальность

Photoshop 7.0

+ Adobe Photoshop 7.0 с русскими шрифтами

СТУДЕНТЧЕСКАЯ БОМБА

ВЫПУСК 8

MP3 192K 44 KHZ STEREO

ТОЛЬКО ЛУЧШИЕ АЛЬБОМЫ

РОМАНТИЧЕСКАЯ ГИТАРА

VOL 1

Высокое качество

7 АЛЬБОМОВ
КАРТИНКИ

профессиональному разработчику под Windows

СОФТ

От идеи до дистрибутива

Разработка программного обеспечения
Разработка справочных систем
Системы создания установочных пакетов
Утилиты для программистов
Полезная информация

100%

ALEX SOFT

ВСЕГДА В ЦЕНЕ

ПУХЛЫЙ СОФТ

алмазный фонд

Электронные словари и переводчики 2003

VIONS

- PROMT XT Office Giant v.6.0.0.36
- ABBYY FineReader 6.0 Professional
- ABBYY LINGVO 8.0 Англо-русская версия
- ABBYY LINGVO 8.6 Многоязычная версия
- 19 специализированных словарей для многоязычной и англо-русской версий ABBYY LINGVO 8.9
- Govorilka 1.40b

100%

VIONS


open.by
Продвижение » сайтов
Digital Shop.By
Canon
Nikon
Olympus
Sony
Minolta
Casio
FujiFilm
Panasonic
Canon
Sony
Отфильтровано!
Интернет - хирургия

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,
Мария 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.Коштоянца, 6 — 233, тел. (095) 105-99-89

Подписано к печати 25.09.2003 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная.

Отпечатано в типографии
ЗАО "Красногорская типография".
Заказ №2862.

радиомир
Ваш компьютер

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

№11/ноябрь/2003

ЧИТАЙТЕ В НОМЕРЕ

Дорогие друзья!

Страницы журнала "Радиомир. Ваш компьютер" открыты для всех, кто хочет и умеет сделать общение с компьютером лучше, интереснее и полезнее. Поделитесь с коллегами своими идеями, схемными решениями, программами и советами.

Присылайте нам и свои вопросы — возможно, кто-то уже знает ответ. Многочисленному форуму наших читателей под силу найти решения самых сложных проблем. В общем, ждем ваших писем.

Редакция.

Полные листинги некоторых программ расположены по адресу `http://radio-mir.com`

Обзор основных видеокодеков для современной видеообработки

А.СНИТКО,
г.Минск, БГУИР, ФКСиС, 1-й курс.
E-mail: snitko_alexey@mail.ru
WWW: http://www.zx4ever.narod.ru

Наверное, многие из читателей журнала увлечены просмотром и коллекционированием фильмов на своих компьютерах, а некоторые, может быть, и сами занимаются видеообработкой собственноручно снятого видео. Есть среди нас и те, кто занимается риппингом видео с DVD для последующей записи на CD (чаще всего для массового тиражирования и продажи). Должен отметить, что это преследуется по закону.

Для начала давайте разберемся, что же такое кодек? Происходит это слово от английской аббревиатуры CODEC, обозначающей CODER-DECODER (кодер-декодер). То есть под кодеком понимают некую программу (чаще всего выполненную в виде динамической библиотеки), предназначенную для кодирования какой-либо информации (в нашем случае — видеопотока) в специальный формат, с возможностью дальнейшего декодирования и представления в первоначальном виде. Большинство кодеков предназначено для уменьшения размера исходной информации до приемлемых размеров. Происходит это или абсолютно без изменения информации при кодировке-декодировке, или с минимальными искажениями этой информации — так, чтобы особенности органов чувств человека не позволяли ему замечать существенных изменений.

Теперь применим понятие кодека к видеоинформации. Видеокодеки служат для уменьшения астрономических размеров стандартного видеопотока до размера, пригодного для записи видео на стандартные бытовые устройства хранения информации. Сейчас я покажу вам, почему возникает необходимость данной операции. Предположим, что мы представляем видео в виде последовательности картинок, каждая из которых представлена в формате, подобном BMP. Тогда одна секунда стандартного видео формата PAL займет:

720×576×25×2=20736000 байт=20250 Кбайт.

Значит, один час займет:

20250×3600=72900000 Кбайт ≈ 70 Гбайт.

Немного поясню расчеты. Стандартное разрешение картинки в системе PAL — 720*576. Частота — 50 Гц, но, так как картинки выводятся полукадрами, частоту можно принять равной 25 Гц. Я считал, что каждая точка может принимать один из 65536 цветов, что соответствует 16-битной цветовой палитре. Поэтому умножаем все еще на два байта.

Естественно, что никто не собирается тратить на час видео 70 гигабайт дискового пространства. К тому же, я не учитывал место, занимаемое звуковой информацией. Даже при монтаже широкоформатных фильмов в Голливуде используют менее емкие способы записи.

Все видеокодеки можно разделить на два больших класса:

- кодеки для промежуточного хранения видео, которое впоследствии будет подвергнуто обработке и сжатию кодеком следующего класса;
- кодеки для конечного сжатия видео, которое уже было обработано, и записи его на какой-нибудь носитель для последующего многократного просмотра.

Кодеки для промежуточного хранения видео, как правило, используют при его захвате для снижения потока данных, которые требуется сохранить на винчестере. Я думаю, что у немногих из читателей есть жесткие диски, способные справиться с потоком несжатого видео с разрешением 720*576. При использовании же подобных кодеков размер захватываемого видео значительно снижается, и работа, которую должен был выполнять винчестер, перекладывается на плечи почти не загруженного при захвате процессора. Такие кодеки должны обладать рядом достоинств: практически не влиять на качество картинки и при этом обеспечивать приличное сжатие; работать достаточно быстро при кодировании (ведь все происходит в реальном времени, а "выпавшие" кадры не вернешь).

Кодеки для конечного сжатия должны обеспечивать наилучшее соотношение величины сжатия и качества получаемого видео. Производительность же данных кодеков практически не критична. Необходимо только обеспечить надлежащую скорость декодирования при просмотре видео даже на слабых машинах.

Я же в этой статье рассмотрю в основном кодеки для конечного сжатия. Здесь будет сделан только небольшой краткий обзор видеокодеков, а более подробно о каждом из них я расскажу в других статьях. Среди кодеков для промежуточного сжатия я выделил один, по-моему, самый интересный и заслуживающий внимания, он будут рассмотрен в конце статьи.

Сразу же разберемся в смысле понятия MPEG-4, которое является ключевым не только в названии, но и в идеологии многих кодеков конечного сжатия. Я хочу опровергнуть расхожее мнение о том, что "MPEG4 — стандарт файлового формата видео, новое слово в области сжатия и качества изображения" [1]. Формат MPEG-4 был окончательно разработан, одобрен и зарегистрирован в декабре 1999 г. Он явился естественным развитием стандартов MPEG-1 и MPEG-2 и в первую очередь предназначался для передачи медиа-контента по сетям, имеющим низкую пропускную способность. Стандарт MPEG-4 задает принципы работы с тремя видами медиа-данных:

- интерактивное мультимедиа;
- графические приложения;
- цифровое телевидение.

Этот формат задает целую объектно-ориентированную среду, т.е. в нем предусмотрены не просто медиа-потоки и медиа-массивы, а новое понятие — медиа-объекты. В MPEG-4 даже определен двоичный язык описания объектов, классов и сцен — BIFS, который разработчики характеризуют как "расширение C++".

Алгоритм кодирования видео в MPEG-4 не отличается от такового в предыдущих версиях формата. Отли-

чие заключается только в том, что формат работает с объектами произвольной формы, а не разбивает всю картинку на прямоугольники.

Итак, перейдем к непосредственному рассмотрению видеокодеков.

DivX ;-)

Данный кодек открыл новую эпоху в области кодирования видео, которая продолжается до сих пор и закончится, видимо, только с массовым нашествием DVD, т.е. когда те станут стоить копейки. Для начала давайте разберемся с названием данного кодека. Первоначально аббревиатурой DivX обозначался коммерческий вариант DVD-дисков, которые можно было смотреть несколько дней, после чего они переставали работать. Стоили эти диски в несколько раз дешевле обычных, однако этот сервис распространения не получил. Нечто подобное представляют внедряемые сейчас самоуничтожающиеся диски EZ-D.

В сентябре 1999 г. двое хакеров под псевдонимами MaxMorice и Gej взломали кодек Microsoft MPEG-4 Video Codec V3. На это дело их подвигло то, что в текущей версии кодека от Microsoft не было возможности кодирования видео в AVI-файл, а только в ASF. Microsoft, по известным причинам, прекратила разработку своего продукта и выпустила версию, предназначенную только для декодирования. Но не будем сильно углубляться в историю этого кодека — этому будет посвящена следующая статья. Хакеры выпустили как бы два различных кодека в одном — DivX ;-) MPEG-4 Fast-Motion и DivX ;-) MPEG-4 Low Motion. Первый (**рис.1**) больше подходит для быстрых сцен, где создает меньше паразитных следов, например, при резком сдвиге камеры. Второй (**рис.2**) хорош для медленных сцен с равномерными заливками.

Кодек очень быстро набрал популярность, и с его помощью кодировались практически все издаваемые в то время фильмы. Последняя версия этого кодека называется DivX ;-) 3.11alpha, хотя были еще и суррогатные 3.2 и 3.22, выпущенные неизвестно кем. Энтузиастами было произведено множество внешних программ и настроек, позволявших значительно улучшать качество изображения. Но, как я уже упоминал, это тема отдельной статьи.

Перейдем к рассмотрению качества видео, закодированного при помощи этого кодека. Многие говорят о сравнимом с DVD качестве. Однако пока отсутствуют технологии, позволяющие десятикратно уменьшить объем и без того сжатого видео и при этом не потерять на качестве. В любом случае все будет зависеть от опытности кодировщика, количества выделенных на фильм дисков и затраченного на кодировку и подготовку к ней времени. В итоге можно сказать, что "качество картинки будет намного лучше, чем на VideoCD (MPEG-1), но хуже, чем на DVD (MPEG-2)". Готовьтесь к потерям — они неизбежны.

Рис. 1

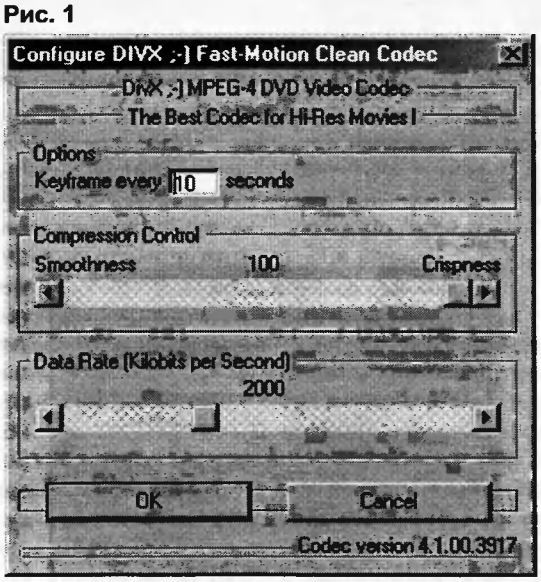
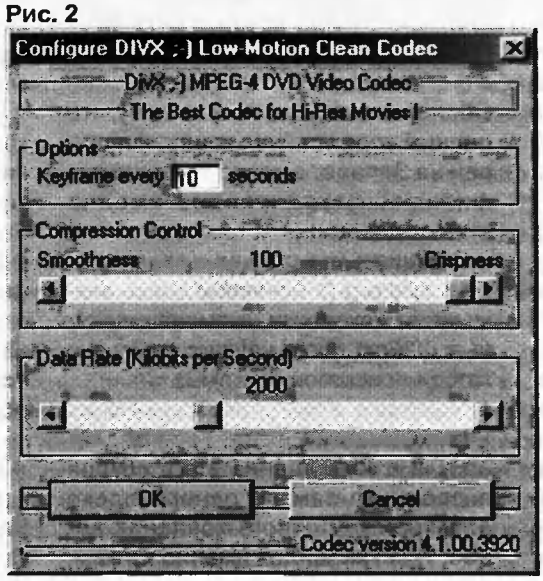


Рис. 2



При кодировании важным является не только высокий битрейт, но и выбор нужного для данного фильма варианта кодека (Low или Fast). Low-кодек позволит вам вместить на 2 CD высококачественный вариант фильма, но все впечатление испортится на размытых быстрых сценах. При применении Fast-кодека вы в любом случае вместите фильм на один диск, но, если в фильме много равномерных сцен, вы будете не в восторге от качества. Поэтому в последнее время получил распространение метод "смешанного кодирования".

DivX™ Codec (от 4.x до 5.0.x)

Чтобы лучше понять концепцию этого кодека, обратимся к истории его создания. Естественно, что грандиозный успех DivX ;-) не мог не привлечь к себе внимания людей, "шустрых" в плане бизнеса. Уже в 2000 г. Джордан Гринхол (бывший администратор MP3.COM) нашел хакера по кличке Gej и создал компанию DivXNetworks. Естественно, они не могли зарабатывать на чужом взломанном кодеке (это все равно, что публично продавать пиратские версии Windows), поэтому в спешном порядке открыли Open Source-проект "OpenDivX". Они преподнесли его как очередной "народный проект". После того как умные люди накачали "OpenDivX" достаточным количеством идей и исходных кодов, руководство в спешном порядке закрыло проект. Даже ребенку ясно, что DivXNetworks изначально открывала "OpenDivX" только для того, чтобы получить работоспособный код для своего детища, а не чтобы сделать новый кодек открытым и общедоступным.

В начале 2001 г. была выпущена первая версия нового кодека — DivX 4.0. Этот кодек содержал множество новых возможностей и новых глюков, а главное — он стал коммерческим продуктом. Существует три версии программы: freeware, adware, shareware. На **рис.3** вы можете увидеть не самую свежую freeware-версию.

К настоящему моменту кодек эволюционировал до версии 5.0.5, но каких-либо изменений в лучшую сторону в качестве кодирования по сравнению с DivX ;-) я не заметил. Артефакты стали другого типа, и их стало больше. В окне настроек появляются новые таинственные пункты, включение которых сулит невероятное повышение качества кодирования, но при этом только снижает его скорость. Популярностью кодек пользуется в кругах тех людей, которые любят новые версии известных программных продуктов, не вдаваясь в подробности наличия у них каких-либо достоинств.

Официальная страница данного проекта — www.divx.com.

Рис. 3

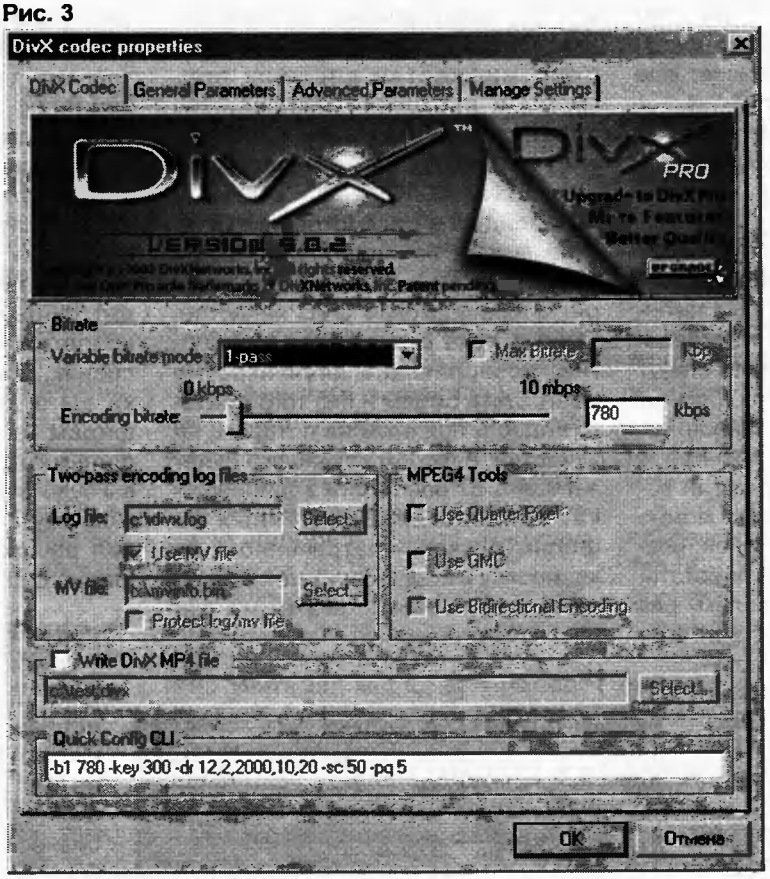
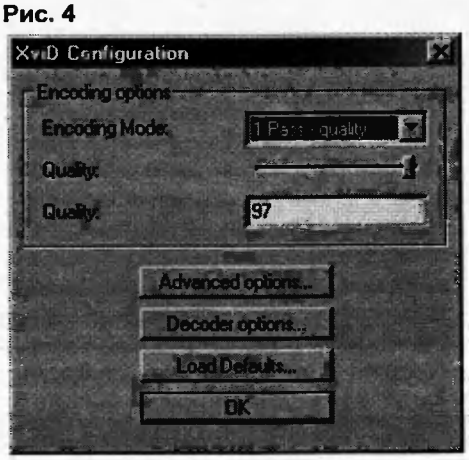


Рис. 4



XviD

Думаю, что вам не составит труда догадаться, как образовалось название этого кодека.

После того как DivX Networks закрыла Open Source-проект "OpenDivX", осталось много людей, которые хотели продолжать работать и делиться свежими исходниками. Они объединились и продолжали работать на www.videocoding.de. Результатом этой работы стал новый высококачественный кодек, который на данный момент является самым прогрессивным и быстроразвивающимся (**рис.4**). Главное достоинство этого кодека в том, что он является полностью бесплатным и свободно распространяемым.

При всем этом XviD ничуть не хуже, а даже лучше всеми любимого DivX'a. В нем добавлено много довольно полезных опций, отсутствующих в DivX. Размер установочного файла при этом довольно компактный — около 400 Кб против нескольких мегабайт конкурента. При правильно установленных настройках он дает гораздо лучшее качество картинки, что выражается в меньшем количестве артефактов. Об этом можно судить и по тому, что данный кодек сейчас очень популярен на Западе, а около трети новых фильмов закодировано именно этим кодеком.

Официальная страница проекта — www.xvid.org (в основном здесь размещены исходники, скомпилированные файлы легко найти с помощью любой поисковой системы).

Huffyuv

Ну, и в завершение этого обзора я расскажу вам об одном кодеке для промежуточного хранения видео. Я остановился именно на нем, потому что он, единственный среди всех других кодеков промежуточного хранения видео, обеспечивает сжатие действительно без потери качества — идентичность "пиксел в пиксел". Получать заранее подпорченное всякими MJPEG'ами видео, а затем редактировать его, мне не очень-то нравится. Тем более, что все распространенные сейчас кодеки конечного видеосжатия очень чувствительны к шуму, источником которого является MJPEG.

Этот кодек работает по такому же принципу сжатия данных, как и все современные упаковщики данных. При этом работает он довольно быстро (сжатие — до 50 Мб видеоданных в секунду на 500 МГц-процессорах), что во многом достигается почти полностью ассемблерным кодом. Кодек оптимизирован для работы с YUY2-палитрой, однако прекрасно справляется и с RGB. В реальных условиях коэффициент сжатия в среднем равен 2 и колеблется в обе стороны в зависимости от содержания картинки.

Как и предыдущий, этот кодек полностью бесплатен, и его исходный код свободно распространяется. Текущая версия кодека — 2.1.1, официальная страница — www.math.berkeley.edu/~benrg.

Ну вот я и рассказал об основных кодеках, которые следует применять в современной видеообработке. При этом по каждому кодеку была дана только обзорная информация, а более подробно я собираюсь рассказать о каждом из них в отдельности в других статьях. Данная статья призвана помочь вам ориентироваться в большом разнообразии современных кодеков.

Литература

1. А.Горячкин. Смотрим видео. — Радиомир. Ваш компьютер, 2003, №4, С.10.

MATLAB. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

Б.КИСЕЛЕВ,
г.Минск,
БГАТУ, каф.ВТ,
тел.264-62-37.

Среди систем компьютерной математики пакет MATLAB наилучшим образом приспособлен для математического моделирования различных объектов, процессов и явлений [1]. В этой статье мы рассмотрим общий порядок построения модели и проиллюстрируем на простейшем примере ряд этапов этого процесса.

Под моделью будем понимать такой материальный или мысленно созданный объект, которым мы заменяем изучаемый реальный объект, сохраняя при этом важнейшие свойства этого реального объекта.

Модель назовем математической, если она представляет собой мысленно созданный объект в виде совокупности математических соотношений, а моделированием — процесс изучения реального объекта, проводимый не на нем самом, а на модели.

Основные этапы математического моделирования:

1. Постановка задачи и определение цели.
2. Математическое описание решения поставленной задачи.
3. Выбор метода решения.
4. Разработка алгоритма решения.
5. Разработка программы.
6. Отладка и тестирование программы.
7. Моделирование на ЭВМ и анализ результатов.

Этапы 1, 2, 7 присутствуют при решении любой оригинальной задачи, т.е. задачу решает пользователь. Остальные этапы могут частично или полностью отсутствовать в зависимости от опыта пользователя, сложности задачи и используемого программного продукта.

Для примера рассмотрим одну из функций сепарирующего транспортера уборочной машины.

Постановка задачи и определение цели

Для интенсивного просеивания примесей (например, картофель с землей) на сепарирующем решете технологические частицы должны двигаться по этому решету в режиме с подбрасыванием. Под действием привода решето совершает гармонические колебания, оставаясь параллельным самому себе. Движение должно быть таким, чтобы находящиеся на нем почвенные комки были разрушены и просеивались сквозь решето, а клубни не повреждались (**рис.1**).

Рис. 1

Воздействие решета на частицы определяется скоростью соударения (V_c) частиц с решетом. Если V_c мала, комки не просеиваются, велика — бьются клубни, следовательно, скорость соударения V_c частиц с решетом должна находиться в пределах, задаваемых свойствами данной сельскохозяйственной (с/х) культуры.

Известны: r — радиус эксцентрика; ω — скорость вращения эксцентрика; допустимые пределы $V_{min} < V_c < V_{max}$.

Нужно определить, будет ли V_c находиться в заданных пределах.

Математическое описание решения поставленной задачи

Обычно угол наклона решета $\alpha = 20...25°$. Поскольку мы исследуем только скорость соударения, то без существенных потерь можно положить $\alpha = 0°$, т.к. движение ленты транспортера (она же — решето) и угол наклона важны при учете обдира кожуры (для простоты модели оставляем наиболее существенные свойства).

Очевидно, что:

$$V_c = V(t_{уд}) + V_к(t_{уд}), \quad (1)$$

где:

- $t_{уд}$ — момент удара подброшенной частицы с решетом;
- $V_к$ — скорость клубня;
- V — скорость решета (при противоположных направлениях скорости должны складываться, а при одинаковых — вычитаться).

Пусть y — вертикальная ось. Тогда для данного механизма вертикальные колебания решета будут определяться следующими выражениями:

- положение — $y(t) = r\sin(\omega t)$; (2)
- скорость — $V(t) = y' = \omega r\cos(\omega t)$; (3)
- ускорение — $a(t) = y'' = -\omega^2 r\sin(\omega t)$. (4)

Чтобы частица была подброшенной, необходимо выполнение условий

$$a(t) < -g;$$
$$\omega^2 r\sin(\omega t) > g. \quad (5)$$

Т.к. $|\sin(\omega)| \le 1$, то для подбрасывания должно выполняться условие

$$\omega^2 r > g. \quad (6)$$

Пусть (6) выполняется.

Если в момент отрыва $t = t_0$, то из (5) следует, что:

$$\omega^2 r\sin(\omega t_0) = g;$$
$$\sin(\omega t_0) = g/(\omega^2 r) = \lambda;$$
$$t_0 = (1/\omega)\arcsin(\lambda).$$

После отрыва клубня в момент t_0, уравнение его полета можно записать так:

$$y_к(t) = y(t_0) + V(t_0)(t - t_0) - g(t - t_0)^2/2, \quad (7)$$

а его скорость в полете:

$$V_к(t) = V(t_0) - g(t - t_0). \quad (8)$$

Из (1) находим скорость соударения:

$$Vc = \omega r\cos(\omega t_{уд}) - \omega r\cos(\omega t_0) + g(t_{уд} - t_0). \quad (9)$$

Найдем $t_{уд}$. В момент удара вертикальные координаты клубня и решета совпадают, следовательно,

$$y_к(t_{уд}) = y(t_{уд}). \quad (10)$$

Из решения этого уравнения определяется $t_{уд}$. Здесь $y_к$ вычисляется из (7), а y — из (2). Заметим, что это уравнение в явном виде решить трудно, поэтому его решают численно, одним из итерационных методов [2].

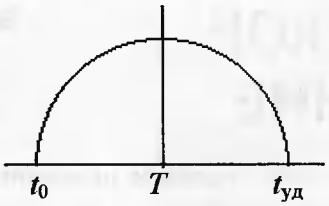
Т.к. все вычисления выполняются с погрешностью ε, то фактически вместо (10) будем иметь:

$$|y_к(t_{уд}) - y(t_{уд})| < \varepsilon.$$

Уравнение (10) удовлетворяется не только в момент $t_{уд}$, но и в момент $t = t_0$, т.е. оно имеет два корня — $t_{уд}$ и t_0. Чтобы найти именно $t_{уд}$, заметим, что падение частицы на решето произойдет после того как она пройдет наивысшую точку траектории свого движения (**рис.2**). Этот момент (обозначим его T) несложно определить. Так как

$$V_к(T) = 0,$$

Рис. 2



то

$$V_к(T) = V(t_0) - g(T - t_0) = 0. \quad (11)$$

Здесь $g(T - t_0) = \omega r \cos(\omega t_0)$;

$$T = t_0 + V(t_0) / g. \quad (12)$$

Таким образом, отрезок, на котором находится корень уравнения (10), лежит правее точки T, следовательно, значение $t_{уд}$ должно удовлетворять ограничению $t_{уд} > T$ или

$$t_{уд} > t_0 + V(t_0) / g. \quad (13)$$

Находя из (10) и (13) значение $t_{уд}$ и подставляя его в (9), найдем V_c.

Соотношения (1)...(13) — это простейшая математическая модель процесса просеивания почвы.

Выбор метода решения

Большинство выражений нашей модели определяется путем непосредственной подстановки значений в формулы. Исключение — уравнение (10), решение которого должно удовлетворять ограничению (13). Уравнение (10) решается одним из итерационных методов [2]. Наиболее часто для этих целей используются разновидности методов Ньютона. В общем случае, выбор метода решения уравнения требует достаточно кропотливой работы — необходимо решить задачу отделения корней и проверить сходимость метода. На практике чаще поступают следующим образом — если уравнение не имеет очевидных особенностей, то сразу применяют одну из встроенных функций используемой компьютерной математической системы, т.к. эти функции достаточно добротны. Мы поступим таким же образом. Если результат окажется подозрительным, тогда займемся исследованием уравнения [2].

Разработка алгоритма

Содержание и форма представления этого этапа зависят от предпочтений и опыта разработчика. Обычно используют графическую интерпретацию. Также пользуются и классическими структурными схемами, и различными типами графов, и т.п.

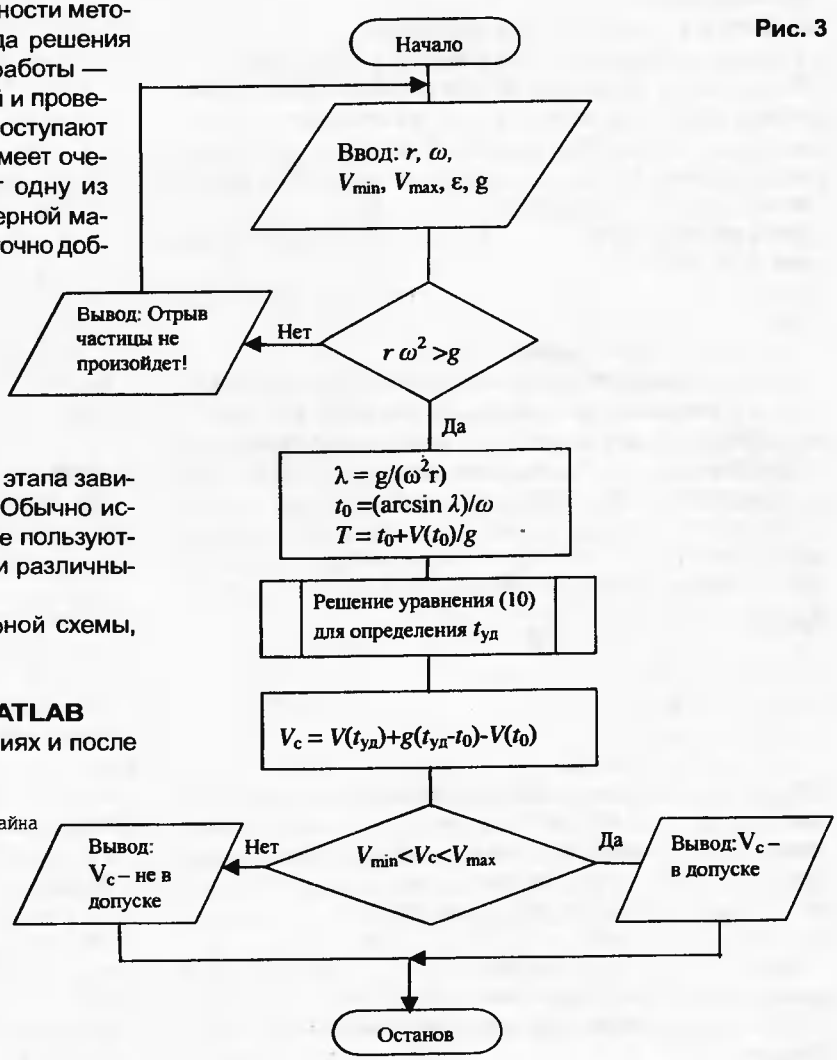
Мы представим алгоритм в виде структурной схемы, приведенной на **рис.3**.

Разработка программы в системе MATLAB

Пояснения к программе даны в комментариях и после ее текста.

```
function kopalka
% Сепарирующий транспортер картофелеуборочного комбайна
% r - радиус кулачка
% om - частота вращения кулачка (рад/сек)
% n - скорость вращения кулачка (об/мин)
% yo - положение клубня и решета в момент отрыва
% vo - скорость клубня в момент отрыва
% to - момент отрыва клубня
% g - ускорение свободного падения
% tud - момент соударения клубня с решетом
% Vmin - минимальная скорость соударения Vc
% Vmax - максимальная скорость соударения Vc
% ---------------------------------------------------
global r om yo vo to g;
g=9.80665; % ускорение свободного падения
while 1
  r=input('радиус кулачка (мм)=');
  n=input('скорость (об/мин)=');
  Vmin = input('min Vc(м/с) =');
  Vmax = input('max Vc(м/с) =');
  r = r/1000; % радиус кулачка в метрах
  om = pi*n/30; % перевод скорости вращения в рад/сек
  om2 = om*om;
  if om2*r <= g  % условие (6) в модели
    disp('Отрыв частицы не произойдет');
  else break
  end;
end;
%
to = asin(g/(om2*r))/om;
yo = y(to);
vo = v(to);
%
% поиск момента соударения tud
T1 = to + vo/g; % левая граница поиска корня уравнения (10)
T2=T1 + (T1 - to)*2; % правая граница поиска корня уравнения (10)
tud = fzero(@root,[T1 T2]); % поиск корня (решение уравнения (10))
%
zz = vc(tud); % скорость соударения Vc
if Vmin < zz & zz < Vmax
    disp('Скорость в допуске')
else
    disp('Скорость не в допуске')
end
% ---------------------------------
```

Рис. 3



```
function transp = y(t)
% положение решета (выражение (2) в модели)
  global r om;
  transp = r*sin(om*t);
%
function kartof = yk(t)
% положение клубня (выражение (7) в модели)
  global yo vo to g;
  kartof = yo+vo*(t-to) - g*(t - to)^2/2;
%
function vel_tr = v(t)
% скорость решета (выражение (3) в модели)
  global r om;
  vel_tr = om*r*cos(om*t);
%
function vel_kart = vk(t)
% скорость клубня (выражение (8) в модели)
  global vo to g;
  vel_kart = vo - g*(t - to);
%
function vel_soud=vc(t)
% скорость соударения (выражение (1) в модели)
  vel_soud = v(t) - vk(t);
%
function f = root(tud)
% выражение для поиска момента соударения (выражение (10) в модели)
  f = yk(tud) - y(tud);
```

Наша программа оформлена в виде m-функции **kopalka** с подфункциями для выражений положения решета и клубня, их скорости и ускорения, а также для поиска корня уравнения (10). Команды **global** объявляют общими переменные в функции и подфункциях. Цикл **while** используется для повторения ввода, если заданные параметры не обеспечивают подбрасывания клубней.

Правила использования встроенной функции для поиска корней **fzero** изложены в [3], кроме того, можно вызвать **help**.

Выбор границ для поиска корня поясняется с помощью рис.2. Параметр T1 — это левая граница отрезка, содержащего корень (это точка *T* на рисунке). Очевидно, что $t_{уд}$ при неподвижном решете расположена симметрично точке t_0. Если решето в момент удара движется вверх, то $t_{уд}$ будет расположена ближе к точке *T*, а если решето движется вниз, то дальше. Чтобы точка *T2* (правая граница отрезка) была наверняка дальше точки $t_{уд}$, возьмем ее на расстоянии $(T - t_0)\cdot 2$ вправо от точки *T*, т.е. $T2 = T1 + (T1 - t_0)\cdot 2$.

Отладка и тестирование программы

Для читателя, знакомого с программированием, этот этап должен быть известен, и мы не будем его комментировать. Отметим только, что при $r = 60$ *мм* $\omega = 240$ об/мин, $V_{min} = 2$ *м/с* и $V_{max} = 2{,}5$ *м/с*, скорость V_c должна быть в допуске.

Моделирование на ЭВМ и анализ результатов

Этот этап определяется целями решаемой задачи, и в нашем случае он очевиден.

Используя полученную модель, можно изучать различные вопросы, например, определить траекторию полета клубня и движения решета. Для этого достаточно дописать в конце программы (перед функциями) следующий текст:

```
x = to:(tud-to)/100:tud; % массив времен для графика с шагом:
                         % (tud-to)/100
plot(x,yk(x),x,y(x));
grid
title('траектории движения решета и клубня')
xlabel('время t')
ylabel('положение решета и клубня')
```

В функции *yk* операцию возведения в степень следует сделать поэлементной:

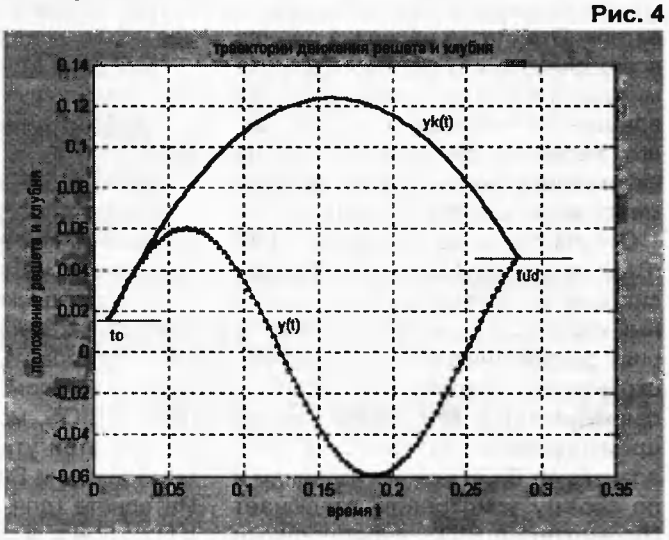
kartof=yo+vo(t-to)-g*(t-to).^2/2;*

В командном окне будет отображено (информация, вводимая пользователем, выделена подчеркиванием):

>> Lab_1_m
радиус кулачка (мм) = 60
скорость (об/мин) = 240
min Vc(м/с) = 2
max Vc(м/с) = 2.5
Скорость в допуске

Графическое окно после редактирования представлено на **рис.4**.

Рис. 4



В заключение отметим, что математическое моделирование — весьма объемное и разностороннее понятие, поэтому данная статья претендует лишь на элементарную иллюстрацию наиболее общих этапов при компьютерном математическом моделировании.

Упражнения

Необходимо модифицировать рассмотренную программу или написать свою для следующих задач (за основу следует взять упомянутые выше данные из примера):

1. Определить максимальный радиус кулачка эксцентрика при постоянной скорости его вращения (увеличивать радиус до тех пор, пока V_c остается в допуске).
2. Определить минимальный радиус кулачка эксцентрика при постоянной скорости его вращения.
3. Определить максимальную скорость вращения (в об/мин) кулачка эксцентрика при постоянном его радиусе.
4. Определить минимальную скорость вращения (в об/мин) кулачка эксцентрика при постоянном его радиусе.
5. Построить график скорости полета клубня.
6. Построить график ускорения полета клубня.

Литература

1. Дьяконов В.П. Компьютерная математика. Теория и практика. — М.: Нолидж, 2001. — 1296 с.
2. Демидович Б.П., Марон И.А. Основы вычислительной математики. — М.: Наука, 1966. — 664 с.
3. Киселев Б.М. MATLAB. Программирование в MATLAB — Радиомир. Ваш компьютер, 2003, №№9, 10.

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

Разблокирование скрытых функций BIOS

Тем, что сейчас каждый рядовой пользователь имеет возможность регулярно обновлять BIOS материнской платы, пожалуй, никого не удивишь. А вот то, что BIOS можно модифицировать, и тем самым включить и отобразить в Setup BIOS скрытые дополнительные параметры — об этом знают лишь немногие опытные пользователи. Эта статья написана прежде всего для тех, кто занимается оптимизацией и постоянно ищет пути для повышения производительности своего "железного друга", кто ставит на своем компьютере смелые эксперименты и не боится рисковать.

Откуда берет свое начало BIOS (Basic Input/Output System) — базовая система ввода/вывода? Программный код BIOS рождается в лабораториях разработчиков. Самые распространенные разработчики BIOS — фирмы Award и AMI. Затем каждый производитель материнских плат адаптирует BIOS к своим изделиям и по своему усмотрению отключает (блокирует) в BIOS определенные функции. Причем заблокированными могут оказаться многие функции, влияющие на тонкую настройку BIOS, которые, в свою очередь, могут оказать влияние на повышение производительности компьютера. Для чего производители материнских плат блокируют эти функции в BIOS? Прежде всего, это делается в целях упрощения процедуры настройки Setup BIOS. Отключаются и те функции, которые в данной модели материнской платы отсутствуют, но могут быть задействованы в следующих ревизиях.

Используя специальное программное обеспечение (утилиты), можно разблокировать практически все заблокированные функции и опции в BIOS. После модификации BIOS результаты необходимо сохранить в отдельном файле и затем произвести обновление BIOS. При загрузке Setup BIOS на экране монитора можно будет увидеть новые дополнительные возможности для настройки. Для BIOS каждой фирмы-разработчика нужна своя утилита. Для обладателей Award BIOS понадобится утилита Modbin, а для тех, у кого на компьютере установлена AMI BIOS, нужно скачать утилиту AMIBCP. Интересен тот факт, что утилита AMIBCP разработана программистами самой фирмы AMI.

А теперь перейдем к самому главному — к практическим занятиям. Прежде всего, перед модификацией BIOS нужно подготовить необходимое программное обеспечение. Итак, для модификации BIOS потребуются:

- утилита для обновления BIOS ("прошивальщик");
- файл с действующей версией BIOS;
- утилита для модификации BIOS.

Утилиту для "прошивки" BIOS можно найти в Интернете на сайте производителя материнской платы или на компакт-диске, прилагающемся к "материнке", после чего ее необходимо скопировать на жесткий диск. Файл, в котором размещена текущая версия BIOS, можно получить с помощью этой утилиты. Утилиты для модификации BIOS можно скачать в Интернете (для Award — по адресу http://www.biosmods.com/download.php, для AMI — http://gate.creckx.com/recenze/hard/download/k7s5a/amibcp75103.rar).

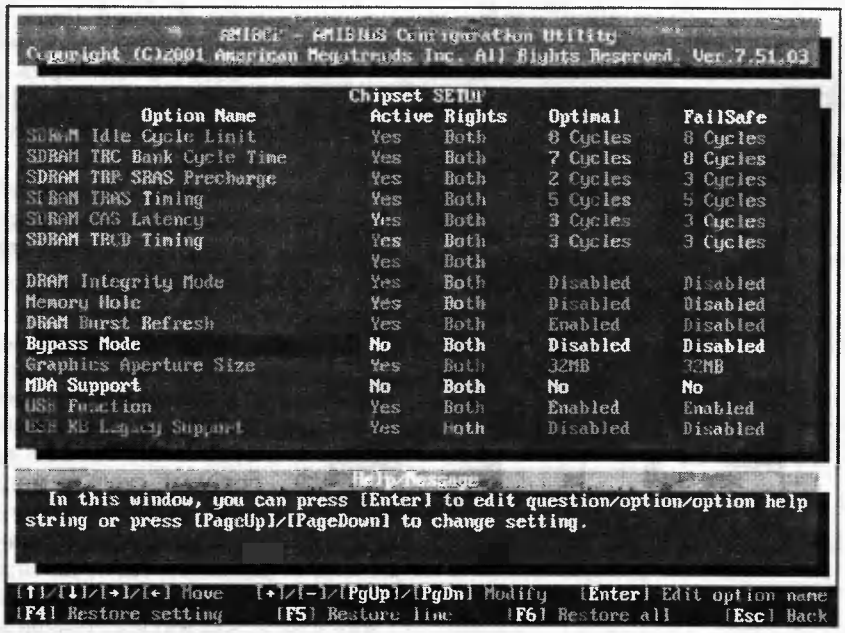
Файл с BIOS желательно разместить на жестком диске в одной директории с утилитой для модификации. Обязательно перед модификацией сделайте резервную копию файла с BIOS на случай "отката". Теперь, когда все готово для модификации BIOS, можно смело загружать утилиту.

В качестве примера рассмотрим случай, когда на компьютере установлена AMI BIOS. Что касается BIOS от Award, то принцип модификации аналогичен, и в этой статье рассматриваться не будет. Утилита AMIBCP является DOS-программой и без проблем загружается под Windows без перезагрузки в режим эмуляции MS-DOS. Объем утилиты AMIBCP (файл **amibcp75.exe**) в распакованном виде составляет 542 Кб.

После загрузки утилиты AMIBCP, необходимо вводом с клавиатуры указать имя файла с BIOS, например, **7vr_f4.bin**. Далее, находясь в главном меню программы, ищем раздел "Configure Setup Data" и по порядку просматриваем все подразделы. Особенно интересны для нас будут подразделы "Chipset Setup" и "BIOS Features Setup". Сама модификация заключается в активации заблокированных опций. В каждом разделе присутствуют столбцы с заголовками "Option Name", "Active", "Rights", "Optimal" и "FailSafe". Заблокированные опции подсвечены, напротив этих опций в столбце "Active" указано "No". Клавишами Page Up и Page Down можно задействовать ту или иную заблокированную установку. Возможны случаи, когда в столбце с заголовком "Option Name" названия опций на экране полностью отсутствуют. В таком случае придется включать все опции подряд вслепую.

Закончив изменения в BIOS, выходят в главное меню программы AMIBCP и по нажатию клавиши F10 сохраняют модифицированный BIOS в файле. После сохранения модифицированного файла BIOS применяют "прошивальщик" — утилиту для обновления BIOS. Ряд производителей материнских плат выпускает свои фирменные утилиты для обновления BIOS непосредственно в среде Windows. К их числу можно отнести утилиты фирм Gigabyte, ASUS, Intel и др. Об утилите Gigabyte @BIOS Writer подробно писалось в [1].

Перед прошивкой BIOS необходимо убедиться в том, что в Setup BIOS разрешено обновление — в разделе BIOS Features Setup параметр BIOS Flash Protection должен находиться в положении Enabled или Auto. После прошивки BIOS производят перезагрузку компьютера и входят в Setup BIOS. Эффект в результате модификации BIOS будет гарантирован. Если говорить на эту тему более конкретно, то в моей практике модификации BIOS были заблокированы настройки, позволяющие увеличивать производительность системы: возможность изменять напряжение на шине AGP в диапазоне 1,5...1,8 В с шагом 0,1 В и на модулях памяти DDR в пределах 2,5...2,8 В с шагом 0,1 В, а также повышать на 5, 7,5 или 10% штатное напряжение на ядре процессора. Имели место случаи, когда в BIOS был отключен режим **S.M.A.R.T.** для жестких дисков и режим **Bypass Mode** для оптимальной работы CPU (см. **рисунок**). Интересной была и



заблокированная опция в подразделе Hardware Monitor — **Slow Down CPU Duty Cycle** — ответственная за уменьшение тактовой частоты CPU при переходе системы в режим Doze. С помощью данной опции можно установить другое значение тактовой частоты CPU в процентах от предыдущего значения тактовой частоты.

В заключение следует заметить, что не стоит из-за полного контроля над BIOS терять голову. И поэтому к модификации и изменению параметров BIOS следует подходить с некоторой долей осторожности, и с пониманием того, что вы делаете и на что вы идете.

Литература

1. Горячкин А. Системная плата MB K7 Gigabyte GA-7IXE4. — Радиомир. Ваш компьютер, 2002, № 6, С.37

С.РЮМИК,
г.Чернигов.

Интернет-калейдоскоп, freeware #1

Познание начинается с удивления.
Аристотель

Интернет — это огромная библиотека с достоверной и не очень достоверной информацией. Каждый ищет в Сети "свое сокровенное", но иногда бывает полезным посмотреть, а чем же интересуются другие?

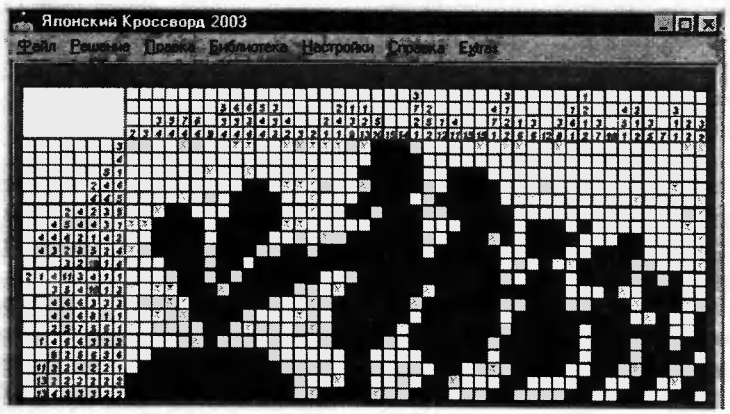
Нонограмма, гриддлер и японский кроссворд

Слова разные, но обозначают они одно и то же — японскую головоломку-рисунок. Изобретателем этого жанра считается Ноно Ишида (Nono Ishida), дизайнер по профессии. Именно она в 1988 г. опубликовала три первых числовых рисунка-загадки. В 1990 г. головоломки появились в Европе под названием "нонограммы", NON — от имени изобретательницы, GRAM — сокращение от слова "диаграмма". В дальнейшем были созданы их многочисленные клоны — цветные, с треугольниками, объемные и т.д., под общим обозначением "гриддлеры" (griddlers). "Японский кроссворд" (ЯК), он же "японский сканворд" — это два нарицательные названия, придуманные нашими соотечественниками. Нелогичность кроется в окончании "...ворд" (от англ. "word" — слово), поскольку в головоломке имеются пересечения закрашенных клеток "кросс...", в которых полностью отсутствуют буквы.

Как бы там ни было, но ЯК пользуются заслуженной популярностью не только у читателей, но и у программистов. Известны программы-"решалки" ЯК для трех платформ: домашние компьютеры (ZX-Spectrum, Amiga), игровые приставки (GameBoy, SNES, PlayStation), персональные компьютеры (IBM PC, Mac).

Теория ЯК подробно изложена в [1], но программой удобнее пользоваться не на ZX-Spectrum, а на IBM PC — здесь и сервис получше, и возможности пошире, и быстродействие повыше. В Интернете из русифицированных программ под Windows 9x/NT наиболее популярны две — "Японский кроссворд 2003" (И.Егоркин, http://enotes.diallink.net/jc.zip, 232 Кб) и "Японские кроссворды 2.1" (Len Stepanov, http://len.binn.ru/download/japanins.exe, 488 Кб). Первая из них, простая в управлении, имеет встроенную библиотеку кроссвордов и оригинальный формат записи файлов *.jcr (**рис.1**), вторая — поддерживает 9 различных типов файлов, конвертирует их друг в друга, быстрее проводит расчеты (**рис.2**). Обе программы снабжены системой подробных подсказок и позволяют решать, составлять, распечатывать, загружать, редактировать, устанавливать точки ветвления и даже автоматически преобразовывать рисунки в кроссворды.

Рис. 1



Ключевые слова для поиска: японский кроссворд, jcr, jpn, jpz, cwd, jc, jcw, griddlers, nonograms.

"Реверс инжиниринг" в программировании

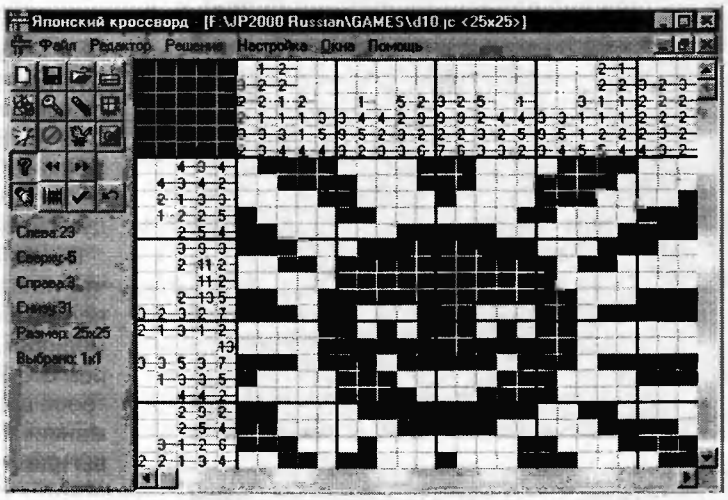
В электронике давно применяется процесс, известный как инженерный анализ или обратное проектирование ("reverse engineering"),

который позволяет создавать новое устройство, имея промышленный образец для копирования. Назвать плагиатом такие действия сложно, поскольку знание того, что должно делать устройство, не означает знание того, как это сделать. Технологию изготовления, а именно она составляет производственный секрет, пользователь разрабатывает самостоятельно.

Как пример, юридическую законность процесса "реверс инжиниринг" смогла доказать ф.Cyrix в судебном разбирательстве против ф.Intel в отношении ее клонов процессора Pentium-II. В 1998 г. между двумя фирмами было заключено мировое соглашение, и ф.Cyrix смогла законно создавать чипы "а-ля" Pentium-II.

В программировании методы "реверс инжиниринг" широко применяют хакеры для вскрытия программного кода. Если отвлечься от неблаговидных целей, которые они иногда преследуют, то сам процесс получения новых знаний заслуживает пристального внимания и изучения. Обычно используют различные дизассемблеры и отладчики. Но при этом на полную расшифровку кода уходит непомерно много времени, да и выяснить структуру программы очень сложно. Облегчить работу помогают декомпиляторы с языков высокого уровня. Один из них, для языка Си, размещен по адресу http://www.backerstreet.com/rec/rec16pc.zip, 195 Кб. Декомпилятор REC (Reverse Engineering Compiler, автор Giampiero Caprino) оптимизирован под диалект GnuC и восстанавливает структуру программного кода файлов с расширением *.exe. Если в программе использовался другой диалект Си, то значительно увеличивается длина листинга. Получаемый файл *.rec (см. пример в **листинге**) аналогичен файлу с расширением *.c. Он не является точной копией текста разработчика, поэтому не может рассматриваться как плагиат.

Рис. 2



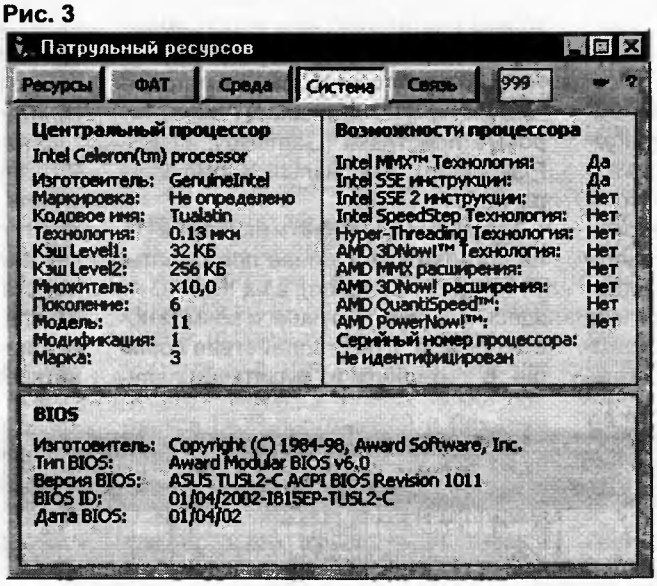
Листинг

```
while(1) {
        ah = si == 0x3fa ? 0xff : 0;
        dx = di;
        for(bx = si; bx != di; bx = bx + 6) {
            if(*(es:bx) != 0xff) {
                if(si == 0x3fa) {
                } else {
                    ah :: *(es:bx+0x1);
                }
```

Многие программисты отмечают ограниченные возможности REC. И вот на одном из форумов появилось сообщение о декомпиляторе REVENGE (REVerse ENGineering Environment). Это интерактивный C/C++ дизассемблер/декомпилятор/делинкер для x86, позволяющий реально воссоздать исходный код из файла *.exe. Восстанавливается не только логика кода (execution flow), но и структуры и типы данных, даже поддерживаются объединения (union) и классы. Задержка за малым — подождать, пока белорусский программист Андрей Шульга (shulgaaa@tut.by) разместит в Интернете демо-версию своего декомпилятора REVENGE.

Ключевые слова для поиска: декомпилятор, C++, decompiler, REVENGE, REC.

Рис. 3



Скажи, процессор, как имя твое?

Перед покупкой компьютера пользователь дотошно изучает по прайсам, сколько мегагерц, мегабайт и мегабит в секунду поддерживает его центральный процессор (ЦП), память, системная плата. В процессе покупки пользователь может удостовериться в соответствии заявленных параметров при помощи встроенных средств Windows: "Мой компьютер — Панель управления — Система — Общие".

А вот предусмотрительный человек принесет с собой еще и дискету с программой "Патрульный ресурсов" (В.Джангл, ЗЕНИТ Технософт, http://zenith.at.tut.by/files/patrul550.rar, 318 Кб). Программа для частных лиц распространяется без ограничения по времени использования. По ней можно не только выяснить точное значение тактовой частоты ЦП, системной шины, объема памяти, но и узнать тонкости производственного процесса. В частности, по кодам, зашитым в служебных ячейках памяти ЦП, можно определить имя, поколение, множитель, технологические нормы, модель ЦП. После тестирования пользователь вправе заявить окружающим: "У меня на борту подлинный (Genuine) Intel Celeron с кодовым именем Tualatin, множителем 10, технологией 0,13 мкм и тактом 1 ГГц" (**рис.3**).

Ключевые слова для поиска: патрульный ресурсов, тестирование процессора.

Литература

1. Рощин И. Японские кроссворды. — Радиомир. Ваш компьютер, 2003, №2, С.39-43.

Возрождение BBS в лучших традициях

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

1. Восход или закат?

Как ни прискорбно это признавать, но на данном этапе своей истории BBS (**Bulletin Board System** — электронные доски объявлений) являют собой довольно жалкое зрелище. На мой взгляд, вовсе не потому, что они устарели, а просто из-за того, что никто их не продвигает в массы, как, например, всевозможные сайты в сети Internet. А ведь идея BBS действительно замечательная! Для тех, кто не понимает, о чем идет речь, я расскажу подробно.

Биба, как ласково называют BBS — это такая программа (tornado, delta, max), которая позволяет с использованием одного модема и одного телефона общаться, обмениваться файлами десяткам людей. Как же это происходит? Довольно просто — скажем так, по очереди. Вначале звонит один человек и оставляет свое послание в форуме, затем звонит другой и отвечает. То же и с файлами. Их может закачивать как СисОп (системный оператор — начальник BBS), так и пользователи станции, тем самым обмениваясь между собой всевозможными занимательными вещами. Этот принцип действия называется "ЭХО", надеюсь, всем легко понять, почему.

Споры вокруг полезности или бесполезности BBS не затихают ни на миг. Кто-то считает их рудиментом прошлого, а кто-то — альтернативой в будущем. Но мы сегодня не будем решать, кто прав, а кто виноват. Проще будет рассказать про все как есть. А читатель пусть сам и решает.

2. Зачем это нужно?

"Зачем тебе это нужно?" — спросили меня, когда я только начал работу по созданию BBS. "Для друзей", — ответил я и счел, что этим все сказано.

Всем известна проблема старых компьютеров AT или XT — и выкинуть жаль, и для работы уже не пригоден. Вот здесь и появляется большое поле для действий. Чтобы мой "старый конь" не пылился в шкафу, решено было сделать из него новую BBS-станцию. Компьютер IBM 286-12 Mhz/4 Mb RAM/170 HDD и, конечно же, модем на 33600 (COM2) для Бибы подходят как нельзя лучше. Сразу оговорюсь, что старые СисОпы, всегда считают своим долгом высказать сомнение в работе станции под MS DOS. Почему-то всех их сразу тянет в далекие дебри OS/2. Ну, да фидошный бог с ними!

После долгих мучений, которые были испытаны при поиске подходящего программного пакета BBS, были обнаружены всего три достойные программы — MAX, DELTA5 и TORNADO BBS. После длительного изучения был сделан вывод, что для компьютера на базе 286-го процессора больше всего подойдет TORNADO, т.к. данный продукт является русскоязычным по своей сути и просто настраиваемым по идее. Отмечу то, что программы для работы с BBS можно скачать с сайта http://www.fdd5-25.narod.ru /network.htm. Там же находится fossil-драйвер, необходимый для работы с com-портом, а следовательно, и с модемом. Без него станция будет работать только в локальном режиме. После всех приготовлений следует перейти к установке TORNADO. Она довольно простая, и, следовательно, этот процесс в описании не нуждается. А вот с настройкой программы все же придется помучиться. После того как программа установлена в каталог (у меня это C:\TORNADO\), сразу же следует заглянуть в файл tornado.ctl — главный файл BBS. Здесь мы обнаружим множество параметров, все они описаны на русском языке, что значительно облегчает настройку. Первые строчки, на которых заострим внимание:

```
HardWare_Flow   Yes   ; Понимает ли модем (и настроен ли он
                      ; соответствующим образом) управление
                      ; потоком данных по сигналам RTS/CTS
                      ; (по умолчанию - Yes).
SoftWare_Flow   Yes   ; Понимает ли модем (и настроен ли он
                      ; соответствующим образом) управление
                      ; потоком данных по XON/XOFF
                      ; (по умолчанию - Yes).
```

Для того чтобы данные параметры работали, в строке инициализации модема файла tornado.ctl пропишем:

```
InitString ATZ| AT/Q3| AT/N5| ATC0|,
```

где ATC0 означает поддержку сжатия данных (0 — выключено, 1 — включено). Если же у вас soft-модем, тогда следует убрать аппаратную поддержку HardWare_Flow No. В строчке BaudRate укажем скорость модема (у меня это 33600). Остальные настройки модема оставим без изменений. Начинающим пользователям не советую далее изменять файл tornado.ctl — это может вызвать неработоспособность программы!

Далее мы создадим папку C:\TMP. В нее TORNADO будет копировать временные файлы. После этого открываем для редактирования файл limits.ctl. Здесь указаны параметры уровней пользователей — задано время пребывания на BBS и максимальный объем информации, скаченной за сутки, в килобайтах. Если вас все устраивает, тогда двинемся дальше. Теперь нам следует создать FILES AREAS — место, где будут храниться разные файлы, доступные пользователям. Для этого мы отредактируем файл Filearea.ctl. Здесь тоже ничего сложного нет, вам лишь следует указать пути и создать нужные папки. У меня это выглядит вот так:

```
[FileArea]
Name            "Старые добрые игры :-)"
DLPath          C:\TORNADO\FILESBBS\games
FileList        C:\TORNADO\FILESBBS\games\files.bbs
ULPath          C:\TORNADO\FILESBBS\UPLOAD
Scan_NewFiles   Yes
DL_Security     1
UL_Security     1
List_Security   1
Show_Security   1
Group           SOFT
;
[FileArea]
Name            "Программы для работы и создания BBS"
DLPath          C:\TORNADO\FILESBBS\BBSSOFT
FileList        C:\TORNADO\FILESBBS\BBSSOFT\files.bbs
ULPath          C:\TORNADO\FILESBBS\UPLOAD
Scan_NewFiles   Yes
DL_Security     1
UL_Security     1
List_Security   1
Show_Security   1
Group           SOFT
```

В каждой папке из FILES AREAS создайте файл files.bbs, там будут находиться описания всего, что есть в данном разделе. Как видите, в этом ничего сложного нет.

Теперь, что касается почтовой области — Msgarea.ctl. Я у себя все оставил по умолчанию, но путь к базе сообщений все же изменил, и вам советую:

```
BasePath C:\TORNADO\FIDO\MSGBASE\j25 ; Путь/имя
                  ; (в зависимости от типа базы)
```

Теперь, что касается защиты. Файл Doorway.ctl отвечает за доступ в виртуальную консоль или, проще говоря, к жесткому диску. Поэтому, чтобы никто не посягнул на ваш хард, поставьте возле каждой опции максимальный уровень доступа, под которым может заходить только SysOp, то есть вы. У меня такой уровень называется 31337. Смотрите пример:

```
Enter_Security    31337  ; Минимальный уровень доступа,
                         ; необходимый для активизации
                         ; Tornado DoorWay
ChDir_Security    31337  ; Минимальный уровень доступа
                         ; для выполнения команды CD (сменить
                         ; текущую директорию)
```

И вообще предпочтительно удалить все ненужные для обычных пользователей опции в меню BBS. Для этого нам следует посетить папки Menus\Russian и Menus\English — соответственно, русскоязычную и англоязычную конфигурацию. В обоих каталогах есть файл main.mnu, где можно прописать доступ пользователя BBS того или иного уровня в одну из областей станции. У меня это выглядит следующим образом:

```
Gosub_Menu     "papirus" 31337 "\14(\15H\14) \02Papirus        |" H
Todays_Callers ""        31337 "\14(\15T\14) \02Сегодня Звонили |" T
Shell          ""        31337 "\14(\15*\14) \02DoorWay         " *
```

В область, для которой указан параметр 31337, может попасть только SysOp или пользователь, которому доверен столь высокий уровень. То же самое справедливо и для файла msg.mnu, в котором можно убрать определенные опции почтового раздела:

```
DownLoad_QWK "" 31337 "\14(\15D\14) \02Скачать почтовый пакет QWK  " D
UpLoad_QWK   "" 31337 "\14(\15U\14) \02Закачать пакет ответов REP ||" U
```

Для файловой области files.mnu:

```
UpLoad      "" 5555N "\14(\15U\14) \02Закачать файл(ы)       "  U
```

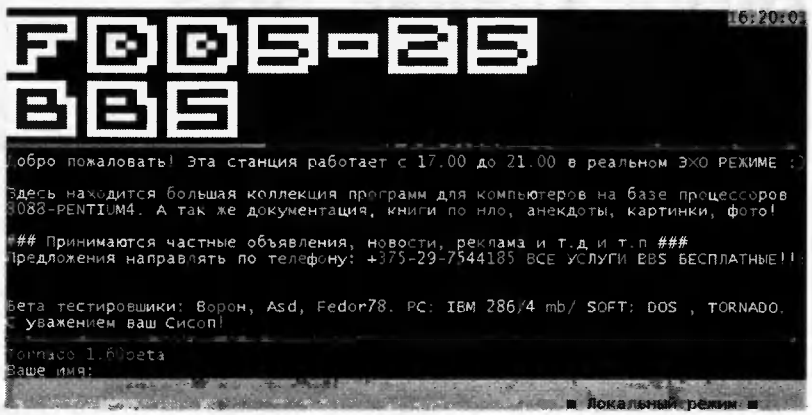
Уровень 5555 тоже реализован мною, это то же, что и 5-й уровень в обычной стандартной настройке TORNADO. Т.е. пользователи уровней 1, 2, 3, 4 файлы на BBS закачивать не могут. Вы же можете изменить этот или другой параметр, как сами того пожелаете:

```
UpLoad_Priv  "" 31337 "\14(\15C\14) \02Закачать для ...    |" C
FileList     "" 0     "\14(\15F\14) \02Список файлов       " F
Show_Raw_Dir "" 31337 "\14(\15R\14) \02Показать каталог    |" R
```

Советую заблокировать все вышеописанные опции.

Создатель программы TORNADO — человек с юмором, это заметно во всем, но особенно в файле Gooduser.ctl, в котором записываются имена тех пользователей, которые при входе на BBS получают полный доступ ко всему без паролей. Советую убрать оттуда всех, на ваш взгляд, неблагонадежных лиц.

Рис. 1



Теперь перейдем к самому главному — логотипу нашей BBS. Я использовал для его создания программу The Draw (www.fdd5-25.narod.ru/multi.htm). Поскольку все мои посетители используют в своей работе только ANSII-терминал, то и логотип буду изменять для него. Для этого заходим в TXTFILE\logo.ans и изменяем текст приглашения и логотип. Вот что получается в итоге (**рис.1**).

После того как вы произвели все вышеописанные действия, вы можете запустить BBS в локальном режиме и понаблюдать, как она работает. Для этого в командной строке наберите — tornado -l.

Я уже писал, что для работы BBS с модемом нам нужен fossil-драйвер. Поскольку он настраивается из командной строки, рискну вам предложить готовый bat-файл:

```
adf COM2 2F8  3  115200  4096  4096  8  3  11
```

Эти параметры справедливы для компьютера класса Pentium с модемом, подключенным к порту COM 2.

Статус зарегистрированного пользователя можно изменить программой USEREDIT.EXE. Чтобы вам было проще заходить с удаленной машины на вашу BBS, зарегистрируйтесь в локальном режиме, а затем просто выставьте для себя самый высокий статус.

Поздравляю! Биба готова на 50%! Осталась самая малость. В папках txtfile\Russian и English создайте файл news.txt или откройте его для редактирования, если таковой имеется. В нем каждый раз после обновления станции вы будете писать свежие новости. Это все, что я хотел написать о самой простой настройке BBS.

3. FAQ

Во время создания и работы с BBS у пользователей возникает множество вопросов, на которые в документации зачастую нет ответов. Поэтому позволю себе описать более или менее распространенные проблемы.

Что должно быть на станции?

На BBS, по моему мнению, должны находиться в основном эксклюзивные вещи, чтобы не повторять бесчисленные сайты Internet. Как это сделать? Очень просто. Постарайтесь выяснить у людей, чего, на их взгляд, не хватает на нашей BBS. Найдите знакомого программиста, который пишет свои программы, и предложите ему раздел на станции, где он сможет их выкладывать. Сделайте ставку на загадочное (книги о НЛО, привидениях и т.д.), также на

редкий soft. Большим успехом пользуются старые игры, программы для взлома в сети, а также коллекции серийных номеров. Например, моя BBS FDD5-25 посвящена всему вышеописанному, и еще там есть фото далеких планет с ftp-сервера НАСА и снимки моего района. Запомните, человек, который заходит к вам, не должен натыкаться на очередную бессмысленную сборку всего подряд! Он должен попасть в ваш круг интересов. И быть может, тогда у вас станет одним хорошим другом больше.

Как посетить BBS?

Для работы с BBS нужен терминал — программа, которая позволит нам установить соединение с удаленной машиной через модем (в нашем случае с BBS). Какими же бывают терминалы? Условно мы их поделим на два типа — под Windows и под DOS. Две самые распространенные программы подобного класса — это Hyper Terminal for Windows и Telemax for DOS, входящий в пакет Norton Commander 5.0. Вначале определимся, с какой из этих программ вам придется работать. Если у вас winmodem, то будем использовать Hyper Terminal, а если же полностью аппаратный, тогда не грех воспользоваться и Telemax. Итак, про все по порядку.

Hyper Terminal

Для работы этой программы с BBS нам следует провести небольшую настройку. Бытует странное мнение о том, что с помощью Hyper Terminal нельзя корректно работать с псевдографикой и русским языком. На самом деле можно! Как? Очень просто. Запустим Hyper Terminal, отменим все выскочившие окошки, кроме того, где появляется название соединения. В нем напишем — BBS. Далее зайдем в Вид/Шрифт и выберем шрифт terminal. Все, ваша псевдографика и русские буквы будут видны лучше некуда! Далее нажмем "OK". Зайдем в Файл/Свойства и уберем галочку с использования кода страны и города. Это для того, чтобы в наш набор номера не прописались непонятные цифры. Теперь звоним на BBS, для этого в окошке печатаем:

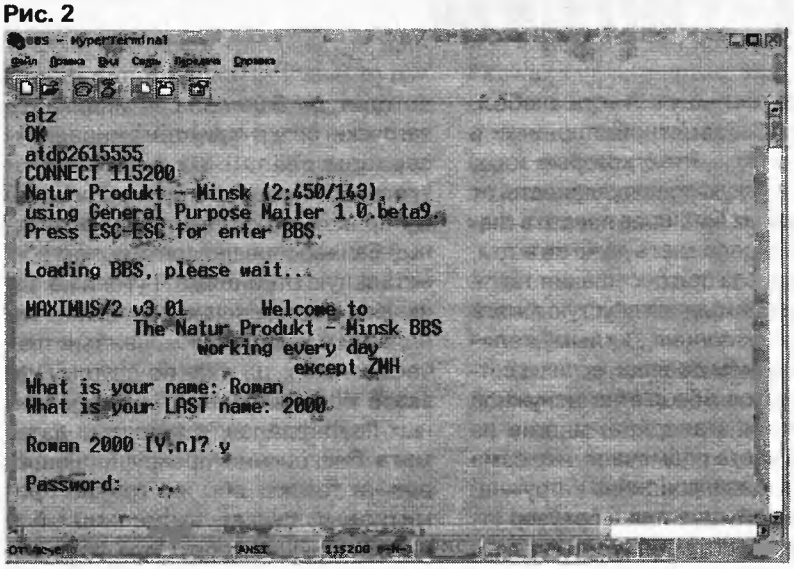
ATZ — строка инициализации модема;

Ok — ответ, что модем проинициализирован;

ATDP(номер телефона) — звоним на BBS.

Все! Если не занято, то вы попали на станцию (**рис.2**)! Ну, чем не Internet Explorer?

Рис. 2



Telemax

Чтобы начать работу с Telemax, надо узнать, к какому COM-порту подключен ваш модем. Если вы пользуетесь Windows 98, то это можно узнать с ее помощью. Щелкните правой кнопочкой мыши на иконке "Мой компьютер", далее "Свойства" — раздел "Устройства"—"Модем"—"Свойства"— раздел "Модем". В строке "Порт" должно быть написано, например, "Последовательный порт (COM2)". Затем в Telemax в появившемся окне с запросом выставляем порт COM2, скорость обмена — 115200. Нажимаем OK. В открывшемся окне печатаем:

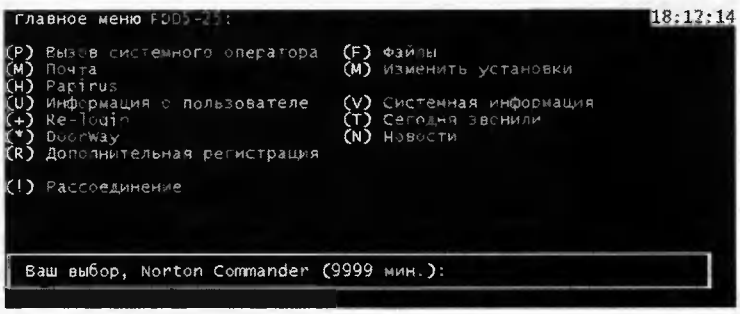
ATZ — строка инициализации модема.

Ok — ответ, что модем проинициализирован.

ATDP(номер телефона) — звоним на BBS.

Если не занято, то вы попадаете на BBS (**рис.3**). Так выглядит моя FDD5-25 BBS, когда на нее заходит пользователь 31337-го уровня, или, проще говоря, SysOp.

Рис. 3



Распространенные проблемы с терминалом

Если программа спрашивает строку инициализации, а вы не знаете, что это такое, то напишите "ATZ" (без кавычек), а затем нажмите Enter.

Если не работает скорость 33600, и ничего не получается сделать, попробуйте использовать только такие числа: 115200, 57600, 38400, 19200, возможно ваш порт работает медленнее, чем вы указали.

Если у вас аналоговая АТС, то вам нужно набирать номер в импульсном режиме командой ATDP — Pulse Dial. Если телефонная станция — цифровая, и работает в тоновом режиме, то пишите ATDT — Tone Dial.

Как себя вести на BBS?

Ведите себя так же как и на сайте в Internet. Желательно при регистрации указывать реальные данные. Обычно посторонним они недоступны. А системному оператору будет легче обновлять BBS, т.к. как он будет знать свою аудиторию и ее интересы. Обязательно оставьте свое мнение о станции в почтовой области.

Как перемещаться по разделам?

После регистрации вы получаете доступ к главному меню BBS. Для переключения разделов используются горячие клавиши. Например, написано (F) Files area — это значит, что клавиша "F" отвечает за переход в это меню. Если написано (!) Разъединение, то нужно нажать Shift+1, чтобы покинуть BBS.

Низкая скорость при скачивании

Многое зависит как от модема, так и от вашей АТС. Если у вас плохое соединение с BBS, не поленитесь перезвонить, оно должно стать лучше. Бывает, что во всем виноват оператор BBS — просто он как следует не протестиро-

вал станцию, а сразу же запустил ее в работу. Тогда проблема, вероятно, в параметрах модема и fossil-драйвера. Скорее всего, строка инициализации не содержит подключения режимов софтовой и аппаратной коррекции соединения, а fossil-драйвер неправильно сконфигурирован. Почему-то все забывают об установках параметров контроля линии, отсутствие которых вызывает при скачивании огромное количество ошибок и, как следствие, низкую скорость соединения. Пример правильной конфигурации fossil-драйвера для компьютера Pentium был приведен выше.

Какой протокол выбирать при скачивании? Их так много...

У себя на станции я по умолчанию использую ZModem. Тем более, что он интегрирован в TORNADO, и, следовательно, мне не нужно устанавливать внешние программы для работы BBS. Протокол ZModem хорош еще и тем, что при потере связи можно начать скачивание не с 0, а с того места, где вы остановились. Это очень удобно.

Сколько стоят услуги BBS?

Многим читателям этот вопрос покажется очень смешным. На самом же деле, некоторые пользователи считают, что BBS — это платный ресурс. Нет, поверьте, все бесплатно. Станции держатся только на рвении или фанатизме их создателей. И если раз в год вы угостите SysOp'а чаем или кофе, я думаю, он не расстроится.

Какой компьютер нужен для работы с BBS?

Компьютер для работы с BBS может быть любым! От 8088 до Pentium 4. В этом одно из преимуществ BBS. Пользователь, скажем, 80386-го компьютера и модема на 28800 всегда найдет на станции поддержку и понимание.

Все приведенные выше вопросы мне встречались в разное время и в разных местах. Я лишь попытался их обобщить и выставить на всеобщее обозрение. Тем не менее, продолжаю считать, что вопросы по работе BBS заслуживают отдельной публикации.

4. Идеологическая обработка

Не секрет, что у всего, созданного руками человека, есть своя идея. Есть она и у BBS. Вопрос только, в чем эта идея выражена. Бродя по бескрайним просторам сети, я наткнулся на один занимательный форум, в котором обсуждалась нужность/ненужность BBS. Люди спорили, доказывали друг другу "плюсы" и "минусы", оскорблялись. Мне это показалось довольно забавным, поскольку я не понял сути проблемы. Что может быть плохого в том, что люди общаются? Какая разница, как? Лишь бы им, то есть вам, дорогие читатели, это нравилось. Ведь вам решать, имеет тот или иной способ общения право на существование. Я сторонник той мысли, что ничего навязывать нельзя, все придет или уйдет со временем. Но если есть возможность задержать хорошие воспоминания, завести новых друзей, в конце концов, просто пообщаться, то почему бы и нет? И способ отнюдь не важен.

BBS — для друзей. Те, кто к нам приходит — это не малолетки из чатов, это продвинутые пользователи или просто любознательные люди, которые хотят узнать больше.

Посетители BBS живут с вами в одном городе или районе. В этом большое преимущество перед Internet. Вы можете встретиться с вашими гостями и просто по-человечески пообщаться.

BBS доступны каждому — и пользователям старенького 386-го, и владельцам Pentium 4.

Предполагаю, что некоторые, из читающих эти строки уже недовольны, мол, кто ходит на BBS... А вы никогда не задавались вопросом, почему BBS, расписанные на всевозможных форумах, никто не посещает? Нет? Да просто их не там рекламируют. Настоящий контингент читает газеты. Вот там, именно там нужно помещать множество объявлений! И конечно же, важно время работы станции. Вы думаете, много найдется желающих посещать BBS с 23.00 до 6.00 утра? Нормальная BBS должна работать в удобное время, скажем, с 17.00 до 23.00. Это тоже важный фактор. Я также надеюсь, что те системные операторы, которые до сих пор содержат BBS, позволят закачивать файлы не только избранным, за пиво, а всем желающим. Чтобы новым пользователям не было досадно и обидно, что там нет частички и их труда.

Интернет-хирургия

В.КУЦ,
E-mail: victor_kootz@mail.ru

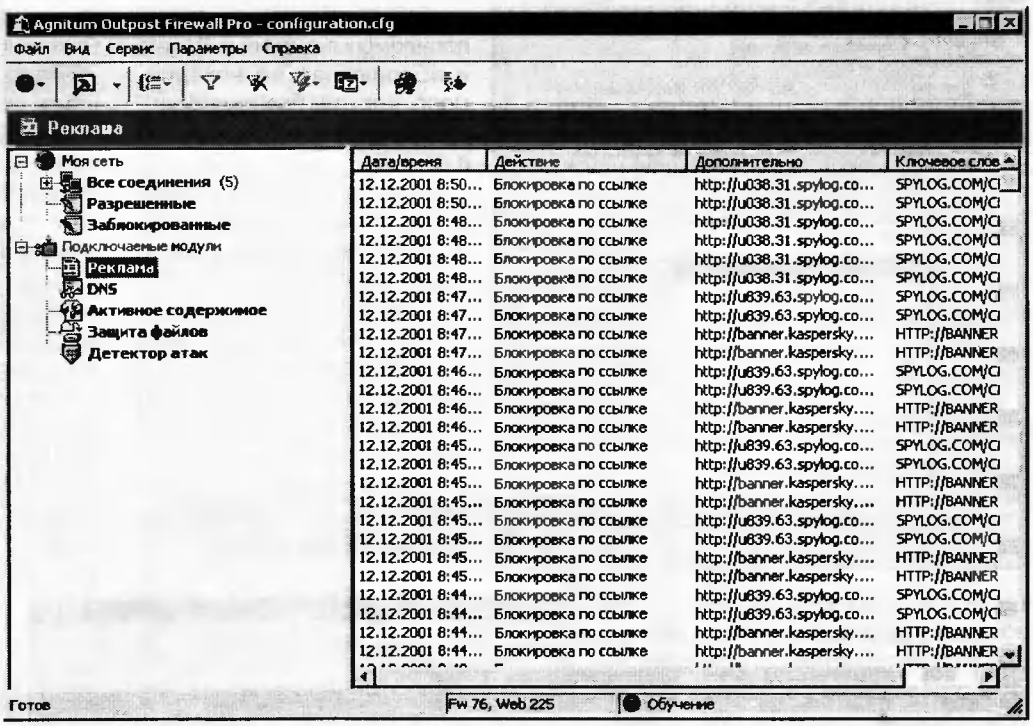
К сожалению, давно прошли те благословенные времена, когда Интернет был средством общения между людьми, инструментом, существенно облегчающим поиск и получение самой различной информации. Теперь же Всемирная Сеть превратилась в одно из средств массовой информации, со всеми их достоинствами и недостатками. Причем, в полном соответствии с законами Мэрфи, на первый план выходят некоторые, далеко не самые симпатичные черты, заимствованные Интернетом у своих предшественников. Да-да, я все о той же рекламе, с каждым годом отъедающей все большую часть любой, мало-мальски заметной странички в Сети, а на тех сайтах, которые завоевали хоть какую-то популярность, от всяких-разных баннеров просто в глазах рябит. И дело здесь даже не в том, что каждый раз при посещении такой странички приходится впустую гонять по нашим довольно "хилым" телефонным линиям десятки, если не сотни килобайтов абсолютно ненужной информации. Наверное, многие из вас очень часто подмечали, что сама страничка, даже прилично нагруженная графикой, грузится в браузер относительно быстро, но вот когда дело доходит до баннеров — индикатор загрузки резко притормаживает — серверы рекламных служб почти всегда сильно перегружены, поэтому приходится ждать этот несчастный баннер гораздо дольше, чем всю остальную страничку. Я уже и не заикаюсь о применении новейших технологий в этих человеконенавистнических целях, да и как по-другому назвать использование анимированных flash-файлов, с тупостью автомата бесконечно прокручивающих один и тот же, обычно крайне примитивный сюжет, выполненный в диссонирующе-ярких тонах (видимо,

чтобы лучше бросался в глаза и глубже "врезался" в подкорку), да еще и возможность остановить этот бесконечный калейдоскоп, мельтешащий перед глазами, чаще всего просто заблокирована остроумными web-мастерами.

Как любое действие непременно рождает ответное противодействие (никуда не деться от законов физики), так и явный перебор с рекламой на страничках в Сети привел к появлению особого вида программ, занимающихся очисткой просматриваемых страничек от надоедливой рекламы. Но перед тем как приступить к рассмотрению некоторых из них, хочу обратить внимание читателя на те внутренние механизмы, с помощью которых можно успешно бороться с баннерами, и которые есть в любой из версий Windows. Этот "механизм" представляет собой всего-навсего обычный текстовый файл с именем HOSTS и без расширения. Он располагается в системной папке Windows 9x (например, C:\Windows\hosts) или в папке C:\Winnt\System32\drivers\etc в Windows 2000/XP. Справку с примерами использования этого файла можно найти там же — в файле HOSTS.SAM. Файл HOSTS является своеобразным локальным DNS-сервером, в котором содержатся определенные сетевые адреса и поставлены их числовые значения. Поэтому, если мы привяжем адреса известных баннерных служб к адресу 127.0.0.1, который, как известно, принадлежит локальной машине, то при обращении по любому из адресов из этого списка браузер будет пытаться загрузиться с ПК, не обращаясь в Сеть, и баннер отображаться не будет. К сожалению, метод с файлом HOSTS работает далеко не всегда. Например, в тех случаях, когда локальная машина выходит в Интернет через прокси-сервер, HOSTS попросту игнорируется.

Да, самые простые способы борьбы далеко не всегда являются и самыми эффективными, поэтому мы рассмотрим несколько интересных программ, имеющих развитые средства борьбы с интернет-рекламой.

Рис. 1



Outpost Firewall 1.0

Первой из них по праву является персональный брандмауэр Outpost Firewall (http://outpost.agnitum.com) российской компании Agnitum, представляющий собой комплексное средство по защите персонального компьютера от всех видов сетевых воздействий. В его основе лежит эффективный межсетевой экран (firewall), дополненный средствами для борьбы с сетевой рекламой. Outpost Firewall достаточно прост в освоении, и был создан для использования людьми, не имеющими специальных знаний в области компьютерной безопасности. Он может использоваться сразу же после установки, без какой-либо предварительной настройки. Возможности Outpost Firewall очень широки. Кроме своей основной функции firewall-а, он умеет кэшировать адреса DNS, защищать файлы, получаемые по E-mail, ограничивать доступ к определенным web-страницам (чаще всего, сомнительного содержания), а также имеет достаточно развитые средства для борьбы с интернет-рекламой. Под последним подразумевается:

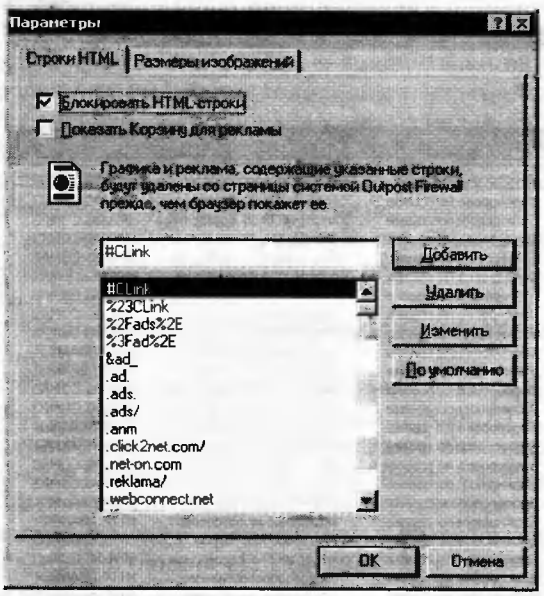
- фильтрация рекламных баннеров, причем не только по их web-адресам, но и по размеру изображения;
- блокировка всплывающих окон (pop-up) в web-страницах;
- ограничение или запрет на использование отдельных программных компонент, встроенных в web-страницы, таких как Java-апплеты и программы на языке JavaScript, ActiveX и т.п.;
- ограничение или запрет на использование cookie.

Перспективность данной программы определяет модульный принцип ее построения, что позволяет встраивать в Outpost новые защитные модули (в том числе и от сторонних разработчиков). А это дает возможность собирать защитную стенку из "кирпичиков"-модулей в конфигурации, наиболее подходящей для насущных потребностей конкретного пользователя, да и облегчает процесс модернизации и наращивания возможностей программы в будущем.

Главное окно Outpost (**рис.1**) выполнено в полном соответствии со всеми канонами построения внешнего вида окошек Windows и предназначено для визуального контроля за работой компьютера в сети, а также для изменения многочисленных настроек программы. В верхней своей части оно содержит меню и панель инструментов, в нижней — строку состояния. Основное рабочее поле занимают две информационные панели, левая из которых содержит иерархический список объектов и модулей, подробная информация о которых отображается в правой.

Для значительного ускорения загрузки содержимого сайтов из Сети, а

Рис. 2



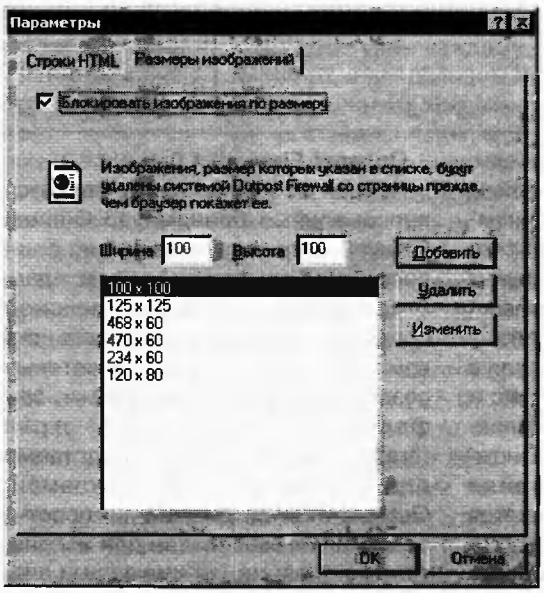
также для сохранения душевного равновесия издерганных отечественным dialup'ом интернетчиков, можно запретить отображение рекламных баннеров на просматриваемых Web-страницах (**рис.2**). Поскольку адреса большинства баннерных служб хорошо известны, то можно просто исключить из HTML-кода загружающейся Web-страницы теги, в которых присутствуют ссылки, указывающие на такие службы. Сразу после установки Outpost содержит большой список адресов рекламных служб, но всегда можно добавить любую ссылку или HTML-строку из Web-страницы, загруженной в данный момент в браузер. Для более оперативного внесения нежелательного HTML-кода в этот список удобно воспользоваться плавающей Корзиной, которая располагается поверх всех открытых окон, поэтому рекламный баннер можно просто перетащить в нее мышкой, и все — дело сделано.

Существует и другой путь выявления рекламных баннеров, основанный на том, что большинство из них имеет строго фиксированный размер графического изображения (**рис.3**). Как и в предыдущем случае, можно не только корректировать заданные по умолчанию размеры удаляемых изображений, но и задавать собственные. Но здесь надо быть осторожным и не перегнуть палку — например, на многих страничках в Сети размер кнопок часто бывает 88 x 31 пункт, что при использовании установок по умолчанию приведет к их повальному удалению как злостных баннеров.

К не самым лучшим чертам Outpost' относится, помимо некоторой тяжеловесности (объясняемой, впрочем, многочисленностью выполняемых ею функций), еще и абсолютная ее нетерпимость к другим брандмауэрам, установленным вместе с ней в системе.

Интерфейс Outpost Firewall поддерживает русский или английский языки, совместим со всеми версиями операционной системы Windows 95/98/ME/NT/2000/XP. Программа Outpost Firewall выпускается в двух вариантах — бесплатный Outpost Firewall и его чуть более "навороченная", но уже shareware-версия Outpost Firewall Pro. Ничего не поделаешь, рыночные реалии.

Рис. 3



Proxomitron Naoko 4.4

При всех своих достоинствах, Outpost Firewall, как многофункциональный продукт, зачастую уступает аналогичным узкоспециализированным решениям. В частности, в деле борьбы с интернет-рекламой практически неограниченными возможностями обладает программа Proxomitron (http://proxomitron.cjb.net/). Принцип ее работы прост, как правда — являясь своеобразным HTTP-прокси-сервером, Proxomitron располагается между web-браузером и Интернетом. Таким образом, браузер посылает HTTP-запросы не удаленному HTTP-серверу, а программе Proxomitron, которая посылает запрос удаленному HTTP-серверу. Сервер отдает файлы программе Proxomitron, она их фильтрует в соответствии с настройками и возвращает отфильтрованные страницы браузеру. Причем все фильтры и правила, в соответствии с которыми Proxomitron осуществляет "препарацию" страниц, написаны на простейшем скриптовом языке, базирующемся на стандартном HTML, и доступны не только для просмотра, но и для редактирования. Также имеется возможность добавления новых фильтров, написанных самим пользователем. Благодаря этому программа отлично работает с любым браузером, будь то Internet Explorer, Opera, Mozilla, Netscape или даже экзотический Lynx. Proxomitron может успешно работать со многими другими типами программ, которые для своей работы используют протокол HTTP, например, оффлайновыми браузерами. Одна из приятных особенностей Proxomitron заключается в том, что она не требует инсталляции, что весьма благотворно сказывается на устойчивости склонных к "падучести" Windows.

После распаковки архива с программой в выбранную папку, необходимо сконфигурировать свой браузер для работы с Proxomitron'ом. Для этого в параметрах настройки удаленного доступа сначала необходимо разрешить использование прокси для HTTP, после этого задать его адрес — **localhost** и номер порта — **8080** (**рис.4**). Все, Proxomitron готов к работе.

Рис. 4

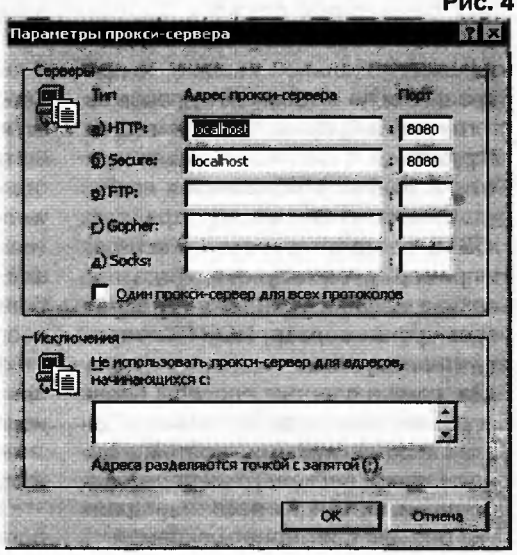


Рис. 5

Главное рабочее окно программы (**рис.5**) очень простое, в левой его части располагается секция Active Filters с выключателями групп фильтров, а в правой — секция Edit Filters с кнопками доступа к соответствующим спискам фильтров и некоторым вспомогательным функциям. Proxomitron имеет несколько категорий фильтров, причем каждая из них может быть быстро заблокирована снятием соответствующего флажка в секции Active Filters:

- **Web-фильтры** — влияют на загрузку web-страниц;
- **Фильтры исходящих заголовков** — влияют на запросы, посылаемые от браузера к серверу;
- **Фильтры входящих заголовков** влияют на ответы, посылаемые от сервера к браузеру;
- **Фильтр, "замораживающий" GIF-анимацию** — загружает только первый кадр анимированного изображения в формате GIF.

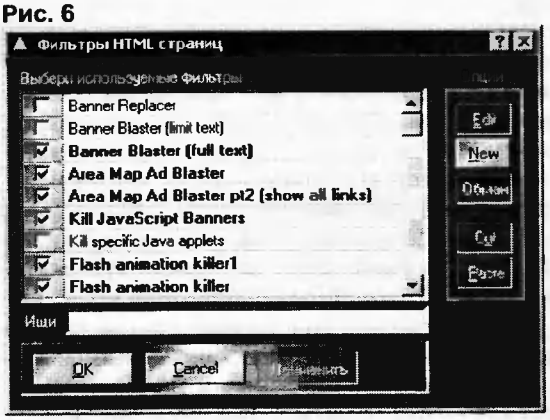
Всю информацию о настройках фильтров Proxomitron сохраняет в файлах конфигурации (имеющих расширение .cfg). Файлы конфигурации — это простые текстовые файлы, и они могут быть вручную отредактированы в любом текстовом редакторе.

Внешний вид рабочих окошек программы, честно говоря, по умолчанию довольно аляповатых, лучше сразу сменить, нажав кнопку Config. На вкладке Visuals можно подобрать более "спокойную" схему, такую как, например, отделка под благородное дерево в стиле Proxomitron Retro. Естественно, для любителей все и вся изменять и украшать нет никаких ограничений на использование собственных bmp-файлов в качестве фоновых рисунков.

Но оставим в покое раскраски и перейдем к основному рабочему инструменту программы — фильтрам. Наиболее важный из них — фильтр web-страниц, в который включено большое количество самостоятельных фильтров, каждый их которых выполняет свою специфическую задачу.

Получить доступ к перечню этих фильтров можно, нажав на кнопку Web page в главном окне. Каждый фильтр, входящий в перечень, можно индивидуально включить или заблокировать, устанавливая или снимая флажок рядом с ним (**рис.6**). В дистрибутив Proxomitron включен внушительный перечень самых разнообразных фильтров, используя которые, можно решить большую часть задач, поставленных перед программой. Для начинающих этого набора вполне может хватить, но в дальнейшем, по мере освоения возможностей программы, можно будет создавать новые фильтры самостоятельно, или изменять существующие, или, для самых ленивых, регулярно пополнять их из коллекции, имеющейся на сайте разработчика (http://proxomitron.cjb.net/). Эта коллекция постоянно обновляется и разрастается как вширь, так и вглубь за счет творчества большого количества энтузиастов. Для иллюстрации сказанного хочется отметить несколько наиболее интересных фильтров из тех, которые включены в состав дистрибутива:

Рис. 6



- **Banner Blaster** — наверное, это самый главный фильтр, удаляющий большую часть рекламных баннеров. Он удаляет картинки рекламных баннеров и на их место помещает текстовую ссылку, воспроизводящую оригинальный текст, содержащийся в теге ALT. Это достаточно эффективный прием, ведь суть рекламного объявления остается доступной пользователю, и, в то же время, меньше страдает остальное содержимое страницы (в тех случаях, когда полезные картинки на странице совпадают со стандартными размерами рекламного баннера или ведут на адреса, напоминающие адреса баннерных служб);
- **Banner Replacer** — еще один фильтр баннеров, в отличие от предыдущего, заменяет баннеры на рамку с прозрачной картинкой в формате GIF. Это позволяет сохранить оригинальную разметку страницы без изменений;
- **Flash animation killer** — преобразует Flash-анимацию в ссылки, ведущие на нее;
- **Counter Killer** — блокирует большинство известных счетчиков, которые часто бывают очень медленными из-за долгого ожидания обновления;
- **Webpage Background Killer** — удаляет общий фон из страницы, но оставляет другие фоны (например, в таблицах), тогда как другой фильтр — **Kill All Backgrounds (even tables)** — удаляет фоновое изображение отовсюду, даже из таблиц;
- **Sound Silencer** — блокирует возможность воспроизведения звуков на странице;
- **Blink Buster (Blink to Bold)** — преобразует мигающие надписи в полужирный текст;
- **Onload unloader** — отключает некоторые автоматически стартующие скрипты;
- **OnUnload unloader** — является чрезвычайно полезным фильтром, так как блокирует запуск очень вредного скрипта, открывающего рекламные окна, после того как вы покидаете просматриваемую страницу;
- **Kill All pop-up windows** — полностью отключает команду Java "window.open", что блокирует все всплывающие окна;
- **Restore pop-up windows after page loads** — дополняет предыдущий фильтр, разрешая всплывающие окна после полной загрузки страницы. Ведь больше всего всплывающие окна досаждают при входе и выходе со странички. В процессе же ее просмотра всплывающие окна обычно служат для получения какой-либо дополнительной полезной информации;
- **Kill add-on JavaScripts** — убирает все JavaScript в нижней части страницы. Обычно такие скрипты добавляют провайдерские компании, предоставляющие бесплатное место под

сайты. Они используют их для вывода своей рекламы или логотипов. С этим фильтром посещение таких страниц станет более приятным;

- Stop status bar scrollers — отключает скрипты, выводящие текст в строку статуса (бегущую строку);

- Disable JavaScript cookies — запрещает отправлять или принимать информацию cookie. В необходимых случаях вы можете разрешить индивидуально для каждого сайта получать или отсылать cookies;

- Hide Browser's Referrer from JS. Referrer — один из видов персональных данных о пользователе, пересылаемых в заголовках. По нему веб-мастер может узнать адрес (в том числе и на жестком диске), с которого вы перешли на его страницу. Фильтр используется для блокирования JavaScript'а, использующегося для получения таких данных;

- Kill window.external methods — блокирует команды JavaScript, позволяющие выполнять различные операции (любая такая операция выглядит крайне подозрительно) за пределами страницы;

- Stop OnMouseOver events — останавливает действия, производимые при наведении мыши на ссылку или картинку;

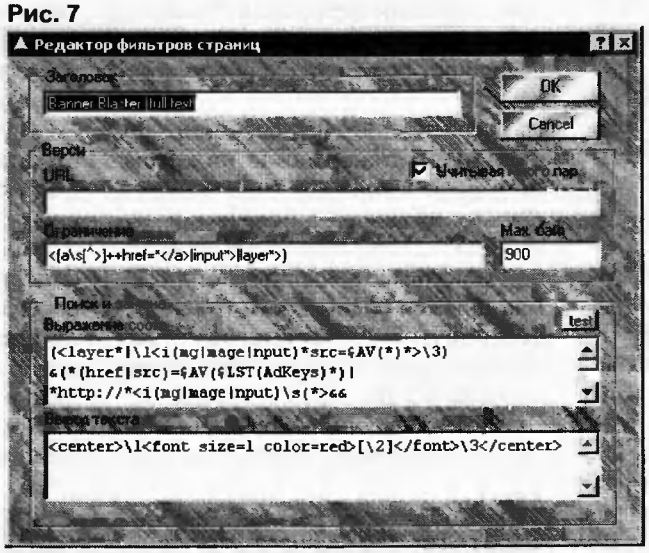
- Kill Style Sheets — отключает внешние таблицы стилей CSS, при этом индивидуальные теги стилей непосредственно на странице остаются неизменными;

- Allow for frame resizing — позволяет изменять размеры фреймов произвольным образом.

Это только небольшая часть из всего многообразия фильтров, включенных в состав программы. Но использование готовых фильтров далеко не всегда может удовлетворить все потребности пользователя, поэтому особую прелесть придают Proxomitron простота и наглядность редактирования имеющихся или создания новых web-фильтров или фильтров заголовков HTTP.

Этот, без сомнения, творческий процесс производится в специальном окне (**рис.7**), где можно изменять правила соответствия, которые позволяют Proxomitron перезаписывать web-страницы. Говоря проще, правила соответствия работают примерно

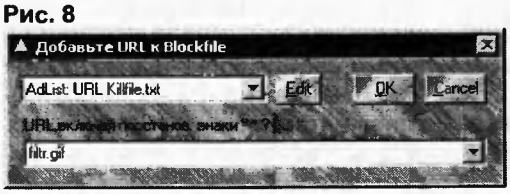
Рис. 7



аналогично функции поиска и замены обычного текстового процессора. Любой текст, соответствующий некоторому определенному выражению, будет заменен другим, заранее введенным текстом. Причем язык сравнения текстов, используемый в Proxomitron, использует всего несколько операторов и очень сильно напоминает подстановочные символы, применяемые в DOS или UNIX, что позволяет быстро и без особых трудозатрат освоить его.

Но, кроме использования готовых фильтров и написания своих собственных, есть еще один довольно простой способ настройки Proxomitron под нужды конкретного пользователя. Я имею в виду запоминание программой определенных URL, вносимых непосредственно в файлы конфигурации (**рис.8**). Используя контекстное меню ярлыка программы в систем-

Рис. 8



ном трее, можно внести URL сайта в список тех, с которыми разрешено обмениваться cookies; блокировать изображения как полностью на указанных сайтах, так и с конкретных адресов; а также указывать URL, на которых работа Proxomitron полностью блокируется.

Программа Proxomitron распространяется совершенно бесплатно, работает под управлением любой из версий Windows, начиная с древней 95-й и вплоть до XP. Версию Proxomitron на русском языке, а также большое количество дополнительных материалов о программе можно найти на сайте http://proxomitron.da.ru/.

Ad Muncher 4.5

Несмотря на то что программа Proxomitron является наиболее эффективным решением для борьбы с сетевой рекламой, все-таки она имеет не так много поклонников, как того заслуживает. И в первую очередь это связано с определенной сложностью программы, ведь даже для того чтобы элементарно освоить ее, требуется приличный уровень компьютерных знаний, а уж писать собственные фильтры — под силу лишь наиболее продвинутым пользователям. А что же остается нам, простым смертным? А остается нам Ad-Muncher — тоже довольно эффективная программа для борьбы с сетевой рекламой, отличающаяся скромными размерами (размер дистрибутива — всего 150 Кб) и гораздо более легкая в использовании, чем Proxomitron. Программа Ad-Muncher (http://www.admuncher.com) умеет блокировать рекламные баннеры по их адресам или размерам (размеры блокируемых изображений задаются в диапазоне, а не устанавливаются точно, как в Outpost Firewall), а на их место помещать произвольный текст. Кроме баннеров, блокируются всплывающие окна и всевозможные счетчики, включая новомодные "скрытые" (размером 1 x 1). Также Ad-Muncher умеет скрывать конфиденциальные данные о компьютере: версию ОС и браузера, URL-адрес, с которого вы пришли на данный сайт. Довольно интересная функция IP Scramble позволяет скрыть от излишне любопытных глаз ваш IP-адрес, заменяя его адресом произвольного прокси-сервера. Хотя такая "секьюрити" для большинства нормальных пользователей попахивает паранойей, однако лучше иметь, но не использовать, чтобы потом, когда вдруг "петух жареный..." Ну, сами понимаете...

Сразу после запуска (а запускаться Ad-Muncher может вместе с системой) в трее появляется симпатичная иконка в виде коровьей мордашки. Хотя настройки программы по умолчанию достаточно толковые, борьба с баннерами — дело сугубо индивидуальное. Поэтому вам, в процессе серфинга, не раз придется их корректировать. Сде-

лать это очень просто — нужно всего лишь кликнуть правой кнопкой мыши по значку и из выпавшего меню выбрать Configure (к сожалению, программа не русифицирована).

В Ad-Muncher есть возможность не только настроить собственные фильтры, введя ключевые слова и символы, по которым выявляется рекламная вставка на web-станицах, но и импортировать их, или экспортировать весь набор фильтров, для того чтобы поделиться с друзьями или сохранить настройки при переустановке системы (**рис.9**).

Почти в каждой программе есть пункт About (или "О программе"), где авторы программы обычно указывают сведения о номере версии своих творений, не забывают они упомянуть и о себе, любимых. В Ad-Muncher этот пункт замечателен тем, что, кроме всего прочего, можно узнать количество "убитых" баннеров и их общий "вес" за все время работы программы (**рис.10**). Очень познавательно!

Заключение

Напоследок мне хотелось бы еще раз остановиться на проблеме взаимоотношения посетителей сайтов с баннерами на страничках. Прежде всего я хочу обратиться к web-мастерам — к тем людям, которые создают и эти самые странички, и раздражающие всех баннеры. Очень часто можно услышать их возмущенные голоса, что недопустимо при просмотре странички удалять рекламные баннеры, ведь это та плата, которой каждый посетитель должен оплатить труд их создателей. Действительно, против такого веского аргумента, что достойный труд должен быть достойно оплачен, трудно что-либо возразить, ведь сделать качественную страничку — удовольствие не из дешевых (а на "шедевры", сляпанные дилетантами за пять минут, я думаю, все насмотрелись досыта). Но, говоря словами известного мультипликационного персонажа — "ребята, давайте жить дружно!". И понимая обиды дизайнеров, я порой совершенно не могу их понять — ведь не кто иной, как те же самые дизайнеры, вкладывающие душу в создание настоящих шедевров, плодят легионы уродцев-баннеров, место которым исключительно в... баннерорезке. Ведь абсолютное большинство интернетчиков крушат баннеры совсем не из своей природной вредности; не потому, что хотят отобрать кусок хлеба у web-дизайнеров, а потому, что они (баннеры, а не дизайнеры) их не устраивают, не способствуют, мягко говоря, комфортному просмотру странички. И если уж на то пошло, то почему бы и рекламным службам, размещающим свои баннеры на сайтах, не проявить хоть немного элементарного уважения к тем, кто является объектом их деятельности? Как пример такого уважения к потребителю интернет-рекламы лично мне очень симпатичны начинания отдельных рекламных служб, в последнее время переориентирующихся на использование текстовых баннеров, которые имеют мизерный размер, а значит, автоматически исчезает одна из причин, порождающих войну с баннерами. Следуя такой тенденции, всегда можно найти разумный компромисс, одинаково устраивающий и рекламодателей, и авторов сайтов, и их посетителей. Да и как "не все йогурты одинаково полезны", так и не вся реклама одинаково раздражает. Лично я, например, ничего не имею против тех баннеров (особенно текстовых), которые непосредственно относятся к теме странички. Даже больше того, нужная ссылка в нужном месте бывает порой просто необходима. Причем это будет вдвойне верно, если баннеры и по своему внешнему виду не особенно диссонируют с общим дизайном web-странички. Но, с другой стороны, все вы знаете, сколько ссылок на примитивно-развлекательные или даже просто порнографические ресурсы можно встретить на просторах Сети, причем в самых неожиданных местах. Так что, как волки выполняют важную функцию санитаров леса, не давая слишком расплодиться остальной живности и поддерживая экологический баланс, так и баннерорезки выполняют не менее важную роль, ограничивая аппетиты рекламных служб и защищая нас, посетителей сайтов, от рекламного беспредела.

Рис. 9

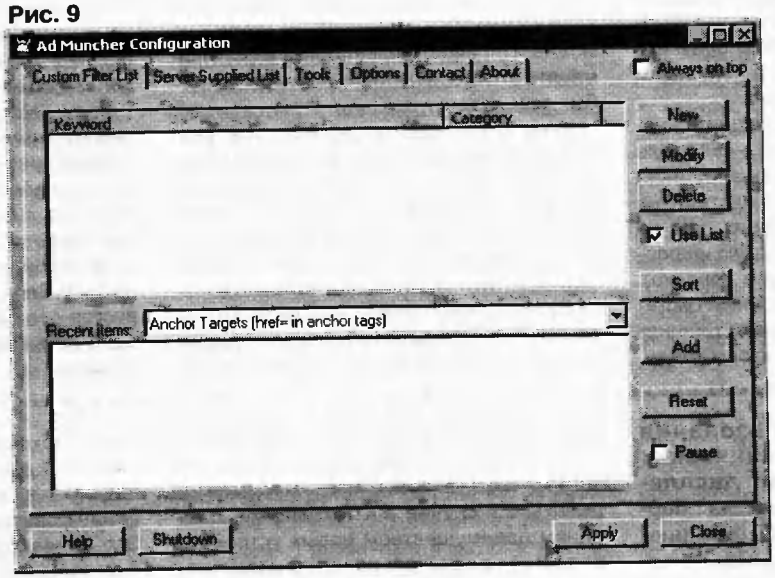
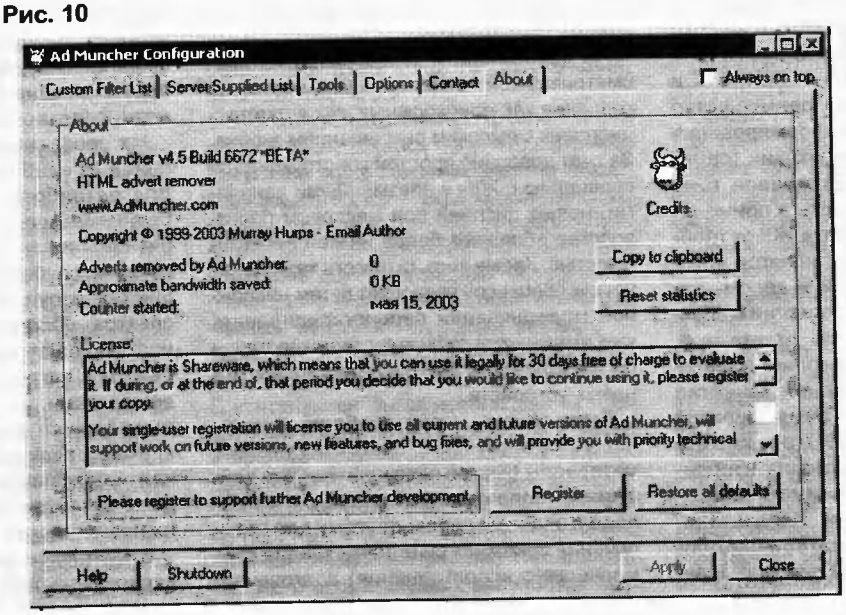


Рис. 10



ЛИСТАЯ СТРАНИЦЫ

Тем, кто все еще использует **ОС Windows 95 или Windows 98**, будет полезно ознакомиться со статьей *А.Егорова* "Организация хранения файлов в Windows 9x/Me" (**Мир ПК, №4/2003, С.111...114**).

Загрузка ОС начинается с помощью загрузочной записи. Она находится в пераом секторе системного диска (при использовании FAT32 загрузочная запись размещается в нескольких секторах). Если не существует загрузочной записи, то она создается и заполняется программой FORMAT. Но если она уже есть, то утилита FORMAT будет использоаать ее информацию.

С загрузочной записью вручную обычно работать не приходится, хотя быаают и исключения. Рекомендуется сохранить и скопироаать сектор загрузочной записи на дискету. В случае краха системы организации хранения информации на диске, этот файл поможет сохранить данные.

Еще один соает: если нужно провести полное форматироаание диска, то сначала с помощью утилиты DiskEdit обнулите сектор загрузочной записи, а уже затем запускайте программу FORMAT.

Следующая ступень организации хранения данных на жестком диске — FAT (File Allocation Table — таблица размещения файлов). Она, как прааило, представлена в даух экземплярах, сладующих друг за другом и содержащих одинакоаую информацию (при условии, что все в порядке). Нужно помнить, что операционные системы DOS и Windows не различают цилиндры, головки и физические секторы диска, который для этих ОС предстает в виде непрерывной последовательности логических секторов или кластеров (групп смежных секторов). Система MS-DOS до версии 7.0 включительно и первые версии Windows могли распознать 65536 логических блоков на диске. Начиная с Windows 95 OSR2, появилась возможность использоаать для нумерации логических элементоа на диске 32-разрядные данные, а значит, число адресуемых элементов теоретически возросло до 4294967295. Для DOS до версии 3.x аключительно размер логического блока равнялся 512 байтам, т.е. размеру физического сектора. Наибольшая емкость диска, с которым могла работать DOS, составляла 32 Мб. Для работы с винчестерами большего размера логические секторы объединялись в группы — так называемые кластеры. Под этим термином понимаатся группа таких смежных секторов, которым соответствует одно значение адреса. Если увеличить размер кластера, то можно будет работать с большими разделами, остаааясь а рамках 16-разрядной адресации. После перехода к 32-разрядной адресации стало возможным (правда, лишь теоретически) применять кластеры любого размера на любых разделах.

Идея FAT очень проста. Все пространство логического диска разбиаается на кластеры, размер которых зависит от емкости диска. Затам составляется таблица, каждому элементу которой соответствует элемент дискового пространства — кластер. Эта таблица линейна — индекс ячайки соответствует номеру кластера. Что касается скорости работы и "потерь" дискового пространства от незаполненных кластеров, то здесь действует принцип: выигрываешь в скорости — проигрываешь на не полностью занятых кластерах, т.е. чем больше размер кластера, тем меньше величина FAT и выше скорость работы с ней.

Корнеаой каталог — это следующая за аторой FAT область на логическом диске, являющаяся последовательностью 32-байтоаых записей. Каждая из них может быть каталогом, именем файла (в том числе и длинным), а также меткой тома. В FAT16 размер корнеаого каталога зафиксирован, а FAT32 — нет, его размер растет по мере необходимости. 32-байтовое имя файла содержит в закодированном аиде привычные имя и расширениа файла, его размер, дату и время создания, атрибуты и номер первого кластера. Программы, работающие с файлом, раскодируют и показыаают имя, дату и т.д.

Основные атрибуты файлоа: "только чтение", "скрытый", "системный", "метка тома", "подкаталог", "архивный". Атрибут "подкаталог" сообщает ОС, что данный файл является подкаталогом, атрибут "метка тома" — что имя является меткой тома. Остальные атрибуты относятся к файлам данных и указыаают, как система должна с ними работать.

Здесь есть один нюанс. Если указать недопустимую комбинацию атрибутов, например, "метка тома"+"каталог"+"скрытый", то DOS перестанет аидеть данную запись, а вместе с ней и тот объект, на который она указывает. Это свойстао используется а Windows 9x для хранения длинных имен файлов. Причем длинное имя кодируется особым сочетанием атрибутов, а само длинное имя хранится в кодироаке Unicode.

Перспективам использования светоизлучающих органических материалов для создания дисплеев посвящена статья *С.Асмакова* "Мерная поступь OLED-дисплеев" (**КомпьютерПресс, №7/2003, С.123...128**).

Дисплеи на светоизлучающих органических материалах обладают целым рядом достоинств по сраанению с широко используемыми в настоящее аремя дисплейными технологиями — ЭЛТ, ЖК и плазменной. Главное их преимуществао — это использование для формироаания изображения люминесцирующих (самоизлучающих) аеществ. Благодаря тому что отпадает необходимость в применении лампы подсветки (как в ЖК-устройствах), такие дисплеи отличаются чрезвычайно малой толщиной и весом, потребляют меньше электроэнергии и практически не аыделяют тепла. Кроме того, подобная конструкция позволяет добиться значительного улучшения качества изображения, обеспечить очень широкий угол обзора (не менеа 160°), а также повысить яркость и контрастность изображения до уровня, недостижимого для современной ЖК-технологии. Использование люминесцирующих материалов позволяет сделать апертуру пиксела практически равной 1 (то есть эффективная площадь пиксела равна его полной площади), чего, а принципе, неаозможно добиться в случае ЖК-технологии. Еще одним преимуществаом ноаых устройста является чрезвычайно малое аремя реакции (не превышающее единиц миллисекунд), причем практически не зависящее от температуры.

На современном этапе развития рассматриваемой технологии уже аозможно создание как монохромных, так и цветных дисплаев с высоким разрешением экрана. За счет доаольно простой конструкции (по сравнению с ЖК- и плазменными панелями) новые дисплеи при массовом производстае обладают более низкой себестоимостью. Кроме того, схожесть технологических процессов позволяет путем несложной модернизации перепрофилировать уже имеющиеся производственные линии по изготовлению ЖК-дисплееа на выпуск новых устройста.

Есть, однако, и ряд проблем. В частности, органические саетоизлучающие материалы быстро разрушаются под действием содержащегося в воздухе кислорода и водяных паров, поэтому для обеспечения приемлемой (с точки зрения коммерческого использования) долгоаечности необходима полная герметизация "начинки" дисплейной панели. Не менее актуальная проблема — деградация (старение) светоизлучающих материалоа. Это проявляется а уменьшении их эффективности (падении яркости при заданном напряжении питания) и изменении спектральных характеристик. Современные светоизлучающие материалы способны излучать свет с очень аысокой яркостью, однако их старение прямо пропорционально яркости излучения.

На протяжении длительного времени одной из наиболае актуальных проблем для разработчикоа цаетных дисплееа было повышение чистоты самого "капризного" из первичных цаетоа RGB-модели — голубого. В силу ряда причин аещества, излучающие сает голубой части спектра, обладают меньшей эффективностью, дают наименее чистый цвет и быстрее деградируют по сравнению с веществаами, излучающими сает красного и зеленого диапазонов. К настоящему времани ведущие разработчики смогли найти пути решения данной проблемы и даже достигли определенных успехоа в их практической реализации.

В настоящее время работы по созданию дисплееа на базе самолюминесцирующих

материалов ведутся параллельно по нескольким направлениям. Основное их различие заключается в структуре используемых светоизлучающих материалов: в случае OLED и SMOLED — это молекулярные органические вещества, а в случае LEP (PLED, PolyLED) — полимеры.

Аналогично ЖК-панелям, в зависимости от способа активации ячеек светоизлучающих элементов, OLED- и LEP-дисплеи подразделяются на активно-матричные и пассивно-матричные. В пассивно-матричном дисплее активация нужной ячейки экрана производится подачей напряжения на соответствующие анод и катод. В дисплее с активной матрицей управление работой ячеек осуществляется при помощи интегрированных электронных компонентов, в частности, тонкопленочных транзисторов. Активно-матричная технология открывает возможность создания цветных дисплеев, пригодных для полноценного воспроизведения видео. Использование активной матрицы позволяет достичь высокой разрешающей способности, малого времени отклика и низкого уровня энергопотребления.

Пионером в разработке приборов на базе светоизлучающих молекулярных органических материалов (Organic Light Emitting Diode, OLED — светодиоды на основе органических материалов) можно считать компанию Eastman Kodak. Отправной точкой в истории OLED-технологии стала публикация в 1987 г. научной статьи о свойствах органических светоизлучающих материалов. Структура простейшего элемента OLED-дисплея, разработанного специалистами Kodak, представляет собой несколько тонких слоев органического вещества, заключенных между пересекающимися частями расположенных перпендикулярно друг другу прозрачного анода и металлического катода. При подаче напряжения (порядка нескольких вольт) на пересекающиеся на данной ячейке анод и катод из соответствующих слоев испускаются положительные и отрицательные заряды, рекомбинация которых в излучающем слое вызывает эффект электролюминесценции. Вещества, из которых состоят органические слои, а также материалы для анодов и катодов подобраны таким образом, чтобы обеспечить максимальный световой поток.

Специалисты Kodak работают над двумя типами OLED-дисплеев — с активной и с пассивной матрицей. Важным достижением разработчиков Kodak в области пассивно-матричных дисплеев стало создание уникального способа изготовления этих устройств. Суть его заключается в использовании специальной "ребристой" структуры, сформированной на фигурных прозрачных анодах. Эта структура воспроизводит рисунок элементов дисплейной панели, что позволяет наносить слои органического материала и металла методом простого напыления без последующей обработки. Основное достоинство описанного процесса — максимальная простота и технологичность, что дает возможность применять его при изготовлении дисплеев большого размера, а также значительно сокращает производственный цикл.

Что касается создания цветных OLED-дисплеев, то в большинстве случаев предпочтение отдается конструкциям с активной матрицей. В настоящее время наиболее актуальными проблемами, стоящими перед разработчиками цветных OLED-дисплеев, являются достижение более широкого цветового охвата, стабильности спектральных характеристик и увеличения срока службы излучающих материалов.

В 1989 г. ученым двух научных центров Кембриджского университета удалось синтезировать полифениленвинилен. В ходе экспериментов выяснилось, что под воздействием электрического поля это вещество начинает излучать свет желто-зеленого цвета.

Первые образцы светоизлучающих полимеров (Light Emitting Polymer, LEP) обладали очень низкой эффективностью (порядка 0,01%), однако по мере совершенствования технологии удалось достичь показателя эффективности в 5%, что соответствует аналогичному параметру полупроводниковых светодиодов.

Дальнейшие исследования позволили синтезировать полимеры, излучающие свет самых различных цветов видимого человеческим глазом спектра — от красного до фиолетового, причем их эффективность составляет не менее 1%. Важным шагом на пути к созданию цветных LEP-дисплеев стало получение высокоэффективных полимерных материалов, излучающих свет первичных цветов аддитивной модели — красного, зеленого и голубого.

По своему внутреннему устройству LEP-дисплей похож на OLED. На тонкую стеклянную или пластиковую подложку с прозрачными электродами наносится слой полимерных материалов (обычно их два), а завершают конструкцию металлические электроды. Прозрачные и металлические электроды расположены взаимно перпендикулярно, образуя ортогональную сетку. При возникновении между электродами электрического поля участок полимера, расположенный в точке их пересечения, начинает светиться.

В начале нынешнего года группе голландских ученых удалось получить люминесцирующее под действием электрического поля вещество, излучающее свет двух различных цветов. В отличие от разработанных ранее LEP-дисплеев, использующих сочетание двух полимерных материалов, была применена новая комбинация заключенных между электродами веществ — полимер с полупроводниковыми свойствами и двухатомный рутениевый комплекс. В конструкции были использованы прозрачные аноды из оксидов индия и олова и золотые металлические катоды. При подаче тока в прямом направлении полимерный слой излучает свет красного диапазона (620 нм), а при изменении полярности питания — зеленого (510 нм). Немаловажно, что в обоих случаях получается чистый цвет — то есть красный без примеси зеленого и наоборот. В настоящее время исследовательская группа ведет работу по созданию прототипов двухцветных дисплеев с иными сочетаниями цветов.

Американская компания Universal Display Corporation (UDC) разработала уникальную технологию производства OLED-дисплеев с прозрачным катодом, получивших название TOLED (Transparent OLED). Данная технология позволяет создавать как односторонние, так и двусторонние прозрачные дисплеи, изображение на которых можно видеть и с внешней, и с внутренней стороны экрана. TOLED-дисплеи можно создавать на самых разнообразных подложках — стеклянной, пластиковой, силиконовой, металлической и пр.

В выключенном состоянии современные образцы TOLED-дисплеев имеют прозрачность не менее 70%, что позволяет интегрировать их, например, в оконные и автомобильные стекла, прозрачные щитки шлемов и даже в обычные очки.

В мае нынешнего года специалисты Samsung SDI, являющейся одним из технологических партнеров UDC, продемонстрировали работающий прототип цветного активно-матричного TOLED-дисплея с диагональю экрана 2,2 дюйма.

Еще одно важное изобретение UDC — это многослойные, или стекированные, OLED-дисплеи (Stacked OLED, SOLED). В отличие от обычных цветных дисплеев, где изображение формируется из пикселов трех первичных цветов, лежащих в одной плоскости, пиксел SOLED-дисплея состоит из трех расположенных друг за другом субпикселов первичных цветов аддитивной модели. Такой подход позволяет в три раза сократить количество пикселов экрана при сохранении той же разрешающей способности. Кроме того, SOLED-дисплей обеспечивает практически 100-процентное использование площади экрана независимо от воспроизводимого оттенка. Конструкция SOLED-дисплея (по крайней мере, теоретически) позволяет очень точно управлять настройкой цветопередачи благодаря возможности изменять электрические параметры отдельно для каждого из слоев субпикселов.

Самым первым коммерческим продуктом, оснащенным монохромным LEP-дисплеем, стала электробритва Sensotec, выпущенная Philips-Norelco в июле 2002 года. А конец прошлого и начало нынешнего года ознаменовались появлением первых коммерческих изделий, оснащенных цветными OLED-дисплеями.

В декабре прошлого года дисплейный OLED-модуль Kodak AM550L был удостоен золотой награды SID (Society for Information Display) в номинации "Дисплей года". Данный продукт стал первым цветным дисплейным модулем на базе OLED, который установлен в серийно выпускаемых продуктах. AM550L, разработанный Kodak совместно с Sanyo, имеет эффективный размер экрана 5,48 см и разрешение 521x218 пикселов. Энергопотребление это-

го дисплея составляет 450 мВт, а вес — всего 8 г. По замыслу разработчиков, сфера применения AM550L чрезвычайно широка — от портативных электронных устройств (таких как мобильные телефоны, цифровые фотокамеры и т.п.) и бортовых компьютеров для автомобилей до разнообразного промышленного оборудования.

В самом начале нынешнего года компания Sanyo Electric представила новую модель мобильного телефона, оснащенную двумя цифровыми камерами и цветным OLED-дисплеем с диагональю 2 дюйма. Первоначально была выпущена партия из 300 таких телефонов для проведения широкомасштабного тестирования. Результаты первых месяцев интенсивной эксплуатации не аыявили каких-либо проблем, связанных с использованием OLED-дисплея, и летом Sanyo Electric планирует запустить данную модель в серийное производство.

Нынешней весной Kodak продемонстрировала первую серийную модель своей 3-мегапиксельной цифровой фотокамеры EasyShare LS633, оснащенную цветным OLED-дисплеем. Отличительными особенностями этого дисплея являются его размер — 2,2 дюйма по диагонали, и эффективный угол обзора — 165°.

На осенних выставках 2002 г. некоторые производители представили прототипы полноразмерных компьютерных OLED-дисплеев. Так, на проходившей в начале октября в Японии выставке CEATEC JAPAN 2002 компании Sanyo Electric и Eastman Kodak продемонстрироаали 15-дюймовый широкоэкранный (16:9) цветной OLED-дисплей (точные размеры экрана — 326,4x183,6 мм; разрешение — 1280x720 пикселов; количество воспроизводимых оттенков — 262144). Представленный на выставке образец имел яркость 300 кд/м² и максимальный угол обзора 165°.

Абсолютный рекорд по размеру экрана OLED-дисплея в настоящее время принадлежит тайваньской компании Chi Mei. В начале нынешнего года ее специалистам удалось создать опытный образец широкоэкранного дисплея с диагональю в 20 дюймов. Разрешение экрана — 1280x768 пикселов, а максимальная яркость — 500 кд/м². Известно, что данный дисплей построен на базе активной OLED-матрицы с транзисторами из аморфного кремния. Для своих размеров дисплей отличается чрезвычайной экономичностью — потребляемая им при работе электрическая мощность не превышает 25 Вт.

Некоторые сведения об уровне электромагнитного излучения современного ПК приведены а заметке "Излучение компьютера опасно?" (**CHIP, №7/2003, С.20...21**).

Наклейка с надписью "CE" на любом приборе, в том числе и собранном а заводских условиях компьютере, подтверждает, что он был аттестован производителем как соответствующий требованиям Евросоюза по электромагнитной совместимости. Это значит, что данное устройство не подвержено влиянию со стороны других устройств и само не является источником электромагнитных помех. Многочисленные инструкции и правила проведения контрольных измерений уроаня электромагнитного излучения устанавливают, каким критериям должно соответстаовать проверяемое устройство, каков порядок проведения измерений и каковы должны быть предельно допустимые значения.

На практике производитель поручает своим специалистам проверить на предмет наличия вредных излучений лишь базовую конфигурацию ПК. Иная ситуация возникает в том случае, когда к компьютеру подключается все, что только можно. Однако в большей степени у специалистов вызывает обеспокоенность тот факт, что производители не обязаны предоставлять свою продукцию на проверку в независимую лабораторию технического контроля. Они вправе самостоятельно маркировать свои устройства знаком соответствия "CE". На сегодняшний день, по мнению одного из сотрудников центра технического контроля, около 30% компьютеров с высокими рабочими частотами процессоров имеют расхождение между заявленными и реальными показателями уровня электромагнитного излучения.

Эксперты предупреждают, что знак "CE" гарантирует всего пишь безупречную работу устройства. При этом показатель электромагнитной соаместимости никак не связан с аоздействием электромагнитного излучения на здоровье человека. И еще, эксперты едины в том, что уровень излучений персонального компьютера очень далек от внушающих опасения показателей: в помещении, где имеются электрические светильники и розетки, излучение от компьютера попросту не воспринимается приборами. В подтверждение этого приводятся величины плотности магнитного потока (в микротеслах) следующих источников:

- лампа дневного саета — до 0,25 мкТл на удалении 1 м;
- электробритва — 15...1500 мкТл на расстоянии 3 см;
- микроволновая печь — 4...8 мкТл на удалении 30 см и до 200 мкТл у корпуса;
- телевизор — 0,01...0,15 мкТл на расстоянии 1 м;
- компьютер — 0,01 мкТл на удалении 30 см.

Возможностям программ, помогающим содержать в порядке ваш винчестер и реестр ОС, посвящена статья *А.Евдокимова* "С глаз долой, с диска вон" (**CHIP, №7/2003, С.116...119**).

Когда мы удаляем ненужные программы привычным способом, с помощью Панели управления, в системе остается много виртуального мусора в виде оставленных этими программами файлов и ключей реестра. Для окончательной очистки лучше воспользоваться специальными утилитами-деинсталляторами.

К сожалению, стандартный деинсталлятор Windows не может удалять из системы то, что разработчики софта хотели бы в ней оставить, хотя для этого было бы достаточно сделать снимок ОС до и после установки приложения. Именно на этом простейшем, по сути, принципе построено большинство альтернативных деинсталляторов. Утилиты, рассматриваемые в статье, тестировались на одной и той же машине с установленной ОС Windows 98 SE. В процессе тестирования они должны были осуществлять мониторинг установки одних и тех жв программ, отобранных по принципу максимально возможного внедрения в систему, то есть добавления ими в ходе инсталляции и первого запуска множества DLL-библиотек в директорию "Windows\System" и значительного количества записей в системный реестр.

Утилита Ashampoo Uninstaller Suite 1.31 разработана Ashampoo GmbH&Co (сайт разработчика — www.ashampoo.com, объем дистрибутива — 4,2 Мб, условия распространения — shareware, цена — 39,99 USD). В пакет входит не только деинсталлятор, но и утилиты для очистки жесткого диска от временных и дублирующихся файлов, пустых каталогов, побочных продуктоа web-серфинга и т.д.

В последней версии Ashampoo появилось два режима — упрощенный (Easy Mode) и расширенный (Expert Mode). В Easy Mode пользователю вообще ничего делать не нужно, только указать запускаемый файл инсталлятора. Если же в качестве такового используется Setup.exe или Install.exe, сопровождение стартует автоматически благодаря резидентному монитору Ashampoo UIWatcher,

Но какой бы вариант вы ни выбрали, отслеживаться инсталлируемые приложения будут очень тщательно. Без внимания не останется ни один файл, ни одна строчка в системном реестре. Во всяком случае, бесплатные программы, выбранные автором а качестве пробы, были деинсталлированы Ashampoo целиком и полностью. Не осталось никаких зримых следов их присутствия в системе за исключением одной общей динамической библиотеки. К сожалению, на операции по контролю Ashampoo тратил слишком много времени. Не понравилось автору и то, что программа сохраняет, помимо записей о произведенных во аремя инсталляции изменениях, еще и довольно большие по объему ECD-файлы с информацией обо всех контролируемых директориях жесткого диска и разделах системного реестра. Лог-файлы деинсталлируемых программ сохранить можно, но повторно использовать нельзя.

Программа Total Uninstall 2.33 (разработчик — Gavrila Martau, сайт разработчика — www.geocities.com/ggmartau, объем дистрибутива — 800 Кб, условия распространения — freeware) не столь знаменита, как Ashampoo, и не выполняет никаких иных функций, кроме самой деинсталляции приложений. Зато она во много раз меньше "Шампуня" и, что немаловажно, абсолютно бесплатна. Но самое главное — Total Uninstall работает значительно быстрее предыдущей утилиты при слежении за установкой программ, сохраняя в виде особых TUN-файлов не все параметры системного реестра, а лишь те из них, что претерпели во время установки изменения. Соответственно, такие логи на диске много места не занимают. Их после удаления программ можно оставить и в дальнейшем использовать, не прибегая к мониторингу при повторной инсталляции.

Последовательность действий в Total Uninstall практически ничем не отличается от Ashampoo в режиме Expert. В окошке, появляющемся при старте, вы должны ввести имя лога будущей установки. Обычно это название устанавливаемого приложения. Далее вам нужно будет выбрать системные директории и разделы реестра, которые Total Uninstall начнет отслеживать. Здесь все просто: чем больше галочек поставите, тем точнее получится результат. При переходе к следующей ступени инсталляции Total Uninstall создаст временные файлы образов. При этом вас попросят ничего в этот момент не предпринимать: не запускать, не закрывать и, тем более, не инсталлировать посторонние программы. Все подобные манипуляции будут записаны и впоследствии могут затруднить деинсталляцию. На предпоследнем этапе вы должны решить, одну ли программу устанавливать или сразу несколько, а также указать путь к запускаемому файлу дистрибутива. Нажав кнопку "Next", вы автоматически запустите процесс собственно инсталляции, в финале которого, как и в Ashampoo, желательно открыть установленное приложение.

Теперь Total Uninstall без проблем определит произошедшие в системе изменения и выдаст их в виде списка, который можно не только просмотреть, но и экспортировать в текстовый файл.

Удаляет программы Total Uninstall чуть хуже Ashampoo, но зато более осторожно. Например, перед удалением неиспользуемых другими приложениями общих DLL-файлов он спрашивает разрешения. Из недостатков Total Uninstall автор отмечает отсутствие резидентного монитора.

Утилита Trash It 1.7 (разработчик — Optimus Software, сайт разработчика — www.trashituninstaller.com, объем дистрибутива — 1,7 Мб, условия распространения — shareware, цена — 24,5 USD) осуществляет мониторинг инсталляций еще быстрее, чем Total Uninstall. И это, по мнению автора, большой плюс. Скорость работы, хотя и не является определяющим параметром для деинсталляторов, тем не менее, очень важна. Чем меньше времени мы будем тратить на рутинное отслеживание процессов при установке программ, тем чаще будем прибегать к помощи подобного рода утилит.

Но с интерфейсом разработчики Trash It немного перемудрили. Три важнейшие кнопки носят весьма странные для деинсталлятора названия: Update database (обновить базу данных), Find changes (найти изменения) и View&delete changes (посмотреть и удалить изменения). На самом деле первая всего лишь запускает режим мониторинга, вторая сохраняет в виде небольшого TRH-файла результаты инсталляции (которую вы должны перед этим самостоятельно запустить), ну а третья непосредственно удаляет выбранное приложение.

Зону контроля пользователь вправе определить по своему усмотрению в разделе "Options". К сожалению, в Trash It нельзя выбирать отдельные разделы реестра. Зато на закладке опций "Scheduler" можно задать периодичность очистки диска от временных файлов и папок, а также перемещения неиспользуемых общих DLL-библиотек. В разделе System clean-up вы сможете самостоятельно освободить систему от этого мусора, а также, при желании, еще и от повторяющихся файлов.

С предложенными автором заданиями Trash It справился. Однако действует он, по мнению автора, чересчур смело и бескомпромиссно. Все, что укажет пользователь, включая ненужные в системе общие DLL, удалит без лишних вопросов. Хорошо еще, что по умолчанию Trash It не стирает все до конца, а только отправляет в Корзину. Логи деинсталлируемых программ он, кстати, также уничтожает без согласования с пользователем. Сохраняются лишь особые TRS-файлы, отражающие текущее состояние системы.

У программы Professional Uninstaller 3.11 (разработчики — И.Дышленко, Д.Мироводин, сайт разработчиков — www.hcsoft.spb.ru, прямая ссылка на программу — www.hcsoft.spb.ru/pun.zip, объем дистрибутива — 290 Кб, условия распространения — freeware) есть два неоспоримых преимущества: во-первых, интерфейс и файл справки этого деинсталлятора в переводе не нуждаются, поскольку они созданы российскими разработчиками, а во-вторых, ему самому не требуется установка. Распакуй в любую папку и пользуйся.

Однако он оказался единственным из рассмотренных, который не сумел справиться с заданием. Точнее, сумел лишь наполовину. Professional Uninstaller корректно, без пропусков и ошибок, стер файлы и папки тестовых программ. Но вот найти и удалить все записи о них в системном реестре при заданном по умолчанию ограниченном режиме слежения он не смог. Когда же автор настроил его на полный контроль, то едва не потерял систему. Спасло резервное копирование файлов реестра.

Удручает и скорость работы Professional Uninstaller. По быстродействию этот деинсталлятор уступил даже громоздкому Ashampoo. На выполнение заданий, на которые у других уходило от силы три-пять минут, Professional Uninstaller умудрялся затрачивать более десяти. Тормозит он главным образом при сканировании ключей и классов системного реестра. К тому же, как и другие бесплатные деинсталляторы, Professional Uninstaller не имеет резидентного монитора.

А это значит, что процесс инсталляции в нем пользователь должен запускать самостоятельно щелчком по кнопке "Сканировать". Но доступна она становится только после создания файла лога предстоящей установки. Изменения, произошедшие в системе, вы сможете зафиксировать, кликнув мышкой по кнопке "Отсеять". Деинсталляция отслеживаемых приложений осуществляется в меню "Файл/Удаление программы", запустив которое вы должны будете выбрать соответствующую учетную запись. На удаление файлов из директории Windows программа запрашивает разрешение.

My Uninstaller 2.16 (разработчик — Arnab, сайт разработчика — www.geocities.com/hirak_99/goodies/uninstaller.html, объем дистрибутива — 280 Кб, условия распространения — freeware), как и предыдущую программу, устанавливать не нужно, достаточно распаковать в какую-либо папку. По скорости мониторинга My Uninstaller практически не уступает "реактивному" Total Uninstall, а по качеству работы, по мнению автора, даже превосходит его и остальные утилиты. Дело в том, что только My Uninstaller перед началом зачистки того или иного приложения запускает встроенный в него деинсталлятор, и лишь затем подключается сам и окончательно удаляет то, что разработчик хотел бы оставить. Это позволяет, помимо всего прочего, отследить использование общих DLL-библиотек стираемым приложением и спокойно, без суеты, решить вопрос с их удалением из системы. Перед деинсталляцией появляется окошко с перечисленными директориями на жестком диске и ключами системного реестра. Нужно поставить галочки лишь у тех пунктов, которые имеют отношение к удаляемому приложению. Текстовые REC-файлы логов занимают не больше 10 Кб. Они не удаляются при деинсталляции, и их можно использовать повторно.

К сожалению, My Uninstaller не имеет резидентного монитора, но мало-мальски продвинутым пользователям он не очень-то и нужен. Тем более, что управление процессом инсталляции в этой программе предельно упрощено. До установки приложения нужно кликнуть по кнопке "Start Noting Changes", а после ее завершения — по кнопке "End Noting Changes". Перед началом мониторинга деинсталлятор запрашивает файл каталога. Если вы его еще не создали, согласитесь с использованием открытого по умолчанию.

А.КОРБИТ, А.ЩЕРБАКОВ,
г.Минск, БГУИР.

Программирование в среде Visual C++ 6.0

Элемент управления ComboBox

Цель этой статьи — научиться использовать элемент управления ***ComboBox****, изучить основные методы класса CComboBox, предназначенные для управления данным элементом.*

1.Класс CComboBox

Элемент управления ComboBox представляет собой список ListBox в комбинации с редактором Edit или надписью Static. Данный элемент называют комбинированным окном. Окно при этом может иметь два состояния — распахнутое (которое отображается постоянно) и свернутое (отбражается при щелчке на элементе "стрелка вниз"). Элементы списка в окне можно выбирать, и тогда они отображаются или в окне редактирования, или в окне статического элемента.

Три модификации компонента определяются его свойством "Type" в закладке окна свойств Style. В таблице приведено сравнение этих трех стилей.

Стиль	Когда отображается ListBox?	Что отображается, Edit или Static?
Simple	Всегда	Edit
Drop-down	После нажатия кнопки со стрелкой	Edit
Drop-down list	После нажатия кнопки со стрелкой	Static

Для управления элементом ComboBox в библиотеке MFC существует класс CComboBox.

Чтобы добавить новую строку в выпадающий список, нужно использовать функцию

```
int AddString( LPCTSTR lpszItem );
```

Здесь параметр *lpszItem* — это указатель на нуль-терминированную строку, добавляемую в список.

Для очистки списка служит функция

```
void ResetContent( );
```

В списке элемента ComboBox пользователь может выбирать элементы, и для того чтобы определить, какой элемент был выбран, нужно воспользоваться функцией

```
int GetCurSel( ) const;
```

Эта функция не принимает параметры и возвращает индекс выбранного элемента, причем индекс начинается с нуля.

Если необходимо из списка по заданному индексу элемента определить его содержимое, можно воспользоваться следующей функцией:

```
void GetLBText( int nIndex, CString& rString ) const;
```

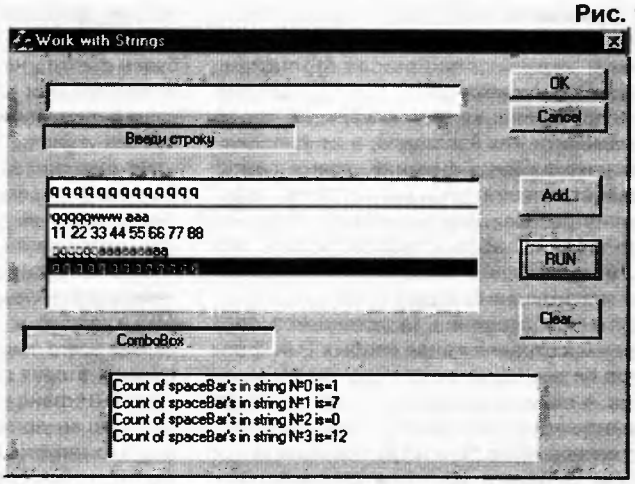
Здесь `nIndex` — индекс элемента, `rString` — строка, куда помещается текст.

2.Пример написания программы

Необходимо создать приложение, осуществляющее следующие действия.

В поле ввода последовательно должны вводиться строки, которые кнопкой Add... добавляются в поле ComboBox (**рис.1**). Затем щелчком левой клавишей мыши должна выбираться соответствующая строка (выбранная строка отобразится в верхнем окошке ComboBox). Щелчок на кнопке RUN должен произвести подсчет пробелов в выделенной строке и отобразить результат в поле вывода.

Рис. 1



Для решения задачи проделайте следующие действия.

1. Создайте при помощи AppWizard(exe) новый проект типа Dialog based (предварительно выбрав для него путь и имя).

2. Используя панель Controls, нанесите на окно Dialog: два элемента типа Editbox (поле ввода строки и поле вывода); три элемента типа Button и один элемент типа ComboBox для вывода добавляемых строк. Используя Static text, нанесите на окно нужные пояснения. Для кнопок и диалоговых элвментов установите нужные свойства и надписи. Для ComboBox в закладке Styles установите Type->Simple и активизируйте CheckBox для Vertical scroll.

3. При помощи ClassWizard через закладку Member Variables свяжите поле Edit1 с переменной m_a типа CString. Для этого следует выбрать ClassWizard/Member Variables..., сделать щелчок на IDC_EDIT1, затем щелчок на Add Variable:..., в закладке ...name: **m_a** ...type выбрать тип CString и подтвердить сделанные изменения, нажав на OK. Аналогичные действия выполните и для поля вывода IDC_EDIT2 — name: **m_b** ...type: CString и в свойствах установите — Multiline (можно добавить вертикальный скроллинг). Элемент ComboBox свяжите с **m_c**, выбрав Category: **Control**, установив тем самым для данного объекта тип **CComboBox** (т.е. для данного объекта можно вызывать методы по всей цепи иерархии данного класса MFC). Далее через ClassWizard/Message maps... увяжите кнопки с событием "один щелчок" и активизируйте обработчики. Так, для стирания окон вывода в обработчик щелчка на кнопке Clear... можно вызвать метод `m_c.ResetContent();` и `m_b=""`.

Обработчик кнопки Add... добавляет в поле вывода ComboBox строку, набранную в поле ввода, одновременно очищая последнюю. Его текст:

```
{
UpdateData(TRUE);
m_c.AddString(m_a);
m_a="";
UpdateData(FALSE);
}
```

Обработчик кнопки RUN выводит в поле вывода номер выбранной строки и количество пробелов в ней. Его текст:

```
CString s1;
int i,j=0;
char str[25];
```

```
CComboBox *p=(CComboBox *)GetDlgItem(IDC_COMBO1);
i=p->GetCurSel();
if (i==LB_ERR)
  AfxMessageBox("String No SELECT!!!");
else
{
  p->GetLBText(i, str);
  AfxMessageBox(str);
}
```

Рис. 2

Объявляем указатель на элемент ComboBox, через него вызываем метод GetCurSel() для определения номера выбранной строки и контролируем ввод — если выбор строки не сделан, то получаем AfxMessageBox с сообщением об этом (**рис.2**). Если же все верно, методом GetLBText копируем выделенную строку списка в str и отбражаем ее в AfxMessageBox.

Приведенный ниже фрагмент программы решает поставленную задачу, т.е. подсчитывает число пробелов в выделенной строке:

```
int k=strlen(str);
for(int i1=0;i1<k;i1++)
      if (str[i1]==" ") j++;
s1.Format("Count of spaceBar's in string №%d
       is=%d%c%c",i,j,13,10);
m_b=m_b+s1;
UpdateData(FALSE);
}
```

В начале текста программы нужно подключить заголовочный файл **string.h**. Кроме того, находясь в режиме конструктора диалога, командой Layout/Tab Order установите необходимую очередность перехода между элементами управления — поле ввода должно иметь №1.

3. Индивидуальные задания

По методике, описанной в примере написания программы, необходимо создать приложение, выполняющее задачу в соответствии с вариантом задания, и протестировать его работу.

1. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран группу, состоящую из пяти символов.
2. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран самую короткую группу.
3. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество символов в самой длинной группе.
3. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран группы с четным количеством символов.
4. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество единиц в группах с нечетным количеством символов.
5. Дана строка, состоящая из букв, цифр, запятых, точек, знаков "+" и "-". Вывести подстроку, которая соответствует записи целого числа (т.е. строка начинается со знаков "+" и "-", и внутри подстроки нет букв, запятых и точек).
6. Дана строка, состоящая из букв, цифр, запятых, точек, знаков "+" и "-". Вывести подстроку, которая соответствует записи вещественного числа с фиксированной точкой.
7. Дана строка, состоящая из букв, цифр, запятых, точек, знаков "+" и "-". Вывести подстроку, которая соответствует записи вещественного числа с плавающей точкой.
8. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести на экран числа строки в порядке возрастания их значений.
9. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести на экран четные числа этой строки.

Литература

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

Решение задач на графах построением максимального потока

М.ДОЛИНСКИЙ, г.Гомель.

(Окончание. Начало в №№8-10/2003)

8. Алгоритм Форда-Фалкерсона для нахождения максимального потока

Неформально этот алгоритм может быть записан следующим образом:

```
Формируем нулевой поток
Пока существует путь из истока в сток, увеличивающий поток
  Выбираем Cmin - минимальную из остаточных пропускных
                  способностей найденного пути
  Для каждой дуги (u,v) пути (из вершины u в вершину v)
    f[u,v]:=f[u,v]+Cmin
    f[v,u]:=-f[u,v]
```

Далее приводится описание одного из вариантов программной реализации метода Форда-Фалкерсона.

Тело программы для решения задачи обычно выглядит следующим образом:

```
begin
  InputData;
  MaxFlow;
  OutputData;
end.
```

При этом процедуры InputData и OutputData отражают специфику задачи. А тело процедуры MaxFlow выглядит следующим образом:

```
begin
 Init;                          {Инициализация нулевого потока}
```


```
While ExistPath(KR,Cmin) do
          {CMin - минимальный из cf(u,v)=c(u,v)-f(u,v)}
  for i:=KR downto 1 do {Для каждой дуги увеличивающего пути}
   begin
    u:= path[i+1];           {U->V - дуга с номером i в пути}
    v:= path[i];
    f[u,v]:=f[u,v]+Cmin;
                          {Увеличиваем поток на минимальную}
    f[v,u]:=-f[u,v];              {величину f(v,u)=-f(u,v)}
    cf[u,v]:=c[u,v]-f[u,v];
          {Перевычисляем остаточную пропускную способность}
    cf[v,u]:=c[v,u]-f[v,u];    {на дугах увеличивающего пути}
   end;
end;
```

Как можно заметить, в теле процедуры MaxFlow вызываются процедура Init — для инициализации нулевого потока, и функция ExistPath, которая возвращает значение "true", если увеличивающий путь существует, и значение "false" — в противном случае. Очевидно, что если функция ExistPath возвращает значение "false", то процедура MaxFlow завершает свою работу (увеличивающих путей больше нет). Если функция ExistPath возвращает значение "true" (найден увеличивающий путь), то через свои параметры она также возвращает:

- Cmin — минимальное значение из остаточных пропускных способностей дуг увеличивающего пути;
- KR — количество дуг в увеличивающем пути.

Кроме того, в глобальном для процедуры MaxFlow массиве path возвращаются номера вершин увеличивающего пути в порядке от стока к истоку.

Если путь найден, то в соответствии с алгоритмом Форда-Фалкерсона для всех дуг увеличивающего пути пересчитывается поток (f[u,v]), строятся обратные дуги с отрицательным потоком (f[v,u]) и пересчитываются остаточные пропускные способности (cf[u,v] и cf[v,u]).

Рассмотрим более подробно содержание процедур Init и ExistPath.

```
Procedure Init;
 begin
  for i:=0 to Finish do
    for j:=0 to Finish do f[i,j]:=0;
  for i:=0 to Finish do
    for j:=0 to Finish do cf[i,j]:=c[i,j];
                          {построение списков входящих дуг}
  for i:=0 to Finish do kia[i]:=0;
  for i:=0 to Finish do                    {По всем вершинам}
    for j:=1 to Finish do ia[i,j]:=-1; {По всем исходящим дугам}

  for i:=0 to Finish do                    {По всем вершинам}
    for j:=1 to ka[i] do             {По всем исходящим дугам}
      begin
        inc(kia[a[i,j]]);
                      {инкремент к-ва дуг, входящих в a[i,j]}
        ia[a[i,j],kia[a[i,j]]]:=i;
                {запоминаем вершину i, как входящую в a[i,j]}
      end;
 end;
```

Здесь вначале строится нулевой поток:

```
for i:=0 to Finish do
  for j:=0 to Finish do f[i,j]:=0;
```

Затем — начальная остаточная пропускная способность:

```
for i:=0 to Finish do
  for j:=0 to Finish do cf[i,j]:=c[i,j];
```

Далее для более удобной работы с "обратными" дугами увеличивающего пути строится "инвертированный" граф таким образом, чтобы номера вершин в "инвертированном" графе оставались прежними, а направления дуг менялись на противоположные. Тем самым мы для каждой вершины получаем список входящих в нее дуг.

- kia[i] = количество дуг, входящих в вершину i;
- ia[i,j] = номер вершины, из которой ведет j-я по счету дуга, входящая в вершину i.

```
        {Построение списков входящих дуг}
                                                {Инициализация}
for i:=0 to Finish do kia[i]:=0;
for i:=0 to Finish do                          {По всем вершинам}
  for j:=1 to Finish do ia[i,j]:=-1; {По всем исходящим дугам}
```

Заметим, что величины ia[i,j] инициализируются значением -1, поскольку нулем инициализировать нельзя! В графе есть дуги, исходящие из вершины 0!

```
for i:=0 to Finish do                          {По всем вершинам}
  for j:=1 to ka[i] do                  {По всем исходящим дугам}
    begin
      inc(kia[a[i,j]]);  (инкремент к-ва дуг, входящих в a[i,j])
      ia[a[i,j],kia[a[i,j]]]:=i;
                  (запоминаем вершину i, как входящую в a[i,j]}
    end;
```

И наконец, более подробно рассмотрим функцию ExistPath.

```
Function ExistPath(var KR,Cmin:integer):boolean;
                                          {Breadth-First Search}
  const
    WHITE = 1;
    GRAY  = 2;
  var
    color, back : array [0..MaxV] of integer;
    Q           : array [0..MaxV] of integer;

    EndQ, BegQ          : integer;
    Find                : Boolean;

  Procedure Put(x:integer);
    begin
      inc(EndQ); Q[EndQ]:=x;
    end;

  Procedure Get(var x:integer);
    begin
      x:=Q[BegQ]; inc(BegQ);
    end;

  begin
    for i:=0 to Finish do color[i]:=WHITE;{Все вершины свободны}
    for i:=0 to Finish do Q[i]:=-1;     {Все вершины свободны}
    color[0]:=GRAY;              {Начальная вершина обработана}
    back[0]:=-1;

    BegQ:=1;                                      {Начало очереди}
    EndQ:=0;                                      {Очередь пуста}
    KR:=0;                                {Количество дуг в пути = 0}
    Put(0);              {Поместить в очередь начальную вершину}
    Find:=false;
    while (BegQ<=EndQ) and not Find do
                         {Пока очередь не пуста и путь не найден}
      begin
        Get(i);                        {Взять вершину i из очереди}
        j:=1;                             {Номер дуги из вершины i}
        while (a[i,j]>0) and
              not Find do                 {Пока дуги не кончились}
          begin
            if (color[a[i,j]]=WHITE) and
               (cf[i,a[i,j]]>0) {Если вершина a[i,j] свободна}
              then begin
                     Put(a[i,j]);      {поставить ее в очередь}
                     back[a[i,j]]:=i;
```

```
                              {в вершину a[i,j] - из вершины i}
                    color[a[i,j]]:=GRAY;
                  {Пометить вершину a[i,j] как использованную}
                    Find := a[i,j]=Finish;
                  end;
          inc(j);                          {Взять следующую дугу}
        end;

                              {Проход по отрицательным ребрам}
      j:=1;                              {Номер дуги из вершины i}
      while (ia[i,j]>=0) and
            not Find do                  {Пока дуги не кончились}
        begin
          if (color[ia[i,j]]=WHITE} and
             (cf[i,ia[i,j]]>0){Если вершина a[i,j] свободна}
           then begin
                  Put(ia[i,j]);     {Поставить ее в очередь}
                  back[ia[i,j]]:=i;
                              {В вершину a[i,j] - из вершины i}
                  color[ia[i,j]]:=GRAY;
                {Пометить вершину a[i,j] как использованную}
                  Find := ja[i,j]=Finish;
                end;
          inc(j);                          {Взять следующую дугу}
        end;

      end;
                              {Восстановление пути из очереди}
    KR:=0;
    i:=Finish;                                  {Начинаем с конца}
    Cmin:= maxint;
    while (i<>0} do                             {Пока не начало}
      begin
        Inc(KR);                       {Увеличиваем к-во дуг в пути}
        path[KR]:=i;          {Заносим номер предыдущей вершины}
        i:=back[i];                    {Меняем текущую вершину}
        if Cmin>cf[i,path[KR]]
          then Cmin:=cf[i,path[KR]];
                                {Минимальный добавляемый поток}
      end;
    path[kR+1]:=0;
    ExistPath:=Find;
  end;
```

Для построения увеличивающего пути используется механизм очереди от истока:

```
...
Put(0);                    {Поместить в очередь начальную вершину}
..
while (BegQ<=EndQ) and not Find do
                        {Пока очередь не пуста и путь не найден}
  begin
    Get(i);                            {Взять вершину i из очереди}
```

Вначале идет попытка увеличить путь за счет прямых дуг:

```
j:=1;                                       {Номер дуги из вершины i}
while (a[i,j]>0} and
      not Find         do                   {Пока дуги не кончились}
  begin
    if (color[a[i,j]]=WHITE) and
       (cf[i,a[i,j]]>0)          {Если вершина a[i,j] свободна}
     then begin
            Put(a[i,j]);                    {Поставить ее в очередь}
            back[a[i,j]]:=i;{В вершину a[i,j] - из вершины i}
            color[a[i,j]]:=GRAY;
                 {Пометить вершину a[i,j] как использованную}
            Find := a[i,j]=Finish;
          end;
    inc(j);                                 {Взять следующую дугу}
  end;
```

Заметим, что здесь:

- массив Color[i] хранит пометки для всех вершин (включена/нет в увеличивающий путь);
- массив back[k] хранит номер вершины, из которой пришли в вершину k в увеличивающем пути;
- переменная Find получает значение "true" если увеличивающий путь достиг стока (a[i,j]=Finish).

Далее идет попытка увеличить путь за счет обратных дуг:

```
                              {Проход по отрицательным ребрам}
j:=1;                                  {Номер дуги из вершины i}
while (ia[i,j]>=0) and
      not Find         do              {Пока дуги не кончились}
  begin
    if (color[ia[i,j]]=WHITE) and
       (cf[i,ia[i,j]]>0)          {Если вершина a[i,j] свободна}
     then begin
            Put(ia[i,j]);                   {Поставить ее в очередь}
            back[ia[i,j]]:=i; {В вершину a[i,j] - из вершины i}
            color[ia[i,j]]:=GRAY;
                 {Пометить вершину a[i,j] как использованную}
            Find := ia[i,j]=Finish;
          end;
    inc(j);                                 {Взять следующую дугу}
  end;
```

После выхода из цикла построения увеличивающего пути остается только восстановить увеличивающий путь в массив path, используя массив back. Попутно подсчитываются количество дуг в увеличивающем пути (KR) и минимальный добавляемый поток (Cmin):

```
                              {Восстановление пути из очереди}
KR:=0;
i:=Finish;                                  {Начинаем с конца}
Cmin:= maxint;
while (i<>0) do                             {Пока не начало}
  begin
    Inc(KR);                       {Увеличиваем к-во дуг в пути}
    path[KR]:=i;          {Заносим номер предыдущей вершины}
    i:=back[i];                    {Меняем текущую вершину}
    if Cmin>cf[i,path[KR]]
      then Cmin:=cf[i,path[KR]]; {Минимальный добавляемый поток}
  end;
path[kR+1]:=0;
ExistPath:=Find;
```

Полное решение задачи "Новогодние вечеринки" с целью экономии печатной площади вынесено на веб-страницу журнала.

9. Некоторые замечания по приведенной реализации метода Форда-Фалкерсона

Возможно, некоторым наиболее опытным читателям покажется чрезвычайно неэкономным использование оперативной памяти в данных решениях. Ничуть не пытаясь оспаривать такие утверждения, автор имеет по этому поводу следующие соображения:

1. Современные 32-битные компиляторы и операционные системы, в которых они работают, снимают привычные для DOS-приложений (например, такого как Turbo-Паскаль,) ограничения на статический размер сегмента данных (64 Кб). Более того, они поощряют использование массивов данных больших размеров оптимизацией работы с оперативной и дисковой памятью на уровне приложений (страничная организация памяти, динамическая подгрузка страниц, оптимальные алгоритмы вытеснения страниц и т.д.).

2. В международных, а соответственно, и национальных олимпиадах по информатике для школьников (IOI) и студенческих командных олимпиадах по программированию для студентов (ACM ICPC) стимулируется использование 32-битных компиляторов (Free Pascal, Delphi, GNU C и др.).

3. Основной целью данной статьи было максимально простое и понятное изложение идей по сведению задач к задачам на построение максимального потока и собственно алгоритма Форда-Фалкерсона для решения данной задачи. Потому и были выбраны наиболее простые, возможно, в некотором смысле и избыточные структуры данных.

4. Как уже упоминалось, задача "Новогодние вечеринки" по правилам олимпиад USACO имеет право решаться с использованием 32-битного компилятора FreePascal. Модификация читателями авторских решений задач "Кубики", "Игра" и "Courses", так чтобы они смогли быть выполнены в среде DOS и с помощью компилятора Turbo Pascal, является неплохим "домашним заданием" для вдумчивых читателей.

5. Автор полагает, что методы оптимального использования оперативной памяти и использование более компактных, хотя и более сложно организованных структур хранения данных безусловно важны и, скорей всего, заслуживают специальной статьи, посвященной рассмотрению только этих вопросов.

10. Задача для самостоятельного решения

(1999-2000 USA Computing Olympiad, Winter Contest, January 26...February 2, 2000, Green Division. Задача 1: Подарки [Hal Burch])

На ферме праздник. Вы купили подарки для коров и теперь пытаетесь выбрать совершенный подарок для каждой коровы. Каждая корова дала вам свой список желаемых подарков. Каждая корова в итоге должна получить ровно один подарок. Корова радуется своему подарку только тогда, если он есть в списке ее желаемых подарков.

Ваша задача — определить наибольшее количество коров, которых можно порадовать заданным множеством подарков. Список подарков может включать некоторые наименования более одного раза, если таких подарков было куплено несколько.

В этой задаче каждый подарок обозначается целым числом в интервале от 1 до 1000.

Формат ввода:

Строка 1: Два целых числа C и G ($1 \le C \le 100$ — количество коров, получающих подарки; $C \le G \le 100$, количество подарков).

Строка 2: G целых чисел, отделяемых друг от друга ровно одним пробелом.

Строки 3..C+2: C строк, содержащих списки желаемых подарков для коров, в формате *n w0 w1 w2 ... wn*, где: *n* — количество подарков, которые корова будет рада получить; *w0, w1, ...* — номера этих подарков.

Формат вывода:

Одна строка с целым числом, которое указывает максимальное количество коров, которые могут быть обрадованы заданным списком подарков.

Пример ввода (файл INPUT.TXT):	Пример вывода (файл OUTPUT.TXT):
5 5 35 35 483 622 801 1 35 1 35 1 35 3 483 622 801 3 35 801 483	4

Заключение

Данная статья продолжает серию публикаций [1...14], ориентированных на самообучение программированию и подготовку к олимпиадам по информатике . Очевидно, что для закрвпления данного материала необходимо проверить полученные знания на компьютере как самостоятельным решением поставленных в данном материале задач, так и применением полученных знаний к другим задачам. Удобно пользоваться сайтом http://dl.gsu.unibel.by для контроля правильности решения задач.

Полезно также ознакомиться с соответствующими материалами учебных пособий, например, [15...18].

Литература

1. М.Долинский. Памятка участнику олимпиады по информатике среди школьников. — Радиолюбитель. Ваш компьютер, 2000, №1, С.20.

2. М.Долинский. Начинаем программировать самостоятельно. — Радиолюбитель. Ваш компьютер, 2000, №№1...6.

3. М.Долинский. Решение задач на тему "Геометрия на плоскости" — Радиолюбитель. Ваш компьютер, 2000, №№8, 9.

4. М.Долинский. Разработка программ с использованием определяемых программистом типов данных. — Радиолюбитель. Ваш компьютер, 2001, №№1, 3.

5. М.Долинский. Решение задач с помощью очереди и стека. — Радиолюбитель. Ваш компьютер, 2001, №№4...6.

6. М.Долинский. Обзор возможностей языка программирования Паскаль. — Радиомир. Ваш компьютер, 2001, №7, С.23.

7. М.Долинский. Алгоритмы генерации комбинаторных объектов. — Радиомир. Ваш компьютер, 2001, №№10, 11.

8. М.Долинский. Некоторые факты из теории чисел. — Радиомир. Ваш компьютер, 2001, №9, С.20.

9. М.Долинский. Использование рекурсивных функций и процедур. — Радиомир. Ваш компьютер, 2002, №№1, 2.

10. М.Долинский . Решение задач с помощью рекуррентных соотношений. — Радиомир. Ваш компьютер, 2002, №№3...6.

11. М.Долинский. Решение задач на графах. — Радиомир. Ваш компьютер, 2002, №№7...12.

12. М.Долинский. Использование динамической памяти в прикладных программах. — Радиомир. Ваш компьютер, 2002, №12; 2003, №1.

13. М. Долинский. Решение задач методом дихотомии. — Радиомир. Ваш компьютер, 2003, №№2...4.

14. М. Долинский. Решение задач на графах построением минимального остовного дерева. — Радиомир. Ваш компьютер, 2003, №№5...7.

15. Котов В.М., Волков И.А., Харитонович А.И. Методы алгоритмизации. Учебное пособие для 8-го класса общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 1996.

16. Котов В.М., Волков И.А., Лапо А.И. Методы алгоритмизации. Учебное пособие для 9-го класса общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 1997.

17. Котов В.М., Мельников О.И. Методы алгоритмизации. Учебное пособие для 10-11-го классов общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 2000.

18. Кормен Т., Лейзерсон Т., Ривест Р. Алгоритмы: построение и анализ. — М.: МЦНМО, 1999, 960 с.

ИССЛЕДОВАНИЕ ПРОГРАММ

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

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КРЭКИНГА

(Продолжение. Начало в №№8-10/2003)

Переменные и константы.

Одна из первых задач, с которыми сталкивается крэкер, заключается в поиске в тексте программы констант или обращений к переменным, которые отвечают за реализацию ограничений в незарегистрированных и демонстрационных программах. Нередко также возникает необходимость извлечь из программы какие-либо данные, либо заменить их своими собственными. Поскольку в данной главе пойдет речь не только об изучении, но и об изменении программ, сразу же ознакомлю вас с одним из важнейших, с моей точки зрения, принципов, которым нужно руководствоваться, если приходится прибегать к модификации данных или исполняемого кода. Это **принцип минимального вмешательства**, который можно сформулировать так:

Чем меньше изменений внесено в логику программы, тем менее вероятно, что эти изменения приведут к некорректной работе программы.

Приблизительный рейтинг изменений по их влиянию на логику работы (от минимального к максимальному) выглядит следующим образом:

- модификация константы;
- изменение значения предварительно инициализированной переменной;
- изменение условия перехода на противоположное;
- удаление линейной (т.е. не содержащей ветвлений и вызовов нетривиальных подпрограмм) последовательности команд;
- дописывание в программу собственного кода или изменение значения, возвращаемого функцией;
- модификация внедренного в программу ресурса, важного для логики программы (т.е. такого, изменение которого меняет поведение программы);
- удаление из программы логического блока (например, вызова нетривиальной функции) или удаление внедренного в программу ресурса.

Этот принцип, как любое другое широкое обобщение, требует достаточно осторожного отношения, и в некоторых случаях может повести вас по неоптимальному пути, но на практике принцип минимального вмешательства обычно работает вполне успешно. Да и с точки зрения простого здравого смысла очевидно, что найти и исправить условие, которое проверяет зарегистрированность программы — прием куда более безобидный и надежный, чем вырывание с корнем nag-screen'ов и триальных ограничений при помощи глубокого патчинга.

Как я уже говорил, многие незарегистрированные программы содержат количественные ограничения. В предыдущей главе мы как раз рассматривали некоторые разновидности таких ограничений и способы их обхода. Но, к сожалению, есть немало случаев, когда манипуляции с системным временем, файлами и реестром не помогают — это всевозможные ограничения на продолжительность одного сеанса работы с программой, на максимальное количество создаваемых или обрабатываемых документов, число записей в базе данных и т.п. Даже число жизней в компьютерной игре является одним из таких ограничений (как ни странно, крэкинг иногда применяется в качестве средства для беспроблемного прохождения компьютерных игр; "вечная жизнь" и "бесконечное оружие" — это тоже один из аспектов крэкинга, возможно, даже самый старый).

Наверное, наиболее распространенным типом данных, используемым в программах, в настоящее время являются целочисленные данные. Абсолютное большинство существующих процессоров, в том числе и наши любимые x86, изначально ориентировались на работу с целыми числами определенной разрядности (в настоящее время на платформе x86 наиболее актуальны 32-разрядные целые со знаком или без). Именно в виде целых чисел чаще всего хранятся всевозможные счетчики, контрольные суммы и даже логические значения (когда я переходил от программирования под ДОС к программированию под Win32, меня сильно удивляла расточительность фирмы Microsoft, отводившей под простую булевскую переменную целых 4 байта. И только более глубокое изучение архитектуры 32-разрядных процессоров и ассемблера расставило все по своим местам).

Что вы будете делать, если вам потребуется найти в программе сравнение чего-либо с целочисленной константой с определенным значением? В принципе, для некоторых целых чисел достаточно обычного поиска блока двоичных данных в файле. Однако двоичный поиск хорошо работает далеко не со всеми числами — скажем, если вы ищете число 3B9ACA00h, вероятность ложного срабатывания будет весьма небольшой, а вот если вы попытаетесь найти в исполняемом файле число 10 или 15, вы скорее всего просто устанете нажимать на кнопку "Найти далее". Кроме того, при таком способе поиска не учитывается структура исполняемого файла программы, т.е. вы ищете нужную вам константу не только в коде программы, но и в PE-заголовке, секции данных, секции ресурсов и прочих областях, имеющих к коду программы самое отдаленное отношение.

Однако есть и другой, более эффективный метод поиска в программе известных заранее значений. Как это ни странно, но в реальных программах широко используется лишь сравнительно небольшой набор целочисленных констант — это, прежде всего, небольшие положительные числа от 0 до 7 (а также небольшие отрицательные от -3 до -1) и степени двойки — 8, 16, 32 и т.д. Другие константы в программах встречаются значительно реже. Попробуйте сами провести эксперимент — дизассемблируйте какую-нибудь достаточно большую программу и попробуйте найти в ней какое-нибудь число, например, 32h. Для демонстрации этого я написал программку на Delphi 7, вся функциональность которой концентрировалась в следующем коде, имитирующем простейшее ограничение на число строк в документе:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if Memo1.Lines.Count>50 then
   begin
    Application.MessageBox('More than 50 items not available',
                                                 'Demo version');
    Close;
   end
  else Memo1.Lines.Add(Edit1.Text);
end;
```

В результате компиляции, этот весьма нехитрый текст превратился в исполняемый файл размером более 350 Кб (я

намеренно использовал режим компиляции без использования runtime packages, и потому мой собственный код составлял в исполняемом файле очень малую долю по сравнению с библиотеками Delphi). Затем я дизассемблировал откомпилированную программу при помощи W32Dasm и получил текст листинга длиной более 180 000 строк. Казалось бы, обнаружить область, где происходит сравнение с числом 50, в этом листинге ничуть не проще, чем найти иголку в стоге сена. Но я воспользовался функцией поиска в тексте строки, в качестве параметра поиска указав 00000032 (так в W32Dasm отображается число 50; заодно это позволило отсеять команды типа mov eax,[ebx+50h], обычно использующиеся для доступа к элементам массивов и полям структур). Результат превзошел самые смелые ожидания — четырехбайтное число 32h встретилось в листинге всего два (!!!) раза:

```
:004478C6 6A32                  push 00000032
```

и

```
:004505BB 83F832                cmp eax, 00000032
```

Чтобы догадаться, что нужное нам сравнение происходит во второй строке, достаточно самых минимальных познаний в ассемблере. Из этого следует вывод: поиск нужной константы в дизассемблированной программе вполне реален, даже несмотря на огромный объем листинга.

Далее. Вам наверняка интересен не сам факт наличия константы где-то в недрах кода, а то, в каком контексте эта постоянная используется. Иными словами, если вы знаете, что программа сравнивает число записей в базе данных с некоторым значением (в нашем случае — 32h), то среди всех строк, в которых присутствует эта константа, в первую очередь следует рассматривать команды сравнения (cmp) и вычисления разности (sub и sbc), и лишь затем — все прочие, менее очевидные способы сравнения, вроде

```
mov ebx,50
cmp eax, ebx
```

или

```
push 50
push eax
call Compare2dwords
```

Ну и, раз уж речь зашла о сравнениях, нельзя не упомянуть об альтернативных вариантах реализации этой, казалось бы, нехитрой операции. Поразмыслим над приведенным выше примером сравнения содержимого регистра eax с числом 50. В самом деле, условия eax>50 и eax≥51 в приложении к целым числам имеют один и тот же смысл, а код

```
cmp  eax,50
jg my_label
```

работает совершенно аналогично коду

```
cmp eax,51
jge my_label
```

Если же eax необходимо сравнить с числом 31, проверка может выглядеть даже так:

```
and eax, 0FFFFFFE0h
jnz my_label
```

Т. е. проверка одного и того же условия может быть реализована несколькими разными способами, и когда вы будете искать константы в дизассемблированном тексте, этот факт тоже надо учитывать.

Если рассматривать области возможного практического применения вышеописанного приема, то лучше всего поиск известной константы в дизассемблированном тексте работает на триальных ограничениях типа "не допускается создание больше N элементов в базе данных". Как правило, N больше 7 и является целым числом, что облегчает поиск нужной константы. Исходя из принципа минимального вмешательства, для обезвреживания таких ограничений я предпочитаю исправлять не команды сравнения, а сами константы. Действительно, если программа для проектирования интерьера комнаты не способна работать более чем с 20 объектами, для практического применения она вряд ли будет пригодна. Но вот та же программа, где максимальное количество обрабатываемых объектов увеличено до двух с хвостиком миллиардов, наверняка удовлетворит даже самого взыскательного пользователя.

Один из наиболее частых вопросов, возникающих у начинающего крэкера, звучит так: "Программа работает 30 дней, но я так и не нашел в листинге сравнения с числом 30. Что делать?". Один из факторов я уже описал выше — там могло быть сравнение не с числом 30, а с числом 31. Однако этим список возможных причин неудачи не исчерпывается. Как мы все знаем, день состоит из 24 часов, каждый из которых состоит из 60 минут, каждая из которых состоит из 60 секунд. Более того, продолжительность секунд также может измеряться во всевозможных "условных единицах", например, в миллисекундах. Тысячные доли секунд, в частности, используются в таких функциях WinAPI как SetTimer (таймеры Windows часто используются для установки ограничений на продолжительность одного сеанса работы с программой) или Sleep. А вот в функциях, возвращающих время в виде структуры типа FILETIME, используются уже другие "условные единицы", равные одной миллионной доле секунды. Так что пресловутые 30 дней — это не только 30 дней, но еще и 720 часов, 43200 минут, 2592000 секунд, ну и так далее. И каждое из этих значений может быть использовано в программе как один из аргументов операции сравнения. Надо отметить, что в "условных единицах" может быть представлено не только время, но и многие другие величины: масса, географические координаты, денежные единицы и т.д.

Раз уж речь зашла о представлении временных отрезков внутри программ, уместно будет рассказать и о тонкостях использования таймеров. Наверняка вы встречали программы, в которых после запуска окно с предложением о регистрации висит на экране в течение некоторого времени (иногда еще в этом окне идет обратный отсчет секунд), и при этом его невозможно закрыть. Подобные "спецэффекты" по усмотрению разработчика могут сопровождать вызов каких-либо функций программы, сохранение файлов или завершение приложения — это не суть важно. Важно другое: все эти временные задержки, так или иначе, используют средства измерения времени. В ОС Windows существует два наиболее популярных способа отсчета отрезков времени — использование функций задержки (в частности, функции Sleep) и использование всевозможных таймеров.

Вообще в Windows существует несколько разновидностей таймеров — кроме обычного таймера, создаваемого функцией SetTimer, существует еще высокоточный мультимедийный таймер и специфические таймерные функции DirectX. Эти таймеры срабатывают с некоторой заданной частотой, вызывая функцию-обработчик (она же callback-функция), внутри которой и выполняются необходимые действия, например, тот же обратный отсчет секунд до исчезновения окна с предложением зарегистрироваться. Периодичность срабатывания таймера почти всегда является константой, однако взаимосвязь между тем, что происходит внутри программы, и тем, что вы можете видеть на экране, не всегда очевидна. Чтобы пояснить эту мысль и заодно продемонстрировать на практике, как можно обращаться с таймерами, приведу несколько примеров.

Первый пример простейший — программа, при запуске показывавшая в течение пяти секунд рекламный баннер, при этом поверх баннера выводился обратный счетчик секунд, регистрация в программе не предусматривалась. Ди-

зассемблирование показало, что таймер срабатывает каждые 1000 миллисекунд, при каждом вызове callback-функции значение переменной, изначально равной пяти, уменьшалось на единицу, и результат проверялся на равенство с нулем. В той конкретной программе баннер можно было просто "выломать", убрав функцию создания и отображения рекламного окна, но в общем случае это решение было бы не лучшим (вспомните принцип минимального вмешательства). И вот почему — на последнее срабатывание таймера могло быть "подвешено" не только закрытие окна с баннером, но и инициализация каких-либо объектов внутри программы или другие критичные действия, без которых программа могла бы работать некорректно. Так что немного усложним задачу — будем считать, что полностью убирать вызов окна с рекламой нельзя. Первое, что нам приходит в голову — уменьшить число секунд, в течение которых показывается баннер. Сказано — сделано, цифру 5 исправляем на единицу. Однако баннер все равно висит целую секунду — ведь первое срабатывание таймера наступает только через секунду после его создания. Теперь уменьшим период таймера до нуля (хотя лучше все-таки до одной миллисекунды, "таймер с периодом 0 миллисекунд", согласитесь, штука довольно странная). В результате мы получили баннер, появляющийся при запуске программы лишь на мгновение и не заставляющий тратить целых пять секунд на праздное разглядывание рекламных лозунгов.

В качестве второго примера я возьму одну из старых версий TVTools. В справке к программе было четко указано, что незарегистрированная версия работает только 10 минут. Дизассемблирование и анализ листинга выявили, что программа создает два таймера с периодами 60 секунд (что навело меня на мысли о защитном назначении этого таймера) и 2 секунды. Без особых сложностей обезвредив первый таймер, я запустил программу и обнаружил, что она все равно больше 10 минут не работала. Тогда я более пристально изучил callback-функцию второго таймера и наткнулся в ней на такой код:

```
inc     dword_40D5A7
cmp     dword_40D5A7, 136h
jbe     short loc_405CAA
```

Нетрудно догадаться, что это увеличение некоего счетчика, который затем сравнивается с числом 310. Поскольку период таймера — 2 секунды, а 310*2=620 (т.е. чуть больше 10 минут), логично было предположить, что это и есть второй уровень защиты, дублировавший первый. Очевидно, что если бы я принял на веру, что программа перестает работать ровно через 10 минут (а не через 10 минут 20 секунд, как это оказалось в действительности) и стал бы искать сравнение с числом 300, я бы не смог обнаружить таким способом вторую проверку времени работы программы. Этот пример демонстрирует один из неочевидных приемов, который может быть использован для реализации такой, казалось бы, простой операции как отсчет 10-минутного интервала. Также из этих примеров следует и другой, не менее важный вывод — далеко не всегда следует искать известную константу, чтобы найти код, в котором она используется. Иногда следует поступать прямо противоположным образом — сначала искать код, выполняющий нужные действия, и лишь затем выяснять, какая константа внутри этого кода ответственна за интересующие нас действия.

(Продолжение следует)

Приложение Windows "голыми руками"

Р.ГАЛЕЕВ /HI-TECH,
Башкортостан, г.Туймазы.

Как известно, для того чтобы небрежно называть себя программистом в компании друзей-чайников, необходимо (и достаточно!) написать программу, выдающую тем или иным способом на экран надпись "Hello, World!". Теперь Windows позволяет сделать это очень просто. Наберите в старом добром Блокноте — MsgBox "Hello, World!", сохраните файл с расширением .vbs (например, 'Hello.vbs'), затем запустите его двойным щелчком. Те, у кого установлен Word из MS Office XP, могут использовать более изощренный вариант:

```
Set w = CreateObject("Word.Application")
w.Visible = True
Set rng = w.Documents.Add.Range(0,0)
With rng
      .InsertBefore "Hello, World!"
      .ParagraphFormat.Alignment = 1
      With .Font
            .Name = "Arial"
            .Size = 48
            .Italic = True
            .Color = 200
      End With
End With
```

Но все это к самой операционной системе имеет лишь отдаленное отношение. Можно ли создать "настоящее" приложение Windows, не используя довольно громоздкие среды разработки, на самом обычном компьютере? Оказывается, можно.

Впервые запустив вновь установленный Windows XP Pro и набрав в командной строке debug, я был весьма удивлен, увидев дефис в консольном окне — знакомое приглашение старого отладчика DOS. Эта любопытная вещица является именно тем инструментом, который нам нужен. Кроме него, нам потребуется немного знаний о формате исполняемых файлов PE и процессе их загрузки в память.

Исполняемые файлы Win32 EXE используют формат файла PE. Как и старый формат EXE для DOS, PE-файл состоит из заголовка и собственно образа исполняемой программы. Образ программы составлен из одного или нескольких объектов или секций, которые по старинке иногда называют сегментами. Однако они не имеют ничего общего со старой сегментной моделью, также, впрочем, как и с объектами в том значении, в котором они используются в языках программирования. Поэтому для обозначения разделов образа программы PE-файла лучше использовать термин "секции".

Разделение на секции нацелено главным образом на оптимизацию управления памятью Windows. По этой причине размеры загруженных в оперативную память секций должны быть кратны размеру страницы памяти (обычно 4 Кб) и выровнены по ее границе. Записанные в файл секции должны быть выровнены по границе "файловых страниц", размер которых кратен размеру сектора (512 байтов) — это также сделано для оптимизации загрузки.

Образ программы загружается в память, начиная с некоторого указанного в заголовке файла базового адреса загрузки, который должен быть выровнен по 64 Кб-границе. Для EXE-файлов он обычно равен 400000h.

Чтобы наши выкладки не выглядели чересчур абстракт-

ными, сразу приступим к конструированию нашего PE-файла. Примем минимально возможные размеры выравнивания: секций — 4000 (1000h) байтов, файла — 512 (200h) байтов. Образ нашей программы будет состоять всего из одной секции, в которой будут размещены и данные, и код, и вспомогательные таблицы для импорта. Таким образом, размер файла уместится в 1 Кб, а размер образа в памяти — в 8 Кб (2 страницы). Первую страницу по адресу загрузки 400000h займет заголовок PE-файла, а со смещения 1000h будет располагаться первая (и единственная) секция нашего файла.

Итак, создадим "болванку". В командной строке набираем "debug". Очистим 1000 (400h — в отладчике debug используется шестнадцатеричная запись чисел) первых байтов памяти, заполнив их нулями:

```
f 0 400 0
```

В этой области мы будем "собирать" наше приложение. Но для удобства набора используем еще одну область по смещению 1000h, чтобы не запутаться с адресами при наборе:

```
f 1000 1200 0
```

Как и для сценария VBS, работа нашего приложения будет заключаться в отображении окна сообщения "Hello, World!", для чего необходимо использовать функцию Win32 API MessageBoxA (из системного модуля USER32.dll). После этого приложение завершает работу (вызвав еще одну функцию API ExitProcess из модуля KERNEL32.dll). Таким образом, необходимо сначала импортировать указанные две функции в наше приложение.

Процесс импорта заключается, во-первых, в отображении (посредством страничной переадресации) нужных нам DLL в адресное пространство нашего процесса и, во-вторых, в сохранении адресов нужных нам функций из этих DLL в специально отведенных для этого местах — таблицах импортируемых адресов (Import Address Table — IAT). Все это автоматически проделывается системой при загрузке файла на исполнение, но для этого необходимо предусмотреть в образе программы ряд вспомогательных таблиц для импорта.

Таблицы импортируемых адресов (IAT) должны располагаться в самом начале секции. Они представляют собой последовательности 4-байтовых (DWORD) полей, в которые загрузчик Windows заносит при связывании с DLL адреса соответствующих импортируемых функций. Порядок расположения функций должен соответствовать порядку функций в таблице поиска (об этом далее). Каждая таблица содержит данные о функциях из одного модуля (DLL); признаком конца таблицы является поле, заполненное нулями. При необходимости, несколько таблиц следуют одна за другой. До связывания с DLL таблица импортируемых адресов должна быть полностью идентична соответствующей таблице поиска, в противном случае загрузчик сообщит об ошибке.

В нашем случае необходимы две IAT: одна — для USER32.dll, другая — для KERNEL32.dll. Из обоих модулей импортируется по одной функции, поэтому размер обеих таблиц будет по 8 байтов (4 — на адрес, 4 — на завершающие 0). Первая IAT будет располагаться по смещению 1000h относительно базового адреса загрузки, вторая — по 1008h. Эти значения мы введем позже.

А пока займемся данными. Функция MessageBoxA принимает в числе прочих параметров адреса двух строк (одна — собственно выводимое сообщение, вторая — заголовок). Выровняем адреса по границе параграфа (это не обязательно, и сделано лишь для удобства, чтобы не запутаться с адресами). По смещению 1010h поместим ASCII-строку 'VBScript' (чтобы заголовок сообщения был аналогичен заголовку сообщения сценария VBS):

```
a 1010
db 56,42,53,63,72,69,70,74
<Enter>
```

По смещению 1020h поместим строку 'Hello, World!' и оставим побольше места (на случай, если потом захотим изменить сообщение):

```
a 1020
db 48,65,6c,6c,6f,2c,20,57,6f,72,6c,64,21
<Enter>
```

По смещениям 1040 и 1050 поместим имена импортируемых модулей USER32.dll и KERNEL32.dll соответственно, на которые будет ссылаться таблица импорта:

```
a 1040
db 55,53,45,52,33,32,2e,64,6c,6c
<Enter>
a 1050
db 4b,45,52,4e,45,4c,33,32,2e,64,6c,6c
<Enter>
```

Необходимо также предусмотреть имена импортируемых функций, но для них используются строки особого формата — первые два байта строки являются "подсказкой" загрузчику, и лишь после них идет собственно имя. Подсказка представляет собой индекс массива указателей экспортируемых имен DLL, по которому предположительно можно обнаружить искомое имя функции. Если в указанном месте нужное имя отсутствует (например, вследствие изменения версии DLL), загрузчик выполняет поиск данного имени по всему списку экспортируемых имен, на что, естественно, требуется больше времени. В нашем случае придется обойтись без подсказок — на их месте оставим нули. По смещению 1060h поместим имя функции — 'MessageBoxA', по смещению 1070h — 'ExitProcess' (необходимо помнить, что, в отличие от имен DLL, в именах функций учитывается регистр символов):

```
a 1060
db 0,0,4d,65,73,73,61,67,65,42,6f,78,41
<Enter>
a 1070
db 0,0,45,78,69,74,50,72,6f,63,65,73,73
<Enter>
```

Поскольку текстовые данные закончились, самое время проверить правильность введенных данных. Для дампа памяти используется команда отладчика d:

```
d 1000
```

Если данные были введены правильно, в правой части экрана должны отобразиться в текстовом виде введенные нами имена. При ошибке нужно повторно ввести данные по тому же адресу.

Далее (по смещению 1080h) разместим таблицы поиска. Аналогично IAT, таблица поиска для импортируемых из одного модуля функций состоит из последовательного ряда 32-разрядных (DWORD) значений, завершающихся нулевым полем. Поля таблицы указывают на способ поиска функций в списках экспорта DLL — по порядковым номерам или по именам. В последнем случае поле содержит смещение на строку "подсказка-имя" для искомой функции. В нашем случае соответствующие смещения равны 1060h для 'MessageBoxA' и 1070h для 'ExitProcess' (к сожалению, debug не признает 32-разрядных чисел, поэтому придется вводить их как пары 16-разрядных; при этом надо учесть, что в PC для чисел применяется обратный порядок следования байтов):

```
a 1080
dw 1060,0,0,0,1070,0,0,0
<Enter>
```

Поскольку IAT при загрузке должна быть идентична таблице поиска, теперь мы можем вернуться и заполнить оставленные ранее пустыми поля:

```
a 1000
dw 1060,0,0,0,1070,0,0,0
<Enter>
```

Мы добрались до самой главной таблицы — таблицы импорта (не путать с таблицей импортируемых адресов). Таблица импорта связывает воедино все подготовленные ранее данные. Каждая строка таблицы импорта состоит из пяти 4-байтовых (DWORD) полей и относится к одному импортируемому модулю (DLL). Первое поле содержит смещение (относительно базового адреса загрузки) таблицы поиска для данной DLL; второе и третье не используются и содержат нули; четвертое — смещение на строку с именем DLL; пятое — смещение на соответствующую IAT. Число входов в таблицу импорта равно числу импортируемых модулей плюс одна строка, в которой все поля заполнены нулями для обозначения конца таблицы. В нашем случае таблица импорта будет состоять из 3 входов (для USER32.dll, KERNEL32.dll и один пустой). Таблица располагается по смещению 1090h и имеет размер 3x5x4=60 (3Ch) байтов:

```
a 1090
dw 1080,0
```

(смещение первой таблицы поиска),

```
dw 0,0,0,0
```

(два пустых поля),

```
dw 1040,0
```

(смещение на строку с именем USER32.dll),

```
dw 1000,0
```

(смещение первой IAT).

Аналогично заполняем вторую строку:

```
dw 1088,0,0,0,0,0,1050,0,1008,0
<Enter>
```

Следующие 20 байтов оставляем пустыми.

Осталось ввести только сам код. Функция MessageBoxA принимает 4 DWORD-параметра: дескриптор окна приложения (в нашем случае окно отсутствует, т.е. 0), указатель на строку сообщения, указатель на строку заголовка и тип окна сообщения (числовая константа; здесь — 0). Параметры функциям Win32 API передаются через стек в обратной последовательности, т.е. первым помещается в стек последний параметр. Поэтому на ассемблере код мог бы выглядеть примерно так:

```
push 0
push offset title        ; здесь - 401010h
push offset message      ; здесь - 401020h
push 0
call IAT[1]              ; адрес функции MessageBoxA
```

Нужно учесть, что в стек помещаются не смещения, а линейные адреса, поэтому к смещениям строк 1020h и 1010h необходимо добавить базовый адрес загрузки (400000h), получив соответственно 401020h и 401010h; а для импортируемого адреса MessageBoxA — 401000h. Поскольку debug не работает с 32-разрядными смещениями, придется кодировать все самостоятельно (не забыв про обратный порядок байтов в числах):

```
a 10d0
db 6a,0
db 68,10,10,40,0
db 68,20,10,40,0
db 6a,0
db ff,15,0,10,40,0
```

ExitProcess (адрес которого хранится во второй IAT по линейному адресу 401008h) принимает лишь один аргумент — код завершения (в нашем случае — 0):

```
db 6a,0
db ff,15,8,10,40,0
<Enter>
```

Вот и вся программа. У нас получился как бы "образ в памяти" (по смещению 1000h), теперь надо перенести его на свое место в файле (по смещению 200h):

```
m 1000 1200 200
```

Осталась самая малость — заполнить заголовок. Заголовок PE-файла можно разделить на "старый" и "новый". "Старый" заголовок, в свою очередь, состоит из несколько модернизированного заголовка EXE-DOS и необязательной программы-заглушки DOS, которая обычно выводит текст "This program cannot be run in DOS mode", когда PE-файл по ошибке пытаются запустить в DOS. Но вместо нее, в принципе, может быть любая другая DOS-программа. Модернизация DOS-заголовка заключается, во-первых, в резервировании по смещению 20h с начала файла места для идентификатора и имени производителя программы, которое практически всегда остается пустым. Во-вторых, и это уже существенно, по смещению 3Ch находится 32-разрядный указатель на PE-заголовок.

Единственное, что нам нужно оставить в заголовке DOS — это сигнатура EXE-файла (ASCII-символы 'MZ'):

```
a 0
db 4d,5a
<Enter>
```

Программу-заглушку мы вообще опустим. Таким образом, PE-заголовок будет следовать непосредственно за 4-байтным указателем по смещению 3Ch, т.е. по смещению 40h. Это значение и запишем в качестве указателя:

```
a 3c
db 40
<Enter>
```

"Новый" заголовок представлен собственно PE-заголовком и так называемой таблицей объектов. PE-заголовок, в свою очередь, делится на стандартный и NT-заголовок. В конце последнего дополнительно выделяют каталог смещений/размеров. Каждый вход каталога представлен парой DWORD-значений, первое из которых содержит смещение соответствующей служебной таблицы относительно базового адреса загрузки, второе — размер таблицы. Если какая-либо из таблиц не используется, соответствующие поля содержат нули. В **табл.1** приведены только те поля PE-заголовка, без которых запуск программы невозможен.

Табл. 1

Смещение	Размер (байтов)	Описание
Стандартный		
00h	4	Сигнатура: ASCII-символы "PE" и два нулевых байта
04h	2	Тип процессора (обычно 14Ch для i386)
06h	2	Число секций в образе программы
14h	2	Размер NT-заголовка вместе с катвлогом смещений; обычно E0h
16h	2	Флаги программы; для Win32-приложений обычно 10Fh
NT-заголовок		
18h	2	"Магическое" значение 10Bh
28h	4	Точка входа в программу (смещение)
34h	4	Бвзовый адрес загрузки (для EXE обычно 400000h)
38h	4	Выравнивание секций в ОЗУ (размер системной страницы памяти, 4 Кб = 1000h)
3Ch	4	Файловая страница: выравнивание секций в файле (кратно 512 (200h) байт)
40h	2	Старший номер версии Windows; обычно 4
48h	2	Старший номер версии подсистемы Windows; обычно 4
50h	4	Рвзмер загруженного образа вместе со всеми заголовками; кратен размеру выравнивания секций
54h	4	Общий размер всех заголовков (старых и новых)
5Ch	2	Тип приложения (2 — графическое, 3 — консольное)
74h	4	Число входов в каталоге смещений/размеров (обычно 10h)
Каталог смещений/размеров		
80h	4	Смещение таблицы импорта
84h	4	Размер таблицы импорта

Примечание. Для запуска нашего приложения достаточно заполнить указанные в таблице поля (проверено для трех версий Windows: 98 SE, 2000 Server и XP Pro). Однако для более сложных программ может потребоваться также заполнить 4-байтные (DWORD) поля по смещениям 60h (резервируемый размер стека), 64h (выделенный размер стека), 68h (резервируемый размер кучи), 6Ch (выделенный размер кучи).

Таблица объектов следует непосредственно за PE-заголовком и описывает секции (объекты) образа программы. Она фактически является картой отображения записанных на диске секций в память. Число входов в таблицу объектов равно числу секций в образе программы и указывается в PE-заголовке в поле со смещением 6. Каждый вход таблицы имеет следующий формат (**табл.2**):

Табл. 2

Смещение	Размер (байтов)	Описание
0	8	Произвольное имя секции (используется при компоновке). Заполняется нулями до 8 байтов.
8	4	Количество памяти, отводимое для загрузки секции
0Ch	4	Размещение секции в памяти, смещение относительно базового адреса загрузки
10h	4	Размер секции в файле, выровненный по границе файловой страницы
14h	4	Смещение секции в файле, выровненное по границе файловой страницы
18h	12	Зарезервировано; используется в объектных файлах
24h	4	Флаги секции. Наиболее употребляемые: 20h — секция кода; 40h — инициализированные данные; 80h — неинициализированные данные; 20000000h — разрешено исполнение; 40000000h — разрешено чтение; 80000000h — рвзрешена запись

Завершим создание заголовка. По смещению 40h находится сигнатура PE:

```
a 40
db 50,45,0,0
```

Процессор — i386, число секций — 1:

```
dw 14c,1
<Enter>
```

Размер NT-заголовка E0h, флаги программы, "магическое" значение:

```
a 54
dw e0,10f,10b
<Enter>
```

Точка входа в программу:

```
a 68
dw 10d0
<Enter>
```

Базовый адрес загрузки 40 0000h (в обратном порядке), выравнивание в памяти — 1000h, в файле — 200h:

```
a 74
dw 0,40,1000,0,200,0
```

Версия ОС — 4.0, версия подсистемы — 4.0; промежутки заполняются нулями:

```
dw 4,0,0,0,4
<Enter>
```

Размер образа с заголовками в памяти — 2000h, размер заголовка в файле — 200h, подсистема 2:

```
a 90
dw 2000,0,200,0,0,0,2
<Enter>
```

10h входов в каталоге смещений:

```
a b4
dw 10
<Enter>
```

В каталоге смещений используем лишь один вход: смещение таблицы импорта — 1090h, размер — 3Ch. Остальные входы оставляем пустыми:

```
a c0
dw 1090,0,3c
<Enter>
```

По смещению 140h начинается таблица объектов, в нашем случае она имеет лишь один вход. Никакого имени секции давать не будем. Секция занимает в памяти 1000h байтов, начиная со смещения 1000h; в файле она занимает 200h байтов и находится по смещению 200h:

```
a 140
dw 1000,0,1000,0,200,0,200,0
<Enter>
```

И, наконец, флаги — секция является кодовой, имеет разрешения на исполнение, чтение и запись. Сумма всех флагов (в данном случае это эквивалентно побитовому OR) составит E0000020h (порядок слов — обратный):

```
a 15c
dw 20,e000
```

Отладчик debug позволяет сохранять файлы лишь в com-формате, при этом первые 100h байтов на диск не записываются. Поэтому необходимо сначала "сдвинуть" весь (400h) образ в памяти на 100h байтов:

```
m 0 400 100
```

Озаглавим наш файл. Т.к. отладчик debug не позволяет записывать файлы в формате exe, сначала сохраним файл с расширением .bin, потом переименуем его в .exe.

```
n hello.bin
```

Количество записываемых байтов заносится в регистр CX, после чего осуществляется собственно запись командой w:

```
r cx
400
w
```

Для выхода из отладчика используется команда q. Но перед этим нужно еще раз тщательно проверить правильность всех введенных данных. При ошибочном расположении чисел в таблицах попытка запуска может вызвать сообщение: "<Имя программы> не является приложением Win32", а то и вообще вызвать сбой системы.

Ну что же, примите мои поздравления — вы создали полноценное Win32-приложение даже не на ассемблере, а в машинных кодах!

Сжатие данных: от битов к байтам

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

Хотите, покажу фокус? Смотрите, имеется bmp-файл длиной 1406 байтов, содержащий вот такой черно-белый рисунок (**рис.1**). На рисунке изображены восемь одинаковых объектов. Налицо явная избыточность, так что, по идее, файл должен хорошо сжиматься.

Попробуем это проверить. Сожмем файл с помощью архиватора PKZIP 2.50:

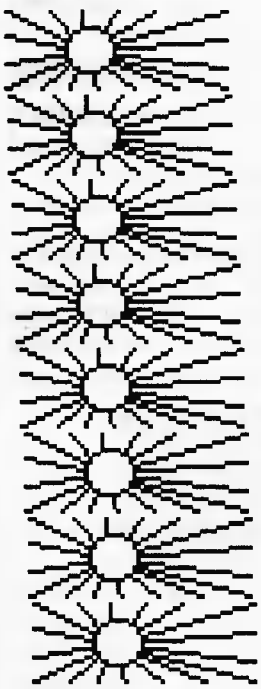
```
pkzip 1.zip 1.bmp
```

В результате получаем архив 1.zip длиной 1073 байта, что всего на 24% меньше длины исходного файла.

А теперь обработаем исходный bmp-файл с помощью "волшебной" программы, текст которой на языке С приведен ниже.

```
/* bit2byte.c */

#include <stdio.h>

void main (int argc, char *argv[])
{
 FILE *f_src, *f_dst;
 int i,a,b;

 if (argc<3)
 (printf ("Syntax: bit2byte.exe source-file destiny-file\n");
                                                    exit();}
 f_src=fopen(argv[1],"rb");
 if (f_src==NULL) [printf ("Can't open source file\n"); exit();}
 f_dst=fopen(argv[2],"wb");
 if (f_dst==NULL) (printf ("Can't open destiny file\n");
                                                    exit();}

 while ((a=fgetc(f_src))!=EOF)
 {
  for (i=0;i<8;i++)
  {
   if ((a&0x80)==0) b=0; else b=1;
   fputc (b,f_dst);
   a=a<<1;
  }
 }

 fcloseall ();
}
```

Рис. 1



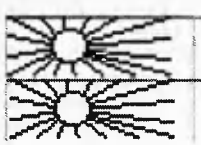
При запуске этой программы в командной строке надо указать имя исходного файла (находящегося в текущем каталоге) и имя формируемого выходного файла (он будет сформирован также в текущем каталоге). Запускаем программу с такими параметрами:

```
bit2byte.exe 1.bmp 1.bbb
```

Программа создаст файл 1.bbb, длина которого в восемь раз больше длины исходного файла и равна 11248 байтам. Зачем нужен этот файл? А вот смотрите, сейчас мы попробуем сжать его тем же PKZIP'ом:

```
pkzip 2.zip 1.bbb
```

Получили архив 2.zip, длина которого... всего 445 байтов! Это приблизительно в 2,4 раза меньше длины первоначального архива!



Естественно, в файле 1.bbb содержится вся информация, которая была в исходном файле 1.bmp. Эту информацию можно извлечь, обработав файл 1.bbb нижеприведенной программой.

```
/* byte2bit.c */

#include <stdio.h>

void main (int argc, char *argv[])
{
 FILE *f_src, *f_dst;
 int i,a;

 if (argc<3)
 {printf ("Syntax: byte2bit.exe source-file destiny-file\n");
                                                    exit();}
 f_src=fopen(argv[1],"rb");
 if (f_src==NULL) {printf ("Can't open source file\n"); exit();}
 f_dst=fopen(argv[2],"wb");
 if (f_dst==NULL) {printf ("Can't open destiny file\n");
exit();}

 while ((a=fgetc(f_src))!=EOF)
 {
  for (i=0;i<7;i++)
  {
   a=a<<1;
   a=a+fgetc(f_src);
  }
  fputc (a,f_dst);
 }

 fcloseall ();
}
```

При ее запуске, как и для предыдущей программы, надо указать в командной строке имена исходного и формируемого файлов. Запускаем:

```
byte2bit.exe 1.bbb 2.bmp
```

Получили файл 2.bmp, полностью идентичный первоначальному файлу 1.bmp (в этом легко убедиться, сравнив файлы, например, с помощью утилиты fc.exe).

В чем же секрет фокуса? Как получилось, что "раздутый" в восемь раз файл сжался гораздо лучше исходного? Что делают программы bit2byte и byte2bit?

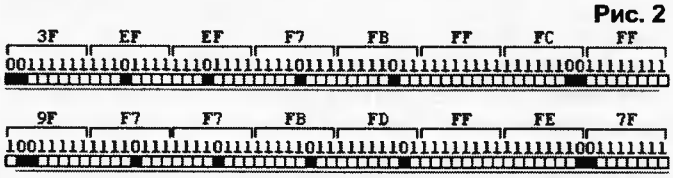
Черно-белое изображение в формате bmp хранится по строкам. Каждая строка представлена последовательностью байтов, а в каждом байте хранится информация о восьми пикселах. Каждому пикселу соответствует один бит: 0 — черный пиксел, 1 — белый.

Рассмотрим, как представлены в файле 1.bmp две строки изображения — строка, соответствующая верху первого объекта, и строка, соответствующая верху второго объекта (**рис.2**).

Что мы видим? Подчеркнутые последовательности пикселов в этих двух строках совпадают, а значит, совпадают и соответствующие последовательности битов в файле. А вот совпадений на байтовом уровне не наблюдается, так как второй объект сдвинут относительно первого на пиксел вправо.

Архиватор PKZIP является байт-ориентированным, т.е. как раз ищет совпадающие последовательности байтов. Совпадений на битовом уровне он просто "не видит". Поэтому он и смог сжать файл 1.bmp только на 24%.

Рис. 2



А что делает программа bit2byte? Она читает байты из исходного файла и после чтения каждого байта записывает в формируемый файл восемь байтов, представляющих собой значения битов прочитанного байта. Например, если из исходного файла прочитан байт 00110101, в формируемый файл будут записаны байты 0, 0, 1, 1, 0, 1, 0, 1.

Таким образом, после обработки файла 1.bmp программой bit2byte, последовательности битов в нем превратились в последовательности байтов в сформированном файле 1.bbb. Это позволило PKZIP'у при сжатии файла 1.bbb обнаружить в нем совпадающие последовательности, в результате чего и произошло значительное повышение степени сжатия.

Программа byte2bit, как вы уже, наверно, догадались, выполняет обратную задачу — прочитав из исходного файла (ранее полученного в результате работы программы bit2byte) очередные восемь байтов, она записывает в формируемый файл один байт, "собранный" из прочитанных значений.

Кроме PKZIP, я протестировал и несколько других архиваторов, сжимая с их помощью файлы 1.bmp и 1.bbb. Архиваторы запускались с параметрами по умолчанию. Результаты тестирования (включая и полученные ранее для PKZIP) вы можете увидеть в таблице.

PKZIP 2.50	1406	(100%)
	1073	(76%)
	445	(32%)
ARJ 2.60	1406	(100%)
	1058	(75%)
	412	(29%)
IMP 1.12	1406	(100%)
	1110	(79%)
	359	(26%)
HA 0.99c	1406	(100%)
	958	(68%)
	258	(18%)
RAR 2.90 beta 4	1406	(100%)
	1056	(75%)
	312	(22%)

Для каждого архиватора в таблице приведены три строки с информацией: первая строка соответствует исходному файлу 1.bmp (чтобы было с чем сравнивать), вторая — архиву, полученному при сжатии этого файла, и третья — архиву, полученному при сжатии файла 1.bbb. Информацию в каждой строке следует понимать следующим образом: первое число — длина файла в байтах, второе число — длина файла в процентах от длины исходного файла 1.bmp.

Как видно из таблицы, существенное улучшение сжатия файла 1.bbb по сравнению с файлом 1.bmp наблюдалось для всех архиваторов, принимавших участие в тестировании. Это можно считать косвенным свидетельством того, что эти архиваторы, как и PKZIP, являются байт-ориентированными.

Какой можно сделать вывод из всего вышесказанного? Если вы пользуетесь каким-либо из указанных выше архиваторов или другим байт-ориентированным архиватором, и сжимаемый файл представляет собой массив элементов, размер которых не кратен байту (элементы могут быть как одной какой-то длины, так и у разных элементов может быть разная длина), и в файле присутствуют совпадающие последовательности элементов (а значит, и совпадающие последовательности битов), то имеет смысл не сжимать такой файл непосредственно, а воспользоваться программой bit2byte и сжимать уже полученный "раздутый" файл (а при извлечении файла из архива — соответственно, воспользоваться программой byte2bit, чтобы восстановить исходный файл).

Исполняемые файлы (для DOS) программ bit2byte и byte2bit вы можете скачать с сайта журнала или с моей web-страницы. Там же доступен и файл 1.bmp (на случай, если вы захотите проверить приведенные в статье результаты или протестировать еще какие-нибудь архиваторы).

Сделай сам: "Вскрывалка" паролей

И.ШИРКО,
E-mail: FDC@tut.by

Каждому пользователю ПК время от времени приходится вводить пароли: для установки программы, для соединения с интернетом, для получения почты и т.д. Как правило, при вводе пароля, вместо нормальных символов появляются звездочки ("*").Такая конспирация используется не только для сокрытия пароля от посторонних глаз, но и для того, чтобы злоумышленник, получивший доступ к компьютеру, не мог узнать пароли, сохраненные в системе. Но, как известно, совершенной защиты нет, поэтому теперь можно найти много программ для извлечения паролей из-под "звездочек". Некоторые программы могут показывать пароль в отдельном окошке, другие же неким чудесным образом заставляют "звездочки" превратиться в нормальные символы. В этой статье мы рассмотрим процесс создания программы, которая умеет "выковыривать" пароли двумя вышеупомянутыми способами.

Но вначале немного теории. Windows, судя по ее названию, это "сборище" окон. Каждое окно обычно содержит другие окна, т.е. является родителем для некоторых других окон (эти "другие окна" называют дочерними). Для нашей программы нам достаточно знать, что то поле, куда мы вводим пароли, является хоть и маленьким, но окном. А значит, как и любое порядочное окно, оно имеет параметр "текст окна" и умеет принимать/отсылать сообщения. Именно на данных весьма полезных свойствах полей ввода и основана работа программ для извлечения паролей из-под "звездочек".

Итак, первый способ для "вскрытия" паролей — нужно просто узнать текст окна, содержащего пароль. Вот и все, никаких сложностей. Правда, работать данный способ будет только в Win9x и, возможно, в WinMe. Осталось только написать функцию для получения текста произвольного окна:

```
Function GetText(WindowHandle: hwnd):string;
var
  txtLength : integer;
  buffer: string;
begin
//узнаем длину текста
  TxtLength := SendMessage(WindowHandle, WM_GETTEXTLENGTH, 0, 0);
  if txtlength>0 then
  begin
    txtlength := txtlength + 1;
    setlength (buffer, txtlength);
    //записываем текст окна в buffer
    sendmessage(WindowHandle, wm_gettext, txtlength,
                                          longint(@buffer[1]));
    result := buffer;
  end else result:='';
end;
```

Теперь, если мы вызовем эту функцию, записав в параметр идентификатор поля ввода пароля, то она возвратит нам сам пароль. Как узнать идентификатор нужного окна, будет рассказано далее. А сейчас поговорим о втором способе извлечения пароля. Только в данном способе мы не запишем пароль в какую-либо переменную, а заставим поле ввода отобразить вместо звездочек нормальные символы. Для этого нужно послать ему сообщение EM_SETPASSWORDCHAR. Вообще-то, это сообщение используется и для того, чтобы нормальный текст заменить каким-либо символом (обычно

используются звездочки). Ниже приведена процедура, которая не просто показывает нормальный текст вместо звездочек, но, если у выбранного поля ввода текст не зашифрован, то мы заменяем его звездочками. Т.е., если применить эту процедуру для поля ввода пароля, то появятся нормальные символы, а если для того же поля повторить операцию, то текст опять превратится в звездочки.

```
procedure ShowPass(h:hwnd);
var
  ch:integer;
  i:integer;
begin
  //узнаем, закодирован ли текст
  ch:=SendMessage(h, EM_GETPASSWORDCHAR, 0, 0);
  //если текст закодирован, то раскодируем его,
  //иначе кодируем звездочками
  if ch>0 then
    i:=0
  else
    i:=ord("*");
  SendMessage(h, EM_SETPASSWORDCHAR, i, 0);
end;
```

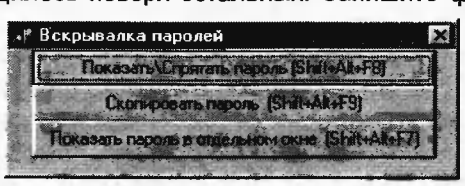
Теперь можно приступить к созданию самой программы, но перед этим нужно оговорить принципы ее функционирования:

Пользователь должен каким-то образом указать программе, какой именно пароль он хочет "вскрыть". Наиболее удобным для юзера способом мне представляется следующий — пользователь наводит курсор мышки на поле ввода с нужным паролем, нажимает определенную комбинацию клавиш и получает взамен "звездочек" нормальный пароль.

Создадим функцию копирования паролей в буфер обмена (этого нет в подобных программах!). Windows не позволяет просто взять и скопировать пароль, поэтому будем делать так:

- заменяем "звездочки" на обычные символы;
- выделяем весь текст в поле ввода;
- копируем выделенный текст в буфер обмена;
- обратно маскируем пароль "звездочками".

Теперь запускаем Delphi и делаем формочку, взяв за образец **рисунок**. В свойстве формы FormStyle установите константу fsStayOnTop, чтобы окно нашей программы находилось поверх остальных. Запишите функцию



GetText и процедуру ShowPass (см. выше). В разделе **Var** нужно объявить несколько глобальных переменных:

```
p:TPoint;
h:HWND;
ch:Integer;
s:String;
```

Процедура обработки нажатия на кнопку "Показать\Спрятать пароль":

```
procedure TForm1.Button1Click(Sender: TObject);
begin
//узнаем координаты курсора мыши
  getcursorpos(p);
//получаем идентификатор окна, находящегося под курсором мыши
  h:=windowfrompoint(p);
//прячем\показываем пароль
  ShowPass(h);
//перерисовываем окно
  InvalidateRect(h, nil, true);
end;
```

Данная процедура позволяет маскировать/демаскировать текст почти любого поля ввода.

Процедура обработки нажатия на кнопку "Скопировать пароль":

```
procedure TForm1.Button2Click(Sender: TObject);
begin
//узнаем координаты курсора мыши
  getcursorpos(p);
//получаем идентификатор окна, находящегося под курсором мыши
  h:=windowfrompoint(p);
//каким символом замаскирован текст в данном окне?
  ch:=SendMessage(h, EM_GETPASSWORDCHAR, 0, 0);
//если пароль замаскирован, то показываем его в нормальном виде
  if ch>0 then
    SendMessage(h, EM_SETPASSWORDCHAR, 0, 0);
//выделяем весь текст
    SendMessage(h, EM_SETSEL, 0, -1);
//копируем выделенный текст
    SendMessage(h, WM_COPY, 0, 0);
//если мы "вскрывали" пароль, то опять маскируем его
  if ch>0 then
    SendMessage(h,EM_SETPASSWORDCHAR, ch, 0);
//перерисовываем окно
    InvalidateRect(h, nil, true);
end;
```

Процедура обработки нажатия на кнопку "Показать пароль в отдельном окне":

```
procedure TForm1.Button3Click(Sender: TObject);
begin
//узнаем координаты курсора мыши
  getcursorpos(p);
//получаем идентификатор окна, находящегося под курсором мыши
  h:=windowfrompoint(p);
//получаем текст окна
  s:=gettext(h);
//если текст не является пустой строкой, то показываем его пользователю
  if s<>'' then
  begin
    setforegroundwindow(form1.handle);
    showmessage(s);
  end;
end;
```

С помощью данной процедуры (работает она только в Win9x) можно узнать не только пароль, скрытый под "звездочками", но и текст практически любого стандартного элемента управления Windows: поля ввода, кнопки, флажка (checkbox) и др. Напомню, что текст копируется в буфер обмена, работа с которым рассматривалась в [1].

В принципе, программа уже готова к эксплуатации. Но для удобства пользователя сделаем, как и собирались, поддержку "горячих клавиш". Для этого мы воспользуемся функцией

```
RegisterHotKey(
    HWND hWnd, // этому окну придет уведомление о нажатии
                  // комбинации клавиш
    int id,       // идентификатор "горячих клавиш"
    UINT fsModifiers, // должны ли быть нажаты клавиши Ctrl,
                  // Shift или Alt?
    UINT vk    // код клавиши, на которую мы будем реагировать
   );
```

Для события OnCreate формы запишите процедуру:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
//регистрируем сочетание Shift+Alt+F9
 If not RegisterHotkey
        (Handle, 1, MOD_ALT or MOD_SHIFT, VK_F9) Then
   ShowMessage("Нельзя использовать данное сочетание клавиш!");
//регистрируем сочетание Shift+Alt+F8
 If not RegisterHotkey
        (Handle, 2, MOD_ALT or MOD_SHIFT, VK_F8) Then
   ShowMessage("Нельзя использовать данное сочетание клавиш!");
//регистрируем сочетание Shift+Alt+F7
 If not RegisterHotkey
       (Handle, 3, MOD_ALT or MOD_SHIFT, VK_F7) Then
   ShowMessage("Нельзя использовать данное сочетание клавиш!");
end;
```

Теперь в секции ***Private*** объявите процедуру, которая будет реагировать на нажатие комбинаций клавиш:

```
Procedure WMHotkey( Var msg: TWMHotkey ); message WM_HOTKEY;
      А вот и сама процедура:
Procedure TForm1.WMHotkey( Var msg: TWMHotkey );
begin
 case msg.hotkey of
//если нажато Shift+Alt+F9, то копируем пароль
  1:begin
     getcursorpos(p);
     h:=windowfrompoint(p);
     ch:=SendMessage(h,EM_GETPASSWORDCHAR,0,0);
     if ch>0 then
       SendMessage(h,EM_SETPASSWORDCHAR,0,0);
     SendMessage(h,EM_SETSEL,0,-1);
     SendMessage(h,WM_COPY,0,0);
     if ch>0 then
       SendMessage(h,EM_SETPASSWORDCHAR,ch,0);
     InvalidateRect(h,nil,true);
    end;
//если нажато Shift+Alt+F8, то прячем/показываем пароль
   2:begin
      getcursorpos(p);
      h:=windowfrompoint(p);
      ShowPass(h);
      InvalidateRect(h,nil,true);
     end;
//если нажато Shift+Alt+F9, то показываем пароль в отдельном окне
   3:begin
      getcursorpos(p);
      h:=windowfrompoint(p);
      s:=gettext(h);
      if s<>'' then
        begin
         setforegroundwindow(form1.handle);
         showmessage(s);
        end;
     end;
 end;
end;
```

Вот и все. Осталось только откомпилировать программу, которая по своим возможностям превосходит все виденные мною аналоги. Правда, интерфейс у нее "хромает", но это уже не совсем из области программирования.

P. S. Разумеется, данную программу нельзя применять ни в каких деструктивных целях.

Литература

1. И.Ширко. Работа с буфером обмена в Delphi.— Радиомир. Ваш компьютер, 2002, №7, С.27.

ViNCE,
E-mail: vincelabs@pisem.net
www.virtuality.nightmail.ru

Реализация криптоалгоритма Вигенера (2)

Предисловие

В первой части статьи мы рассмотрели процесс реализации алгоритма Вигенера в среде программирования Delphi. Процесс кодирования оказался довольно прост в реализации и понимании, что, в свою очередь, позволило создать простую программу-шифратор. Теперь мы напишем алгоритм декодирования, который наглядно продемонстрирует обратный процесс — шифрации. Приступим.

Алгоритм декодирования

Итак, у нас есть закодированный текст, и нам необходимо узнать его информационную нагрузку, или смысл. Кликаем по кнопке bDeCrypt, и Delphi создаст новую процедуру обработки нажатия на кнопку алгоритма расшифровки.

В алгоритме шифрации мы использовали обьявление вспомогательных переменных, здесь мы также не отступим от правил и обьявим следующие переменные в разделе var (variables):

- buf — для хранения временных данных;
- i, b — для организации циклов (for);
- dectxt — расшифрованный текст;
- a1 — для хранение алфавита;
- chrA, chrB — для хранения порядковых номеров символов зашифрованного текста и ключа.

Помимо локальных переменных, область видимости которых ограничена процедурой, в которой они обьявлены, мы также добавим четыре глобальные переменные:

- Chr_Crypted — содержит текущий зашифрованный символ;
- Chr_DeCrypted — хранит текущий расшифрованный символ;
- global_txt — содержит зашифрованный текст;
- global_txt_chr — содержит текущий символ зашифрованного текста;
- global_key — содержит ключ;
- global_key_chr — содержит текущий символ ключа.

Переменные обьявлены, теперь необходимо провести их инициализацию:

```
dectxt:='';
cellfill();                          //Заполняем таблицу кодирования
b:=1;     //Устанавливаем цикл для ключа в начальное состояние
global_txt:=mText2.Text;         //Считываем зашифрованный текст
global_key:=eKey2.Text;                          //Считываем ключ
                                              //Определяем алфавит
a1:='ABCDEFGHIJKLMNOPQRSTUVWXYZ0987654321 .,:;<>_-|"/\*+?';
```

Все переменные установлены в "начальные" положения и готовы к использованию. Теперь можно приступить к непосредственному алгоритму декодирования:

```
//Расшифровка
  for i:=1 to length(global_txt) do
                        //От 1 до длины зашифрованного текста...
   begin
    if b>length(global_key) then   //Если b больше длины ключа
     b:=1;
    //Сбрасываем счетчик или переходим к первому символу ключа
     global_txt_chr:=global_txt[i]; //Записываем текущий символ
     global_key_chr:=global_key[i];
                              //Записываем текущий символ ключа
     ChrToNum(global_txt[i]);       //Получаем код буквы текста
     chrA:=ChrToNum_Code;                        //Сохраняем код
     ChrToNum(global_key[b]);        //Получаем код буквы ключа
     chrB:=ChrToNum_Code;                        //Сохраняем код
     SearchCryptChr(chrB);
               //Ищем зашифрованную букву и получаем ее номер
                                  //Получаем расшифрованный символ
     jcrypt(ChrToNum_Code,chrB);
     dectxt:=dectxt+Chr_DeCrypted;
                                        //Получаем расшифрованный текст
     b:=b+1;
     (ShowMessage('Символ #:'+IntToStr(i)+' dectxt='+dectxt+'
global_txt[i]='+global_txt[i]+'
global_key[i]='+global_key[i]+'  chrA='+IntToStr(chrA)+'
chrB='+IntToStr(chrB)+'
ChrToNum_Code='+IntToStr(ChrToNum_Code));}
    end;
```

Как видим, алгоритм расшифровки довольно прост, но все же требует некоторых комментариев.

Сперва мы запускаем главный цикл расшифровки, он

работает от первого символа зашифрованного текста и до последнего.

Далее сразу же идет код проверки цикла ключа. Если b имеет значение больше длины ключа, то b сбрасывается в единицу.

Затем следует запись текущих символов зашифрованного текста и ключа. Для дальнейшей расшифровки нам необходимо получить порядковые номера этих символов, что достигается с помощью функции ChrToNum. Эту функцию мы рассматривали в первой части статьи.

Для сохранения полученных порядковых номеров используем переменные ChrA (для зашифрованного текста) и ChrB (для ключа).

Следущей строкой идет функция SearchCryptChr, которая реализует основной поиск расшифрованного символа в таблице кодирования.

Для дальнейшего понимания алгоритма декриптования мы рассмотрим функцию SearchCryptChr.

```
//Поиск из строки нужного символа и получение его номера
function SearchCryptChr(chr:integer):string;
var fchr:string;              //Для временного хранения символов
    i:integer;                              //Для организации цикла
(for)
begin
 for i:=1 to 85 do
  begin
   //Получаем текст текущей ячейки
   fchr:=Form1.StringGrid1.Cells[i,chr];
   {ShowMessage('fchr='+fchr);}
       //...если найденный символ равен символу зашифрованного
       // текста, то вычисляем его номер
   if fchr=global_txt_chr then
    begin
     //План:
     //1. Получаем номер i найденного символа
     //   (зашифрованного текста)
     //2. Используем номер столбца и получаем имя символа,
     //    находящегося по координатам (i,0)
     fchr:=Form1.StringGrid1.Cells[i,0];
     Chr_DeCrypted:=fchr;
     ChrToNum(Chr_DeCrypted);
     {ShowMessage('Расшифрованная буква (fchr): '+fchr);}
    end;
  end;
end;
```

Как работает функция SearchCryptChr? Вначале запускается цикл от единицы до количества символов в алфавите. Только после этого в цикле считывается первый символ первого столбца и chr-строки. Что за "chr-строка"? В коде процедуры нажатия на кнопку DeCrypt мы передаем функции SearchCryptChr порядковый номер текущего символа ключа. Именно в рассматриваемой функции и используется этот порядковый номер.

Далее следует код проверки: если полученный символ (из предыдущей команды) равен текущему символу зашифрованного текста, то считываем из ячейки StringGrid'a символ, используя координаты [i,0], где i — столбец, а 0 — строка. Полученный символ записывается в переменную Chr_DeCrypted. Сохранив символ, мы находим его порядковый номер с помощью функции ChrToNum. Все, код функции SearchCryptChr завершен, и программа возвращается в процедуру bDeCrypt.OnClick;.

После выполнения функции SearchCryptChr идет следующий код:

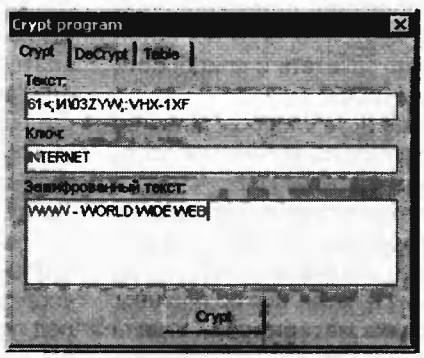
```
jcrypt(ChrToNum_Code,chrB);
dectxt:=dectxt+Chr_DeCrypted; //Получаем расшифрованный текст
b:=b+1;
end;
```

В функцию jcrypt, которая также использовалась в алгоритме кодирования, мы получаем расшифрованный символ. Символ получен, и он добавляется в переменную dectxt. Замечу, что процесс заполнения dectxt осуществляется путем суммирования значения dectxt с переменной Chr_DeCrypted. Если написать dectxt:=Chr_DeCrypted, то расшифрованный текст будет состоять из одного (последнего) символа (также расшифрованного)! Для перехода к следующему символу ключа мы инкрементируем значение переменной b.

Реализация алгоритма дешифрации завершена, осталось вывести результаты работы программы:

```
//Выводим зашифрованный текст
   mText2.Clear;
   mText2.Text:=dectxt;
```

Результат работы программы представлен на **рисунке**.



Усовершенствования

Созданная нами программа не является образцом реализации крипто-алгоритма Вигенера, она всего-навсего пример. Чего не хватает в программе? Любую программу можно хоть чуть-чуть, да усовершенствовать. В нашем проекте можно изменить дизайн, сделать программу более интерактивной. Компоненты для ввода незашифрованного и зашифрованного текста желательно заменить компонентами Memo. В закладке "Table" мы видим компонент StringGrid, который очень желательно заменить обыкновенным двумерным массивом, что соответственно уменьшит общий обьем исполняемого файла (exe) и увеличит скорость работы программы. В данном случае нет необходимости стараться увеличивать скорость шифрации/дешифрации, так как рассматриваемые алгоритмы предельно просты, и для их выполнения требуются минимальные "усилия" процессора.

Итоги

Итак, мы написали простейшую программу, реализующую алгоритм шифрования Вигенера и обратный процесс, т.е. дешифрацию. Полный исходный текст проекта можно найти на веб-странице журнала.

К достоинствам созданного проекта можно отнести:

- простоту программного кода;
- быстроту шифрования/дешифрования.

Однако, наряду с положительными качествами программы, следует отметить один недостаток, который перекрывает все достоинства — это очень простой алгоритм, вследствие чего его очень легко расшифровать. Поиск ответа на вопрос, как написать "подбиратель" (bruteforcer) паролей, я оставляю вам к качестве "домашнего задания".

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

На "материнку" с паяльником

Ремонт материнских плат с поврежденной BIOS

Театр начинается с вешалки, а компьютер — с BIOS'а.
Измышлизм

В [1] была описана методика замены микросхемы BIOS, выполненной в корпусе DIP и установленной в панельке. Публикация этой статьи вызвала читательский отклик, и ко мне стали поступать письма с вопросами. Основные из них:

- как сменить микросхему BIOS, если она впаяна своими выводами в отверстия на материнской плате?
- как сменить микросхему BIOS, если она припаяна поверхностным монтажом?
- где можно заказать и приобрести микросхемы BIOS?
- какие еще есть способы прошивки микросхем BIOS, кроме тех, которые описаны в статье [1]?

В этой статье я решил дать ответы на все эти вопросы. По сути, эта статья является логическим продолжением статьи [1]. Итак, все по порядку.

Диагностика материнской платы

Прежде чем браться за ремонт "материнки", нужно быть твердо уверенным в том, что причина неработоспособности компьютера действительно заключается в отказе микросхемы BIOS. Неплохо, если диагностику проведет специалист по ремонту компьютеров или опытный "железячник" со стажем. Иначе вы рискуете нажить себе лишние проблемы. Основные внешние признаки в случае выхода из строя BIOS — экран монитора после включения компьютера остается черным, как безлунная ночь, а спикер молчит, как партизан на допросе.

Микросхемы BIOS в корпусе DIP

Если микросхема BIOS выполнена в корпусе DIP, т.е. имеет двухрядное расположение выводов и впаяна своими выводами в отверстия на материнской плате, а не вставлена в панельку, в этом случае поступают так. Неисправную микросхему не выпаивают из "материнки", а выкусывают, так чтобы выводы микросхемы по возможности целиком остались на плате. К этим оставшимся выводам сверху припаивают исправную микросхему. Припаивают микросхему к старым выводам потому, что материнские платы — многослойные. К двум внешним слоям печатной платы доступ есть, но нет доступа к ее внутренним слоям. Слои — это проводники печатной платы, изготовленные из медной фольги. Конечно, выглядит припаянная таким образом микросхема не очень красиво, зато контакт с платой будет надежным. Для удобства смены микросхемы можно вместо самой микросхемы припаять панельку, в которую будет вставляться микросхема. Панельку, как и микросхему, надо припаивать к выводам "выкушенной" микросхемы. Для того чтобы при установке микросхемы был упор снизу, нужно заранее подложить под панельку "кирпичик" из диэлектрического материала соответствующих размеров.

Микросхемы BIOS в корпусе PLCC

Если микросхема, выполненная в корпусе PLCC, вставлена в панельку, то это упрощает ее замену. Для демонтажа такой микросхемы понадобится специальный ключ, с помощью которого микросхему вытаскивают из панельки. Для этого в панельке по ее углам находятся специальные выемки для захвата микросхемы.

Если микросхема BIOS припаяна на материнской плате поверхностным монтажом (как правило, такие микросхемы имеют четырехрядное расположение выводов и выполнены в корпусе PLCC), то в таком случае ее можно попытаться демонтировать с помощью паяльника. Один из способов демонтажа микросхем описан в [2].

Как и в случае с микросхемой в корпусе DIP, вместо микросхемы желательно припаять панельку PLCC. Необходимо учесть, что в продаже имеются PLCC-панельки двух видов — под монтаж в отверстия и под поверхностный монтаж. В нашем случае понадобится второй вариант. Также учтите, что сами микросхемы BIOS, выполненные в корпусе PLCC, тоже бывают двух видов. В первом случае выводы микросхем отформованы под поверхностный монтаж, во втором — под панельку. Не пытайтесь сами переформовать выводы, ничего хорошего из этого не получится, так как выводы у микросхем легко обламываются. Так что, сначала определитесь, какой вариант для вас предпочтителен, а уж потом начинайте поиск нужной вам микросхемы BIOS.

Где можно заказать и приобрести микросхемы BIOS

Сейчас достать микросхему BIOS — не проблема, были бы, как говорится, деньги. Стоимость такой микросхемы составляет от 50 до 100 рублей. Можно поискать ее на радиорынках, компьютерных барахолках. Но надежнее всего посетить специализированные фирменные магазины, например, фирмы "Промэлектроника". Что такое "Промэлектроника"? Это фирма с десятилетним стажем. Ее головной офис находится в г.Екатеринбурге. Имеется развитая сеть дилеров и филиалов. Ассортимент достиг 40 тысяч наименований. Если

Табл. 1

Название	Сайт	Адрес	Почта	Телефон
"Промэлектроника"	www.promelec.ru	г.Екатеринбург, ул. Колмогорова, д.70	pr@promelec.ru	(3432) 45-44-88
"DESSY"	www.dessy.ru	107113, г.Москва, а/я 10	post@dessy.ru	тел/факс (095) 304-72-31
"Симметрон"	www.symmetron.ru	195196, г.Санкт-Петербург, а/я 49	npo@symmetron.ru	(812) 278-84-84
"СМД Компонент"	www.smd-component.ru	103031, г.Москва, ул. Б. Лубянка, 13/16, стр.1	smd@smd-component.ru	(095) 928-83-98
ООО "СМП"	www.smd.ru	-	sale@smd.ru	(095) 158-73-96
"ЧИП и ДИП"	www.chip-dip.ru	129110, г.Москва, а/я 996	sales@chip-dip.ru	тел/факс (095) 973-70-73

в магазине отсутствует нужная микросхема, то ее можно заказать, предварительно оплатив ее стоимость. Если вы проживаете вдали от крупных городов, в таком случае заказ можно попробовать сделать почтой, или связаться с организацией "Промэлектроника" по электронной почте. В общем, принцип такой же, как в старые времена, когда радиодетали заказывали по Посылторгу.

У каждого должен быть выбор. Кроме фирмы "Промэлектроника", можно обратиться в другие торгующие организации. Например, в почтовое агентство "DESSY", посетив его интернет-магазин. Также можно обратиться в фирму "СМД Компонент", ООО "СМП" и др. (**табл.1**).

Довольно много рекламной информации по электронным компонентам размещено в журнале "Радио". В общем, стучитесь во все двери. Как говорится, за спрос по "фэйсу" не бьют, а кто ищет — тот всегда найдет. Учтите, что в каталогах и справочниках микросхемы BIOS обозначены как микросхемы Flash-памяти или как микросхемы EEPROM с параллельным доступом. В своей заявке на микросхему BIOS указывайте не ту маркировку, что сверху нанесена на этикетке микросхемы, а ту, что находится на самом корпусе микросхемы, под этикеткой. При выборе микросхемы не помешает знать о том, что микросхемы Flash-памяти, производимые фирмами ATMEL и SST, при всех равнозначных параметрах стоят дешевле по сравнению с продукцией фирм AMD, Intel и Winbond.

"Прошивка" микросхем BIOS

Новые микросхемы BIOS продаются торгующими организациями "непрошитыми". "Прошивать" микросхемы придется либо самим, либо обращаться в сервисный центр. В первом случае понадобится работоспособный компьютер, на материнской плате которого микросхема BIOS установлена в панельку. Можно "прошить" микросхему и на программаторе. В случае, если микросхема выполнена в корпусе PLCC, понадобится специальный переходник для программатора. Этот переходник идет в продаже как "адаптер DIP — PLCC". Во всех случаях потребуется файл с версией BIOS от вашей "материнки", который можно скачать с сайта производителя вашей материнской платы.

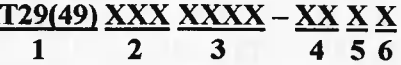
О программаторах

Приобретать программатор для прошивки одной микросхемы очень накладно. Стоимость программатора составляет порядка 3000 рублей и выше. Для тех же, кто интересуется, где можно приобрести программатор для прошивки микросхем, сообщаю следующее. НПО "Симметрон" предлагает программатор ChipProg для программирования продукции фирмы ATMEL. Данный программатор поддерживает внутрисхемное программирование. За дополнительной информацией можно обратиться по телефону (812) 278-84-21.

Также для программирования Flash-памяти можно приобрести почтой программатор UniProg с подключением к LPT-порту. Подробное описание можно найти в Интернете на сайте www.programmator.ru. Поставляет программаторы UniProg фирма "МикроАрт", ее полный адрес: 123022, г. Москва, а/я 76, ООО "МикроАрт", телефоны (095) 180-85-98; 189-28-01.

Маркировка микросхем Flash-памяти

В маркировке микросхем Flash-памяти пока единых стандартов не наблюдается, но все же общие принципы прослеживаются. На **рисунке** по-



казано, как маркируются, например, микросхемы Flash-памяти фирмы ATMEL.

Здесь:
- **1** — префикс;
- **2** — напряжение питания;
- **3** — емкость (**табл. 2**);
- **4** — время доступа, нс;
- **5** — тип корпуса:
  P — DIP;
  J — PLCC;
  T — TSOP;
- **6** — температурный диапазон:
  C — 0...+70°C; I — -40...+85°C.

Табл. 2

Маркировка	Организация
512	64 K x 8
010 (011)	128 K x 8
1024	64 K x 16
020	256 K x 8
2048	128 K x 16
040 (004)	512 K x 8
4096	256 K x 16
080 (008)	1 M x 8
8192	512 K x 16
1604	1 M x 16
1614	2 M x 8
3208	2 M x 16

Данные в колонке "Организация" означают *объем памяти, Кб x разрядность памяти, бит.*

Полезные советы

Пайку производите при помощи низковольтного паяльника на 12, 24 или 36 В с заземленным жалом. Также необходимо использовать браслет с заземлением для снятия статического электричества. При демонтаже и пайке микросхем удобно использовать увеличительные очки. Во-первых, видно все гораздо лучше, во-вторых, руки свободны, а в-третьих, эти очки предохранят глаза от попадания брызг расплавленного припоя.

Литература

1. Горячкин А. Восстановление и обновление BIOS. — Радиомир. Ваш компьютер, 2002, №8, С.36.

2. Рюмик С. Способ демонтажа SMD-микросхем. — Радиомир. Ваш компьютер, 2001, №7, С.32.

Как запрограммировать AT89C4051

С.РЮМИК,
г.Чернигов.

К чему душа лежит, к тому и руки приложатся.
Русская пословица

Микроконтроллер AT89C4051 фирмы Atmel по времени появился позже своих собратьев — микроконтроллеров AT89C1051 и AT89C2051. Как следствие, "младшему братцу" не досталось отдельной программы обслуживания в пакете драйверов фирменного программатора [1]. А жаль, ведь AT89C4051 имеет увеличенную до 4 Кб память при полной программной и аппаратной совместимости.

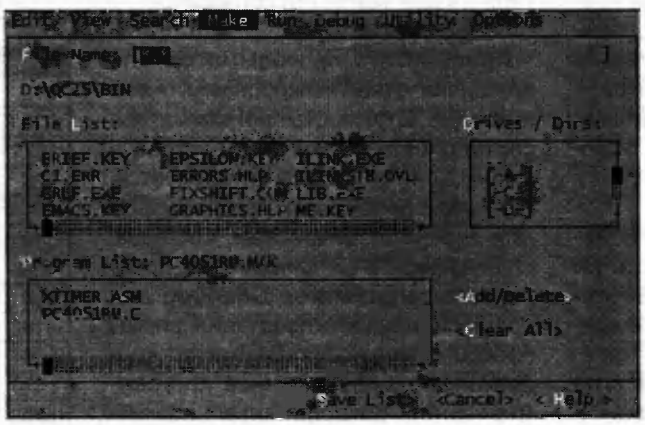
Если прошивка занимает объем FLASH-памяти меньше 2 Кб, то выход из положения заключается в использовании драйвера "pc2051.exe", предназначенного для AT89C2051. При этом ни в схеме фирменного программатора, ни в методике программирования контроллера изменений делать не надо. Запись будет производиться в младшую половину программной памяти AT89C4051, старшая половина останется незаполненной.

А как быть, если программа занимает от 2 до 4 Кб памяти? Здесь самое время приложить руки программисту и изменить коды драйвера "pc2051.exe", увеличив в нем в 2 раза доступное адресное пространство. В том, что эта идея осуществима, убеждает наличие в пакете программ (http://www.atmel.com/dyn/resources/prod_documents/APCPGM.EXE, 276 Кб) файла "pc2051.c". В нем приведен исходный текст программы на языке Си с подробными комментариями. Если заменить в строке 24 константу "2048", определяющую объем памяти в байтах, константой "4096", и сохранить текст под именем "pc4051.c", то получится исходная заготовка для программирования AT89C4051.

Одна незадача — в "исходнике" не указан диалект компилятора языка Си, с которым работает программа "pc2051.c". Более того, попытка откомпилировать ее в среде "Turbo-C", "Borland C++" или "Watcom C" наталкивается на многочисленные сообщения об ошибках. К разгадке ведет название графической библиотеки "graph.h", которая встречается в компиляторе "Microsoft QuickC".

Немного истории. В середине 80-х годов фирма Microsoft располагала отличным компилятором языка Си — быстрым, компактным, хорошо совместимым как со стандартом K&R, так и с ANSI. Только один недостаток смущал программистов — высокая цена пакета "Microsoft C" (500...700 USD). Рынок отреагировал незамедлительно, и в 1986-87 годах появилось несколько десятков новых компиляторов по цене в 3...10 раз ниже. Чтобы не упустить клиентов, фирме Microsoft пришлось выпустить простой и дешевый "QuickC" по цене 99 USD — ровно столько стоил конкурирующий пакет "Turbo C" фирмы Borland.

Известно несколько модификаций "QuickC". "Прадедушкой" называют версию 1.0 образца 1987 г. Затем в течение трех лет появились улучшенные версии 1.01, 2.00, 2.01, 2.50, 2.51. В них постепенно устранялся главный недостаток компилятора, заключавшийся в практически полном отсутствии системы оптимизации кода.



Применительно к рассматриваемому файлу "pc4051.c" следует отметить, что не любая версия "QuickC" подходит для работы с ним. Например, "прадедушка" не понимает комментариев, начинающихся с двух наклонных черточек "//", часто называемых "прямой слэш" (от англ. "slash" — рубец, разрез), и отказывается производить компиляцию.

Успешно откомпилировать файл "pc4051.c" можно с помощью "QuickC" версии 2.51 (ftp://ftp.atnet.ru/pub/OS/msdos/lang/qc25.zip, 3392 Кб), и то не с первого раза. Дело в том, что кроме стандартных встроенных библиотек, файл "pc4051.c" использует 5 внешних функций: extern void tinit(void), tend(void), tread(void), disable_traps(void) и enable_traps(void).

Где их искать? Оказывается, в том же пакете фирменного программатора APCPGM.EXE, но в файле "Xtimer.asm". Соединить Си-программу и ассемблерный код можно в файле проекта с расширением *.mak по следующей методике.

Предварительно нужно подготовить опции компилятора "QuickC": "Options — Make — Release — Compiler Flags — Small — MS Extensions — OK".

Затем сделать копию файла "Life.mak", входящего в каталог примеров "Samples", и переименовать его в "pc4051.mak".

Открыть в главном меню "QuickC" окна: "Make — Set Program — List — pc4051.mak — OK — No".

Очистить mak-файл, скопировать в него две программы и сохранить его на диске: "Make — Edit Program List — File Name *.* — Clear All — Xtimer.asm — Add/Delete — pc4051.c — Add/Delete — Save List" (см. **рисунок**).

Откомпилировать весь проект: "Make — Set Program List — pc4051.mak — OK — Make —Rebuild All".

После этих действий, в текущем каталоге "QuickC" должен появиться исполняемый файл "pc4051.exe" длиной 44936 байтов, готовый к работе с контроллером AT89C4051.

Тем, кто хотел бы русифицировать драйвер, ничто не мешает заменить английский текст в Си-программе родной кириллицей в операторах "printf" и "puts". Более того, можно ввести двуязычие выбором соответствующей опции при загрузке программы. Полученный "исходник" переименовывается в "pc4051ru.c" и компилируется по приведенной выше схеме в файл "pc4051ru.exe".

Программирование микросхем с новым драйвером ничем не отличается от методики, принятой в фирменном программаторе. Ввод команд осуществляется маленькими или большими буквами латинского алфавита. Хорошо проработанная система подсказок и сообщений не дает ошибиться в действиях. Новый драйвер "pc4051ru.exe" обладает универсальностью. С его помощью можно программировать три типа микросхем: AT89C4051, AT89C2051, AT89C1051. Отличаться будут лишь размеры вводимых файлов прошивок, соответственно, 4, 2 и 1 Кб.

Копия двуязычного драйвера "pc4051ru.exe" и текст программы "pc4051ru.c" размещены на сайте журнала http://radio-mir.com (архив "pc4051ru.zip, 39 Кб).

Литература

1. Рюмик С. Замена микросхем в фирменном программаторе. — Радиомир. Ваш компьютер, 2003, №5, С.32.

Телетест на Sinclair

Е.ИЛЯСОВ,
412302, г. Балашов,
ул. Красина, 82.

Глаза есть, а видеть — нету!
Дерсу Узала.

В процессе использования бытовых компьютеров, формирователи видеосигналов (видеопроцессоры) которых основаны на комплексе телевизионных стандартов [1], время от времени возникает необходимость проверки (и, если по результатам такой проверки требуется, то и подстройки, корректирования) параметров устройства видеовывода. Для применения в качестве такого устройства подразумевается, естественно, бытовой телевизор либо видеомонитор — что, по большому счету, одно и то же. Видеомонитор, например, монохромный "Электроника" МС 6105 или цветной "Электроника" МС 6113.02, это в своей основе тот же телевизор, только без радиоканала.

К основным техническим показателям, доступным для визуальной проверки по изображению на экране цветного кинескопа, относятся: чистота цвета свечения экрана; статический и динамический баланс белого; точность совмещения трех растров; верность воспроизведения белого, основных и дополнительных цветов; матрицирование и качество цветовой синхронизации. Значения этих основных и многих других параметров определены в [2].

В профессиональной практике ремонта и настройки телевизионных приемников цветного и черно-белого изображения для оценки основных характеристик и проверки качества изображения наряду с контрольно-измерительными приборами используют различные испытательные сигналы и таблицы (например, [3, 4]). Наиболее распространенные и важные испытательные сигналы для проверки перечисленных показателей — это сигналы "сетчатое поле", "шахматное поле", "серая шкала" и "цветные полосы".

Сигнал "сетчатое поле" воспроизводит на темном фоне экрана кинескопа светлую сетку, которая состоит из перекрещивающихся горизонтальных и вертикальных линий. Число линий сетки по вертикали и горизонтали может меняться в широких пределах. С помощью этого сигнала можно оценить статическое и динамическое сведение лучей, проверить центровку изображения, геометрические и нелинейные искажения растра, а также прохождение высокочастотных составляющих спектра видеосигнала.

Изображение сигнала "шахматное поле" состоит из черных и белых квадратов. С его помощью можно проверить работу блока разверток, оценить нелинейности по горизонтали и вертикали, стабилизацию и размер изображения, геометрические искажения растра и отсутствие цветных окантовок.

Изображение, формируемое сигналом "цветные полосы", образуется восемью вертикальными цветными полосами одинаковой ширины с нормализованными уровнями яркости и насыщенности (75%), которые размещаются слева направо в определенной последовательности — белая, желтая, голубая, зеленая, пурпурная, красная, синяя, черная. Этот тест используется для проверки прохождения сигналов через канал цветности, проверки значений цветоразностных сигналов (правильности матрицирования), оценки цветовой насыщенности в смежных строках, устойчивости цветовой синхронизации и некоторых других настроек. В частности, большинство осциллограмм, приводимых на принципиальных схемах телевизоров в цепях усиления и формирования сигналов яркости и цветности, соответствуют приему сигнала цветных полос.

При тестировании монохромных устройств видеовывода этот испытательный сигнал может заменить сигнал "серая шкала", обычно используемый для проверки и регулировки баланса белого и проверки правильности воспроизведения градаций серого в черно-белых телевизорах и монохромных видеомониторах.

Для формирования на экране телевизора этих и некоторых других сигналов применяются специализированные приборы — генераторы испытательных сигналов (например, [5]), из арсенала радио- и телемехаников ремонтно-обслуживающих предприятий соответствующего профиля. Видеомониторы в качестве устройств видеовывода, как правило, также допускают подключение к таким приборам.

В целом ряде случаев оценить основные параметры и качество изображения устройства видеовывода можно без использования сложной и дорогостоящей специальной техники — с помощью самого компьютера.

На первый взгляд, неполное использование площади экрана в Sinclair-совместимых компьютерах значительно ограничивает возможности программного способа генерации испытательных сигналов. Но не следует забывать, что телестандартами допускаются различные специфические искажения изображения, которые наиболее сильно выражены именно на краях кинескопа, но не являются при этом признаками неисправности, если в сумме не превышают допустимых, заданных стандартами пределов. Причем, если в процессе отображения динамичных сцен (например, телевизионных фильмов или же активных компьютерных игр) эти искажения менее заметны, то в режиме отображения символьной информации на краях экрана они приводят к резкому несоответствию между параметрами изображения и физиологическими особенностями органов зрения человека [6], со всеми вытекающими отсюда последствиями. Таким образом, видеоконтроллер Sinclair-совместимых компьютеров является еще и наиболее "щадящим" среди машин такого же, "домашнего" класса. Поэтому и "охота" за всей площадью экрана в "синклеровском" случае просто не имеет смысла. Дополнительным независимым свидетельством в пользу разумности "синклеровского" подхода к использованию площади экрана является сходная реализация такого "нагружающего" зрение режима работы телевизора как телетекст.

В простейшем варианте можно воспользоваться тестом, близким к сигналу "цветные полосы", который "прошит" в ПЗУ всех Sinclair-совместимых компьютеров, имеющих не менее

КОМПЬЮТЕРНЫЕ ХРОНИКИ

CAFe'2003

Итак, 23...24 августа 2003 г. вот уже в четвертый раз в Казани прошел фестиваль компьютерного искусства — Computer Art Festival, или просто CAFe. Для "ZX-Spectrum" проводились конкурсы в номинациях Demo, AY music, Graphics, 512 bytes intro, 4K intro, Game.

Итоги конкурсов

Demo (всего 7 работ): 1 — "Weed" by Triebkraft (370), 2 — "RESURRECTION" by Jocker, Klim/Omega Hackers Group (303), 3 — "caprize" by milytia (208).

AY music (всего 29 работ, преселект прошли 20): 1 — "Retrochip 2" by Gasman/Hooy-Program/AY Riders (258), 2 — "its life" by mmcm^sage (233), 3 — "fioletto" by c-jeff/gbg+bwt (231).

Graphics (всего 29 работ, преселект прошли 20): 1 "Unreal" by Rion (345), 2 — "Alien" by G.D./Triebkraft, 4th Dimension (290), 3 — "Village People" by Diver/4th Dimension (287).

512 bytes intro (всего 10 работ): 1 — "i512" by psb/halloween (265), 2 — "DRAWING" by Siberian Group (237), 3 — "_kosmos_" by voodoo/delirium tremens (206).

4K intro (всего 1 работа): "Requiem of a Sun" by KIM/PoS-WT.

Game (всего 6 работ): 1 — "Fire&Ice" by n-Discovery (206), 2 — "Lethargy" by Studio Stall (204), 3 — "Darkwing Duck" by Virtual Masters (200).

Подробности можно узнать на официальном сайте CAFe — http://cafeparty.org.ru. Результаты и работы для "ZX-Spectrum" также должны быть доступны на крупных спектрумовских сайтах.

Подготовил И.Рощин.

128 Кб оперативной памяти и встроенную дисковую систему TR-DOS. В таких машинах тест запускается кнопкой <RESET> при одновременно нажатой клавише <BREAK> [7]. На экран выводятся цветные полосы, чередующиеся в стандартном порядке и разделенные по вертикали на зоны с повышенной и нормальной яркостью. Кроме того, стандартный однобитный звуковой формирователь компьютера генерирует прерывистый сигнал для тестирования канала звука.

Более разнообразные тестовые сигналы можно получить с помощью прикладных программ, созданных специально для этой цели. Одна из самых распространенных программ-генераторов испытательных сигналов — это "TELETEST" (автор — В.Рабинович), написанная в 1988 г., в городе, тогда еще, Ленинграде. Помимо "цветных полос", "шахматного поля" и "сетчатого поля", в программе генерируются сигналы, содержащие точечную матрицу для проверки фокусировки и астигматизма электронного луча на полезной площади экрана, и вертикальные полосы разной ширины для оценки "частотной синхронизации", или, правильнее — частотной характеристики канала яркости. Есть возможность закрашивания экрана в один из основных цветов (равномерные цветные поля, служащие для проверки чистоты цвета и выявления дефектов маски кинескопа) и некие подобия телевизионных испытательных таблиц. Русскоязычный интерфейс пользователя в программе достаточно удобен и понятен. Эта тестирующая программа распространялась, в частности, московским МКП "Инфорком" в составе серии "СД" (системные дискеты) на дисковом сборнике под кодом "СД-2".

Эта же программа (авторское название — "Телевизионные испытательные таблицы") была использована в нашем компьютерном кружке для тестирования любезно предоставленного "спонсором" цветного видеомонитора отечественного производства "Электроника" 32ВТЦ-202. В ходе тестирования попутно выявились и некоторые недостатки этой программы, затруднившие оценку параметров изображения.

В частности, имитация испытательного сигнала "сетчатое поле" (пункт в программе — "Проверка сходимости") выводится на экран, в отличие от стандарта, темными линиями на светлом фоне и без вмешательства в текст программы. "Раскладку" цветов изменить нельзя. Пункт "Частотная синхронизация" малоинформативен, поскольку на экране присутствуют всего пять градаций вертикальных штрихов, соответствующих частотам, начиная от 0,062(!) МГц, да и на практике необходимость в таком испытательном сигнале встречается крайне редко. Подобия испытательных таблиц "Моноскоп Б" и особенно "Моноскоп А" также малоинформативны и присутствуют больше "для красоты" и напоминания об авторстве, чем для облегчения процесса проверки. Есть и другие, менее значительные огрехи. Например, из-за недоработки в самой программе (строка 130), "точки" большого размера (2 на 2 пиксела) в пункте "Точечный растр\малая плотность" выводятся лишь наполовину — только нижний пиксельный ряд в каждой "точке". Также невозможно без вмешательства в текст программы изменить цвет бордюра в некоторых пунктах, и др.

Эти обстоятельства подтолкнули нас к созданию альтернативной версии "генератора испытательных сигналов" — программы "TV-test", в которой перечисленные недоработки были бы устранены. Попутно решалась не менее важная учебная задача — используя межпредметные связи (в данном случае из области физиологии зрения, физики и иностранного языка), взаимодействуя с кружками других технических направлений (в частности, радиоэлектроники), продолжить развитие навыков самостоятельного прикладного программирования у кружковцев, аккумулировать, углубить и закрепить знания, полученные по другим предметам школьного курса, повысить уровень общетехнической грамотности и стимулировать творческий интерес учащихся к практическим приложениям информационных технологий.

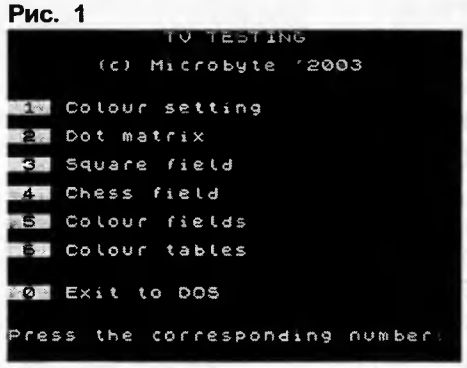
Поскольку диалоговая система пользовательского интерфейса "TV-test" несложна, интуитивно и контекстно понятна, то на этапе обсуждения концепции программы было решено не заниматься специально ее русификацией. Тем не менее, если в этом возникнет необходимость, задача русификации не будет сложной, так как текстовых сообщений здесь не так уж много.

Программа проектировалась максимально прозрачно, для облегчения понимания логики работы и взаимодействия отдельных ее составляющих. В ней не использовались какие-либо особо "навороченные" приемы программирования и экономии памяти. Тем не менее, как обычно, особое внимание было уделено надежности работы программы и удобству управления. Она не страдает функциональными излишествами (как, например, в [8]), но обладает, на наш взгляд, разумным набором испытательных сигналов, необходимым и достаточным для разносторонней оценки качества изображения устройств видеовывода.

Также можно обратить внимание на ряд полезных моментов, способных заинтересовать как учащихся, так и любителей прикладного программирования и пользователей. Рассмотрим вкратце некоторые особенности работы "TV-test".

В программе используются два комплекта цветовых настроек — для визуализации интерфейса с пользователем (задается в строке 110) и действующий во время генерации испытательных сигналов (строки 20 и 1000...1100). Назвать их глобальными и локальными цветами язык как-то не поворачивается, но на практике такое разделение оказалось очень даже удобным.

Рис. 1

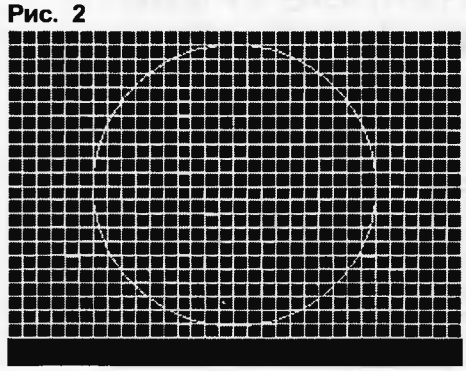


Так, например, для испытательных сигналов устанавливается светлое изображение на темном фоне — как и положено по стандарту. Но если потребуется, по ходу работы цвета можно изменить, выбрав из главного меню (**рис.1**) установку цветов (пункт 1, "Colour setting") и ответив нужным образом на подсказки в нижней части экрана. Причем, для восстановления стандартных цветов достаточно еще раз выбрать этот же пункт меню, и на все подсказки, используя принцип задания параметров "по умолчанию", ответить нажатием клавиши <ENTER>.

Программа "TV-test", так же как и "TELETEST", генерирует "точечный растр" (пункт 2, "Dot matrix") с возможностью выбора размера "точек", "сетчатое поле" (пункт 3, "Square field") и "шахматное поле" (пункт 4, "Chess field"), также с двумя возможными размерами клеток.

Дробные значения в шагах циклов, выводящие изображения точечного и сетчатого полей, подобраны таким образом, чтобы наиболее полно "вписать" размеры формируемых изображений в размеры экрана (**рис.2**). Конечно же, видеоконтроллер не станет выводить пикселы с "дробными" координатами между линиями развертки, но зато Бейсик постарается при выводе округлить эти вычисленные координаты до ближайшего целого числа.

Рис. 2



Центр изображения в обоих ва-

риантах сигнала "сетчатое поле" является условным и находится выше физического центра экрана на восемь пиксельных линий (линий развертки). Это связано с недоступностью двух нижних строк редактирования из стандартного интерпретатора Бейсика.

Пункт 5 — "цветные поля" ("Colour fields") — содержит изображения равномерных цветных полей трех основных цветов: красного, зеленого и синего (Red, Green, Blue). Кроме того, есть возможность вывести на экран равномерное серое поле (Grey) для проверки статического баланса белого и чистоты цвета. Если же вдруг потребуется "залить" экран каким-либо другим, дополнительным цветом, то это можно сделать, установив нужный цвет во всех трех подсказках пункта 1 ("Colour setting") и выбрав затем для вывода любой испытательный сигнал (кроме цветных полей и полос, разумеется). Вернуться к стандартным цветам по умолчанию можно, опять же, через пункт "Colour setting" и троекратное <ENTER>.

Рис. 3

Формирование сигнала "цветные полосы" (пункт 6, "Colour tables") в программе возможно в трех вариантах. Первый из них довольно близок к стандартному, другой напоминает тест, встроенный в ПЗУ (c) Sinclair research 1986, с разнояркими полосами (**рис.3**). Кроме этих основных, реализован еще один вариант "цветных полос", встречавшийся на измерительно-настроечной видеокассете для настройки канала изображения видеомагнитофонов формата VHS. И наконец, после завершения вывода любого из вариантов "полос", нажатие клавиши <SPACE> будет циклически переключать цвет бордюра, а любая другая клавиша (но не <BREAK>!) вернет управление в основное меню.

```
    "TV-test" <B>
    1 REM (c) Microbyte
    2 REM version for iS-DOS
    3 REM last edition: 30.05.03
    4 REM
   10 REM - setting up -
   20 LET b$="0": LET p$="0": LET i$="7"
   30 LET main=100: LET cols=1e3: LET dots=2e3: LET sqar=3e3:
      LET ches=4e3: LET colf=5e3: LET colt=6e3
  100 REM - main menu -
  110 BORDER 0: PAPER 0: INK 6: BRIGHT 0: FLASH 0: CLS
  120 PRINT TAB 11;"TV TESTING"''
  130 PRINT TAB 6;"(c) Microbyte '2003"''''
  140 PRINT "[1] Colour setting"''
  150 PRINT "[2] Dot matrix"''
  160 PRINT "[3] Square field"''
  170 PRINT "[4] Chess field"''
  180 PRINT "[5] Colour fields"''
  190 PRINT "[6] Colour tables"''''
  200 PRINT "[0] Exit to DOS"''''
  210 PRINT "Press the corresponding number:"
  220 BEEP .1,10
  230 LET o$=INKEY$
  240 IF o$="1" THEN BEEP .1,10: GO TO cols
  250 IF o$="2" THEN BEEP .1,10: GO TO dots
  260 IF o$="3" THEN BEEP .1,10: GO TO sqar
  270 IF o$="4" THEN BEEP .1,10: GO TO ches
  280 IF o$="5" THEN BEEP .1,10: GO TO colf
  290 IF o$="6" THEN BEEP .1,10: GO TO colt
  300 IF o$="0" THEN BEEP .1,10:.e
  310 GO TO 230
 1000 REM - colour setting -
 1010 INPUT "Border colour: (0...7), [0] ";b$
 1020 IF b$="" THEN LET b$="0"
 1030 IF b$<"0"AND b$>"7" THEN GO TO 1010
 1040 INPUT "Paper colour: (0...7), [0] ";p$
 1050 IF p$="" THEN LET p$="0"
 1060 IF p$<"0"AND p$>"9" THEN GO TO 1040
 1070 INPUT "Ink colour: (0...7), [7] ";i$
 1080 IF i$="" THEN LET i$="7"
 1090 IF i$<"0"AND i$>"9" THEN GO TO 1070
 1100 GO TO main
 2000 REM - dot matrix -
 2010 INPUT "Dots: small/big (1/0), [1] ";s$
 2020 IF s$="" THEN LET s$="1"
 2030 IF s$<> "0" AND s$<> "1" THEN GO TO 2010
 2040 BORDER VAL b$: PAPER VAL p$: INK VAL i$: CLS
 2050 IF s$="0" THEN GO TO 2140
 2060 REM - small dots -
 2070 FOR x=0 TO 255 STEP 15.92
 2080 FOR y=0 TO 175 STEP 15.9
 2090 PLOT x,y
 2100 NEXT y
 2110 NEXT x
 2120 BEEP .1,10
 2130 PAUSE 0: GO TO main
 2140 REM - big dots -
 2150 LET e=0: LET r=0
 2160 FOR v=0 TO 1
 2170 FOR n=0 TO 1
 2180 FOR y=0 TO 175 STEP 21.7
 2190 FOR x=0 TO 255 STEP 21.2
 2200 PLOT x+e,y+r
 2210 NEXT x
 2220 NEXT y
 2230 LET e=e+1: NEXT n
 2240 LET e=1: LET r=r+1: LET e=e-1: NEXT v
 2250 BEEP .1,10
 2260 PAUSE 0: GO TO main
 3000 REM - square field -
 3010 INPUT "Square: big/small (1/0) [1] ";s$
 3020 IF s$="" THEN LET s$="1"
 3030 IF s$<> "0" AND s$<> "1" THEN GO TO 3010
 3040 BORDER VAL b$: PAPER VAL p$: INK VAL i$: CLS
 3050 IF s$="0" THEN GO TO 3130
 3060 REM - big squares -
 3070 FOR i=0 TO 175 STEP 15.9: PLOT 0,i: DRAW 255,0: NEXT i
 3080 FOR i=0 TO 255 STEP 15.93: PLOT i,0: DRAW 0,175: NEXT i
 3090 CIRCLE 127,87,80
 3100 PLOT 120,87: PLOT 135,87
 3110 BEEP .1,10
 3120 PAUSE 0: GO TO main
 3130 REM - small squares -
 3140 FOR i=0 TO 175 STEP 7.95: PLOT 0,i: DRAW 255,0: NEXT i
 3150 FOR i=0 TO 255 STEP 7.96: PLOT i,0: DRAW 0,175: NEXT i
 3160 CIRCLE 127,87,80
 3170 PLOT 123,83: PLOT 131,83: PLOT 123,91: PLOT 131,91
 3180 BEEP .1,10
 3190 PAUSE 0: GO TO main
 4000 REM - chess field -
 4010 INPUT "Table: big/small (1/0), [1] ";s$
 4020 IF s$="" THEN LET s$="1"
 4030 IF s$<> "0" AND s$<> "1" THEN GO TO 4010
 4040 BORDER VAL b$: PAPER VAL p$: INK VAL i$: CLS
 4050 IF s$="0" THEN GO TO 4220
 4060 REM - big table -
 4070 LET s=0: LET y=0
 4080 FOR p=0 TO 5: FOR a=0 TO 1
 4090 FOR x=0 TO 30 STEP 4
 4100 INK VAL i$: PRINT AT y+s,x;"■"
 4110 NEXT x: LET s=s+1
 4120 NEXT a: LET s=s+2
 4130 NEXT p: LET s=2
 4140 FOR p=0 TO 4: FOR a=0 TO 1
 4150 FOR x=2 TO 32 STEP 4
 4160 PRINT AT y+s,x;"■"
 4170 NEXT x
 4180 LET s=s+1: NEXT a
 4190 LET s=s+2: NEXT p
 4200 BEEP .1,10
 4210 PAUSE 0: GO TO main
 4220 REM - small table -
 4230 LET s=0: LET y=0
 4240 FOR p=0 TO 10
 4250 FOR x=0 TO 30 STEP 2
 4260 PRINT AT y+s,x;"■"
 4270 NEXT x: LET s=s+2
 4280 NEXT p: LET s=1
```

```
4290 FOR p=0 TO 10
4300 FOR x=1 TO 31 STEP 2
4310 PRINT AT y+s,x;"■"
4320 NEXT x
4330 LET s=s+2: NEXT p
4340 BEEP .1,10
4350 PAUSE 0: GO TO main
5000 REM - colour fields -
5010 CLS
5020 PRINT TAB VAL "8";"Select colour:"'''
5030 PRINT "■R■ Red"''
5040 PRINT "■G■ Green"''
5050 PRINT "■B■ Blue"''''
5060 PRINT "■0■ Grey"''''
5070 PRINT "Press the corresponding key:"
5080 REM BEEP .1,10
5090 LET o$=INKEY$
5100 IF o$="r" OR o$="R" THEN GO TO 5200
5110 IF o$="g" OR o$="G" THEN GO TO 5300
5120 IF o$="b" OR o$="B" THEN GO TO 5400
5130 IF o$="0" THEN GO TO 5500
5140 GO TO 5090
5200 REM - red field -
5210 PAPER 2: BORDER 2: CLS
5220 GO TO 5600
5300 REM - green field -
5310 PAPER 4: BORDER 4: CLS
5320 GO TO 5600
5400 REM - blue field -
5410 PAPER 1: BORDER 1: CLS
5420 GO TO 5600
5500 REM - Grey field -
5510 PAPER 7: BORDER 7: BRIGHT 0: CLS
5600 REM - end & main -
5610 BEEP .1,10
5620 PAUSE 0: GO TO main
6000 REM - colour table -
6010 CLS
6020 PRINT TAB VAL "9";"Select table:"'''
6030 PRINT "■1■ Standard"''
6040 PRINT "■2■ Sinclair"''
6050 PRINT "■3■ Special"''''
6060 PRINT "Press the corresponding key:"
6070 REM BEEP .1,10
6080 LET o$=INKEY$
6090 IF o$="1" THEN GO TO 6200
6100 IF o$="2" THEN GO TO 6300
6110 IF o$="3" THEN GO TO 6400
6120 GO TO 6080
6200 REM - Standard -
6210 BORDER VAL b$: PAPER VAL p$: CLS
6220 FOR i=0 TO 21
6230 FOR j=7 TO 0 STEP -1
6240 PRINT PAPER j;"    ";
6250 NEXT j
6260 NEXT i
6270 GO TO 6500
6300 REM - Sinclair -
6310 BORDER VAL b$: PAPER VAL p$: CLS
6320 FOR i=0 TO 21
6330 FOR j=7 TO 0 STEP -1
6340 PRINT PAPER j;BRIGHT 1;"  ";BRIGHT 0;"  ";
6350 NEXT j
6360 NEXT i
6370 GO TO 6500
6400 REM - Special -
6410 BORDER VAL b$: PAPER VAL p$: CLS
6420 FOR i=0 TO 11
6430 FOR j=7 TO 0 STEP -1
6440 PRINT PAPER j;"    ";
6450 NEXT j
6460 NEXT i
6470 FOR i=12 TO 21
6480 PRINT PAPER 0;"          ";
     PAPER 7;BRIGHT1;"          ";
     PAPER 0;"          ";
6490 NEXT i
6500 REM - Change border -
6510 LET b=VAL b$
6520 BEEP .1,10
6530 LET o$=INKEY$ : IF o$="" THEN GO TO 6530
6540 IF o$<> " " THEN GO TO main
6550 LET b=b+1: IF b>7 THEN LET b=0
6560 BORDER b: GO TO 6520
```

Приведенный листинг для самостоятельного набора получен путем конвертирования отлаженной Бейсик-программы, работающей под операционной системой iS-DOS, непосредственно в текстовый файл. Если же требуется обеспечить работу программы в среде TR-DOS, то желательно изменить строку с номером 300, записав ее в таком виде:

```
300 IF o$=»0" THEN BEEP .1,10: CLEAR 65367:
    RANDOMIZE USR 15619: REM: RUN
```

и внести соответственные изменения в строку с номером 2.

Несколько рекомендаций и советов, которые могут оказать помощь в самостоятельном наборе и отладке этой полезной программы:

- надо внимательно следить за количеством апострофов в концах строк, отвечающих за вывод на экран основного и промежуточных меню (строки 120...200, 5020...5060, 6020...6050). Это поможет правильно расположить на экране все строки, из которых состоят меню;

- контрастные элементы, из которых выстраивается "шахматное поле" (кавычки в строках 4100, 4160, 4260, 4310), и номера пунктов всех меню можно ввести как инвертированные пробелы, либо как символы псевдографики с кодом 143 (закрашенный квадрат);

- сигналы "цветные полосы" формируются с помощью групп пробелов, окрашенных в цвет фона. При этом в строке 6240 в кавычках должно быть 4 пробела, в строке 6340 — по 2 пробела, в строке 6440 — 4 пробела. В строке 6480 (нижняя, черно-белая часть специального варианта сигнала "цветные полосы") в первой и третьей группе должно быть по 11 пробелов, а во второй группе — 10 пробелов.

При работе над "TV-test" были использованы следующие программно-аппаратные средства:

- ОС BASIC-48, (c) Sinclair research Ltd., 1982;
- редактор Бейсик-программ Plus v2.5, (c) Маслов С., 1995;
- ОС iS-DOS, (c) IskraSoft, 1999;
- iS-DOS Basic, (c) IskraSoft, 1993;
- текстовый редактор iS-Editor, (c) IskraSoft, 1995, 1999;
- Sinclair-совместимый компьютер KAY-1024 turbo.

И наконец, следует особо отметить, что разработка и использование подобных программных средств контроля качества изображения устройств видеовывода вовсе не означает приглашения к самостоятельному их ремонту и настройке. Такая "самодеятельность" может привести к поражению электрическим током или травмам в результате возможного разрушения вакуумного прибора кинескопа, или электролитических конденсаторов. Телевизоры и видеомониторы не являются устройствами, безнаказанно допускающими неквалифицированное вмешательство в домашних или школьных условиях, и требуют предельной аккуратности в обращении и неукоснительного соблюдения правил техники безопасности.

Выражаем особую благодарность А.Карныгину за квалифицированные технические консультации по вопросам телевидеотехники и помощь в подборе литературы по исследуемой теме.

Литература.

1. Скутин В.Г. Откуда взялся бордюр. Элементарные сведения о концепции Spectrum'а. — Радиолюбитель, 2001, №10, С.19...22.

2. ГОСТ 18198-85. Телевизионные приемники.

3. Ельяшкевич С.А. Цветные стационарные телевизоры и их ремонт: Справочное пособие. — 2-е изд., перераб. и дополн. (Массовая радиобиблиотека, вып. 1151). — М.: Радио и связь, Москва, 1990, С.176...177.

4. Бродский М.А. Переносные телевизоры. — Мн.: Вышейшая школа, СП "Авест", 1994, С.305...308, 339...340.

5. Виноградов В.А. Обслуживание и ремонт стационарных цветных телевизоров. — СПб.: "Кристалл", 1996, С.47...52.

6. Блум Ф., Лейзерсон А., Хофстедтер Л. Мозг, разум, поведение. Перевод с английского. — М.: Мир, 1988.

7. Системные программы для ZX-Spectrum 128K. Справочник. — М.: "ВА Принт", 1993, С.3.

8. Рюмик С. "Dendy" — генератор испытательных телевизионных сигналов. Новая версия. — Радио, 2002, №10, С.29...32.

Падал прошлогодний снег

А.ВЕНДИЛОВСКИЙ,
г.Минск.

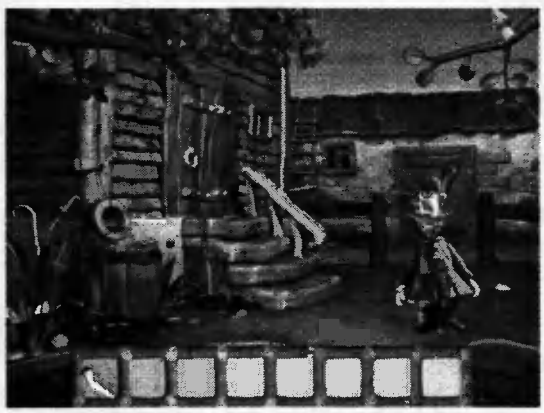
Эй! У Вас тут все дома?
"Падал прошлогодний снег". Мультфильм

В староглиняные времена, в одной пластилиновой местности, жила-была ворона... нет, помнится, собака, а может быть, корова? Нет, в этот раз в статье происходит все что угодно, то пепица, то нелепица. Пластилиновая она, нет, не статья, местность. А тут лиса бежала... Постойте, где лиса и ворона? Ладно, другое расскажу. Один компьютер был навечно прикован... хотя не надо, напомните мне, я об этом вам в следующий раз расскажу. А один человек очень любил арбузы, и вот однажды... оп-па, и он пропал с экрана монитора. Ну и сказочка, хотел вам о новом пластилиновом квесте рассказать, а тут такой бардак творится! Хотя постойте, это все присказка была, а сейчас настоящая сказка начнется. Вот идет наш главный герой — Мужик. Орел-мужчина, правда, работник из него никакой, да и был он большим любителем побездельничать, зато жена попалась ему строгая. Узнали? Ну да, тот самый, что за елкой ходил!

... Елку он принес, но это было уже весной, и жена его отправила обратно в лес, отнести ее на место. На этот раз все обошлось без лишних приключений, хвосты не вырастали, вернулся быстро. А тут и зима прошла, весна настала. Приснился ему как-то странный сон, что пришел к нему белый медвежонок с шайкой, да в каких-то проводах запутался. Проснулся Мужик, опять ничего не понимает. Рассказал сон жене, а она его в баню отправила... в хорошем смысле этого слова, помыться. Просто день банный наступил. А Мужик и рад стараться, дровишки притащил, веничек с чердака достал, пиво холодненькое с рыбкой приготовил. Начал было баньку топить, как бухбарах! Все трясется, ходуном ходит, шайка на голову падает, а сам Мужик на улице оказывается. Но самое главное — бани нет! Украли, окаянные! И домой не вернешься — дома жена... "Без бани" — говорит — "не возвращайся! А с баней — возвращайся!" Ну что оставалось делать Мужику, пошел он свою баню в своем пластилиновом краю искать...

"Когда желудь спелый, его любая свинья слопает"
"П.П.С."

Русских квестов выходит много, благо, сделать квест, наверное, проще всего технически и дешевле. Берешь в руки такое "чудо", посмотришь, покрутишь часа два, отчаянно пытаясь понять логику разработчика, подивишься примитивизму сюжетному, да и забросишь этот кошмар куда-нибудь за шкаф, где он и будет пылиться, ожидая своей очереди превратиться в элегантную подставку для кофе. Да и покупатель стал разборчивее, все чаще и чаще носом крутит, слово "наш квест" уже не оказывает на него волшебного действия — качество ему подавай. Вздохнули разработчики и стали использовать уже раскрученные бренды из анекдотов, сказок и мультфильмов. Вот только ничего особо хорошего из этого не получалось. Поэтому я с большой опаской взял этот квест — уж больно мне оригинальный мультфильм нравится, я даже его себе в коллекцию переписал, не хотелось портить впечатление. Скажу сразу, огорчен я не был, хотя в разряд "пожизненных" хитов эта игра все-таки не попадет. Но лучше все по порядку.



Для любого квеста большую роль играет движок игры. Мы сразу определяем: трехмерный, мультяшный, рисованный, или использовано живое видео. Каждый выбирает по вкусу. В данном случае эта игра сделана как двумерный пластилиновый квест, в стиле "Neverhood". Должен сразу отметить, что его создавали те же люди, что и оригинальный мультфильм, поэтому профессиональность видна в каждой детали. Яркость цветов, качество проработки персонажей НИЧЕМ не отличается от мультфильма, а все движения — это просто сказка! Просто не к чему придраться... поначалу. После прохождения игры у меня возникло стойкое ощущение, что где-то к середине работы их менеджер посмотрел на калькуляцию, что выдали им девелоперы, и произнес: "Хочешь не хочешь, а маловато будет!", И дал приказ уложиться в сроки и стоимость, чего бы это ни стоило. На последних уровнях "задники" начинают беднеть, число вариантов решения задач резко уменьшается — словом, торопились разработчики. Так и хочется высказать это "1С": "Неужели вам, господа, полноценный хит не нужен, который будет продаваться годами?"

Ладно, перейдем к сюжету. Тут все понятно и, как всегда, слегка замороченно — какая местность, такие и жители, какие жители, такие и задачи с их решениями :). Главное, приноровиться и просто проникнуться духом мультфильма. Не спорю, временами логика начинает "опускать глаза", как это делала в небезызвестной серии про Петьку и Василия Ивановича, и приходится сильно напрягаться, чтобы понять, ну чего же от тебя хотят. Зато потом можно весело посмеяться над тем, как герой решает свои проблемы. Да и лица на первых этапах все будут знакомые, дружной толпой перекочевавшие из ВСЕХ популярных пластилиновых мультиков Александра Татарского. И "на десерт" — персонажа и Автора озвучивает неподражаемый Станислав Садальский, да-да, тот самый, что озвучивал мультфильм!

Головная боль любого квестомана — предметы, их поиск и применение. О, этот навязший в зубах пикселхантинг, и медленное вождение мышкой по экрану... Тут, с одной стороны, все просто, предметы такие большие, что промазать по ним просто невозможно, да и на каждой локации их немного. А вот их применение... Не знаю, кому пришла в голову такая "гениальная" идея, которую разрекламировали как особенную "фичу" игры — для выполнения некоторых действий требуется неоднократное повторное применение одного и того же предмета... Ведь как действует наш брат-квестовик: щелкнул одним предметом по другому, получил отрицательный ответ — берет следующий предмет... А тут, поди догадайся... Многие мои знакомые застряли уже через пять минут, с открытыми ртами глядя на безвыходную по их меркам ситуацию. Хорошо, что я был предупрежден, и попытался одно и то же действие сделать несколько раз. И на пятый у меня получилось. Вот и щелкал после этого всю игру мышью, как дятел, теплым словом поминая того, кто это придумал. Ведь после энного раза это начинает порядком утомлять!

В общем, всего в этом квесте хватает — и хорошего, и плохого, и смешного, и грустного. Не получился полноценный хит, но и провала тоже нет — добротный, хороший квест получился, в который поиграть я рекомендую всем.

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***

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,
Голивац 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 К";
- 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, видеокарта 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" (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, 8 — 12	2 — 8, 10 — 12	4, 6, 7, 9, 11, 12	20	800	5
2001	2 — 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 — 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 "Свеловский" (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 Независимости").

Падал прошлогодний снег