10•2002
http://radio-mir.com
Индексы: 48925, 45995

Радиомир

Ваш компьютер

В НОВЫЙ ВЕК —
С НОВЫМ НАЗВАНИЕМ!

АНТИВИРУСЫ И ЗАЩИТА 2002

АНТИВИРУСНЫЕ ПРОГРАММЫ
СЕТЕВАЯ ЗАЩИТА
СНИФФЕРЫ
КРИПТОГРАФИЯ И ЗАЩИТА ДОКУМЕНТАЦИЯ

Золотой Софт 2002

Вирусов НЕТ!
УНИВЕРСАЛЬНЫЕ ПРОГРАММЫ
Microsoft Windows Me
Официальная русская версия
Microsoft Windows 98 Se
Официальная русская версия
Microsoft
+ Новые ут
+ Новые об
+ Новые ар
+ Новые ан
и Многое
Золотой Софт 2002
ЗАГ

1000 мелочей
Все для Windows XP

Утилиты для украшения
Утилиты для ускорения
Утилиты для настройки и т
Полезные программы дл
Документация

ГИГАНТЫ CAKEWALK v.2.0

Cakewalk Music Creator 2002 Edition + Home Studio XL
Cakewalk Guitar Tracks Pro v2 + Cakewalk Plasma + Sonar XL v3.0
Cakewalk Pyro MP3 and CD Maker v1.5 + FruityLoops v3.4
Expansion VST-DX Adapter Standard Edition v3.302 + The Tassman v2.1
Expansion DR-008 DXi v1.01 + Rhythm N Chords Pro Gold v2.031

Русская и английская версии

Mathcad PROFESSIONAL 2001i

с внешним
возможностью
платформу Windows XP!

PYRO MP3 & CD

АСПИРИН 2002

Millenium Collection
Рекомендовано всем компьютерам!
коллекция профессионала
МУЛЬТИЗАГРУЗОЧНЫЙ ДИСК
700Мб - огромной пользы!!!
РАБОТАЙ - НАСЛАЖДАЙСЯ!!!!
Microsoft Windows Millennium русская версия
Microsoft Office XP русская версия
WinAMP 2.78 +русификатор
DirectX v8.1 русская версия
WinFax v10.0 Professional русская версия
Goverilka 1.38b+банк словарей русская версия
ACDSee v4.01+Русификация
VentaFax&Voice 4.43 русская версия
Apple QuickTime v5.0.1 Final
IrfanView 3.6 русская версия
Различные тесты

ФАКС-МОДЕМ

дома и в офисе

лучшие программы для работы с факс

ABBYY SOFTWARE HOUSE
LINGVO® 8 Multilingual Edition
FINEREADER® 6 Professional

2 in 1

ИЗДАТЕЛЬСТВО Media 2000

ОБУЧЕНИЕ РАБОТЕ С INTERNET

+ СИСТЕМА КОНТРОЛЯ ЗНАНИЙ

С помощью этой программы
Вы освоите работу в Интернет.
Курс лекций построен таким образом,
чтобы дать пользователю основные
сведения, позволяющие
эффективно работать с Интернетом
без больших затрат времени
и усилий на изучение работы

РАБОТАЕМ
УЧИМСЯ
СМОТРИМ

Глоссарий современных 3D-терминов

Рис.10

Рис.7

Рис.6

Рис.11

Рис.8

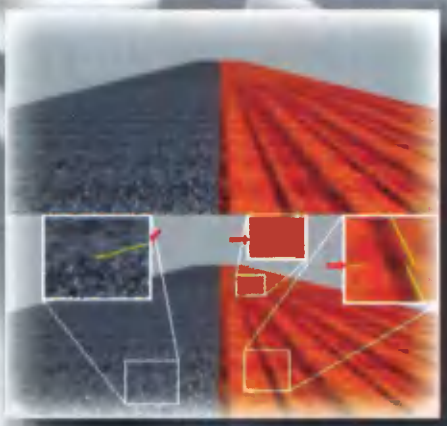


Рис.14

Рис.9

Рис.2

Рис.1

Рис.4

Рис.3

Рис.12

Рис.5

Рис.13

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 графике —
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,
факс (017) 221-01-10

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

Наши платежные реквизиты:
получатель: ООО "Радиомир Пресс",
ИНН 7729404741,
р/с 40702810900010000084
в ООО КБ "Агропромкредит"
ф-л Центральный, г.Москва,
корр. счет 30101810500000000109
БИК 044525109
Адрес банка: 113054, г.Москва,
ул.Зацепа, 43Б

Материалы для публикации принимаются в рукописном, печатном и электронном вариантах

Требования к графическим материалам рекламного характера в электронном виде: CorelDRAW до 9.0, 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 23.08.2002 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная.

Отпечатано в типографии
ЗАО "Красногорская типография".
Заказ 2215.

радиомир
Ваш компьютер

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

№10/октябрь/2002

ЧИТАЙТЕ В НОМЕРЕ

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

Сам себе телережиссер

Телевизор смотрят все. И практически все недовольны тем, что по нему показывают. Тем не менее — смотрят. Смотрят, и нередко хотят вмешаться в происходящее — попеть с певцами, потанцевать с танцорами, в конце концов, просто "поговорить за жизнь" с тем или иным гостем очередной телепередачи.

На этой почве в России родилась и быстро расцвела такая телепрограмма как "Сам себе режиссер". Тысячи и тысячи россиян любых возрастов, профессий, мест обитания снимают себя, своих друзей, детей, прочих родственников, рассчитывая затем увидеть свою видеопленку с экрана национального телевидения, ибо попасть в "ящик" — заветная мечта многих. Но у передачи "Сам себе режисссер" есть один существенный недостаток — она позволяет увидеть только заранее снятый сюжет, не давая при этом возможности для интерактивного участия в происходящем.

В США эту проблему успешно решила TechTV — популярная кабельная сеть, которая освещает технологические достижения, передает новости и развлечения 24 часа в сутки. TechTV была основана в 1997 г. компанией Ziff-Davis, специализирующейся на публикациях технологических обзоров. Исследования показали, что TechTV охватывает более 30 миллионов семейств в США, а подготовленные ею передачи распространяются в 70 странах мира. Ее онлайновое представительство — TechTV.com — принимает более полутора миллионов посетителей в месяц и служит порталом для общения.

Суть идеи очень проста: рядом с телевизором ставится компьютер, а на него — Web-камера. После этого можно пересылать через Интернет в телестудию как отснятые видеосюжеты, так и собственное изображение в данный момент времени. Благодаря возможности передачи живого или записанного потокового видео зрители становятся еще и участниками действия, находясь дома, в офисе — да где угодно (есть даже любители присылать репортажи из автобуса). Они задают вопросы участникам викторины, обмениваются советами относительно оборудования, узнают котировки акций, представляют публике свои домашние видеофильмы и поддерживают свое онлайн-сообщество. Короче, участвуют в любой технической акции в любое время и из любого места.

Как и всюду, в Америке многие хотят похвастаться своими видеороликами. Желающих послать свои работы так много, что составители программ TechTV создали для этого специальный портал. На этом Web-сайте размещаются все поступления, посетители могут их просматривать и голосовать за своих фаворитов. Потом проводится виртуальная церемония награждения, и победители звонками в эфир, как водится, благодарят своих матерей, отцов, жен, мужей (стандартный список читателям известен и так).

Проблема лишь в том, что передача качественного видеоизображения означает необходимость пересылать 24 кадра в секунду (это стандартная скорость кинотрансляции), а каждый кадр — это десятки мегабит информации. Таким образом, чтобы передавать через компьютер изображение с телевизионным качеством, надо ежесекундно обрабатывать сотни мегабит данных. Сегодня такие нагрузки под силу только процессору Intel Pentium 4!

Во времена запуска TechTV четыре года назад при работе с сетью проводились эксперименты со множеством программных продуктов с целью улучшения передачи видеопотока. Были испробованы несколько лицензионных и свободных программных средств, но качество оставляло желать лучшего. Робин Хьюсвельдт, менеджер TechTV Netcam Network, вспоминает: "Наши зрители уже начали привыкать к низкой частоте смены кадров Web-камер, но мы постоянно искали способ повысить качество видеоизображения, передаваемого непосредственно в эфир".

И, как говорится, "тут пришел Intel", точнее — процессор Intel Pentium 4. Впрочем, Intel в TechTV присутствовал с самого начала, ведь именно сеть Intel TechTV Netcam Network позволяет зрителям попасть на телеэкран с помощью Web-камер, подключенных к их компьютерам.

Так вот, Чад Баркер (Chad Barker), менеджер Intel по стратегическим проектам, как раз в то время занимался продажей программного обеспечения одного из независимых разработчиков — компании Inetcam. Баркер искал возможность объединить это ПО с мощностью и производительностью процессора Intel Pentium 4. В ходе этих поисков он установил контакт с TechTV. Остальное читателю уже известно.

Качество изображения на экране — очень важное его свойство. Избалованные хорошими телевизорами зрители просто не станут смотреть сюжет, идущий с меньшей частотой кадров и со смазанными фигурами, или картинку маленького размера. Процессор же Intel Pentium 4 не просто улучшает видеоизображение — он предоставляет небывалые прежде возможности.

TechTV использует программное обеспечение Inetcam iVISTA, которое обеспечивает безопасную аудио- и видеотрансляцию на базе Windows через одноранговую сеть. Как показали исследования, человек, просматривающий видеоряд с помощью iVISTA, получает изображение непосредственно из ПК отправителя. Это значит, что обладатель компьютера на базе процессора Intel Pentium 4 может посылать видеоизображение с более высокой частотой смены кадров, с большим размером картинки и более ровным изображением одновременно большему количеству зрителей.

В январе 2002 г. TechTV объявила о двухмесячных испытаниях ПО для Web-вещания iVISTA "Смотри и слушай меня прямо сейчас" ("Hear and See Me Now"), которое пользовалось спросом у зрителей. Инженеры Inetcam оптимизировали коды для разных процессоров Intel. Пользователи, имеющие процессор Intel Pentium 4, получили наилучшее по качеству изображение. "Pentium 4, несомненно, лучший процессор для просмотра видео, — считает Лео Вольфсон (Leo Volfson), президент и технический директор Inetcam, — новая архитектура позволяет усовершенствовать процесс обработки сигнала".

Рассказанный случай — еще одна иллюстрация лозунга, провозглашенного главой корпорации Intel Крейгом Барреттом: с появлением процессора Intel Pentium 4 ПК превратился в центр цифровой Вселенной. Эта формулировка включает в себя управление цифровой фотографией и фотоальбомами, цифровым видео, цифровой музыкой, цифровыми коммуникациями, играми и многим другим, что уже существует, и что появится в будущем. Сегодня все это так или иначе связано с компьютерами на базе процессора Intel Pentium 4, а в дальнейшем будет работать с ПК, основанными на последующих, еще более мощных процессорах Intel, о появлении которых на свет мы слышим едва ли не ежемесячно.

По материалам www.intel.ru

3DNEWS Daily Digital Digest

Глоссарий современных 3D-терминов

Д.ЧЕКАНОВ,
E-mail: dmitry@vspu.kirov.ru,
FRANCHY (ЗоринD),
E-mail: dima@bws.rosugol.ru,
С.ВАСИЛЬЕВ,
E-mail: stanislav@3dnews.ru,
А.ГУЛЯЕВ,
E-mail: jeff3d@delphicity.net

Введение

Этот словарь был создан для разъяснения многих 3D-терминов, с которыми вам приходится сталкиваться. Каждый термин первоначально дается в английском варианте, далее следует общепринятый перевод и толкование. Мы старались сделать объяснения как можно проще и доступнее, используя аналогии из обычной жизни и метафоры.

A3D (Aureal3D) — API, создающий 3D-позиционирование звука и эффект Доплера с использованием только двух колонок. Другие реализации 3D-звука (аппаратные или программные) работают с четырьмя или более колонками. Довольно интересен тот факт, что A3D был создан с помощью разработанных NASA алгоритмов. Для чего они были разработаны изначально, остается только догадываться. Какова же практическая ценность A3D для пользователя? Как правило, поддержка A3D подразумевает программную реализацию звука в играх и программах. Причем технология не только позволяет ощутить местоположение источника звука в пространстве (3D-позиционирование), но и оценить удаленность источника от слушателя (эффект Доплера). Например, когда вы приближаетесь к источнику звука в 3D-игре, звук становится громче, когда же вы удаляетесь — тише. В реальной жизни такой эффект обусловлен расхождением звуковых волн при удалении от источника.

AC-3 — технический термин, принятый в стандарте объемного звука Dolby Digital 5.1 Surround Sound. AC-3 является алгоритмом, раскладывающим звуковой поток на пять отдельных каналов, каждый канал посылается на свою колонку. Чем полезен AC-3? Когда вы смотрите фильм на большом экране, именно AC-3 делает звук объемным. Например, в фильме "Скорость" можно почувствовать, как автобус выезжает откуда-то сзади. На самом деле, звук движущегося автобуса постепенно перемещается с задних колонок на передние. Алгоритм является еще одним способом создания 3D-звука. Технология Aureal3D реализует тот же самый эффект, но при меньших требованиях к оборудованию. В общем, "Dolby Surround" — это и есть AC-3.

Accelerated Graphics Port (AGP), ускоренный графический порт — шина расширения, разработанная Intel для подключения видеокарты. Современные видеокарты чаще всего используют именно этот разъем, хотя до сих пор выпускаются и PCI-варианты. В чем различие между шинами PCI и AGP? Если судить по скорости, AGP может посылать центральному процессору информацию в четыре раза быстрее, чем PCI. Частота 4X AGP составляет 266 МГц, (термин "мегагерц" в примитивном понимании обозначает единицу скорости обработки и передачи информации). Хотя многие современные карты умеют работать с AGP только на скорости 2X (133 МГц), большинство материнских плат сейчас оснащаются 4x AGP-портом. Большинство видеокарт и видеоускорителей выпускаются для двух шин — AGP и PCI. AGP быстрее, чем PCI, что положительно влияет на производительность. Перед приобретением видеокарты убедитесь в наличии в вашем компьютере шины AGP.

Accelerator, ускоритель, акселератор — это карта или плата, расширяющая возможности компьютера. Обычно она является аппаратным решением, самостоятельно обрабатывающим какую-либо информацию. При этом освобождаются ресурсы центрального процессора. Примером такой информации можно считать обработку видео, звука или декодирование DVD. Хотя для нормальной работы акселератора он должен задействовать некоторые ресурсы центрального процессора, при его использовании происходит ощутимое повышение скорости и качества выполнения ускорителем задач. Представьте, что вашему другу необходимо сделать кучу работы, перед тем как он сможет пойти на вечеринку. Если вы ему немного поможете, то вы будете выступать в роли ускорителя, а он — в роли процессора. Два самых популярных вида акселераторов — видеоускорители и аудиоускорители. Часто они совмещают в себе звуковую и видеокарты. Почти каждый ПК сегодня оснащен видеоускорителем — 2D, 3D или 2D/3D совместно.

Algorithm, алгоритм. В двух словах — это последовательность действий для выполнения чего-либо. Алгоритмы используются, к примеру, в алгебре или в информатике. Предположим, вам нужно поджарить яичницу. Для этого следует купить яйца, поставить сковородку на огонь, разбить яйца, вылить их на сковородку. Эта последовательность действий и есть алгоритм.

Algorithmic Procedure Texturing, алгоритмическое процедурное текстурирование — способ рендеринга изображений с виртуально бесконечной детализацией. Проще говоря, представьте себе алгебраическую формулу, которая позволяет показать картинку на экране с бесконечным уровнем детализации, ограниченным только возможностями компьютера. Чтобы понять принцип работы, посмотрите на определение алгоритма. Слово "процедурное" означает последовательность действий. А "текстурирование" — это просто создание изображения с многочисленными свойствами.

Alpha-Blending, альфа-смешение — создание полупрозрачных объектов, возможность задать изображению или отдельному пикселу специальный атрибут, определяющий, как будет выглядеть изображение — сплошным (не пропускающим свет), невидимым (прозрачным), или полупрозрачным. Текстура, наносимая на объект, может содержать, помимо информации о цвете (Red, Green, Blue), информацию о прозрачности (Alpha). В зависимости от величины коэффициента Alpha разные части объекта приобретают различную степень прозрачности, т.е. при использовании совместно с полигонами альфа-смешение позволяет созда-

вать стекло, воду или другие виртуально прозрачные элементы (**рис.1**, все рисунки на 2-й странице обложки). Как правило, смешивание цветов перекрываемого объекта и полупрозрачного объекта (с альфа-прозрачностью) происходит по следующей формуле: *(alpha) * (значение цвета объекта с прозрачностью) + (alpha-1) * (значение цвета покрываемого объекта)* при *0≤alpha≤1*.

Anisotrophic Filtering, анизотропная фильтрация. Перед чтением узнайте, что такое текстурирование. Анизотропная фильтрация — самый совершенный тип фильтрации, она фильтрует (или смешивает) данную текстуру, учитывая три измерения объекта. Остальные способы фильтрации просто усредняют цвет выводимого пиксела, принимая во внимание цвет исходных пиксвлов, что делает картинку или слишком размытой, или слишком резкой (**рис.2**).

При усреднении анизотропная фильтрация принимает во внимание трехмерную модель объекта, а конкретно — нужный полигон. Однако это преимущество над трилинейной фильтрацией обходится весьма дорого и может очень сильно замедлить работу графического процессора. Кстати, Riva TNT была первым 3D-ускорителем, способным выполнять анизотропную фильтрацию.

Anti-allasing, сглаживание. В применении к 3D-технологии этот термин подразумевает сглаживание характерных "ступенек" линий, прорисованных под углом. Посмотрите на стену в Quake II. Вы наверняка заметите "ступеньки" на линиях, показывающих границу стены. При аппаратной прорисовке 3D-сцены сглаживание может замещаться или дополняться билинейной и трилинейной фильтрацией, поскольку их технологии уже реализуют некоторое подобие сглаживания (**рис.3**).

API, Application Programming Interface — программный интерфейс приложения. Он присутствует в любой операционной системе, для того чтобы программисты использовали обращение к API вместо дублирования существующего кода. Что это значит применительно к 3D? Например, игра была написана на своем коде для вывода 3D-изображения. Таким образом, она становится несовместима с большинством 3D-ускорителей — авторы заложили в игру работу только с определенным ускорителем. В итоге, изготовители игр и изготовители оборудования пришли к созданию API, который каждый смог бы использовать. Это означает, что драйверы ускорителя будут работать с API, и игра будет работать с этим же API. Проще говоря, игра и ускоритель могут общаться на одном языке. Самые популярные 3D API — OpenGL, Direct3D и Glide3D. Хорошие игры и ускорители для большей совместимости поддерживают сразу несколько API. Подумайте, кто будет покупать ускоритель, если он не будет работать с каждой игрой, и кто будет покупать игру, если она не будет работать с вашим ускорителем?

API Extension, расширение API. Сначала познакомьтесь с термином API. Расширение API является некоторой надстройкой или добавлением к существующему API для оптимизации под требуемое аппаратное или программное обеспечение. Например, это Creative Labs EAX (работающий совместно с Microsoft DirectSound3D) и оптимизированный под 3dfx OpenGL (программное расширение для старого доброго API OpenGL).

Aspect Ratio, формат экрана. Отношение ширины экрана к его высоте. Например, разрешение 800x600 имеет формат экрана 4:3.

Artifact, артефакт. Результат плохой реализации компрессии текстур. Если вы наблюдаете сжатую текстуру, и некоторые ее части кажутся "смазанными", то это и будет артефактом. На двумерных и трехмерных изображениях артефакты могут образовываться на стыке цветов.

Auto Texture Compression, автоматическое сжатие текстур. Такая функция обычно является прерогативой графических ускорителей. Они автоматически сжимают текстуру для уменьшения ее размера. Таким образом, графика работает быстрее без ощутимой потери в качестве.

Bilinear Filtering, билинейная фильтрация. Эта возможность реализована в большинстве 3D-ускорителей. Билинейная фильтрация позволяет получать графику с менее заметными пикселами и с менее выделяющейся блочной структурой. Она применяется для получения более сглаженных текстур, которые визуально приятнее (**рис.4**).

Как она работает? Берутся четыре соседних пиксела (или тексела), их цвета усредняются. В результате мы получаем один пиксел с усредненным цветом. Однако билинейная фильтрация может привести к тому, что объект потеряет "виртуальную" глубину или свой естественный вид. Здесь приходит на помощь другой вид фильтрации — трилинейная или более продвинутая — анизотропная.

Bitmap, битовое изображение или битовая карта. Под этим понимают практически любое изображение, которое мы видим на экране компьютера. Оно состоит из массива точек, упорядоченных в вертикальные и горизонтальные ряды. Каждая точка (или пиксел) имеет атрибут — цвет. Совокупность точек формирует изображение. Чем больше битов выделяется под атрибут цвета, тем больше цветов мы увидим на картинке, тем более естественно она будет выглядеть.

Bitmapping, формирование битового изображения. Формирование битового изображения осуществляется вашим оборудованием. Оно создает цельную картинку, прорисовывая каждый пиксел в рядах и строках.

Blocky Filtering, блочная фильтрация. На самом деле это не метод фильтрации, а термин. Он описывает появление крупных "блочных" пикселов на текстурированных объектах. Если вы не совсем понимаете, о чем идет речь, то включите Quake II в режиме software, и все увидите.

Bump-Mapping, отображение шероховатости поверхности. Визуальный прием, используемый для придания поверхностям объекта характерных неровностей. Для этого разработчики привязывают по две текстуры к каждому полигону. Одна из них является обычной, базовой текстурой. Вторая — текстура смещения, описывающая неровности объекта (**рис.5**).

Возникает вопрос, почему бы просто не прорисовать эти неровности на текстуре объекта? Все упирается в освещение. При bump-mapping отражение света от неровностей меняется в зависимости от угла зрения, как в реальном мире. Поэтому сейчас произошел переход от использования текстур с прорисованными неровностями к использованию bump-mapped-текстур.

Bump-Maps, две текстуры, используемые для отображения шероховатости поверхности. С помощью этих текстур отображаются различные неровности объекта (см. bump-mapping).

Clock Cycle, такт. Это одна операция микропроцессора. При этом с помощью транзисторов производятся логические операции. Современные процессоры (в том числе и процессо-

ры видеоускорителей) обычно выполняют несколько сотен миллионов операций в секунду.

Clock Frequency, Clock Speed — тактовая частота. Параметр, показывающий скорость выполнения целочисленных операций. Тактирование производится с помощью генератора, и она помогает синхронизировать операции внутри чипа. Тактовая частота выражается в мегагерцах или миллионах колебаний в секунду. Например, при тактовой частоте 200 МГц процессор выполняет 200 миллионов тактов в секунду.

Collision Detection, обнаружение столкновений. Возможность 3D-объектов взаимодействовать с другими 3D-объектами естественным образом. Когда вы в 3D-игре видите коробку на полу, то коробка "знает", что ей не следует проваливаться сквозь пол. Эти "знания" являются результатом работы по обнаружению столкновений. Процесс обнаружения столкновений постоянно совершенствуется, так что игры становятся все более "как живые". Сейчас даже маленькие объекты могут взаимодействовать с другими объектами весьма реалистично.

Color Convergence, совмещение цветных растров. Для получения различных оттенков в цветном мониторе происходит совмещение трех растров — красного, зеленого и синего. Компьютерный монитор может осуществлять очень точное совмещение цветных растров.

Colored Lighting, цветное освещение. Для отбрасывания теней может использоваться не только белый свет, но и цветной. Многие игроки сейчас считают этот эффект само собой разумеющимся, без него игры выглядели бы более серыми.

Composite, композитный сигнал. В телевизионном сигнале комбинируются красный, синий и зеленый цвета. Такой термин чаще всего используют для обозначения композитного входа/выхода 3D-ускорителя. К такому выходу можно подключить телевизор и смотреть на нем компьютерное изображение либо передавать изображение в компьютер.

Compression, сжатие. Сжатие приобретает все большее значение в компьютерной индустрии. Любой слушатель MP3-музыки вам это подтвердит. Как применяется компрессия в 3D-видео? Сжатие — это возможность уменьшения размера файла без потери значительных графических деталей. 3D-ускоритель быстрее работает со сжатыми текстурами. Если желаете узнать больше, почитайте про автоматическую компрессию текстур.

CPU, Central Processing Unit, центральный процессор. Хотя некоторые аудио- и видеокарты умеют независимо обрабатывать данные, почти все операции компьютера проходят через центральный процессор. Если сравнить компьютер с человеком, то центральный процессор — это мозг человека.

Daughter Card, дочерняя плата. Так называется карта, для работы которой требуется еще одна плата. Примером дочерней карты является видеоускоритель Voodoo2. Он может работать только совместно с видеокартой. Связь между устройствами осуществляется по специальному кабелю. Voodoo2 ускорял только 3D-графику, поэтому для прорисовки двумерного изображения требовалась отдельная карта.

DDR-SDRAM, Double Data Rate SDRAM, память SDRAM с двойной скоростью передачи данных. Такая память способна передавать сигнал по обоим фронтам тактирующего синхроимпульса. Таким образом, скорость памяти виртуально увеличивается — например, от 100 МГц до 200 МГц. DDR SDRAM сейчас используется в некоторых видеоускорителях nVidia GeForce или других устройствах, требующих быстродействующей памяти.

Dedicated Frame Buffer, выделенный кадровый буфер. Это некоторое количество памяти, используемое для хранения данных кадрового буфера и/или Z-буфера. Такая архитектура памяти называется раздельной (split memory architecture). Большинство 3D-карт используют другую архитектуру памяти, где память для хранения текстур не отделена от памяти кадрового буфера. Естественно, там уже не существует "выделенного кадрового буфера".

DVD, Digital Video/Versatile Disc, цифровой видеодиск, многослойный диск. DVD хранит 4,7 Гб информации на одной стороне в одном слое, что почти в семь раз больше стандартного CD-ROM (в среднем, 650 Мб).

Direct3D. Является графической частью Microsoft DirectX API. Почти все трехмерные ускорители и игры поддерживают Direct3D. Поэтому Direct3D играет значительную роль в современной компьютерной индустрии, а производители видеоускорителей пытаются достичь высоких результатов в тестах Direct3D. Кроме Direct3D, существуют и другие API для работы с трехмерной графикой, например, OpenGL или Glide.

DirectSound. Является API производства Microsoft для работы со звуком. Разработчики игр могут оптимизировать свои продукты не под конкретные звуковые карты, а под DirectSound. Ну а так как это API, то подробнее про его работу вы можете прочитать в соответствующем разделе. DirectSound является частью API DirectX.

DirectSound3D. Что же привносит окончание 3D в DirectSound API? Microsoft пытается реализовать трехмерное позиционирование звука. Первые варианты были не совсем удачны, но команда разработчиков делает свое дело, и API становится все лучше.

DirectX. API производства Microsoft. Стал стандартом де-факто, причину вы, думаем, понимаете. Сейчас этот API поддерживает множество устройств и приложений. Две самые важные части DirectX — это DirectSound3D и Direct3D.

Displacement Map, карта смещения. Вторая текстурная карта, используемая при отображении шероховатостей поверхности. Карта показывает, каким образом на оригинальной текстуре отбрасываются тени от неровностей.

Distortion, искажение, дисторция. Под дисторцией понимают искажение звуковых частот, производимых звуковой картой и колонками.

Dithering, сглаживание переходов между цветами. Является зрительным артефактом, появляющимся при уменьшении количества используемых цветов (или уменьшении глубины цвета). К примеру, в результате сглаживания переходов текстура теряет свою четкость, на ней будет заметна пикселизация.

Doppler Effecting, эффект Доплера. Заключается в повышении высоты звука при приближении объекта и понижении — при удалении. В играх эффект Доплера играет значимую роль, он делает игру реалистичнее (см. Aureal3D).

Driver, драйвер. Программное обеспечение, являющееся посредником между различными устройствами и другими программами. Без драйверов оборудование будет невозможно использовать. Поэтому важной задачей операционной системы является управление драйверами.

D3D (Direct3D) — см. Microsoft's Direct3D API.

EAX, Environmental Audio, естественный звук окружающей среды. Является собственной разработкой Creative по позиционированию звука, а также одновременно и API. Что самое приятное — стандарт базируется на реверберации.

Electro-planar, электро-планарный. Так называется технология, используемая в плоских колонках. Электромагнитное поле заставляет материал плоских колонок вибрировать и излучать звук.

Engine, движок. Движок — рабочая основа игры, создаваемая программистами. В зависимости от качества программирования и используемых возможностей, движок может реализовывать весьма "навороченные" эффекты и графику. Хороший движок обеспечивает высокую частоту смены кадров и скорость игры.

Environment-mapping, использование карт окружающей среды. Текстура с использованием этой технологии будет аккуратно отражать окружающие текстуры и объекты.

Fill Rate, скорость заполнения. Показывает скорость прорисовки пикселов на экране монитора. Чем больше скорость заполнения, тем лучше. Скорость современных видеокарт измеряется в миллионах пикселов в секунду.

Filtering, фильтрация. Бывает билинейной, трилинейной и анизотропной. С помощью фильтрации ускоритель сглаживает текстуры, усредняя цвет пиксела с окружающими пикселами.

Flat shading (плоское затенение) — каждая грань объекта закрашивается определенным цветом. Для этого строятся вектора нормалей к поверхностям полигонов и определяется цвет (в зависимости от угла между нормалями и направлением на источник света). Все выглядит граненым (**рис.6**).

Floating-Point, числа с плавающей точкой, десятичные дроби. Число, состоящее из двух частей — целой части и дробной части — называется числом с плавающей точкой (не углубляясь в математику). Очевидно, что такие числа точнее целых. Процессор, хорошо выполняющий операции с плавающей точкой, сможет обеспечить лучшую игровую производительность. Современные игровые движки в значительной части опираются на вычисления с плавающей точкой. Как вы помните, в 1998 г. AMD выпустила набор инструкций 3DNow! Этот набор инструкций ускорял операции с плавающей точкой. Оптимизированные для 3DNow! игры улучшали производительность на 5...10 кадров в секунду.

FMV (Full Motion Video). Заранее созданный и не интерактивный видеоклип. Возможность Sony Playstation проигрывать FMV-ролики из игр была важной вехой в борьбе против Nintendo64. Естественно, такой возможностью обладает и PC.

Fogging, создание тумана. Туман является еще одним хорошим спецэффектом, реализуемым 3D-ускорителями. Разработчики игр могут помещать туман для усиления других эффектов, например, кипящей воды. Или, что довольно грустно, туман может скрывать мелкие текстуры при удалении от них. Разработчики просто поленились создать крупные текстуры. Вспомним Acclaim и Turok: Dinosaur Hunter.

FPS (Frames per Second), количество кадров в секунду. Очень часто производительность видеокарты измеряют по количеству кадров, которые она прорисовывает в секунду. Если вы посмотрите определение кадра, то поймете смысл. Чем быстрее ускоритель, тем быстрее он отрисовывает кадр. Хорошая видеокарта выдает в среднем 60 fps. Не хочется полемизировать, но я так и не смог на глаз отличить 60 fps от 120 fps во время игры. Обычно же игры не выдают больше 40...50 fps. Кстати, чем выше разрешение, тем меньше количество кадров в секунду.

Frame, кадр. Кадром называют одну статическую картинку, отрисованную на вашем мониторе. Если вы знакомы с мультипликацией, то поймете, в чем тут дело. Следующий кадр немного отличается от предыдущего. Постоянная смена кадров создает эффект движения на экране. Благодаря смене кадров вы смотрите кинофильм, играете в игру или перематываете вниз страницу в браузере.

Frame Buffer, кадровый буфер. Некоторое количество памяти на видеоускорителе выделяется для хранения временных кадров, которые будут затем выведены на экран. Это количество и называется кадровым буфером. Больший буфер позволяет хранить кадры с большим количеством цветов и большим разрешением.

(Окончание следует)

Зачем император Нерон поджег Рим?

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

Советы начинающим "прожигателям" компакт-дисков

Какова бы ни была емкость жесткого диска — рано или поздно свободный объем дискового пространства, доступный для хранения данных, уменьшится до критического уровня. Установка современного программного обеспечения стремительно поглощает мегабайты свободного места на винчестере. Свою "лепту" вносят и постоянно пополняющиеся коллекции видеофильмов, клипов, звуковых файлов, графики и игр. Проблемы, связанные с эффективным, надежным и длительным сохранением данных, резервным копированием и оперативным переносом больших объемов информации являются одними из наиболее распространенных и актуальных проблем для активных пользователей PC. Поиск решения вышеуказанной проблемы приводит к следующим вариантам:

1. Архивация файлов на дискеты. В этом случае понадобится большое количество дискет и много времени, да и надежность сохранения файлов вызывает большие сомнения.

2. Применение стримеров. Однако похоже, что на эти медлительные устройства спрос в последнее время заметно упал, и они постепенно уходят в небытие.

3. Использование магнитооптических накопителей и других устройств со сменными носителями информации — диски Zip и Jaz компании Iomega. Этот вариант характеризуется высокой себестоимостью хранения информации.

4. И, наконец, не только на мой взгляд, самое оптимальное решение — применение приводов CD-R (CD-Recorder) и CD-RW (CD-ReWriter). Это универсальные устройства, по-

зволяющие записывать на компакт-диски CD-R и CD-RW объем информации 650 Мб (74 мин.) и 700 Мб (80 мин.). Записанные компакт-диски, как правило, должны считываться на всех приводах CD-ROM, CD-R и CD-RW. Разница между приводами CD-R и CD-RW заключается в том, что приводы CD-RW позволяют многократно перезаписывать однажды записанные диски CD-RW (ReWritable). За последнее время цены на эти устройства снизились до $90...100, что делает их все более привлекательными для применения в офисных и домашних компьютерах.

Вот некоторые полезные советы, которые могут пригодиться при покупке и установке нового устройства CD-R/RW. Стоит остановить свой выбор на внутреннем приводе (int.) с интерфейсом IDE. Внутренние приводы, в отличие от внешних, не занимают лишнего места. Внешние приводы (ext.) с интерфейсом LPT работают медленно и сильно загружают процессор. Внешние или внутренние приводы с интерфейсом SCSI стоят дороже, и особых преимуществ в качестве записи и в скорости не дают. Приобретать лучше перезаписывающий привод (CD-ReWriter), так как возможностей у него больше, чем у привода CD-R, а цены практически сравнялись. Все современные приводы CD-RW должны "уметь" читать диски CD-ROM, CD-R и CD-RW. Приводимые в прайс-листах данные на приводы CD-RW 8x/4x/32x означают, что скорость записи дисков CD-R — 8x, скорость записи-перезаписи дисков CD-RW — 4x и скорость чтения дисков CD-ROM, CD-R и CD-RW — 32x. Все приводимые значения скоростей — максимально возможные.

Подключение приводов CD-R и CD-RW не должно вызывать никаких проблем. Нужно только учесть следующее — приводы CD-ROM и CD-R/RW рекомендуется подсоединять к разным каналам IDE, а не к одному, т.е. каждый привод должен присоединяться своим отдельным шлейфом. Это необходимо для того, чтобы при колировании данных с CD-ROM на CD-R/RW или при клонировании компакт-дисков не возникало задержек с передачей информации, грозящих опустошением буфера привода CD-R/RW. Исходя из вышеуказанного, возможны два варианта установки устройств с интерфейсом IDE (EIDE), которые указаны в **таблице**:

Вариант 1		Вариант 2	
IDE 1	Master — жесткий диск Slave — CD-R/RW	IDE 1	Master — жесткий диск Slave — CD-ROM
IDE 2	Master — CD-ROM	IDE 2	Master — CD-R/RW

Какой из двух предложенных вариантов вам больше подходит — решайте сами.

Первый вариант предпочтителен при частом копировании данных с CD-ROM на жесткий диск, при инсталляции разнообразных программ с компакт-дисков. В этом случае возрастает скорость передачи информации с CD-ROM на винчестер, поскольку CPU не будет ожидать завершения операций с винчестером, прежде чем он получит доступ к приводу CD-ROM.

Второй вариант, наоборот, предпочтителен при частом записывании данных с жесткого диска на привод CD-R/RW.

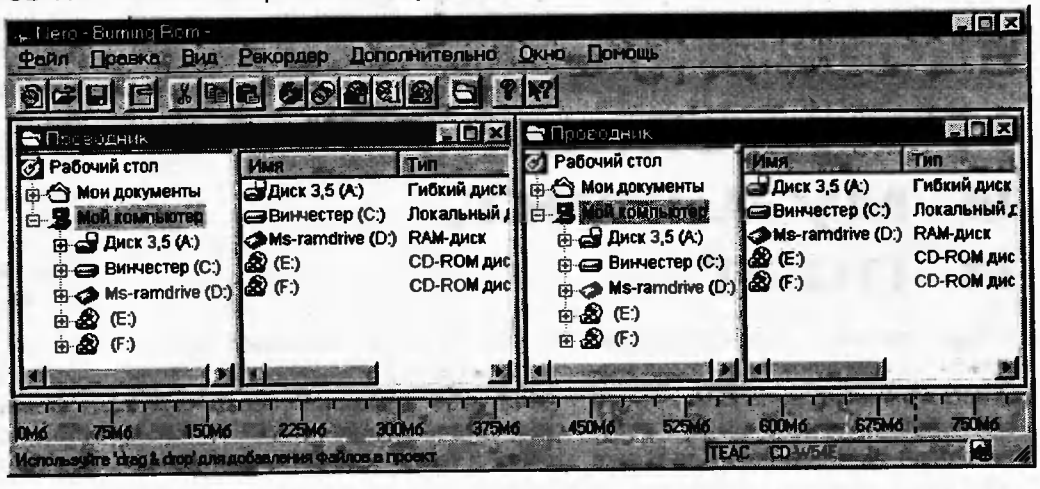
После установки и подключения, CD-рекордер будет определен операционной системой Windows 9x/Me как устройство чтения компакт-дисков. Это обстоятельство не должно вас смущать. Дело в том, что для того чтобы можно было записывать информацию на диски CD-R и CD-RW с помощью приводов CD-R и CD-RW, требуется специальная программа.

Перед покупкой привода стоит предварительно поинтересоваться у продавца комплектацией устройства. Если комплектация — OEM, то возможно, что, кроме привода, к нему больше ничего не прилагается, а это означает, что программу, которая позволит записывать на диски, придется приобретать отдельно. Возможен и такой вариант — с приводом поставляется программа, но версия этой программы устарела, и после ее установки обнаруживается, что программа не поддерживает новые модели приводов CD-R/RW, в том числе и тот, который вы купили. Возникает новая проблема — где взять и какую лучше использовать программу, позволяющую производить запись на диски CD-R/RW.

Для начинающих "прожигателей" компакт-дисков я советую использовать популярную программу **Nero Burning Rom**. На момент создания этой статьи в Интернете на сайте разработчика программы была размещена последняя версия Nero Burning Rom v5.5.6.4, релиз которой состоялся 4 декабря 2001 г. Разработчик этого программного продукта — фирма Ahead Software (http://www.nero.com). Для установки программы нужно скачать файл nero5564.exe, который находится в Интернете на странице http://www.nero.com/en/download.htm

Файл nero5564.exe "весит" 10,8 Мб, после инсталляции Nero Burning Rom займет на жестком диске 24,4 Мб. Программа удобна и проста в использовании, тем более, что ее можно легко русифицировать. Внешний вид рабочего окна программы приведен на **рисунке**.

Интересен тот факт, что свое название программа получила в честь древнеримского императора Нерона, жившего в I веке. Прославился он тем, что поджег столицу Великой Римской империи — город Рим. В названии программы заложен особый смысл, основанный на игре слов. Рим по-английски пишется Rome, а схожая с ним по написанию аббреви-



атура ROM — это, как известно — память только для чтения, в данном случае — CD-ROM. А поскольку Рим за всю историю своего существования, как утверждают историки, был сожжен только один раз (по приказу императора Нерона), то приводится аналогия с тем фактом, что компакт-диски CD-R прожигаются тоже только один раз. Если внимательно всмотреться в иконку программы Nero Burning Rom, то можно увидеть изображение горящего Колизея — исторической достопримечательности Рима и символа былой могущественности Великой Римской империи.

Как русифицировать программу Nero Burning Rom v5.5.6.4? Для этого нужно скачать файл-русификатор программы, который называется n5564rus.exe, и запустить его на исполнение. Найти этот файл в Интернете можно на той же странице, где размещена программа Nero. Русификатор сам по себе небольшой, всего 412 Кб. Обратите внимание на то, что для каждой версии программы нужен свой русификатор. После запуска программы-русификатора, русификация произойдет автоматически. Если этого не произошло, то причина, скорее всего, заключается в том, что был применен русификатор не от "родной" версии (русификаторы от разных версий не взаимозаменяемы), либо программа Nero Burning Rom расположена в другом каталоге (не по умолчанию), и русификатору надо указать путь, где установлена программа Nero.

Nero Burning Rom v5.5.6.4 можно установить под следующие операционные системы — Windows 95/98/Me/NT4/2000. Перед записью компакт-дисков надо обязательно провести проверку и дефрагментацию жесткого диска. Это необходимо, для того чтобы данные для записи компакт-диска считывались в буфер привода CD-R/RW без дополнительных задержек, связанных с чтением фрагментированных файлов винчестера. Записывать компакт-диски с помощью программы Nero Burning Rom просто, особенно, если вам все-таки удалось ее русифицировать. После запуска программы свои услуги предложит Помощник, который проведет вас за руку по всем требуемым установкам и упростит создание вашего CD-R/RW-диска. Если вы будете клонировать компакт-диски с CD-ROM на рекордер, используйте опцию "Быстрое копирование на лету". В этом случае копирование будет происходить напрямую, без предварительного сохранения данных на жестком диске.

И в заключение несколько общих советов, которые могут пригодиться начинающим. При покупке "болванок" обращайте внимание не только на цену, но и на маркировку, нанесенную на нерабочую поверхность диска. На ней, кроме фирмы-производителя, должна быть указана предельно допустимая скорость записи, например — BASF/Emtec CD-R 74min/650Mb 1x-8x. Не стоит рисковать при использовании дешевых безымянных дисков, записывая их на максимальной скорости рекордера. Перед записью дисков CD-R/RW очень уместно вспомнить две старинные русские поговорки: "Тише едешь — дальше будешь" и "Поспешишь — людей насмешишь".

Также в настройках программы есть возможность непосредственно перед самим процессом прожига имитировать этот процесс с целью проверки считывания данных с источника и передачи их на привод CD-R/RW, правильности всех установок, а главное — проконтролировать работу буфера. Не пренебрегайте этой возможностью. Лучше затратить вдвое больше времени и со спокойной душой записать диск, чем портить "болванки" и свое настроение. Используйте для записи своих данных 650-мегабайтные CD-R-диски, а 700-мегабайтные CD-R лучше применять для копирования или клонирования дисков, аналогичных по объему. Записывать диски лучше всегда целиком за одну сессию, потому что в режиме multisession при закрытии каждой сессии на диске расходуется от 13 до 20 Мб для размещения в TOC (Table of Contents) — аналоге FAT для CD-ROM. Так что сначала накопите на винчестере полный объем информации в 650 Мб, потом проведите дефрагментацию жесткого диска, и только после этого записывайте.

CD-RW-диски стоит применять только для переноса или временного хранения большого объема данных, а также для пробных и тестовых записей. После того как будет завершена запись какого-либо компакт-диска, можно для уверенности сделать проверку на сравнение содержимого копии и оригинала, например, с помощью файловых менеджеров Far или Windows Commander.

Весьма желательно подключать питание к компьютеру через источник бесперебойного питания (UPS). Использование источника бесперебойного питания во время записи компакт-дисков может во многих случаях спасти ваш труд и "болванку" не только при кратковременных, но и при длительных сбоях в электропитании, в том числе и при полном его отключении. Особенно, если сразу после отключения электроэнергии во время записи CD-R/RW выключить питание монитора (для более экономного расхода энергии аккумулятора ИБП). После отключения монитора работу программы можно контролировать по светодиодам, находящимся на передней панели привода CD-R/RW, и звуковым сигналам.

А все-таки, зачем же император Нерон поджег Рим? ;-))

И художник, и музыкант, и поэт...

С.РЮМИК,
г.Чернигов.

Снег холоден,
И всякий покой глубок,
И нет тихой елки в рождество...
ЭВМ, одно из первых стихотворений.

Людей, одинаково талантливых в живописи, музыке и поэзии, история знает не так уж и много. Человеку волей-неволей приходится в чем-то специализироваться, чтобы добиться успеха в жизни. А вот от компьютера требуют всего и сразу. Логика простая — если ЭВМ умеет мгновенно перемножать многомиллиардные цифры, то, смотришь, и картинку вместо человека нарисует, и мелодию напишет, и стихи сочинит. Но так не бывает.

Современные компьютеры пока не столь "самостоятельны" в действиях, как того хотелось бы писателям-фан-

тастам. Человек в настоящее время может использовать ЭВМ только в качестве пунктуального и добросовестного помощника, хотя иногда компьютерная программа может стать источником творческого вдохновения для людей искусства. За примерами обратимся к бесплатно распространяемым программам из Интернета.

Backgrounds By Chance (версия 1.0, freeware, http://www.ladlen.com/Download/BackgroundsByChance-1-0.exe, 1495 Кб) — помощник художника-дизайнера, автор — Дмитрий Долгов. Программа служит для генерации случайных фонов и текстур по заданному шаблону и для записи полученных изображений в формате BMP или JPEG. Пример фона, полученного из имеющихся в программе шаблонов, приведен на **рис.1**. Шаблоном может являться любой графический файл, созданный или введенный пользователем.

Особенность получаемых изображений в том, что они совместимы с противоположных сторон (справа и слева, сверху и снизу). Их можно использовать в качестве фона для Windows или для web-страниц, либо как текстуры для трехмерных объектов. Думается, что программа заинтересует специалистов текстильного и отделочного производства при разработке узоров тканей и рисунков на обоях. На сайте автора http://www.ladlen.com можно дополнительно скачать различные варианты генераторов фонов и фильтры для них, которые инсталлируются в виде плагинов.

Industrial Strength Color-Note Organ (версия 1.4, freeware, hthttp://hem.passagen.se/rasmuse/CoagulaLight14i.zip, 1078 Кб) — помощник звукорежиссера, автор — Rasmus Ekman. В данном случае речь идет не о банальном музыкальном редакторе, которых известно великое множество, а о генераторе производственного шума. Согласитесь, есть разница в шумовой атмосфере кузнечного цеха и швейного производства, на поточном конвейере и в центре испытаний ракетных двигателей. Все эти, а также многие другие шумовые эффекты можно получить путем кропотливой настройки многочисленных параметров. Изюминкой программы является визуализация шумовой картинки, в качестве которой на **рис.2** выбрана стандартная заставка экрана Windows (файл "Установка.bmp"). Любой введенный в качестве шаблона графический файл сначала оцифровывается, а затем превращается в звуковой сигнал с шумовыми ритмами. Например, заставка Windows "звучит" как камнедробильная машина. В программе имеется встроенный графический редактор, позволяющий видоизменять исходные шаблоны, добиваясь колоритности шумового "оркестра". Не случайно в названии программы имеется слово "орган", что позволяет сравнивать ее с оригинальным музыкальным инструментом.

Рис. 1

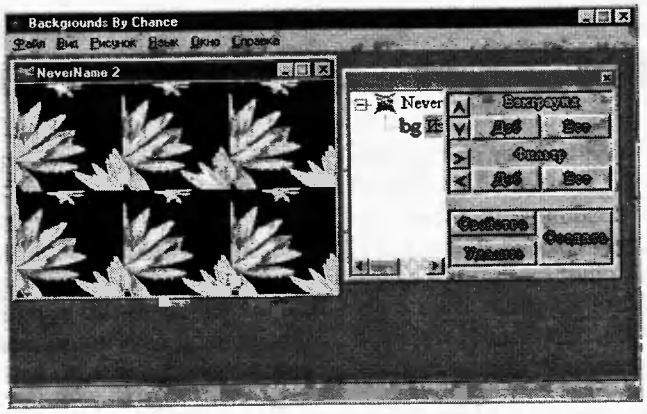


Рис. 2

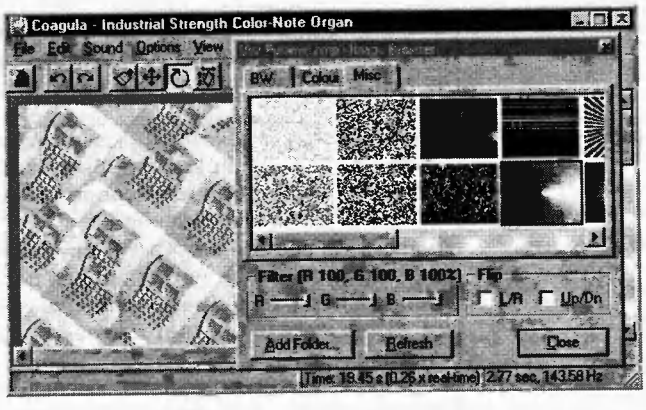
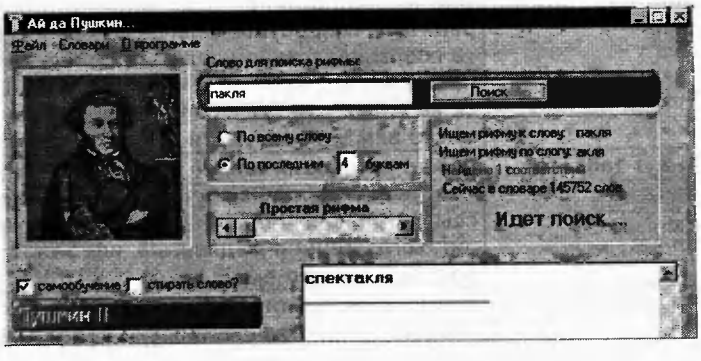


Рис. 3



Программа имеет два подключаемых словаря с общим словарным запасом в 145752 словоформ. Предусмотрена функция самообучения, когда добавляются новые слова и рифмы. Из необычных особенностей следует отметить возможность генерировать фразы в стиле Небуляр — смеси психоделики и абсурда.

"Ай да Пушкин" (версия 2.0, freeware, http://ftpsearch.rambler.ru/db/ftpsearch/go.html?href=ftp://ftp.ware.ru/win/pushkin2_2.zip, 853 Кб) — помощник поэта, автор — В.Кондратьев. Если вы думаете, что программа сочиняет стихи в стиле "а-ля Пушкин", то вы ошибаетесь. Машинная поэзия, к сожалению, еще не вышла за рамки экзотического эксперимента, и многими вообще не воспринимается как отдельное направление в искусстве. Программа "Ай да Пушкин" помогает находить рифмы к словам по строгому и нестрогому соответствию, да так мастерски, что любой поэт позавидует. Например, она без проблем нашла максимально созвучную рифму к небезызвестному слову "пакля" (**рис.3**). Для тех, кто успел распрощаться с детством, напомним, что рифму к слову "пакля" не смогли придумать ни поэт Цветик, ни коротышка Незнайка из замечательной книги Николая Носова "Приключения Незнайки и его друзей".

Г.БЕЛОНОГОВ.

Ошибки в Internet

При работе во всемирной сети Интернет достаточно часто на экран монитора вместо желаемой страницы выводятся какие-то "нехорошие" сообщения об ошибках. Может закрасться сомнение: "С моим компьютером что-то не так?". Чаще всего, причина ошибки связана с сервером, к которому вы пробуете обращаться, а не с вашим компьютером. Вообще говоря, ошибки в Интернете — это обычное явление, они происходят достаточно часто, и к ним надо относиться философски. В большинстве случаев, единственное, что вы можете сделать в этом случае — попытаться связаться с сервером еще раз, спустя некоторое время. Возможно, причина ошибки уже устранена. В таблице приводится список сообщений Интернет об ошибках и их расшифровка. Надеюсь, приводимая информация поможет более осмысленно путешествовать по просторам WWW.

Сообщение	Значение	Кто виноват, и что делать
400 Bad File Request	Неправильный запрос файла	Обычно означает, что синтаксис, используемый в URL, является некорректным. Например, использован символ верхнего регистра, а должен быть символ нижнего регистра; неправильные знаки препинания и т.д.
401 Unauthorized	Несанкционированный	Сервер ожидает некоторый ключ шифрования от клиента и не получает его. Также, возможно, был введен неверный пароль. Попробуйте снова, обращая особое внимание на регистр вводимого текста (верхний или нижний)
403 Forbidden/Access Denied	Доступ запрещен	Аналогично сообщению 401. В случае, если данный ресурс требует предварительной регистрации, для доступа к сайту необходимо специальное разрешение — пароль и/или username. Другими словами, вы не имеете соответствующих разрешений, установленных на сервере
404 File Not Found	Файл не найден	Сервер не может найти запрошенный вами файл. Данный файл был или перемещен, или удален; возможно, вы задали неправильный URL или имя документа. Проверьте URL, при необходимости исправьте, и попробуйте еще раз. Если это не сработает, то попробуйте последовательно удалять в адресном поле символы между группами слэшей, начиная с конца адреса, и связываться с сайтом, пока не появится страница 404 без ошибки. Может быть, с этого места вам удастся найти нужную страницу
408 Request Timeout	Тайм-аут запроса	Клиент остановил запрос прежде, чем сервер закончил его обрабатывать, т.е. нажал кнопку остановки, закрыл браузер или нажал на ссылку, раньше чем страница загрузилась. Обычно это происходит, когда сервер медленный, или запрашиваемый файл имеет большой размер
500 Internal Error	Внутренняя ошибка	Невозможно найти HTML-документ из-за проблем конфигурации сервера. Обычно рекомендуют связаться с администрацией сайта по E-mail
501 Not Implemented	Не реализовано	Web-сервер не поддерживает запрошенное свойство
502 Service Temporarily Overloaded	Сервис временно перегружен	Перегрузка сервера; слишком много соединений; высокая нагрузка на сервер. Продолжайте попытки, пока страница не загрузится
503 Service Unavailable	Сервис недоступен	Сервер занят; возможно, что сайт переехал, или потеряно соединение с Интернет (ваш провайдер отключил вас от сети)
File Contains No Data	Файл не содержит данных	Страница есть в наличии, но показывать нечего; ошибка в документе; неверное форматирование таблиц, или вырезана информация заголовка
Bad File Request	Неверный запрос файла	Возможно, браузер не поддерживает формы или другое кодирование информации
Failed DNS Lookup	Неудачный поиск DNS	Сервер доменных имен не может перевести ваш запрос домена в допустимый адрес Интернет. Сервер может быть занят или не работает, или был введен неправильный URL
Host Unavaliable	Хост недоступен	ЭВМ сервера не работавт. Нажмите перезагрузку страницы (Refresh) или попытайтесь зайти на сайт позже
Unable to Locate Host	Невозможно определить хост	ЭВМ сераера не работает, потеряно соединение с Интернет, или неверно введен URL

ЛИСТАЯ СТРАНИЦЫ

Скорость выполнения многих приложений ограничивается не тактовой частотой процессора ПК и не объемом оперативной памяти, а быстродействием винчестера. Для ускорения работы можно предложить использовать в этом случае более дорогой диск с интерфейсом SCSI. Однако теперь есть более дешевая альтернатива — **винчестеры ATA/100 с буфером объемом 8 Мб.** Именно о таком устройстве пишет *А.Кузнецов* в статье "WD1000JB — винчестер с большой "обоймой" **(ПЛ: компьютеры, №5/2002, С.42).**

Все винчестеры стандарта IDE со скоростью вращения шпинделя 7200 об/мин и некоторые модели со скоростью 5400 об/мин выпускались до последнего времени с буфером объемом 2 Мб. В прошлом году компания Western Digital сделала очередной шаг по пути усовершенствования продукции — линейку высокопроизводительных жестких дисков пополнило семейство специальных (special edition) моделей с увеличенным до 8 Мб объемом буфера. Фигурирующий в их названии индекс JB позволяет отличать special edition от винчестеров линейки BB, также имеющих скорость вращения шпинделя 7200 об/мин. Объем новых дисков составляет 100 и 120 Гб. Уже это свидетельствует об их позиционировании для решения серьезных задач, например, для обработки цифрового видео или работы в составе файл-сервера. Именно в подобных приложениях требуются большие объемы дискового пространства и высокое быстродействие. Компания Western Digital не стала применять в этих моделях новый интерфейс ATA/133. В ближайшем будущем производители жестких дисков ожидают появления принципиально нового стандарта Serial ATA, возможности которого позволят пренебречь разницей между ATA/100 и ATA/133.

В новых винчестерах реализована уже доказавшая свою надежность и эффективность фирменная технология Data Lifequard, предназначенная для защиты от потери данных.

Технические характеристики диска WD1000JB	
- объем диска	100 Гб;
- количество пластин	3;
- количество рабочих поверхностей	6;
- количество головок	6;
- скорость вращения шпинделя	7200 об/мин;
- объем буфера	8 Мб;
- среднее время поиска (чтение/запись)	8,9/10,9 мс;
- скорость считывания с диска в буфер	525 Мбит/с;
- интерфейс	ATA/100;
- защита от ударного воздействия в рабочем состоянии (чтение/запись)	65/20 g;
- защита от ударного воздействия в нерабочем состоянии	200 g;
- розничная цена	200 USD.

Проверка эксплуатационных качеств нового жесткого диска проводилась при помощи двух специализированных программ: Winbench 99 ver.1.2 компании Ziff-Davis и PCMark2002, недавно выпущенной компанией Mad-Onion.

Компьютер, на котором проводились испытания нового винчестера, был собран на базе материнской платы ASUS TUSL2-C с процессором Intel Pentium III 1 ГГц. Оперативная память — PC133 PQL объемом 256 Мб, видеокарта — ASUS V7100 Pro. В качестве системного был использован жесткий диск IBM IC35L объемом 40 Гб, работавший в режиме UDMA/5. Тестирование проходило под управлением ОС Windows 98 SE с установленным DirectX 8.1. Испытывавшийся жесткий диск был установлен на второй IDE-канал в режиме Master. Приведенная ниже оценка PCMarc2002 HD Test рассчитана как среднее арифметическое результатов, полученных в результате пяти последовательно проведенных измерений:

- PCMarc2002 HD Test — 1124;
- WinBench 99 Business-Disk Benchmark, Мб/с — 11,8,
- WinBench 99 High-End Disk Benchmark, Мб/с — 27,2;
- WinBench 99 Disk Access Time, мс — 13,6.

По результатам тестирования автор отмечает, что технические нововведения носят скорее эволюционный, нежели революционный характер — скачкообразного роста важных эксплуатационных параметров не наблюдалось. Однако, благодаря увеличенному объему буфера, жесткие диски с индексом JB вполне могут оказаться востребованы при решении ряда задач, требующих максимальной производительности дисковой подсистемы компьютера. Интерфейс IDE по-прежнему остается наиболее предпочтительным для пользователя, ищущего компромисс между высокой стоимостью и производительностью, ведь цены на винчестеры с интерфейсом SCSI пока остаются сравнительно большими. Не исключено, что в будущем и другие производители жестких дисков последуют примеру Western Digital, снабдив свои новые модели с интерфейсом IDE буфером увеличенного объема.

О самых больших на сегодняшний день ЖК-мониторах рассказывает *М.Бабенков* в статье " Новейшие мультимедийные ТВ/мониторы SyncMaster 211MP и SyncMaster 241MP компании Samsung Electronics" **(КомпьютврПресс, №5/2002, С.112...114).**

Компания Samsung Electronics, один из признанных лидеров в области разработки и производства ЭЛТ- и ЖК-мониторов, развивая свой успех в области ЖК-технологии, выпускает абсолютно новые плоские модели мониторов SyncMaster 211MP и 241MP с размером диагонали 21 и 24 дюйма соответственно. Каждая из моделей реализована по принципу "четыре в одном", то есть это готовое устройство, которое выполняет функции монитора компьютера и телевизора (обе модели снабжены тюнером), а также служит монитором для проигрывателя цифрового видеодиска (DVD-плейер), кассетного видеомагнитофона (VHS), видеокамеры или любого телевизионного устройства. Их основные параметры приведены в **табл.1**.

Табл. 1

Параметры	SyncMaster 211MP	SyncMaster 241MP
Экран	TFT цветной 21,3" (54,1 см)	TFT цветной 24,06" (61,1 см)
Размер изображения, мм	432 x 324	518,4 x 324
Формат изображения	4:3	8:5
Максимальное разрешение	1600 x 1200	1920 x 1200
Максимальное количество цветов, млн.	16,7	16,7
Размер пиксела, мм	0,27 x 0,27	0,27 x 0,27
Угол обзора по горизонтали/по вертикали	170/170	170/170
Контрастность	500:1	500:1
Яркость, кд/м^2	250	270
Время реакции пиксела, мс	25	25
Энергопотребление, Вт	90	83
Габариты с подставкой, мм	568 x 472 x 213	618 x 472 x 213
Вес монитора, кг	11,3	13,8

Samsung использует запатентованную технологию, включающую самые последние достижения в жидкокристаллической технологии, которые улучшают четыре главных критерия оценки монитора: угол обзора, яркость, контрастность и время реакции пиксела. Так, например, угол обзора по вертикали и по горизонтали в данных моделях равен 170°. Обе модвли имеют сверхвысокий коэффициент контрастности 500:1 и исключительно малое время реакции пиксела, которое составляет менее 25 мс.

Samsung SyncMaster 211MP предлагает максимальное разрешение 1600x1200, а Samsung SyncMaster 241MP допускает максимальное разрешение 1920x1200. Однако, несмотря на то что монитор SyncMaster 241MP совместим с свмыми популярными видеоадаптерами, он работает в режиме WUXGA (1920x1200), к которому следует относиться очень осторожно. Поскольку для режима WUXGA промышленный стандарт отсутствует, изготовители видеоадаптеров используют различные конфигурации, которые могут приводить к интерпретации ложных видеорежимов монитора.

Дисплеи предлагают большие возможности по выбору форматов изображения. Например, широкоформатный полнометражный фильм посредством настроек монитора легко масштабируется. С помощью функций мониторов легко добиться увеличения любой части экранного изображения до 17х.

Обе модели снабжены устройством дистанционного управления, мягкими сенсорными кнопками управления, обладают объемным трехмерным звуком, который обеспечивается внешними динамиками, удаленными от самого монитора либо установленными непосредственно на нем. И, конечно же, мониторы обладают функцией Plug and Play. Кроме того, каждый монитор имеет функцию автотестирования, которая позволяет проверить правильность его работы.

На передней панели монитора находится инфракрасный приемник сигнала от пульта дистанционного управления и скрытые небольшой крышкой 11 сенсорных кнопок управления, одна из которых — кнопка включения/выключения питания монитора, а остальные служат для реализации функций ТВ/монитора и для их настройки.

Одна из главных функций — Source — служит для выбора источника видеосигнала. Индикация, появляющаяся в левом верхнем углу монитора, указывает, какой из источников в данный момент активирован. Источники видеосигнала сменяются в такой последовательности: TV/Video/S-Video/DVD/DTV/PC.

Функция PIP активизирует окно PIP (Picture-In-Picture, картинка в картинке). Данная функция служит для одновременного просмотра изображения, создаваемого в ПК, и изображения, формируемого от источника видеосигнала. При этом одно из изображений (от ПК или от источника видеосигнала) будет развертываться в небольшом окне, наложенном на другое изображение. В памяти монитора фиксируется предыдущее состояние источника видеосигнала и ПК. Таким образом, если в предыдущем состоянии видеосигнал ПК поступал с разъема S-Video, и в текущий момент во весь экран развернуто изображение, поступающее от ПК, то в наложенном окне будет развернут видеосигнал, поступающий с разъема S-Video.

Очень интересна функция PBP, с помощью которой реализуется режим Picture-By-Picture, то есть режим последовательной развертки. При активизации клавиши PBP экран монитора делится пополам по вертикали, и в каждой половине по желанию пользователя происходит выбор источника видеосигнала. При повторном нажатии на эту же клавишу изображения меняются местами.

Все остальные клавиши служат для поиска и изменения параметров мониторов в меню. Некоторые из клавиш являются быстрыми и служат в основном для наиболее часто используемых функций ТВ/мониторов, таких как быстрое увеличение/уменьшение громкости, переключение каналов и т.д.

На задней панели каждого из мониторов расположены: один компьютерный вход (15-пиновый D-Sub), один аудиовход (mini jack), один вход компонентного видео и один раздельный аудиостереовход (левый и правый каналы), два интерфейса для подключения внешних источников видеосигнала, таких как DVD-плейер (Y, Pb, Pr, Audio(L), Audio(R)), один S-Video, один аудиостереовыход, а также аудиовыход для подключения наушников. Такой богатый набор входных интерфейсов исключает возможность возникновения каких-либо проблем с подключением к ТВ/монитору любых внешних источников видеосигнала.

Обе модели снабжены системой энергосбережения, которая называется PowerSaver. Эта система позволяет сокращать расход электроэнергии, переключая монитор в режим низкого потребления электроэнергии, в случае если он не используется в течение определенного промежутка времени. Имеются следующие режимы работы монитора: "Включен" (On), "Ожидание" (Standby), "Сон" (Sleep) и "Глубокий сон" (Deep Sleep). Данная система управления расходом электроэнергии работает с VESA DPMS-совместимой видеокартой, установленной в компьютер. Для установки данной функции используется соответствующая программа-утилита, имеющаяся в компьютере. Оба дисплея также соответствуют требованиям строгих стандартов MPR-II и TCO-95.

Питание мониторов осуществляется от сети 220 В частотой 50 или 60 Гц, при этом потребляемая мощность составляет не более 90 Вт для Samsung SyncMaster 211MP и не более 83 Вт для Samsung SyncMaster 241MP.

В случае использования данных ТВ/мониторов с компьютером, снабженным функциями VESA DPMS, аппараты функционируют в соответствии с требованиями EPA Energy Star и NUTEK.

Корпуса мониторов имеют очень удобную систему установки и регулировки положения ЖК-матрицы. Мультимедийность мониторов подчеркивают два довольно мощных (5 Вт каждый) стереодинамика. Оригинально и крепление динамиков: один из вариантов предусматривает непосредственное крепление на мониторе, а второй предоставляет пользователю возможность закрепить каждый динамик на металлической подставке, выполненной в том же стиле, что и подставка на самом мониторе, и расположить их по своему усмотрению.

При тестировании с помощью люксметра/яркомера ТКА-04/3 определялась яркость мониторов и равномерность цветового заполнения. При измерении яркости и равномерности свечения использовалась методика ANSI.

Первоначальная настройка геометрических параметров и фокусировки производилась по стандартной методике, используемой в редакции журнала "КомпьютерПресс". Затем проводилась проверка на наличие дефектных пикселов и инерционности изображения с помощью тестового пакета Meko Monitors Matter CheckScreen. Кроме того, был запущен ряд офисных и мультимедийных приложений. На этом этапе новые мониторы получили самые высокие оценки и заслужили самые лестные отзывы.

После этого производились измерения яркости на белом цветовом поле. Измерения делались в пяти точках — в четырех угловых и в середине экрана, причем каждый из замеров производился три раза, после чего результаты усреднялись. Затем рассчитывались средняя яркость и среднее отклонение, определяемое как максимальное отклонение от среднего значения в процентах.

Рассчитывался также и запас по яркости для каждого монитора. Для этого использовались максимальная яркость свечения на белом фоне и разность между максимальной яркостью и яркостью, соответствующей настройке по шкале ANSI. Хотя на новом мониторе при максимальной яркости никто не работает, однако по мере эксплуатации яркость приходится постепенно увеличивать, и в этом смысле разница между максимальной яркостью и яркостью, соответствующей настройке по шкале ANSI, характеризует запас по яркости, гарантирующий использование ТВ/мониторов в течение долгого времени. Результаты тестирования приведены в **табл.2**.

В заключение автор отмечает, что мультимедийные ТВ/мониторы компании Samsung, обладающие высокой функциональностью и отличными техническими характеристиками, могут стать незаменимыми помощниками для профессионального использования в области дизайна и верстки, поскольку имеют оптимальную площадь рабочего стола. Дисплеи можно порекомендовать и для общего нелинейного телевизионного монтажа, ведь они оснащены богатым набором интерфейсов. Также они станут настоящим открытием для любителей домашних кинотеатров, а созданные на их основе системы обеспечат возможность комфортного просмотра фильмов на огромном экране. А представьте, какое удовольствие вы получите, поиграв на таком мониторе в какой-нибудь автомобильный симулятор или 3D-шутер.

Табл. 2

Параметры	SyncMaster 211MP	SyncMaster 241MP
Средняя яркость, кд/м²	169,2	173,8
Максимальная яркость в центре экрана, кд/м²	207	245
Максимальная неравномерность по яркости, %	17,6	25,8
Запас по яркости, кд/м²	37,8	71,2

Теперь каждый меломан может легко создавать и, главное, носить с собой целую фонотеку. Для этого ему понадобится устройство, о котором рассказывает С.*Маленкович* в статье "Archos Jukebox Recorder 20: фонотека мвломана" **(ПЛ: компьютеры, №5/2002, С.34...35).**

Archos Jukebox — это аудиоплеер, имеющий размеры 115x83x34 мм и вес 350 г. На первый взгляд, это много. Но нужно учесть, что в нем установлен жесткий диск емкостью 20 Гб! С учетом того, что файлы на нем хранятся в формате MP3, легко подсчитать, что фонотека, постоянно доступная для прослушивания, будет насчитывать около 330 часов непрерывного звучания. А это примерно две недели самой лучшей музыки, отобранной владельцем Jukebox, отмечает автор.

На лицевой панели плеера расположены управляющие кнопки и жидкокристаллический экран. На торцах какие-либо элементы управления отсутствуют, есть только четыре разъема: mini-jack (для наушников), линейный вход, цифровой вход/выход и разъем для сетевого питания.

Устройство работает от четырех сменных аккумуляторов Ni-MH типоразмера AA. Для полной зарядки элементов питания требуется 6 часов, но если плеер в процессе зарядки эксплуатируется — 15 часов.

В комплект входят: зарядное устройство, сумочка для ношения на поясе, складные наушники с затылочным оголовьем, короткий кабель USB (на обоих концах одинаковые разъемы USB type A), переходник для подключения к внешней аудиотехнике, компакт-диск с драйверами и руководство по эксплуатации на английском языке.

Восьмистрочный экран отображает вполне приличный менеджер файлов, поэтому на дисплее одновременно видны имя текущего каталога, пять строк со списком содержимого, строка подсказки и строка статуса. В последней отображается уровень заряда батарей, громкость, номер песни и режим воспроизведения.

Для управления плеером, кроме пяти кнопок навигации (четыре направления плюс выбор), используются три клавиши, расположенные под экраном. Чтобы узнать, какие функции выполняет каждая из них, необходимо посмотреть на строку подсказки. Принцип контекстных кнопок заимствован у производителей сотовых телефонов и в данном случае вполне уместен.

Менеджер файлов позволяет запустить воспроизведение музыки из определенного файла и каталога, а также выполнить простейшие операции над данными (переименование и удаление файлов и каталогов). Одна из контекстных клавиш в качестве подсказки имеет индикацию текущего времени (нажав на нее, можно скорректировать часы).

В режиме воспроизведения на экране видны все те же строчки подсказки и статуса, а также имя исполнителя, название песни и альбома. Длинные названия непрерывно прокручиваются вперед-назад. Оставшиеся три строчки заняты индикацией времени воспроизведения, а также уровнем сигнала в левом и правом каналах.

Подсоединенный к компьютеру плеер определяется системой как "съемный жесткий диск". Производительность винчестера заслуживает всяческих похвал. Сравнительно высокая скорость передачи данных достигается за счет того, что устройство общается с компьютером по интерфейсу USB 2.0. Так, например, тестовые файлы общим объемом 150 Мб загрузились в плеер за 2 мин 20 с. т.е. реальная скорость передачи данных по интерфейсу USB 2.0 у Archos Jukebox составляет порядка 1 Мб/с (теоретическая максимальная пропускная способность шины USB 1.1 — около 1,5 Мб/с).

Режимов воспроизведения — шесть: нормальный, повтор первого трека, повтор всех треков, случайный порядок треков, очередь, режим знакомства. В режиме знакомства вы сможете услышать первые 30 с из каждой песни. Регуляторов громкости — два (Volume и Loudness), имеются простой темброблок (bass, treble) и усиление басов (bass boost).

При всем разнообразии полезных настроек и режимов воспроизведения, насладиться качеством звучания вам вряд ли удастся, если использовать наушники, поставляемые в комплекте. Несмотря на их компактность, наличие регулятора громкости и привлекательный дизайн, качество звучания оставляет желать лучшего. Даже простые "затычки" за 300 руб. способны воспроизводить более широкий диапазон частот с требуемым уровнем громкости. Когда плеер был подключен к музыкальному центру через аналоговый выход, оказалось, что предусилитель недостаточно мощный, и на высоком уровне громкости при включенном усилении басов происходит перегрузка. Поэтому плеер следует подключать к музыкальному центру через цифровой выход.

Очень приятно, что Archos Jukebox можно использовать без компьютера. Функция записи позволяет оцифровывать музыку от внешнего источника (например, музыкального центра) или через встроенный микрофон с заданным качеством. По окончании сессии вы сможете ввести имя автора и название композиции, поэтому пополнять коллекцию очень легко. Но здесь есть одна недоработка — для создания программ воспроизведения необходим компьютер и программа MusicMatch Jukebox (поставляется в комплекте). С плеера доступно лишь минимальное редактирование.

Подводя итоги, автор отмечает, что у Archos получилось отличное устройство, которое по характеристикам может претендовать на почетное звание "Мечта меломана". Собрать очень большую коллекцию музыки, хранить ее в компактном устройстве и пополнять при помощи любого внешнего источника без помощи компьютера теперь может каждый. Не хватает только русского алфавита и хороших наушников.

Как правильно купить ноутбук? Этот вопрос обсуждает *С.Самарин* в статье "Тонкий выбор" **(ПЛ: компьютеры, №5/2002, С.70...76).**

Сначала несколько советов. Если внешний вид и дизайн приобретаемого компьютера определяются вкусом пользователя, то технические характеристики стоит обсудить. Прежде всего, *процессор*. В ноутбуки ставятся мобильные, обеспечивающие энергосбережение процессоры ф.Intel. Процессоры AMD встречаются редко. Но в ноутбуках могут использоваться и процессоры для настольных систем. Это следует учитывать.

Объем ОЗУ. Здесь экономить не следу-

Табл. 1

Модель	S1300	3090	Evo N400c	LifeBook S-5582	OmniBook500	ThinkPad T23	M722
Частота процессора, Мгц	933	900	700	800	750	866	900
Диагональ экрана, дюймы	13,3	13,3	12,1	13,3	12,1	13,3	12,1
Видеоадаптер	Intel 82830M	SIS 630/730	ATI Rage Mobility-P AGP	ATI Rage Mobility PCI	ATI Rage Mobility-M AGP	S3 Graphics SuperSavage/IXC SDR	ATI Rage Mobility PCI
Видеопамять, Мб	8	8	8	8	8	16	4
ОЗУ, Мб	128	128	128	128	256	128	128
Аудиосистема	Intel AC'97 — Audio Controller	SiS 7018 Audio Driver	ESS Allegro PCI Audio	Yamaha AC-XG Audio	ESS Maestro PCI Audio	Crystal WDM Audio	Crystal WDM Audio
Емкость винчестера, Гб	10	10	20	20	30	14	10
CD(DVD)-ROM	ASUS SCD-2400	MATCHITA CD-ROM CR-177	Опционально	Hitachi DVD-ROM GD-S250	QSI DVD-ROM SDR-081	LG CD-ROM CNR-8245B	TEAC CD-224E
Встровнный факс-модем	Generic SoftK56 Data Fax	HSP56 World MicroModem	Lucent Win Modem	Lucent Technologies Soft Modem AMR	3Com 56K V.90 Mini PCI Modem	Опционально	HSP56 MR
Сетевая плата	Realtek RTL8139 PCI Fast Ethernet NIC	Davicom 9102/A PCI Fast Ethernet Adapter	Intel PRO/100 P Mobile Combo Adapter	Realtek RTL8139 PCI Fast Ethernet NIC	3Com 10/100 Mini PCI Ethernet Adapter	Intel PRO/100 VE Network Connection	Realtek RTL8139 PCI Fast Ethernet NIC
ИК-порт	Есть	Нет	Есть	Есть	Есть	Есть	Есть
Порт IEEE1394	Есть	Нет	Нет	Есть	Нет	Нет	Есть
Емкость батареи, А·ч	2,9	3,1	1,96	3,4	3,1	3,6	3,6
Габариты, мм	296 x 240 x 21	295 x 239 x 28	266 x 242 x 22	293 x 236 x 33	277 x 225 x 24	305 x 254 x 36	297 x 239 x 39,5
Вес с батареями, кг	1,8	2,0	1,58	1,71	1,5	2,5	2,9
Предустановленная ОС	Windows 98 SE	Windows 98 SE	Windows 98 SE	Windows ME	Windows 98 SE	Windows 98 SE	Windows 98 SE
Стоимость, USD	1565	1345	1380	2540	1665	1650	1645

Табл. 2

Модель	S1300	3090	Evo N400c	LifeBook S-5582	OmniBook500	ThinkPad T23	M722
Общий дизайн	5	2	5	4	5	4	2
Эргономика	4	3	5	4	4	4	2
Удобство работы с клавиатурой	5	4	4	4	5	5	4
Коммуникационные возможности	5	4	5	5	4	4	5
Комплект поставки	5	4	3	5	5	4	3
Дополнительные свойства	4	3	5	4	5	4	3
Температурный режим	5	3	5	4	5	3	3
Итоговая оценка	5	3	5	4	5	4	3

ет, особенно если вы собираетесь использовать новейшие ОС и прикладные пакеты. При недостатке памяти некоторые программы не запустятся, а ОС создаст на винчестере огромных размеров файл подкачки. Постоянная работа двигатвля жесткого диска приведет к значительному расходу энергии батарей.

К *видеоподсистеме* не предъявляется особых требований, т.к. портативные компьютеры используются как правило для решения офисных задач.

Жесткий диск. Потребление электроэнергии не зависит от его объема, поэтому лучше выбирать диск побольше.

Коммуникационные средства (сетевая карта и инфракрасный порт). Их наличие необходимо. С ними вы всегда сможете подключиться к корпоративной сети и сбросить на диск требуемую информацию.

Модули расширения. При выборе ноутбука следует поинтересоваться, есть ли у него стыковочный разъем, и сколько есть слотов PCMCIA. Это повышает гибкость и универсальность компьютера.

Исходя из сказанного, обсуждаются результаты тестирования ноутбуков ASUS S1300, BLISS 3090, Compaq Evo N400c, Fujitsu-Siemens LifeBook S-5582, Hewlett-Packard OmniBook500, IBM ThinkPad T23 и Mitac M722. В них используются процессоры Intel Pentium III и ЖК-матрицы с разрешением 1024x768. Остальные параметры приведены в **табл.1**, оценки редакции журнала "ПЛ: компьютеры" — в **табл.2**.

Подводя итоги, автор отмечает, что больше всего понравился портативный компьютер ASUS S1300, который получил значок "Выбор редакции". Более того, именно этот ноутбук набрал самое большое количество баллов по тесту ZD Content Creation Winstone 2002, оставив позади всех конкурентов. Нельзя не отметить и правильный выбор графической подсистемы, позволяющей пользователю работать не только с офисными, но и с графическими пакетами, максимально нагружающими компьютер. ASUS S1300 также позволит своему владельцу запускать современные 3D-игры.

Приз в виде значка "Лучшая покупка" был отдан ноутбуку Compaq Evo N400c как самой стильной машине теста. Элегантность и стиль ноутбука, вкупе с продуманной системной конфигурацией, оставили самые положительные впечатления. И пусть в базовой конфигурации аппарат лишен некоторых периферийных средств, таких как оптический привод и порт-репликатор, однако сам подход компании Compaq к проектированию портативного компьютера, несомненно, заслуживает самых добрых слов. В наш век коммуникаций и мобильности ноутбук Evo N400c послужит настоящим проводником своему хозяину.

Приятные впечатления оставил Fujitsu-Siemens LifeBook. Модели от IBM и Hewlett-Packard не отличаются новизной дизайна и являются продолжателями популярных серий. Корпоративный стандарт, узнаваемость марки и консервативность аппаратов — вот те три кита, на которых держится их популярность среди людей бизнеса.

Портативные компьютеры BLISS 3090 и Mitac M722 также не блещут дизайнерскими изысками. Мощь системной конфигурации, высокая тактовая частота процессора и исчерпывающие коммуникационные возможности — вот основные достоинства этих сравнительно недорогих моделей. Именно по этой причине такие ноутбуки пользуются спросом.

Известная всем фирма Microsoft ухитряется едва ли не каждый год выпускать **новые ОС. Но стоит ли их использовать?** Ответ на этот вопрос, по крайней мере, относительно Windows XP, дает Стив Басс в статье "Windows XP — ваше будущее?" **(Мир ПК, №5/2002, С.110...111).**

Установка этой ОС может потребовать достаточно много времени, денег и отдельных аппаратных средств. Хотя многие из новых особенностей XP и сделали компьютер автора более простым в применении, тем не менее, докучливый и притом драконовский процесс активации этой ОС не позволяет ему рекомендовать ее всем. Автор перешел к XP после модернизации системы, работавшей в среде Windows 98 SE и объединенной в сеть с еще одним компьютером посредством DSL-маршрутизатора BroadGuard компании SOHOware. И вот первая неприятность — пропал доступ и к Интернету, и к локальной сети. Адаптер USB-to-Ethernet ф.Belkin, благополучно уживавшийся с Windows 98, отверг ухаживания XP.

Автору пришлось купить новую сетевую плату, поскольку XP обеспечивала лишь ограниченную поддержку сетевого протокола, используемого вторым в сети ПК. На восстановление сети ушел целый день, наполненный дурацкой суетой со всякими деталями, перезагрузками системы и консультациями с Microsoft.

Для звуковой платы Turtle Beach также понадобился новый драйвер. А еще автору пришлось переинсталлировать Outlook 2000 и обновить версии трех утилит.

Есть две очень важные вещи, которые автору в XP действительно понравились — стабильность и интерфейс. Windows XP позволяла одновременно просматривать на ПК, оснащенном 648 МГц-процессором AMD и ОЗУ 128 Мб, два здоровенных видеофайла. И это одновременно с 19 другими программами. Такая замечательная стабильность — одна из тех многих черт, которые XP унаследовала от Windows 2000.

Дело здесь вовсе не в том, что приложения никогда не сбиваются, подчеркивает автор. Это случается. Просто XP способна помешать подобного рода событиям обрушить систему.

Интерфайс XP обеспечивает фантастическую экономию времени. Логическая организация Главного меню, например, делает перемещение между приложениями совсем простым делом. Подстроить интерфейс под себя также оказалось легче легкого. Перенос большинства значков с Рабочего стола в Главное меню обеспечит быстрый доступ к программам.

Система Windows XP находит все больше приверженцев, и, возможно, вы уже подумываете о ее использовании. Автор был бы рад определенно высказаться "за" или "против", но в XP есть одна черта, вызывающая у него беспокойство. Это активизация продукта — процедура, призванная отравить жизнь программным пиратам путем ограничения числа возможных инсталляций XP.

Если вы не свяжетесь с компанией Microsoft по телефону или через Интернет в течение 30 дней после установки XP, она перестанет работать. Когда вы активизируете XP, она берет "отпечаток пальца" вашей системы, и если когда-нибудь потом ОС решит, что вы сменили слишком много системных компонентов, вам придется регистрироваться заново. Что же дальше — обязательная ежегодная плата?

Итак, если вы единственный пользователь своего ПК, аварийные отказы у вас случаются редко, и вы пользуетесь всего лишь несколькими программами, то лучше сэкономьте 100 USD и проигнорируйте XP. Однако если вы часто боретесь с неполадками склонного к аварийным отказам ПК, работающего в среде Windows 9x, то XP должна повысить его производительность. Эта ОС также позволяет нескольким пользователям с легкостью делить один ПК, настраивая под себя Рабочие столы и фиксируя свои предпочтения в пользовательских профилях.

М.ДОЛИНСКИЙ,
г.Гомель.

Решение задач на графах

(Продолжение. Начало в №№7-9/2002)

4. Сильносвязанные компоненты и доминантные множества

Подграф графа называется его сильносвязанной компонентой, если каждая из вершин подграфа достижима от любой из вершин подграфа. Доминантное множество — это минимальное множество вершин графа, из которого достижимы все вершины графа. Алгоритмы построения сильносвязанных компонент и доминантных множеств графа рассмотрим на примерах.

4.1. Задача "Карточки"

(Республиканская олимпиада по информатике, 1994 г.)

Есть N карточек. На каждой из них черными чернилами написан ее уникальный номер — число от 1 до N. Также на каждой карточке красными чернилами написано еще одно целое число, лежащее в промежутке от 1 до N (некоторыми одинаковыми "красными" числами могут помечаться несколько карточек).

Например, N=5 и 5 карточек помечены следующим образом:

"Черное" число	1	2	3	4	5
"Красное" число	3	3	2	4	2

Необходимо выбрать из данных N карточек максимальное число карточек таким образом, чтобы множества "красных" и "черных" чисел на них совпадали.

Для примера, приведенного выше, это будут карточки с "черными" номерами 2, 3, 4 (множество красных номеров, как и требуется в задаче, то же — {2,3,4}):

ВВОД: <N≥ N, N≤50

<"Черный" номер 1, "красный" -> "красное"_число_1

.....

<"Черный" номер N, "красный" -> "красное"_число_N

ВЫВОД:

<В выбранном множестве элементов количество элементов> S

<"Черные" номера выбранных карточек> a1, ..., aS

Пример ввода	Пример вывода
6	6
1 2	1
2 3	2
3 4	3
4 5	4
5 1	5
6 6	6

Фактически в задаче требуется найти все сильносвязанные компоненты представляемого карточками ориентированного графа и добавить к ним карточки, на которых и красным, и черным цветом написаны одни и те же числа.

Опишем решение задачи более подробно, опираясь на исходный текст соответствующей программы.

Тело главной части программы будет выглядеть так:

```
readln(N);                              {Читаем количество карточек}
M:=0;                                   {Количество карточек в результате}
for i:=1 to N do                        {Для всех карточек}
  begin
    readln(v,u);                        {Читаем карточку}
    inc(kG[u]); G[u,kG[u]]:=v;          {Пополняем по ней граф}
    if u=v                              {Если красное число равно черному}
      then
                                        {Заносим карточку в результат}
        begin inc(M); T[M]:=u;end;
  end;
BuildDominators;                        {Строим множество доминаторов}
Reverse;                                {Обращаем граф}
BuildDominators2;                       {Строим множество доминаторов}
                                        {обращенного графа, попутно получая}
                                        {все сильносвязанные компоненты графа}
```

Рассмотрим процедуру BuildDominators:

```
Procedure BuildDominators;
  begin
    Time:=0;                            {Время обработки вершин = 0}
    for i:=1 to N do                    {Для каждой вершины i помечаем}
        begin
          Color[i] := WHITE;            {Вершина свободна}
          Dom[i]   := WHITE;            {Доминаторов нет}
          CurC[i]  := WHITE;            {Свободна на текущем шаге}
          F[i]     := 0;                {Время завершения обработки - 0}
        end;
     for v:=1 to N do                   {Для всех вершин}
       if color[v]=WHITE                {Если вершина v свободна}
         then
          begin
            DFS(v);                     {Поиск в глубину от вершины v}
            AddDominator(v);            {Добавить найденного доминатора}
          end;
  end;
```

Таким образом, в ходе обработки вершин графа на этом этапе у нас есть три множества:

- Dom — найденные на текущий момент доминаторы графа;
- Color — все обработанные вершины;
- CurC — вершины, обработанные во время поиска в глубину от текущей базовой вершины V.

Для данной задачи рекурсивная процедура поиска в глубину выглядит таким образом:

```
Procedure DFS(i:byte);
  var
    j : byte;
  begin
    color[i]:=GRAY;                     {Вершина i обработана вообще}
    curC[i]:=GRAY;                      {Вершина i обработана на текущем шаге}
    inc(Time);                          {Увеличить время обработки вершин}
    for j:=1 to kG[i] do                {Пока у вершины есть наследники}
      if curC[G[i,j]]=WHITE             {Если наследник не обрабатывался}
                                        {на текущем шаге}
      then DFS(G[i,j]); {Запустить поиск в глубину от наследника}
    inc(Time);                          {Увеличить время обработки вершин}
    if F[i]=0                           {Если вершине i не устанавливалось время}
       then F[i]:=Time;  {завершения обработки - то установить}
  end;
```

Процедура AddDominator(v), также вызываемая из процедуры BuildDominators, добавляет в список доминаторов вершину v следующим образом — все вершины, достигнутые из текущей (Curc[i]<>WHITE), вычеркиваем из доминаторов; текущую добавляем в список доминаторов (dom[Domi]:=GRAY;).

```
Procedure AddDominator (Domi:longint);
  begin
    for i:=1 to N do                    {Для всех вершин}
      if (CurC[i]<>WHITE)               {Если она достигнута от текущей}
         then dom[i]:=WHITE;            {Вычеркиваем ее из доминаторов}
    dom[Domi]:=GRAY;                    {Вносим текущую в список доминаторов}
    for i:=1 to N do                    {Для всех вершин устанавливем отметку}
    curC[i]:=WHITE;                     {"не достигнута от текущей"}
  end;
```

Следующий шаг после построения доминаторов исходного графа — транспонирование графа (другими словами, смена стрелок на дугах на противоположные). Это делается с помощью процедуры Reverse. Процедура Reverse по исходному графу G[1..N,1..M], kg[1..N] строит граф B[1..N,1..M], kB[1..N]:


```
Procedure Reverse;
  Begin
                        {Обнуляем количества дуг из вершины i}
    for i:=1 to N do KB[i]:=0;
      for i:=1 to N do
        for j:=1 to kG[i] do
          begin
                {Инкрементируем количество дуг из вершины i}
            inc(kB[G[i,j]]);
                              {Добавляем в граф обратную дугу}
            b[G[i,j],kB[G[i,j]]]:=i;
          end;
  end;
```

Процедура BuildDominator2 строит множество доминаторов на транспонированном графе, параллельно формируя сильносвязанные компоненты (ССК) исходного графа:

```
Procedure BuildDominators2;
  begin
    Time:=0;
    for i:=1 to N do                              {Для всех вершин}
      begin
        Color[i]:=WHITE;                          {Свободна вообще}
        CurC[i] :=WHITE;                {Свободна в текущем цикле}
        Kart[i]:=WHITE;                           {Не входит в ССК}
      end;
    SortF;                  {Сортируем в порядке убывания времени}
                                {завершения обработки вершин при}
                              {построении множества доминаторов}
                            {исходного графа - результат Q[1..N]}
    for v:=1 to N do        {В порядке убывания времен обработки}
      if color[Q[v]]=WHITE                  {Если вершина свободна}
         then
           begin
             DFS2(Q[v]);                     {Построить доминатор}
             AddDominator2(Q[v]);             {Добавить доминатор}
           end;

    for i:=1 to N do                              {Для всех вершин}
       if (Kart[i]<>WHITE)                       {Если входит в ССК}
         then  begin
                 inc(M);                        {вносим в результат}
                 T[M]:=i;
               end;
    SortT;                  {Сортируем результирующее множество,}
    writeln(M); for i:=1 to M do writeln (T[i]);  {выводим его}
  end;
```

Процедура DFS2 поиска в глубину по транспонированному графу выглядит следующим образом:

```
Procedure DFS2(i:byte);
   Var  j : byte;
   begin
       color[i]:=GRAY; curc[i]:=GRAY;
       for j:=1 to kB[i] do
          if color[B[i,j]]=WHITE then DFS2(B[i,j]);
   end;
```

Процедура AddDominator2, обеспечивающая корректное построение доминаторов транспонированного графа и сильносвязанных компонентов исходного (и транспонированного) графов, такова:

```
Procedure AddDominator2 (Domi:longint);
   var
     MinT, MinK, KS : longint;
     FlDom  : boolean;
  begin
    KS:=0;                      {Количество вершин в текущей ССК}
    for i:=1 to N do                              {Для всех вершин}
    if Curc[i]<>WHITE                  {Если вершина достигнута,}
       then inc(KS);                      {инкрементировать к-во}
                                           {вершин в текущей ССК}
    if  ks>1                           { Если в текущей ССК вершин}
      then                                        {больше чем одна,}
        for i:=1 to N do               {то внести их всех в KART}
          if CurC[i]<>WHITE then Kart[i]:=GRAY;
    for i:=1 to N do curC[i]:=WHITE;      {Сделать все вершины}
                                       {свободными для нового шага}
  end;
```

Полный текст программы, решающей поставленную задачу, приводится ниже:

```
program by94d1t1;
Const
     WHITE =  1;
     GRAY  =  2;
var
   G,B                    : array [1..100,1..100] of byte;
   Color,kG,kB,Dom,CurC,Kart: array [1..100] of byte;
   D,F,T,Q                : array [1..100] of longint;
   N,i,j,Time,v,u,M,GRA_Y : longint;

   Function max(a,b:longint):longint;
       begin
          if (a>b) then max:=a else max:=b
       end;
  Procedure Reverse;
     begin
        for i:=1 to N do KB[i]:=0;
        for i:=1 to N do
        for j:=1 to kG[i] do
            begin
                inc(kB[G[i,j]]);
                b[G[i,j],kB[G[i,j]]]:=i;
            end;
     end;
  Procedure AddDominator (Domi:longint);
     begin
        for i:=1 to N do
           if CurC[i]<>WHITE then dom[i]:=WHITE;
        dom[Domi]:=GRAY;
        for i:=1 to N do curC[i]:=WHITE;
     end;
  Procedure DFS(i:byte);
     Var  j : byte;
     begin
        color[i]:=GRAY;  curC[i]:=GRAY; inc(Time);
        for j:=1 to kG[i] do
           if curC[G[i,j]]=WHITE then DFS(G[i,j]);
        inc(Time);
        if F[i]=0 then F[i]:=Time;
    end;
  Procedure BuildDominators;
     begin
        Time:=0;
        for i:=1 to N do
           begin
                Color[i]:=WHITE; Dom[i]:=WHITE;
                CurC[i]:=WHITE; F[i]:=0;
           end;
        for v:=1 to N do
           if color[v]=WHITE
              then begin DFS(v); AddDominator(v); end;
     end;

  Procedure SortF;
     var
         i,j,MaxD,k,Temp :longint;
     begin
        for i:=1 to N do Q[i]:=i;
        for i:=1 to N-1 do
            begin
               MaxD:=F[i]; K:=i;
               for j:=i+1 to N do
                  if F[j]>MaxD
                     then begin MaxD:=F[j]; k:=j; end;
               F[k]:=F[i]; F[i]:=MaxD; Temp:=Q[k];
               Q[k]:=Q[i]; Q[i]:=Temp;
            end
     end;

  Procedure SortT;
    var
        i,j,MaxD,k,Temp :longint;
    begin
       for i:=1 to M-1 do
          begin
              MaxD:=T[i]; K:=i;
```

```
            for j:=i+1 to M do
               if T[j]<MaxD
            then begin MaxD:=T[j]; k:=j; end;
            T[k]:=T[i]; T[i]:=MaxD;
         end
   end;

 Procedure DFS2(i:byte);
    var
        j : byte;
    begin
        color[i]:=GRAY; curc[i]:=GRAY;
        for j:=1 to kB[i] do
           if color[B[i,j]]=WHITE then DFS2(B[i,j] );
    end;

 Procedure AddDominator2 (Domi:longint);
    var
        MinT, MinK, KS : longint;
        FlDom  : boolean;
    begin
        KS:=0;
        for i:=1 to N do
           if Curc[i]<>WHITE then inc(KS);
        if ks>1
           then
              for i:=1 to N do
                 if CurC[i]<>WHITE then Kart[i]:=GRAY;
        for i:=1 to N do curC[i]:=WHITE;
   end;

 Procedure BuildDominators2;
     begin
         Time:=0;
         for i:=1 to N do
             begin
                Color[i]:=WHITE;
                CurC[i] :=WHITE; Kart[i]:=WHITE;
             end;
         SortF;
         for v:=1 to N do
             if color[Q[v]]=WHITE
                then begin DFS2(Q[v]); AddDominator2(Q[v]);
end;
         for i:=1 to N do
             if (Kart[i]<>WHITE)
                then begin inc(M); T[M]:=i;end;
         SortT;
         writeln(M);
         for i:=1 to M do writeln (T[i]);
   end;

begin
    assign(input,'input.txt'); reset(input);
    assign(output,'output.txt'); rewrite(output);
    readln(N);
    M:=0;
    for i:=1 to N do
        begin
            readln(u,v);
            inc(kG[u]); G[u,kG[u]]:=v;
            if u=v then begin inc(M); T[M]:=u;end;
        end;
  BuildDominators;
  Reverse;
  BuildDominators2;
  close(input); close(output);
end.
```

4.2. Задача "Межшкольная сеть"

(IOI'96 — Международная олимпиада по информатике, Венгрия)

Некоторые школы связаны компьютерной сетью. Между школами заключены соглашения — каждая школа имеет список школ-получателей, которым она рассылает программное обеспечение всякий раз, получив новое бесплатное программное обеспечение (извне сети или из другой школы). При этом, если школа B есть в списке получателей школы A, то школа A может и не быть в списке получателей школы B.

Требуется написать программу, определяющую минимальное количество школ, которым надо передать по экземпляру нового программного обеспечения, чтобы распространить его по всем школам сети в соответствии с соглашениями (подзадача A).

Кроме того, надо обеспечить возможность рассылки нового программного обеспечения из любой школы по всем остальным школам. Для этого следует расширить списки получателей некоторых школ, добавляя в них новые школы. Требуется найти минимальное суммарное количество расширений списков, при котором программное обеспечение из любой школы достигло бы остальных школ (подзадача B). Одно расширение означает добавление одной новой школы-получателя в список получателей одной из школ.

Входные данные

Первая строка файла INPUT.TXT содержит целое число N — количество школ в сети ($2 \le N \le 100$). Школы нумеруются первыми N положительными целыми числами. Каждая из следующих N строк задает список получателей. Строка i+1 содержит номера получателей i-й школы. Каждый такой список заканчивается нулем. Пустой список содержит только 0.

Выходные данные

Ваша программа должна записать 2 строки в файл OUTPUT.TXT. Его первая строка должна содержать одно положительное целое число — решение подзадачи A. Вторая строка должна содержать решение подзадачи B.

Пример ввода и вывода

INPUT.TXT	OUTPUT.TXT
5	1
2 4 3 0	2
4 5 0	
0	
0	
1 0	

Подзадача A требует найти доминантное множество — т.е. минимальное множество вершин исходного графа, из которого есть цепи ко всем остальным вершинам графа.

Обходом в глубину, до тех пор пока есть свободные вершины, находим все вершины, которые достижимы из свободных. Из множества доминант вычеркиваем все достижимые на текущем ходу вершины и добавляем в множество доминант исходную свободную.

Подзадача B требует найти количество ребер, которые нужно добавить, чтобы исходный ориентированный граф стал сильносвязанным. По исходному графу строится транспонированный (стрелки на дугах меняются на противоположные). И для нового (транспонированного) графа опять строится доминантное множество.

Если и первое, и второе доминантные множества состоят из одного и того же элемента — вершины 1, то граф был сильносвязанный, и добавлять ребра не нужно. Иначе, необходимо добавить максимальное из количеств элементов в построенных доминантных множествах.

Поскольку практически почти такая же задача полностью рассмотрена в предыдущем разделе, полное описание решения здесь не приводится.

Другое решение этой же задачи

Для подзадачи A. Методом Флойда строим матрицу достижимости.

В результате все вершины разбиваются на группы:

- истоки (в них нет входящих дуг);
- стоки (из них нет исходящих дуг);
- промежуточные (есть и входящие, и исходящие дуги).

Количество истоков — ответ подзадачи A.

Для подзадачи B. Если нет ни истоков, ни стоков, то граф — ССК (сильносвязанная компонента), и добавлять ребра не нужно. Иначе, выбираем максимальное из количеств истоков и стоков — это и есть количество ребер, которые нужно добавить.

Пояснение. Добавляем связи между истоками и стоками, в результате чего граф и становится сильносвязанной компонентой.

4.3. Задача "Винни-Пух и Пятачок"

(Республиканская олимпиада по информатике 1995 г., автор — Котов В.М.)

Винни-Пух и Пятачок нанялись защищать компьютерную сеть от хакеров, которые выкачивали из компьютеров секретную информацию. Компьютерная сеть Винни-Пуха и Пятачка состояла из связанных между собой больших ЭВМ, к каждой из которых подключалось несколько терминалов. Подключение к одной из больших ЭВМ позволяло получить информацию, содержащуюся в памяти этой ЭВМ, а также всю информацию, доступную для ЭВМ, к которым данная ЭВМ могла направлять запросы. Хакеры и раньше нападали на подобные компьютерные сети, и их тактика была известна. Поэтому Винни-Пух и Пятачок разработали специальную программу, которая помогла принять меры против готовившегося нападения.

Тактика хакеров

При нападениях хакеры всегда получают доступ к информации всех ЭВМ в сети. Добиваются они этого, захватывая некоторые ЭВМ таким образом, чтобы от них можно было запросить информацию у оставшихся ЭВМ в сети. Вариантов захвата существует множество, например, захватить все ЭВМ. Но хакеры всегда выбирают такой вариант, при котором суммарное количество терминалов у захваченных ЭВМ минимально.

Примечание. В сети Винни-Пуха и Пятачка ни у каких двух ЭВМ количество терминалов не совпадает.

Техническое задание

Необходимо написать программу, входными данными которой было бы описание сети, а выходными — список номеров ЭВМ, которые могут быть выбраны хакерами для захвата сети согласно их тактике. Ввод осуществляется из файла с именем INPUT.TXT. Возможен ввод с клавиатуры.

Формат ввода

Количество ЭВМ в сети: N
ЭВМ #1 имеет терминалов: T[1]
ЭВМ #2 имеет терминалов: T[2]
...
ЭВМ #N имеет терминалов: T[N]
Права на запрос :
A[1] B[1]
A[2] B[2]
...
A[K] B[K]
0 0

Где A[i] и B[i] — номера ЭВМ; последняя строка "0 0" обозначает конец списка права на запрос; каждая пара A[i] B[i] означает, что ЭВМ с номером A[i] имеет право запрашивать информацию у ЭВМ с номером B[i] (A[i] не равно B[i]).

При вводе числа N и T[i] — натуральные, T[i] ≤1000, N≤50, K≤2450.

Формат вывода

Номера захватываемых ЭВМ: C[1] C[2] ... C[M]. Количество захватываемых ЭВМ: <M>

Пример

```
INPUT.TXT        OUTPUT.TXT
5                1 4
100              2
2
1
3
10
1 3
3 2
4 3
4 5
5 4
0 0
```

Поиском в глубину находим множество доминаторов сети (истоки графа). Однако для оптимального выбора ЭВМ по количеству терминалов необходимо выбрать лучшую ЭВМ (с наименьшим количеством терминалов) в каждой сильносвязанной компоненте.

Для этого инвертируем граф (меняем направление всех дуг графа на противоположное) и поиском в глубину по инвертированному графу находим все его сильносвязанные компоненты (последовательные деревья, которые получаются поиском в глубину). Нас интересуют только сильносвязанные компоненты с количеством вершин больше 1. В этом случае находим среди них вершину с минимальным количеством терминалов и заменяем соответствующий доминатор (если от этой ССК ранее в доминаторы попала другая вершина).

Полное решение этой задачи не приводится, т.к. оно очень похоже на полное решение предыдущих двух задач.

(Окончание следует)

Serrgio /HI-TECH,
http://hi-tech.nsys.by/
E-mail: serrgio@gorki.unibel.by

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

(Продолжение. Начало в №№9-12/2001, №№1-9/2002)

Программирование под WIN32

Вдогонку к главе 4

Ну конечно же, все догадались, что локальную переменную

```
LOCAL lpszBuffer :DWORD ;адрес буфера
```

надлежит удалить из кода.

Изменяем наш исходник, и только потом читаем дальше.

6. Стек и локальные переменные

#1. Представьте себе картину — накачанный "колесами" и протеиновыми коктейлями "шварценеггер" бьет изо всей силы по боксерской груше. Та отлетает в сторону, а потом по каким-то абсолютно не физическим законам кинематографа возвращается назад, и бьет этого боксера по лицу, в результате чего "шварценеггер" отлетает на несколько метров, и, обязательно задев что-нибудь из мебели, "размазывается" по стене — к неописуемому удовольствию зрителя. Посмотрев на такую картину, Станиславский бы сказал: "Не верю!" А вот дZенствующей программер, увидя это безобразие, подумал бы: "Ба! Да совсем как стек. Помнится, в одной из своих кулхацкерных прог я на похожие грабли как раз и напоролся".

Краткое содержание предыдущей части: мы написали программу, использующую локальные переменные, которые существуют только тогда, когда процедура получает управле-

ние, и располагаются в стеке. Однако каким образом происходит создание и работа с подобными переменными, я предложил исследовать самостоятельно — при помощи Иды и Сайса, которым были посвящены две предыдущие главы. Получилось ли у вас что-нибудь толковое — не знаю.

В главе 1.7 мы уже разобрались с очередностью записи в стек и чтения из него. Напомню, что доступ к стеку осуществляется в соответствии с принципом LIFO (Last In First Out — Последним Пришел, Первым Ушел). Однако это отнюдь не единственное, что нам нужно знать о стеке — конечно же, если мы не собираемся время от времени получать "отдачу" от "боксерской груши" ;).

Нам уже хорошо известно, что стек можно использовать для временного хранения данных — с его помощью мы выкручивались из такой проблемы как недостаточное для полета нашей фантазии количество регистров. Т.е. мы временно сохраняли значения регистров в стеке, активно юзали их для наших нужд, а потом снова восстанавливали "статус кво", за исключением, в большинстве случаев, одного-единственного регистра, в котором хранился результат проделанной работы.

Также известно, что в инструкциях процессоров от Intel переменные (которые в памяти) не могут выступать в качестве приемника и источника одновременно, то есть инструкция:

```
mov  [dwVar1],[dwVar2]
```

не проходит. А выкрутиться из такой ситуации можно двумя способами — либо использовать в качестве посредника регистр (которых, как всегда, мало):

```
mov eax,[dwVar2]
mov [dwVar1],eax
```

либо задействовав стек:

```
push [dwVar2]
pop  [dwVar1]
```

Кстати, недавно в почтовой рассылке RTFM_helpers прошло обсуждение того, как можно копировать из памяти в память — там было упомянуто, например, использование movs. А если подумать, можно найти и другие нетривиальные способы.

Для тех, кто еще не понял. Вот этот кусок кода:

```
push 1
push 2
push 3
pop  eax
pop  ebx
pop  ecx
```

делает то же самое, что и следующий код (если только не учитывать разницу в скорости, размере кода и побочные эффекты):

```
mov  eax,3
mov  ebx,2
mov  ecx,1
```

Сомневающиеся могут проверить под отладчиком :). Также попробуйте сравнить:

```
push 1
pop  eax
```

и психоделическое:

```
sub  esp,4
mov  dword ptr [esp],1
mov  eax,[esp]
add  esp,4
```

Вы можете мне не поверить, но эти два куска кода тоже "делают" одно и то же, хотя последний и кажется плодом больного воображения.

- Что за ***esp*** такой? — спросите вы.

Поширe откройте глаза и слушайте — сейчас я поведаю вам страшные тайны ;).

#2. Помните мою аналогию с блинами от штанги и вделанным в пол штырем, на который они надевались для хранения? Так вот, адрес самого верхнего блина — это ***вершина стека***, и хранится он (адрес) в регистре ***esp/sp***. Другими словами, *вершина стека — это адрес последнего занесенного в стек элемента*.

Давайте посмотрим на поведение ***esp*** при трассировке следующего кода:

```
Main proc
 push 1
 push 2
 push 3
 pop  eax
 pop  ebx
 pop  ecx
 invoke ExitProcess,0
Main endp
```

При загрузке мы видим, что регистр ***esp*** инициализирован значением 12FFC4 (в других условиях стартовое значение может быть другим, но суть от этого не меняется). Давайте выполним один шаг (команда "t" — trace). В результате этого в стек "ляжет" 1, а значение регистра ***esp*** поменяется на 12FFC0. Трассируем дальше и наблюдаем, как изменяется ***esp***:

```
push 1    ;esp=12FFC0
push 2    ;esp=12FFBC
push 3    ;esp=12FFB8
pop  eax  ;esp=12FFBC
pop  ebx  ;esp=12FFC0
pop  ecx  ;esp=12FFC4
```

Протрассировав первые три строчки, мы можем сделать вывод, что *стек "растет" в сторону младших адресов* (12FFC0 > 12FFBC > 12FFB8, логично?), а шагом изменения регистра ***esp*** является 4. И это правильно, так как при 32-битном режиме адресации в стеке сохраняются двойные слова, они же — 4 байта (хотя допускается класть в стек также и 2-байтные слова — при этом шаг равен 2). К слову сказать, если бы мы писали под DOS (точнее, в реальном или виртуальном режиме), то стек у нас адресовался бы регистром ***sp***, и изменялся бы он на ± 2.

Обобщаем. Алгоритм работы команды ***push <источник>*** следующий:

1. Уменьшение значения указателя стека ***esp/sp*** на 4/2.
2. Запись значения источника по адресу ***ss:esp/sp*** (вершина стека).

Об алгоритме работы команды ***pop*** догадайтесь сами...

Для последователей дZена предлагаем тему для исследования — какое значение будет лежать в стеке после инструкции **push esp**.

ПРЕДУПРЕЖДЕНИЕ: ни в коем случае не принимайте результаты отдельных экспериментов в конкретных условиях за абсолютную истину, верную всюду и всегда!

Теперь, после вышесказанного, вы легко сообразите, какое значение примет регистр **eax** в следующем извращенном случае:

```
push 1    ;esp=12FFC0
push 2    ;esp=12FFBC
push 3    ;esp=12FFB8
add  esp,4
pop  eax  ;esp=12FFC0
pop  ebx  ;esp=12FFC4
```

Правильный ответ — 2 :).

#3. Есть еще один регистр, ассоциируемый со стеком — ***ebp/bp***, и описывается его функция так, что выговорить страшно — ***указатель базы кадра стека.*** Такое название этого регистра я нашел в книжке Юрова и Хорошенко "ASSEMBLER, учебный курс". Нет, конечно же, можно назвать калоши "мокроступами", а bitmap или растр "двоично-точечной картинкой", но... "*У меня нет слов, у меня есть только выражения в адрес того, кто заворачивает такие коленца*" (© А.Белоусов). А посему давайте заменим словосочетание "кадр стека" простым народным :) словом "фрейм", а "указатель базы" заменим на просто "база" (или "указатель" — по вкусу).

В результате подобных терминологических "подстановок" получается следующая картина — есть у нас в компьютере некие "фреймы", располагаются они в стеке, и адреса этих, пока что непонятных нам штук, завязаны с регистром ***ebp/bp***.

Дело в следующем. Любая процедура/функция в терминах любого процедурного языка имеет (может иметь) ноль и больше параметров и локальных переменных. *Область памяти, создаваемая (выделяемая) при вызове процедур для аргументов и локальных переменных*, и называется фреймом. А чтобы процедура могла быть рекурсивной (т.е. могла вызывать саму себя или вызываться из других процедур, которые она вызывает) или реентерабельной (т.е. чтобы код процедуры мог использоваться в параллельных процессах), фреймы для нее должны размещаться в стекоподобной структуре данных — "стеке". А поскольку процедурные языки ближе к "естественному" мышлению, то в наборы инструкций процессоров и ассемблеров вводят поддержку подобных высокоуровневых конструкций.

Существуют нюансы и различия в реализациях, например:

- сишные компиляторы для PC генерят код, в котором часть фрейма с аргументами после вызова процедуры удаляется вызывающим кодом, а в паскалевских соглашениях о вызове фрейм всегда удаляется самой процедурой (что экономит на размере кода, поэтому это соглашение было принято как стандартное в Windows);

- в Паскале существует понятие локальных процедур, которые могут обращаться к параметрам и локальным переменным всех родительских (в смысле статического размещения, а не динамического порядка вызовов) процедур и при этом могут быть рекурсивными (родительские процедуры тоже могут быть рекурсивными, и доступ к переменным должен идти к последнему, активному экземпляру). Для поддержки этой идеи во фреймы добавляются указатели на родительские фреймы, обычно организованные в виде списка;

- Фортран вообще язык не рекурсивный, поэтому IBM в реализации Фортрана на IBM/360 (где, кстати, нет поддержки стека) для каждой процедуры заводила фрейм статически, во время компиляции.

Подобные "нюансы" сущности фрейма, конечно же, не меняют. Приведены они по одной единственной причине — чтобы вы поняли некоторую условность такого термина как "фрейм".

Теперь, собственно, о регистре ***ebp/ep***. В случае стекового фрейма его адрес не фиксирован, поэтому адресация параметров и переменных в нем должна быть "базисно-индексной", относительно начала фрейма. На PC под это идеально подошел (или изначально проектировался) ***bp/ebp*** — в отличие от ***sp***, он может служить базой, и также адресуется относительно регистра **ss**.

Я знаю, что вы мало что поняли из всего этого бреда, а посему будем познавать истину путем долгой и продолжительной медитации.

#4. Мы привыкли к тому, что извлекать данные из стека можно только повинуясь очередности LIFO. А что, если нам понадобилось обратиться к произвольному элементу стека? Один из возможных способов мы уже рассмотрели — это изменение значения регистра ***esp/sp*** плюс команда ***pop***. Это далеко не совсем хорошая идея, и вам не стоит издеваться над стеком таким изощренным способом. Если, конечно же, вы не хотите уподобляться Штирлицу и Мюллеру, которые стреляли по очереди...

Напомню, регистр ***ebp/bp*** ведет себя приблизительно таким же образом, как и хорошо нам знакомый ***ebx/bx***. То есть он может выполнять роль ***базы***.

Напомню, что, например,

```
mov ebx,12FFC0h
mov al,[ebx]
```

присвоит регистру ***al*** значение байта по адресу 12FFC0 из сегмента, задаваемого регистром ***ds***.

Точно таким же образом можно использовать и регистр ***ebp/bp***. Говоря другими словами, это один из немногих регистров, которые можно брать в квадратные скобки, не увеличивая при этом размер инструкции :). Т.е. он позволяет работать с ячейкой памяти, адрес которой находится в регистре.

Проиллюстрирую это на простом примере. Допустим, вы занесли в стек кучу разных параметров:

```
адрес     значение
0012FFC4  77E7EB69
0012FFC8  0047E4AC
0012FFCC  0012DAB4
0012FFD0  7FFDF080
```

И возжелалось вам в силу какой-нибудь нездоровой производственной необходимости прочитать "здесь и сейчас", например, предпоследний элемент. В этом случае делаем вот что:

```
mov ebp,esp
mov eax,[ebp+4]
```

Расшифровываю. Мы принимаем адрес самого последнего из записанных в стек элементов за точку отсчета, и путем прибавления к этой "точке отсчета" четверок можно легко получить доступ к тому элементу стека, который нашей программерской душеньке возжелался. В принципе, в данном примере можно было бы использовать ***esp*** вместо ***ebp***, но, во-первых, должны же мы были показать использование ***ebp***. А во-вторых, в больших фрагментах кода использование ***esp*** непосредственно имеет свои недостатки (больший размер инструкций и необходимость отслеживать изменение вершины стека).

Вот именно такое безобразие и называется "организация произвольного доступа к данным внутри стека".

#5. Перед тем как мы пойдем дальше и разберемся-таки с локальными переменными — пара слов для особо "продвинутых".

В защищенном режиме (или в реальном режиме с префиксами смены разрядности) базой может служить любой регистр, поэтому, если мы точно знаем состояние регистра ***esp*** (т.е. мы точно знаем, сколько раз мы делали ***push***, а сколько ***pop***), то для доступа к фрейму можно использовать ***esp*** (при этом индексы одной и той же перменной в разных местах процедуры могут отличаться из-за промежуточных ***push/pop***). Подобного рода оптимизацией занимаются, насколько я знаю, последние версии BC и VC. А в их "асмах" появилась директива "фраме-поинтоммисинс", как раз для таких извратов и предназначенная.

Однако здесь есть недостатки по сравнению с использованием более или менее статичного ***ebp***, как об этом уже было упомянуто выше. Во-первых, с ***[esp]*** инструкции длиннее, во-вторых, нужно быть очень аккуратным в подсчете промежуточных ***push/pop***, чтобы верно подставлять смещение до аргумента или переменной в ***[esp+xx]*** (а ведь есть относительно непредсказуемые инструкции вида ***push [esp+xx]***). Наконец, поскольку индекс ***xx*** может постоянно меняться, использовать встроенные директивы типа ***args*** или даже вручную расставленные ***equ*** становится малореальным. Поэтому возможность использования ***esp*** в качестве базы (при ручной кодогенерации — "глюкалово" полное) отнюдь не умаляет полезности ***ebp***.

#6. Ну вот, мы и подошли к самому интересному. Сейчас мы готовы проанализировать нашу программу на предмет того, что она там вытворяет с локальными переменными.

```
:00401000 NumberOfCharsWritten= dword ptr -90h
:00401000 nNumberOfCharsToWrite= dword ptr -8Ch
:00401000 hConsoleInput   = dword ptr -88h
:00401000 hConsoleOutput  = dword ptr -84h
```

```
:00401000 Buffer          = byte ptr -80h
:00401000
:00401000 push   ebp
:00401001 mov    ebp, esp
:00401003 add    esp, 0FFFFFF70h
:00401009 push   offset aInputOutput  ; lpConsoleTitle
:0040100E call   SetConsoleTitleA
:00401013 push   0FFFFFFF5h           ; nStdHandle
:00401015 call   GetStdHandle
:0040101A mov    [ebp+hConsoleOutput], eax
:00401020 push   0                    ; lpReserved
:00401022 lea    eax, [ebp+NumberOfCharsWritten]
:00401028 push   eax                  ; lpNumberOfCharsWritten
:00401029 push   11h                  ; nNumberOfCharsToWrite
:0040102B push   offset aTypeSomething ; lpBuffer
:00401030 push   [ebp+hConsoleOutput] ; hConsoleOutput
:00401036 call   WriteConsoleA
:0040103B push   0FFFFFFF6h           ; nStdHandle
:0040103D call   GetStdHandle
:00401042 mov    [ebp+hConsoleInput], eax
:00401048 push   0                    ; lpReserved
:0040104A lea    eax, [ebp+nNumberOfCharsToWrite]
:00401050 push   eax                  ; lpNumberOfCharsRead
:00401051 push   80h                  ; nNumberOfCharsToRead
:00401056 lea    eax, [ebp+Buffer]
:00401059 push   eax                  ; lpBuffer
:0040105A push   [ebp+hConsoleInput]  ; hConsoleInput
:00401060 call   ReadConsoleA
:00401065 push   0                    ; lpReserved
:00401067 lea    eax, [ebp+NumberOfCharsWritten]
:0040106D push   eax                  ; lpNumberOfCharsWritten
:0040106E push   0Ch                  ; nNumberOfCharsToWrite
:00401070 push   offset aYouTyped     ; lpBuffer
:00401075 push   [ebp+hConsoleOutput] ; hConsoleOutput
:0040107B call   WriteConsoleA
:00401080 push   0                    ; lpReserved
:00401082 lea    eax, [ebp+NumberOfCharsWritten]
:00401088 push   eax                  ; lpNumberOfCharsWritten
:00401089 push   [ebp+nNumberOfCharsToWrite];nNumberOfCharsToWrite
:0040108F lea    eax, [ebp+Buffer]
:00401092 push   eax                  ; lpBuffer
:00401093 push   [ebp+hConsoleOutput] ; hConsoleOutput
:00401099 call   WriteConsoleA
:0040109E push   7D0h                 ; dwMilliseconds
:004010A3 call   Sleep
:004010A8 push   0                    ; uExitCode
:004010AA call   ExitProcess
```

Начнем с того, что полегче ;). Нетрудно заметить, что команда ***addr*** в применении к локальным переменным (в контексте ***invoke***) во всех случаях генерит следующий код:

```
lea eax,[ebp-X]
push eax
```

Предвидя "грабли", на которые может натолкнуться начинающий, сразу оговорюсь, эти строки **принципиально** отличаются от

```
push [ebp-X].
```

Вы не поверите, но почему-то многие новички здесь путаются... Как, вы тоже? ;))

Первое — заносит в стек ***указатель на***. А второе — ***значение***. Всем медитировать!

Обратите внимание, что Ида услужливо вынесла вверх листинга блок констант, каждая из которых имеет отрицательное значение. Но мы-то с вами еще со школы умеем решать простенькие задачки на сложение отрицательных чисел и без труда вычислим, что система уравнений

a = -84

b = x+a

имеет более тривиальное решение:

b = x-84

Хе-хе... Слышала бы меня сейчас моя школьная учительница математики ;)).

#7. Очевидно, что ***ebp*** — это некая "точка отсчета", *относительно* которой адресуются локальные переменные. А что же у нас в ***ebp***? Смотрим на начало процедуры, и ищем там строчки

```
:00401001 mov     ebp, esp
:00401003 add     esp, 0FFFFFF70h
```

И медленно ("брюки превращаются... брюки превращаются...") приоткрываем завесу над тайной. Ассемблер смотрит, сколько мы задекларировали локальных переменных, и какова размерность каждой. На основании этой информации определяется размер фрейма. В нашем случае задекларировано (смотрим исходник):

```
LOCAL InputBuffer[128]   :BYTE  ;буфер для ввода
LOCAL hOutPut            :DWORD ;хэндл для вывода
LOCAL hInput             :DWORD ;хэндл для ввода
LOCAL nRead              :DWORD ;прочитано байтов
LOCAL nWriten            :DWORD ;напечатано байтов
```

То есть 128 байтов плюс 4 двойных слова по 4 байта в каждом. Итого, для всего этого "богатства" нужно выделить 144 байта памяти.

Ну и замечательно! Сохраняем адрес вершины стека (память под переменные еще не отведена!) в регистре ***ebp***. Теперь это у нас вовсе не вершина, а "точка отсчета" для локальных переменных. А саму вершину стека передвигаем на 144 байта "вверх", в сторону младших адресов (там у нас будет область для хранения локальных переменных). Как видите, все очень просто :).

	0012FF20	
	0012FF24	
	0012FF28	
esp	0012FF2C	**ВЕРШИНА**
[ebr - 90h]	0012FF30	**nWriten**
[ebr - 8Ch]	0012FF34	**nRead**
[ebr - 88h]	0012FF38	**hInput**
[ebr - 84h]	0012FF3C	**hOutPut**
[ebr - 00h]	0012FF40	**InputBuffer[128]**
	0012FF44	
	0012FF48	
	0012FF4C	
	0012FF50	
	0012FF54	
	0012FF58	
	0012FF5C	
	0012FF60	
	0012FF64	
	0012FF68	
	0012FF6C	
	0012FF70	
	0012FF74	
	0012FF78	
	0012FF7C	
	0012FF80	
	0012FF84	
	0012FF88	
	0012FF8C	
	0012FF90	
	0012FF94	
	0012FF98	
	0012FF9C	
	0012FFA0	
	0012FFA4	
	0012FFA8	
	0012FFAC	
	0012FFB0	
	0012FFB4	
	0012FFB8	
	0012FFBC	
ebp	0012FFC0	**БАЗА ФРЕЙМА**
	0012FFC4	
	0012FFC8	
	0012FFCC	
	0012FFD0	

Задание для медитации — чем будут отличаться генеримые инструкции для директивы addr в случае, если addr будет применяться к **аргументам** процедуры, а не к **локальным переменным**?

Если вас смущает то, что для "поднятия планки" используется инструкция add и столь большое число, то вернитесь к главе о кодировании отрицательных чисел и особенностях дополнительного кода. Либо поставьте в Иде указатель на смущающее вас число 0FFFFFF70h и нажмите на Ctrl+, после чего долго и упорно медитируйте :).

Если подключить фантазию и пару раз протрассировать это безобразие под отладчиком, то получится такая картинка:

Распечатайте ее и повесьте над своей кроватью.

(Продолжение следует)

Трактать о Dynamic Link Libraries

А.ИВАНЧИКОВ, И.ИВАНЧИКОВА,
г.Минск.

(Продолжение. Начало в №9/2002)

Явная компоновка с DLL

Динамическая библиотека в рассмотренном примере строилась и, соответственно, использовалась клиентской программой по методу, который имеет название "с явной компоновкой" или в оригинальной нотации — "runtime dynamic linking".

Отличительными особенностями данного метода являются:

- усложнение программного кода клиентских приложений за счет явной загрузки динамической библиотеки, получения адресов импортируемых функций и освобождения библиотеки, когда она уже не нужна;
- весь ход работы с подгружаемыми DLL полностью контролируется клиентским кодом благодаря анализу возвращаемых значений функций ядра;
- DLL загружается в оперативную память, а именно в адресное пространство клиентского процесса, только в момент обращения к функции LoadLibrary;
- по окончании работы с DLL она может быть выгружена из памяти функцией FreeLibrary.

При реализации DLL, альтернативой DEF-файлу является указание перед именем экспортируемой функции выражения __declspec(dllexport). Данное выражение делает то же самое — сообщает компилятору, что имя будет экспортировано из DLL. Таким образом, если в файле QuickSort.cpp функция QuickSort будет иметь вид

```
extern "C" __declspec(dllexport) void QuickSort(
    void* pvArray,
    unsigned unCount,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) )
{
    RecSort((char*)pvArray,
        (char*)pvArray + (unCount - 1) * unWidth,
        unWidth, compF);
}
```

то файл QuickSort.def из проекта QuickSort можно удалить.

Явную загрузку динамической библиотеки осуществляет функция LoadLibrary, формат которой:

```
HMODULE LoadLibrary(LPCTSTR lpFileName);
```

Данная функция грузит динамическую библиотеку по полному имени файла — lpFileName. Если путь к файлу DLL не задан, то применяется стандартная для Windows процедура поиска, при которой файл последовательно ищется по следующим путям:

1. Каталог, из которого было загружено клиентское приложение.
2. Текущий каталог.
3. Windows 95/98/Me: системный каталог Windows (возвращается функцией GetSystemDirectory).
4. Windows NT/2000 и позже: 32-битный системный каталог Windows (SYSTEM32 — возвращается функцией GetSystemDirectory).
5. Windows NT/2000 и позже: 16-битный системный каталог Windows (SYSTEM — возвращается функцией GetSystemDirectory).
6. Каталог Windows (возвращается функцией GetWindowsDirectory).
7. Список каталогов из переменной окружения PATH.

В случае успешного завершения, функция LoadLibrary возвращает описатель загруженной DLL, иначе — значение NULL.

После того как базовая часть DLL загружена в адресное пространство процесса, необходимо получить адреса отдельных используемых функций. Для этого используется функция GetProcAddress, формат которой:

```
FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName);
```

В качестве параметров функции передаются описатель загруженной DLL и строка с именем экспортируемой функции или переменной. В случае успеха функция GetProcAddress подгружает код импортируемой функции, и возвращается указатель на функцию или переменную. Если получить адрес функции/переменной не удалось, возвращается значение NULL.

После использования DLL она может быть удалена из адресного пространства процесса функцией FreeLibrary, формат которой:

```
BOOL FreeLibrary(HMODULE hModule);
```

Неявная компоновка с DLL

Динамическая библиотека, подключаемая по методу "с неявной компоновкой" (load-time dynamic linking), характеризуется следующим:

- все подключенные DLL загружаются всегда, даже если в течение всего сеанса работы программа ни разу не обратится ни к одной из них;
- если хотя бы одна из требуемых DLL отсутствует (или DLL не экспортирует хотя бы одной требуемой функции), то загрузка исполняемого файла прерывается системным сообщением "Unable To Locate DLL". Это происходит, даже если отсутствие конкретной DLL некритично для исполнения программы;
- клиентский код избавлен от необходимости работать с DLL — импортируемые функции вызываются как обычные функции в составе приложения;
- имеется возможность экспортировать не только функции и переменные, но и классы.

Для демонстрации данного метода работы с DLL в имеющемся окружении проектов создадим еще один проект с именем TestSortLTDL, представляющий собой Win32 Console Application. Данный проект, как и TestSort, изначально создается пустым. После создания проекта следует добавить в него файл TestSortLTDL.cpp, который будет иметь следующий вид:

```
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  TestSortLTDL.cpp
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
#include <stdio.h>
#include <stdlib.h>
#include "../QuickSort/QuickSort.h"
// размерность массива
#define SIZE_TETS_ARRAY 10

// Функция сравнения для QuickSort
```

```
//-----------------------------------------------
int _cmp(
    const void* ppvElem1,
    const void* ppvElem2 )
{
    return *(int*)ppvElem1 - *(int*)ppvElem2;
}

//-----------------------------------------------
void main(void)
{
    // демонстрация работы QuickSort
    int nArray[SIZE_TETS_ARRAY];

    for ( int i = 0; i < SIZE_TETS_ARRAY; i++ )
        printf("%7d", nArray[i] = rand());

    printf("\n");

    QuickSort(nArray, SIZE_TETS_ARRAY,
              sizeof(int), _cmp);

    for ( i = 0; i < SIZE_TETS_ARRAY; i++ )
        printf("%7d", nArray[i]);
}
```

Исходный код данной версии тестового приложения в два раза короче, чем исходный код аналогичного приложения TestSort. Но так как теперь в коде используется реальное имя внешней функции, ее необходимо описать. Именно по этой причине, а также для того чтобы добиться некоторой универсальности, необходимо создать заголовочный файл, который будет общим и для DLL, и для клиентского кода. В этот заголовочный файл включается ifdef — блок, который является стандартным средством создания макроса, упрощающего экспорт из DLL. При этом все файлы текущей DLL должны компилироваться с определенным макроименем. В нашем случае это QUICKSORT_EXPORTS. Кстати, это имя автоматически добавляется AppWizard при создании проекта DLL к списку определения препроцессора. Для любого из проектов, использующих эту DLL, данное макроимя не определяется. Таким образом, любой проект, в который включен этот заголовочный файл, видит функции QUICKSORT_API как импортируемые из DLL, тогда как сама DLL эти же функции видит как экспортируемые.

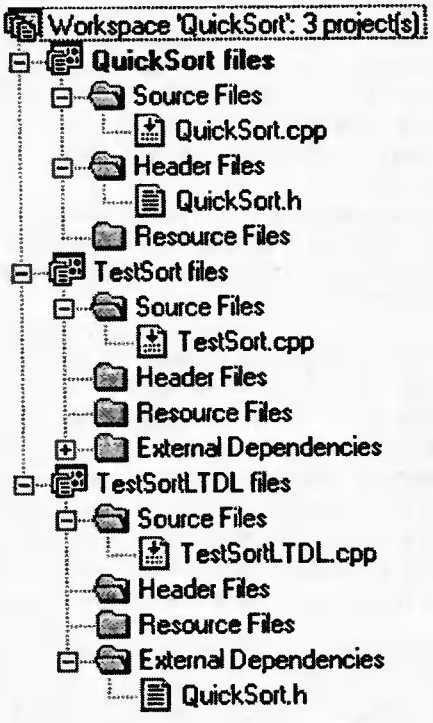
Исходный код заголовочного файла QuickSort.h имеет вид:

```
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  QuickSort.h
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//-------------------------------------------------
#ifdef QUICKSORT_EXPORTS
  // описатель экспортируемой функции
  #define QUICKSORT_API __declspec(dllexport)
#else
  // описатель импортируемой функции
  #define QUICKSORT_API __declspec(dllimport)
  // необходимость компоновки с библиотекой
  #pragma comment(lib, "QuickSort.lib")
#endif

//-------------------------------------------------
extern "C" QUICKSORT_API void QuickSort(
    void* pvArray,
    unsigned unCount,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) );
```

Файл QuickSort.h можно разместить в основном каталоге проекта, и для полного соответствия классической модели следует включить файл QuickSort.h в файл QuickSort.cpp, а также заменить в определении функции QuickSort выражение __declspec(dllexport) на QUICKSORT_API, хотя этого (включения файла и замены в определении функции), разумеется, можно и не делать.

На рис.3 показан внешний вид окна окружения проектов после добавления очередного проекта.

Рис. 3



Остается несколько последних штрихов.

Во-первых, необходимо сообщить компоновщику приложения TestSortLTDL, где будет находиться QuickSort.lib. Так как настройки проета QuickSort не изменялись, то библиотеки обеих конфигураций по умолчанию складываются в каталоги Debug и Release. Для задания этих путей необходимо открыть диалог *Project Settings* выбором пункта главного меню *Project→Settings...*; перейти в закладку *Link*, и в окне *Project Options:* добавить к имеющимся строку /libpath:"../Debug" для Debug-конфигурации проекта TestSortLTDL, и /libpath:"../Release" для его конфигурации Release/. Эти строки являются указанием компоновщику, где осуществлять поиск необходимых статических библиотек.

Во-вторых, как и для проета TestSort, будет полезно помещать результитрующие EXE-файлы в каталоги основного проекта Release/Debug.

Теперь можно собрать проекты QuickSort и TestSortLTDL и, выполнив TestSortLTDL.exe, убедиться, что все работает.

Выполнив команду

```
dumpbin -imports TestSortLTDL.exe
```

среди прочего можно наблюдать такой результат:

```
Section contains the following imports:

    QuickSort.dll
              4060B0 Import Address Table
              4065B0 Import Name Table
                0 time date stamp
                0 Index of first forwarder reference
              0  QuickSort
```

Это лишний раз свидетельствует о том, что функция QuickSort импортируется приложением из динамической библиотеки.

Работа с DLL с отложенной загрузкой

В MS Visual C++ 6.0 появился новый способ подключения DLL, сочетающий в себе почти все достоинства двух рассмотренных ранее методов — отложенная загрузка DLL (delay-load DLL). Отложенная загрузка не требует поддержки со стороны операционной системы (а значит, будет работать даже под Windows 95), и реализуется компоновщиком MS Visual C++ 6.0.

Динамическая библиотека, построенная и подключаемая по методу "с отложенной загрузкой", характеризуется следующим:

- DLL загружается в оперативную память (в адресное пространство клиентского процесса) только в тот момент, когда приложение обращается к одной из содержащихся в ней функций;
- клиентский код избавлен от необходимости работать с DLL — импортируемые функции вызываются как обычные функции в составе приложения;
- имеется возможность экспортировать не только функции и переменные, но и классы;
- для выгрузки библиотеки можно воспользоваться функцией __FUnloadDelayLoadedDLL, иначе библиотека останется в памяти процесса до его завершения;
- имеется возможность контролировать вызов импортируемых функций с соответствующей обработкой ошибок.

Для демонстрации отложенной загрузки будет необходим еще один проект. Назовем его TestSortDL. Как и два предыдущих проекта, изначально это будет пустое консольное приложение, к которому затем добавляется файл TestSortDL.cpp, исходный текст которого приведен ниже.

```
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  TestSortDL.cpp
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
#include <stdio.h>
#include <stdlib.h>
#include <wtypes.h>
#include <Delayimp.h>
#include "../QuickSort/QuickSort.h"
// размерность массива
#define SIZE_TETS_ARRAY 10
// Функция сравнения для QuickSort
//-------------------------------------------------
int _cmp(
    const void* ppvElem1,
    const void* ppvElem2 )
{
    return *(int*)ppvElem1 - *(int*)ppvElem2;
}

//-------------------------------------------------
void main(void)
{
    // демонстрация работы QuickSort
    int nArray[SIZE_TETS_ARRAY];

    for ( int i = 0; i < SIZE_TETS_ARRAY; i++ )
        printf("%7d", nArray[i] = rand());

    printf("\n");

    __try
    {
        QuickSort(nArray, SIZE_TETS_ARRAY,
                  sizeof(int), _cmp);
    }
    __except ( EXCEPTION_EXECUTE_HANDLER )
    {
        return;
    }

    for ( i = 0; i < SIZE_TETS_ARRAY; i++ )
        printf("%7d", nArray[i]);

    // выгрузка DLL
    __FUnloadDelayLoadedDLL("QuickSort.dll");
}
```

Нетрудно заметить, что данный файл отличается от своего предшественника (TestSortLTDL.cpp) немногим: во-первых, вызов QuickSort помещается теперь в блоке __try/__except, что позволяет обработать ситуацию, связанную или с отсутствием самой библиотеки, или с отсутствием в ней конкретной функции; во-вторых, для удаления библиотеки из памяти процесса в последнем коде использована функция __FUnloadDelayLoadedDLL, которая описана в заголовочном файле Delayimp.h.

Для того чтобы использовалась отложенная загрузка библиотеки, в настройках приложения, грузящего ее, необходимо установить опцию компоновщика /DelayLoad:"QuickSort.dll" (пункт главного меню *Project→Settings...*, закладка *Link*, окно *Project Options:*). Кроме того, необходимо прилинковать библиотеку Delayimp.lib, для чего следует добавить это имя к уже имеющимся в окне *Object/library modules:*, закладки *Link*. Для корректной работы функции __FUnloadDelayLoadedDLL нужно установить ключ компоновщика /Delay:unload, а также проследить, чтобы имя динамической библиотеки, переданное параметром __FUnloadDelayLoadedDLL, в точности соответствовало указанному в ключе /DelayLoad.

Если теперь собрать проект TestSortDL, то корректная работа ему гарантирована. Вызов

```
dumpbin -imports TestSortDL.exe
```

позволит наблюдать следующую картину:

```
Section contains the following delay load imports:

QuickSort.dll
                    0 Characteristics
                40990C Address of HMODULE
                4098F8 Import Address Table
                40653C Import Name Table
                406550 Bound Import Name Table
                406558 Unload Import Name Table
                   0 time date stamp
                40138 B QuickSort
```

Промежуточный вывод

Организация самой DLL на языке C++ является стандартной процедурой и не накладывает отпечаток на последующую работу с ней. Однако клиентское приложение, создаваемое с помощью MS Visual C++ 6.0, имеет три возможности использования DLL, у каждой из которых есть свои минусы (способ с отложенной загрузкой минусов, пожалуй, не имеет) и плюсы.

(Продолжение следует)

П.СОКОЛЕНКО /HI-TECH,
E-mail: pts@icomm.ru
http://hi-tech.nsys.by/
http://www.macro.aaanet.ru

Я ВЫБИРАЮ АССЕМБЛЕР?

(Полемические заметки)

Средства, которые мы применяем, оказывают глубокое (и тонкое) влияние на наши способы мышления и, следовательно, на нашу способность мыслить.
Эдсгер Дейкстра

Нельзя сказать, что ассемблер относится к числу незаслуженно забытых языков программирования, тем не менее, негативное мнение о его возможностях и целесообразности применения получило весьма широкое распространение. Если такой точки зрения придерживаются начинающие программисты или "продвинутые пользователи", то это можно отнести к недостатку практических навыков и ограниченности профессиональных знаний. Но отрицательные высказывания, которые можно встретить в некоторых книгах и статьях, лично у меня вызывают недоумение. Более того, я считаю, что подобная неконструктивная критика любого языка просто вредна. Это обстоятельство и послужило поводом для написания данных заметок.

Сравнивать язык команд с алгоритмическими языками имеет смысл только в отношении сложности и трудоемкости процесса программирования. Безусловно, при таком сравнении ассемблер уступает алгоритмическим языкам, но это не означает, что его не следует знать и уметь применять в случае необходимости.

Как известно, универсального языка программирования пока не придумали, и потребность в нем совсем не очевидна. Поэтому, для расширения сферы деятельности и повышения профессионального уровня, программист должен владеть несколькими языками (быть полиглотом), и желательно, чтобы одним из них был ассемблер. Ниже я попытаюсь обосновать эту точку зрения.

Немного истории

Ассемблер (assembly language, assembler) является машинно-ориентированным языком программирования. В переводе с английского слово assembly означает собрание, сборка, монтаж и т.п. Выбор такого названия объясняется тем, что первые трансляторы с Алгола, Фортрана, Кобола и PL/I использовали ассемблер в качестве промежуточного языка при сборке программы из отдельных модулей.

В свое время создание ассемблера было первым и очень важным шагом на пути автоматизации процесса программирования. Он избавлял программистов от необходимости записывать тексты программ в виде последовательности восьмеричных или шестнадцатеричных кодов. Появилась возможность использовать вместо чисел имена и специальные символы для обозначения машинных инструкций, их операндов и адресов. Кроме того, трансляторы, а в дальнейшем компиляторы позволяли применять специальные директивы для описания переменных, оформления макроопределений и сегментирования программ. Все это существенно упрощало рутинную работу программиста.

Ассемблер недолго оставался в группе лидеров, его потеснили алгоритмические языки. Они требовали от программиста меньшей специальной подготовки, а их машинная независимость позволяла организовать широкий обмен алгоритмами и программами и их публикацию в печати. Ассемблер же постепенно становился языком профессионалов.

С появлением персональных компьютеров (ПК) маятник качнулся в обратную сторону, и на некоторое время ассемблер вновь оказался в числе лидеров. Это произошло потому, что ограниченные возможности первых моделей ПК не позволяли применять компиляторы с языков высокого уровня. Но технический прогресс постепенно все расставил по своим местам, и программисты вновь стали отдавать предпочтение алгоритмическим языкам.

Разные языки ассемблера появляются, развиваются и отмирают вместе с семействами процессоров или микропроцессоров, для программирования которых они предназначены. Поэтому версии соответствующих компиляторов обновляются при появлении новых моделей процессоров или микропроцессоров. А отечественные и зарубежные издательства регулярно выпускают новые книги по программированию на языке ассемблера.

Специфические особенности языка

Ассемблер является языком программирования низкого уровня, составленная на нем программа представляет собой последовательность команд **конкретной ЭВМ, чаще — конкретного семейства ЭВМ**, записанных в некой условной мнемонической форме. Именно машинная ориентация определяет достоинства и недостатки этого языка.

Неоспоримым достоинством ассемблера является возможность составления программ, рационально использующих все особенности системы команд конкретной ЭВМ. Он предоставляет неограниченные возможности для различного рода трюков (в хорошем смысле этого слова), тут все зависит от профессиональных навыков программиста и его изобретательности.

Другим положительным свойством является универсальность языка — он позволяет составить программу для любой задачи, которая имеет решение и может быть решена на машинах данного семейства. Это утверждение основано на том очевидном факте, что любая программа, составленная на языке высокого уровня, при компиляции преобразуется в последовательность машинных команд.

Очевидным недостатком является низкий уровень абстрагирования от особенностей конкретной ЭВМ, необходимость знать и учитывать эти особенности. В то же время, при работе с языками высокого уровня программист может полностью сосредоточить свое внимание на особенностях реализуемого алгоритма.

Другим недостатком является большое, а иногда и очень большое количество команд, выполняемых конкретной машиной. Например, у микропроцессора Pentium 4 их около 500! Это вынуждает пользоваться при работе специальными справочниками — не дер-

жать же в голове все разнообразие имен команд и особенностей их выполнения.

Ассемблер и макроассемблер

С точки зрения программиста, язык ассемблера является подмножеством языка макроассемблера. Последний обязательно компилирует полный набор инструкций конкретного семейства микропроцессоров и, кроме того, исполняет большой набор операторов и директив (далее макросредств) различного назначения. У программистов, работающих на компьютерах IBM PC, наибольшей популярностью пользуются макроассемблеры MASM (компании Microsoft) и TASM (компании Borland), имеющие примерно одинаковые возможности и во многом (но не полностью) совместимые. В данном разделе мы будем говорить об особенностях MASM.

Прежде всего, следует отметить, что в чистом виде язык ассемблера применяется только при оформлении вставок в тексты программ, составляемых на языках высокого уровня. Без использования макросредств невозможно составить даже простую подпрограмму, не говоря уже о завершенных программах, а вот без использования команд ассемблера это возможно.

При определенных условиях исходный текст программы, выполняющей достаточно сложные действия, может содержать только макросредства. В нем не будет ни одной команды ассемблера! О какой сложности и трудоемкости программирования можно говорить в подобных случаях?

В руководствах по MASM, в том числе и в HELP, обычно описываются 8 типов операторов и 12 типов директив — начиная с версии MASM 6.10, их количество не изменялось. В рамках данной статьи не имеет смысла приводить полное описание макросредств, поэтому мы ограничимся их общей характеристикой.

Директивы макроассемблера предназначены для выполнения трех основных категорий действий:

- управление распределением оперативной памяти выполняют директивы, предназначенные для описания простых переменных, структур данных, записей и сегментирования программ;

- управление работой с внешними и внутренними подпрограммами, фрагментами программ и макроопределениями выполняют директивы их явного или предварительного описания и вызова;

- управление процессом компиляции программы выполняют директивы условной компиляции. Среди них встречаются такие знакомые каждому программисту имена как IF, ELSE, FOR, WHILE, REPEAT, GOTO.

Независимо от конкретного назначения, все директивы и операторы исполняются на стадии компиляции, при этом сами они **не преобразуются в команды**. В результате их исполнения в объектный модуль включаются или не включаются группы команд, находящиеся в исходном тексте программы или во внешних библиотеках. Кроме того, в объектный модуль включаются данные, необходимые компоновщику (LINK) для сборки и построения задачи.

Таким образом, при обсуждении достоинств и недостатков программирования на ассемблере не следует забывать, что на практике мы, как правило, работаем с языком макроассемблера, а это далеко не одно и то же.

Техника программирования

Ахиллесовой пятой ассемблера является трудоемкость программирования на этом языке. Что может сделать программист для упрощения собственного труда?

Единственным возможным решением является накопление и использование собственного и чужого опыта, оформленного в виде библиотек или отдельных модулей, желательно, с критической оценкой этого опыта. Макроассемблер позволяет применять при разработке задачи готовые исходные и объектные модули. Первые включаются в исходный текст программы с помощью директивы Include, а вторые подключаются к списку объектных модулей на стадии построения задачи компоновщиком. Есть разные источники объектных и исходных модулей, одни из них более доступны, а другие менее. Выбор зависит от ваших возможностей.

1. В состав полной версии системы программирования на макроассемблере (дистрибутивного пакета) входит набор макроопределений и подпрограмм различного назначения и примеры их использования. Внимательно просмотрите все подкаталоги вашей версии макроассемблера, наверняка в них найдутся полезные и интересные решения, которые пригодятся в вашей работе.

2. Не забывайте о том, что при составлении программ можно использовать модули библиотек, входящих в состав систем программирования на языках высокого уровня, например, на Си. Правда, для этого надо иметь описание этих библиотек, но без него невозможно их использование и в рамках той системы программирования, для которой они созданы.

3 Если у вас есть доступ к сети Internet, то на разных ее сайтах можно найти множество готовых решений буквально на все случаи жизни. Только обращайте внимание на даты разработки программ и подпрограмм. Многие из них морально устарели и при использовании новых инструкций микропроцессоров могут быть существенно упрощены.

4. В процессе работы с ассемблером у вас постепенно будет накапливаться набор собственных решений, которые пригодятся при новых разработках. Для удобства их последующего использования старайтесь при написании программ составлять как можно больше подпрограмм, оформленных в виде отдельных модулей, лучше исходных, а не объектных — первые проще модифицировать в случае необходимости. Применение подпрограмм несколько увеличивает размеры задачи и замедляет процесс ее выполнения. Но взамен вы получаете возможность отлаживать большую программу по частям и использовать отлаженные подпрограммы в других своих разработках.

5. Не забывайте о существовании такого эффективного средства как макросы — макроопределения и макровызовы. Они позволяют сокращать не только исходный текст программы, но и количество возможных ошибок при его наборе. Кроме того, макровызовы удобно применять при запросе системных функций, которые выполняют много полезных действий, таких как работа с окнами, с файловой системой, ввод и вывод данных и пр.

Таким образом, при наличии опыта и практических навыков, затраты труда при программировании на ассемблере можно свести к такому уровню, **при котором окажется возможной разработка достаточно больших**

проектов. Безусловно, прежде чем приступать к их реализации, надо взвесить все доводы "за" и "против" и учесть свои реальные возможности.

Когда ассемблеру нет альтернативы

Традиционно ассемблер применяется для написания подпрограмм и вставок в программы, составленные на алгоритмических языках. За последние несколько лет целесообразность такого его применения не только увеличилась, но и приобрела новый смысл. Это объясняется тем, что задачи, получаемые при программировании на алгоритмических языках, крайне неэффективно используют ресурсы современных микропроцессоров. Разумеется, в этом виноваты не языки, а компиляторы, которые не поддерживают работу с групповыми операциями и не учитывают особенности конвейерной обработки. Все старания инженеров и проектировщиков, направленные на повышение производительности микропроцессоров, оказываются бесполезными, как и ваши финансовые затраты на приобретение нового оборудования.

Обвинять в этом разработчиков компиляторов не имеет смысла — проблема не столь тривиальна, и для ее решения нужно не только улучшать технику компиляции, но и разрабатывать принципиально новые языковые средства. Мы привыкли к тому, что арифметические и логические операции выполняются над одной парой операндов, иногда над одним операндом (смена знака или отрицание). А групповые операции современных микропроцессоров (не только семейства Pentium) могут оперировать сразу с несколькими парами операндов, количество и разрядность которых не фиксированы.

В этой ситуации альтернативы ассемблеру просто не существует. Недаром в технической документации к компилятору "Intel C++ Compiler" рекомендуются три варианта работы с новыми командами:

- прямые вставки ассемблерного текста в тело программы, написанной на Си;
- составление процедур (подпрограмм) на языке ассемблера;
- использование Intrinsic (встроенных функций).

Последние являются альтернативной формой записи групповых операций и, по существу, не упрощают (а скорее усложняют) задачу программиста.

Кроме введения групповых операций, для повышения производительности современных микропроцессоров применяется конвейерная обработка инструкций. В целях расширения ее возможностей непрерывно усложняется микроархитектура, что неизбежно увеличивает стоимость изделий.

Использовать преимущества конвейерной обработки не так просто, как это может показаться. При составлении программы, неважно на каком языке, мы исходим из предположения о том, что машина сначала выполняет одну команду, и только потом приступает к выполнению другой. На самом деле современные процессоры уже давно так не работают — они пытаются полностью или частично совместить выполнение нескольких команд. Очевидно, что обычные задачи не приспособлены для этого, и при их выполнении производительность процессора не соответствует его реальным возможностям.

В связи с этим, у старой проблемы оптимизации появились новые аспекты — надо выявлять и изменять те места задачи, в которых производительность процессора занижена, желательно исключить из программы или свести к минимуму условные ветвления (например, по результату сравнения операндов) и т.д. **Решать подобные проблемы в процессе компиляции программы, составленной на алгоритмическом языке, пока невозможно, и мы снова обращаемся к ассемблеру.**

Компания Intel разработала и продает специальный анализатор производительности микропроцессора в процессе выполнения конкретной задачи (VTuneTM Performance Analyzer 6.0). Он позволяет выявить в ней слабые места (критические участки кода задачи), и затем исправить их с помощью ассемблерных вставок.

Подробнее о новых возможностях микропроцессоров вы можете прочитать в моей предыдущей статье [1].

Резюме

В начале статьи я уже писал о том, что в процессе работы программисту время от времени приходится осваивать новые языки. При этом он может преследовать разные цели — от простого удовлетворения любопытства и расширения кругозора до сугубо утилитарных, в виде решения задач, для которых создан новый язык.

Если вы собираетесь расширять свой кругозор, то вспомните высказывание Э.Дейкстры о влиянии языка на способ мышления. Осваивая только алгоритмические языки, вы привыкаете думать при программировании формальными категориями, основанными на синтаксисе и семантике. Я не считаю, что это плохо, скорее наоборот — хорошо, но несколько однобоко. В конце концов, мы имеем дело не с виртуальной, а с реальной техникой, и желательно представлять, как она работает. В этом случае вам будет проще понять, что можно, а что и почему нельзя сделать с помощью средств того или иного алгоритмического языка.

Чтобы оставаться на острие технического прогресса и уметь создавать задачи, рационально использующие возможности самых современных компьютеров, надо в совершенстве владеть искусством программирования ЭВМ. Овладеть этим искусством без знания языка ассемблера, на мой взгляд, невозможно.

Литература

1. П.Соколенко. Pentium глазами программиста. — Радиомир. Ваш компьютер, 2002, №8, 9.

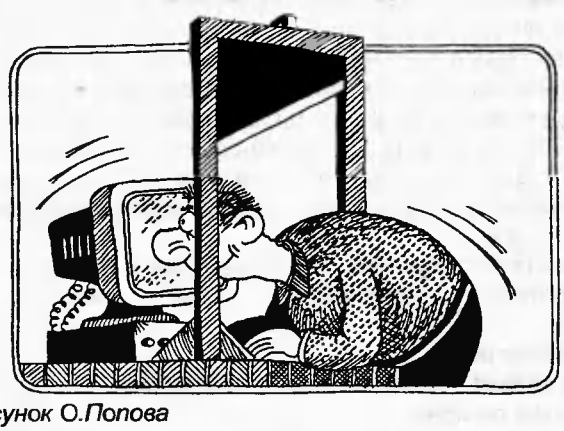


Рисунок О.Попова

А.СНИТКО,
Беларусь, г.Мосты,
E-mail: snitko_alexey@mail.ru.

Работа с непрямоугольными формами в среде Delphi

Давно прошли те времена, когда программы с Windows-интерфейсом были в диковинку, а на их разработку тратилось много сил и времени. Появились мощные интегрированные среды разработки приложений под Windows (Delphi, C++ Builder и др.), в которых на разработку не очень сложного приложения зачастую тратится всего несколько дней. Соответственно возросли и требования пользователей к интерфейсу используемых ими программ. Некоторые пользователи вообще не обращают внимания на приложения со стандартным Windows-интерфейсом. А зря, так как именно среди таких программ встречаются настоящие шедевры, но автор, работая над кодом, не уделил должного внимания красоте интерфейса. Красивые программы, наоборот, в большинстве своем редко могут похвастаться хорошим кодом. Пример тому — Windows Media Player последних версий.

Чтобы облегчить начинающим программистам разработку интерфейса их приложений, я приведу информацию о создании непрямоугольных форм — основного элемента красивого интерфейса.

Создать непрямоугольную форму гораздо легче, чем это может показаться на первый взгляд. Это под силу даже программисту, который работает в Delphi всего неделю. В WinAPI для этого предусмотрена функция **SetWindowRgn**, которой мы и воспользуемся:

SetWindowRgn(HWND *hWnd,* **HRGN** *hRgn,* **BOOL** *bRedraw*).

Параметры:

- *hWnd* — дескриптор окна, регион которого будет установлен;
- *hRgn* — дескриптор региона;
- *bRedraw* — логический параметр, который определяет, будет ли операционная система перерисовывать окно после установки региона.

Думаю, что лишь немногие из тех, кто программирует на Delphi, знают, что такое регион. В документации к Windows это понятие объясняется так: "Регион — это прямоугольник, многоугольник, эллипс (или комбинация двух или более из перечисленных форм), которые могут быть залиты, зарисованы, перевернуты и использованы для тестирования положения курсора". Для создания регионов в WinAPI предусмотрен целый ряд функций. Коротко расскажем о каждой из них.

CreateRectRgn(int *nLeftRect,* **int** *nTopRect,* **int** *nRightRect,* **int** *nBottomRect*) — создает прямоугольный регион.

Параметры:

- *nLeftRect* — горизонтальная координата верхнего левого угла региона;
- *nTopRect* — вертикальная координата верхнего левого угла региона;
- *nRightRect* — горизонтальная координата нижнего правого угла региона;
- *nBottomRect* — вертикальная координата нижнего правого угла региона.

CreateRectRgnIndirect(CONST RECT **lprc*) — создает прямоугольный регион, определенный структурой RECT.

Параметр *lprc* — указывает структуру RECT, которая включает координаты верхнего левого и нижнего правого углов прямоугольника, который определяет регион.

Функция **CreateRectRgnIndirect** полностью аналогична функции **CreateRectRgn**, за исключением того, что в **CreateRectRgnIndirect** координаты углов прямоугольника вынесены в структуру RECT.

CreateRoundRectRgn(int *nLeftRect,* **int** *nTopRect,* **int** *nRightRect,* **int** *nBottomRect,* **int** *nWidthEllipse,* **int** *nHeightEllipse*) — создает прямоугольный регион с закругленными углами.

Параметры:

- *nLeftRect* — горизонтальная координата верхнего левого угла региона;
- *nTopRect* — вертикальная координата верхнего левого угла региона;
- *nRightRect* — горизонтальная координата нижнего правого угла региона;
- *nBottomRect* — вертикальная координата нижнего правого угла региона;
- *nWidthEllipse* — ширина эллипса, используемого для создания закругленных углов;
- *nHeightEllipse* — высота эллипса, используемого для создания закругленных углов.

CreateEllipticRgn(int *nLeftRect,* **int** *nTopRect,* **int** *nRightRect* **Int** *nBottomRect*) — создает эллиптический регион.

Параметры:

- *nLeftRect* — горизонтальная координата верхнего левого угла прямоугольника, ограничивающего эллипс;
- *nTopRect* — вертикальная координата верхнего левого угла прямоугольника, ограничивающего эллипс;
- *nRightRect* — горизонтальная координата нижнего правого угла прямоугольника, ограничивающего эллипс;
- *nBottomRect* — вертикальная координата нижнего правого угла прямоугольника, ограничивающего эллипс.

CreateEllIpticRgnIndirect(CONST RECT **lprc*) — создает эллиптический регион, определенный структурой RECT.

Параметр *lprc* указывает структуру RECT, которая включает координаты верхнего левого и нижнего правого углов прямоугольника, ограничивающего эллипс.

CreatePolygonRgn(CONST POINT **lppt*,**Int** *cPoints,* **int** *fnPolyFillMode*) — создает многоугольный регион.

Параметры:

- *lppt* — указывает массив структур POINT, которые определяют вершины многоугольника;
- *cPoints* — определяет число структур POINT в массиве;
- *fnPolyFillMode* — устанавливает режим заливки многоугольника (ALTERNATE или WINDING).

CreatePolyPolygonRgn(CONST POINT **lppt,* **CONST INT** **lpPolyCounts,* **int** *nCount,* **Int** *fnPolyFillMode*) — создает регион, включающий серию многоугольников. Многоугольники могут пересекаться.

Параметры:

- *lppt* — указывает массив структур POINT, которые определяют вершины многоугольника. Многоугольники определены последовательно. Предполагается, что каждый многоугольник замкнут, и каждая вершина определена только один раз;
- *lpPolyCounts* — указывает массив integer-чисел, каждое из которых определяет число структур POINT в одном из многоугольников массива, определенного в *lppt*;
- *nCount* — определяет общее число integer-чисел в массиве, указанном *lpPolyCounts*;
- *fnPolyFillMode* — устанавливает режим заливки многоугольника (ALTERNATE или WINDING).

Я дал очень краткую информацию о функциях, предназначенных для создания регионов. Она имеет чисто ознакомительный характер. Более полную информацию можно найти в справочной системе Win32 SDK Reference, которая входит в поставку Delphi [2].

Естественно, для создания более или менее красивого приложения, одного региона будет недостаточно. Необходимо использовать комбинацию из нескольких регионов. В WinAPI есть функция **CombineRgn,** которая комбинирует два региона и выдает результат в виде третьего региона:

CombineRgn(HRGN *hrgnDest,* **HRGN** *hrgnSrc1,* **HRGN** *hrgnSrc2,* **int** *fnCombineMode*).

Параметры:

- *hrgnDest* — устанавливает новый регион с размерами, определенными комбинированием двух других регионов;
- *hrgnSrc1* — устанавливает первый из двух регионов, которые должны быть скомбинированы;
- *hrgnSrc2* — устанавливает второй из двух регионов, которые должны быть скомбинированы;
- *fnCombineMode* — определяет режим комбинации двух регионов.

Параметр *fnCombineMod* может принимать следующие значения:

- RGN_AND — создается регион, который является пересечением двух комбинируемых регионов;
- RGN_COPY — создается регион, который является копией региона *hrgnSrc1*;
- RGN_DIFF — создается регион, который является частью региона *hrgnSrc1*, но не является частью региона *hrgnSrc2*;
- RGN_OR — создается регион, который является объединением двух комбинируемых регионов;
- RGN_XOR — создается регион, который является объединением двух комбинируемых регионов, исключая области их пересечения.

Для работы данной функции достаточно двух регионов. Параметр *hrgnDest* может быть равен *hrgnSrc1* или *hrgnSrc2*, если сохранение одного из начальных регионов не обязательно. Это один из шагов на пути к оптимизации кода.

От теории перейдем к практике. То, что в теории не вызывает затруднений, на практике может оказаться невыполнимым. Поэтому я детально опишу создание непрямоугольной формы применительно к среде Delphi.

Создадим в Delphi новое приложение. В процедуру обработки события формы OnCreate вставим код, который отвечает за создание региона и присвоение его форме приложения. Свойству формы *BorderStyle* придаем значение *bsNone*.

Для того чтобы форму можно было "таскать" за любую из ее точек, в исходный код необходимо вставить небольшой программный "изворот". Кто его автор, я не знаю, но данный способ очень эффективен. В исходнике этот кусочек выделен жирным шрифтом.

Для придания форме красивого вида можно свойству формы Color придать соответствующее значение. Чтобы сделать форму исключительно красивой, можно поместить на нее картинку в формате BMP. Сделать это можно, применяя соответствующие компоненты независимых производителей, или средствами WinAPI.

Далее на форму устанавливаем необходимые компоненты — в общем, делаем то, что делали, разрабатывая обычные приложения. Особенно красивый интерфейс получается при применении BMPButton-компонентов (кнопок на базе BMP-картинок).

Я не рекомендую применять еще какие-либо компоненты. Это раздувает размер программы до астрономических размеров. Мало кому захочется скачивать такую вашу программу из сети. Да и программистский подход при этом совершенно теряется. Лучше поищите необходимые функции в WinAPI.

Для демонстрации возможностей стандартных функций, предназначенных для работы с регионами, я приведу изображение моей небольшой, еще не законченной программки ThunderPlayer. Вы можете сами убедиться, что описанные методы работы с регионами работают:

Вот пример исходного кода программы с применением регионов:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
    procedure WMLButtonDown(var Msg: TMessage); message
WM_LBUTTONDOWN;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  hrgn1,hrgn2:HRGN;

implementation

{$R *.DFM}
```

```
procedure TForm1.WMLButtonDown(var Msg: TMessage);
begin
  Perform(WM_NCLBUTTONDOWN, HTCAPTION, Msg.LParam);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  hrgn1:=CreateRoundRectRgn(5,40,300,130,20,20);
  hrgn2:=CreateEllipticRgn(5,5,300,83);
  CombineRgn(hrgn1,hrgn1,hrgn2,RGN_OR);
  SetWindowRgn(Form1.Handle,hrgn1,False);
end;

end.
```

В заключение приведу текст очень полезной функции, которая создает регион на базе растровой BMP-картинки. Вы просто задаете прозрачный цвет, и точки с этим цветом не будут входить в регион. Кто автор этой процедуры, не знаю. Но думаю, он не против ознакомления с ней широкой программистской общественности. Эта функция довольно проста, и ее понимание не должно вызвать затруднений.

```
function BitmapToRegion(Bitmap:TBitmap;TransColor:TColor):HRGN;
var
    X,Y:Integer;
    XStart:Integer;
begin
    Result:=0;
    with Bitmap do
       for Y := 0 to Height - 1 do
          begin
             X := 0;
             while X < Width do
                begin
                   while (X < Width) and (Canvas.Pixels[X, Y]
                                                = TransColor) do
                      Inc(X);
                      if  X >= Width then
                         Break;
                      XStart := X;
                      while (X < Width) and
                      (Canvas.Pixels[X, Y] <> TransColor) do
                         Inc(X);
                      if Result = 0 then
                        Result := CreateRectRgn(XStart, Y,
                                                   X, Y + 1)
                      Else
                         CombineRgn(Result,Result,
                  CreateRectRgn (XStart, Y, X, Y + 1), RGN_OR);
                end;
          end;
end;
```

На мой взгляд, материала, приведенного в статье, достаточно, чтобы создавать очень красивые приложения на Delphi. Нужно только уметь его применять и не бояться экспериментировать. Если у кого-нибудь возникнут вопросы по этой статье, я с удовольствием отвечу на них.

Возможно, кому-то эта статья покажется всего лишь адаптированной под Delphi информацией из [1]. Но это совершенно не так. Майский номер "Радиомир. Ваш компьютер" со статьей А.Корбита, в которой рассказывалось о реализации непрямоугольных окон на C++, пришел ко мне, когда статья уже была готова к отсылке.

Литература

1. А.Корбит. Боремся с однообразием — круглое окно!!! — Радиомир. Ваш компьютер, 2002, №5, С.34.

2. Win32 SDK Reference – Help files for Delphi

В.ЗУБКО /HI-TECH,
E-mail: vzubko@mail.ru
WWW: http://hi-tech.nsys.by

Уязвимость в защите Excel

Как устроена защита файлов в Excel?

Коротко о том, как реализована защита паролем файлов в Excel.

При сохранении файла с паролем, Excel шифрует содержимое файла и создает своеобразный хеш-блок, по которому проверяется пароль. Хеш-блок всегда находится по абсолютному смещению в файле (0222h) и представляет собой группу из 48 байтов, логически разделяемую программой на три равных блока по 16 байтов. При этом алгоритм содержит такие известные криптоалгоритмы как RC4 и хеш-функцию MD5. Назовем три последовательных блока по 16 байтов — buff1, buff2 и buff3. Тогда приблизительный алгоритм программы при проверке пароля может быть таким:

- прочесть первые 16 байтов из 48 (buff1);
- получить MD5-хеш от пароля;
- от этого хеша и первых 16 байтов образовать новый MD5-хеш, фактически от ("MD5(пароль) + 16 байтов" x 16);
- еще раз (!) от полученного хеша и пароля образовать новый MD5-хеш;
- полученный хеш использовать как пароль для формирования таблицы подстановок RC4;
- прочесть вторые 16 байтов из 48 (buff2);
- выполнить RC4-расшифровку этих 16 байтов, таблица подстановок уже получена;
- прочесть третьи 16 байтов из 48 (buff3);
- выполнить RC4-расшифровку и этих 16 байтов;
- от уже расшифрованных RC4 двух блоков по 16 байтов получить MD5-хеш;
- и только сейчас сравнить его с расшифрованным по RC4 третьим блоком из 16 байтов!

Где-то внутри этого алгоритма также происходит формирование буфера, необходимого для расшифровки самого файла — например, той же таблицы подстановок для RC4.

При этом алгоритм использует следующие подпрограммы (версия MS Office 97):

- код по адресу 309AD389 представляет собой подпрограмму инициализации таблицы подстановок RC4 (передается адрес пароля, его длина и адрес инициализируемой таблицы подстановок);
- код по адресу 309AD416 представляет собой подпрограмму шифрации блока данных по алгоритму RC4 (передается адрес сформированной таблицы подстановок, адрес и длина шифруемого блока);
- процедуры по адресам 309AD4D2, 309AD4FC и 309A5A9 представляют собой инициализацию контекста MD5, добавление к контексту буфера данных и формирование дайджеста (хеша) MD5.

Сам код проверки пароля содержится в процедуре по адресу 308B774C, которая получает адрес пароля (второе слово в стеке) в Unicode. Таким образом, для написания простого подборщика паролей к файлам Excel необходимо воссоздать оригинальный код процедуры 308B774C со всеми подпрограммами, из нее вызываемыми (порядка десяти), и в цикле вызывать его, передавая ему перебираемые пароли.

Время выполнения проверки одного пароля на Pentium120 в среде Windows 98 составляет в среднем около 800 мкс. Т.е для проверки всех вариантов паролей длиной 2 символа необходимо примерно:

100x100x0,0008 = 8 секунд!

Более подробно о защите Excel вы можете прочитать в моей статье "Защита файлов в Excel" [1]. Там вы сможете найти и имитацию оригинального кода процедуры проверки пароля (подпрограммы 308B774C).

А как же без "дырок"?

Итак, мы создали простой код, подбирающий пароли к файлам Excel методом "грубой силы". Очевидно, что перебор всех паролей достаточно "серьезной" длины (например, 10 символов) займет огромное время. Попытаемся проанализировать оригинальный код Excel на предмет "багов", наличие которых может существенно сократить время подбора пароля.

Обратим внимание на тот факт, что сам пароль используется в процедуре проверки ***всего один раз*** — в подпрограмме 308B75D5. Сначала всего лишь вычисляется MD5-хеш от пароля:

```
; Подпрограмма 308B75D5
P308B75D5 proc
     push ebp
     mov ebp,esp
     sub esp,68h      ; Выделить локальный буфер в 68h=104 байта
     push ebx         ; 10h
     push esi
     push edi
     lea ecx,[ebp-68h]        ; Адрес локального буфера в 68h
     call P309AD4D2           ; MD5Init
     mov eax,[ebp+0Ch]
     movzx ecx,word ptr [eax] ; ECX=5, если пароль длиной 5
     shl ecx,1                ; x 2
     push ecx
     lea edx,[eax+02]         ; <- АДРЕС ПАРОЛЯ
     lea ecx,[ebp-68h]        ; Адрес локального буфера в 68h
     call P309AD4FC           ; MD5Update
     lea ecx,[ebp-68h]        ; Адрес лок.контекста[0062B958]
     call P309AD5A9           ; MD5Final
```

Назовем этот хеш hp.

А что происходит дальше? А дальше — самое интересное. Программа формирует следующий хеш, но только от 5 байтов первого, hp:

```
     lea ecx,[ebp-68h]
     call P309AD4D2          ; MD5Init
     mov ebx,[ebp+08]
     push 10h
     lea esi,[ebx+116h]
     pop edi
L308B7616:
     push 5          ; Длина буфера для MD5Update РАВНА ПЯТИ!!
     lea edx,[ebp-10h]
     lea ecx,[ebp-68h]
     call P309AD4FC          ; MD5Update
     push 10h                ; Длина буфера для MD5Update
     mov edx,esi             ; buff2 - буфер для MD5Update
     lea ecx,[ebp-68h]
     call P309AD4FC          ; MD5Update
     dec edi                 ; 10h->0Fh->...
     jnz L308B7616           ; Вызывать 10h раз MD5Update
     lea ecx,[ebp-68h]
     call P309AD5A9          ; MD5Final
```

В первом приближении, дальнейший код зависит ***только*** от этих пяти байтов hp! Да, действительно, в начале формируется 16-байтный хеш hp от пароля, что никак не ослабляет мощность введенного пароля, но вот его усечение до пяти символов — это уже что-то. Давайте проверим эту гипотезу. Создадим некий файл, защищенный паролем "1". Откроем его, установив брейкпойнт в точке 308B7616:

```
bpx cs:308B7616
```

Только что сформированый хеш hp находится по адресу [ebp-10h]:

```
d ss:ebp-10
```

И мы видим первые пять байтов хеша от пароля "1":

```
06 D4 96 32 C9 ...
```

Запомним их и продолжим выполнение программы. Естественно, если пароль был верным, файл нормально откроется. А теперь соль эксперимента — откроем еще раз этот же файл, только вот вместо правильного пароля введем какую-нибудь чепуху. Опять попадем в отладчик в той же точке 308B7616 и заменим первые пять байтов нового хеша (видно, что он отличается от прежнего) на те самые первые пять байтов от оригинального пароля (data 0 — заменить пять байтов по адресу [ebp-10h]). И что мы видим? Файл открылся как ни в чем не бывало!

Итак, мы нашли очевидный баг. Для подбора пароля, оказывается, нет нужды перебирать все пароли длиной от 1 до 15 символов, достаточно перебрать первые 5 байтов хеша hp! А это уже совсем другое число вариантов — $256^5 \approx 10^{12}$! Теперь алгоритм перебора может быть переписан в следующем виде — вызываем нашу процедуру обработки пароля, передаем ей какой-нибудь пароль (произвольный!), но дописываем небольшой код, подменяющий создаваемые оригинальным кодом первые пять байтов hp на перебираемые значения:

```
.data
; Тестовый "пароль"
TestFiveBytes db 000h,000h,000h,000h,000h

@@FindPassword:
; ...
  push 0826BB25Ch
  push offset Password
  push offset LocBuffer
  call P308B774C                ; Вызов главного кода
  jz @@PasswordFound
  xor ebx,ebx
@@NextLetter:
  inc byte ptr TestFiveBytes[ebx]
  jnz @@FindPassword
  inc ebx
  cmp ebx,MaxPassLen
  jb  @@NextLetter
; ...
  call P309AD5A9        ; MD5Final
; Подставляем переборочный хеш, причем ТОЛЬКО 5 байтов!!!
  pushad
  lea edi,[ebp-10h]     ; <-62B9B0 - UN1- адрес буфера для MD5Update
  mov esi,offset TestFiveBytes
  mov ecx,5
  cld
  rep movsb
  popad
; ...
  pop edi
L308B7616:
```

Резюме

Оказывается, для расшифровки файла Excel надо перебирать "всего" 256^5 вариантов первых пяти байтов хеша hp. Много это или мало? Во-первых, существенно меньше первичной реализации "тупого" подбирателя любых возможных паролей, если, конечно, не принимать в расчет словарную атаку, которая, впрочем, бесполезна против паролей, специально созданных генератором псевдослучайных чисел. Число возможных вариантов пароля составляет примерно 10^{12}. При скорости подбора 10000 хешей в секунду для подбора пароля необходимо 10^8 секунд, что составляет примерно 40 месяцев. Это не так уж и много для особо "желающих", и, что самое главное — это вполне обозримое время.

Если подключить к делу сотни мощных компьютеров, то подбор может занять вообще несколько дней, что не говорит в пользу защиты Excel.

Следует отметить, что, во-первых, сам пароль можно будет найти после нахождения верного хеша путем атаки на MD5 перебором и сравнением первых пяти байтов получаемых хешей с найденными ранее. А, во-вторых, потенциальное наличие в алгоритме защиты ***дублирующихся*** паролей (с вероятностью примерно 10^{-5}), т.е. пароли, у которых первые пять байтов MD5-хеша равны, для Excel будут эквиваленты!

Литература

1. В.Зубко /Hi-Tech. Защита файлов в Excel. — Радиомир. Ваш компьютер, 2002, №7, С.28.

О.ВАЛЬПА,
E-mail: sandh@narod.ru

Работа с графикой

Графические возможности современных компьютеров достигли отличных показателей. На сегодняшний день компьютеры позволяют отображать рисунки с высоким разрешением и огромным количеством цветовых оттенков. Использование графики делает программы более привлекательными и наглядными. И хотя сегодня существуют мощные визуальные компиляторы для создания программ в среде Windows, начиналось все с простых графических функций в операционной среде MS DOS. Именно с азами графических программ я и хочу познакомить начинающих программистов.

Вниманию читателей предлагается программа "Draw", написанная на языке программирования C++. Эта программа позволит вам познакомиться с некоторыми основными графическими функциями и их применением в ваших программах. Кроме того, программа сама по себе является законченным графическим приложением, позволяющим создавать на экране компьютера простые рисунки. Она также включает в себя функции драйвера мыши, что позволило сделать процесс рисования на экране очень удобным.

Для освоения процесса создания графических программ и изучения некоторых графических функций, ниже приводится полный текст этой программы.

Текст программы "Draw"

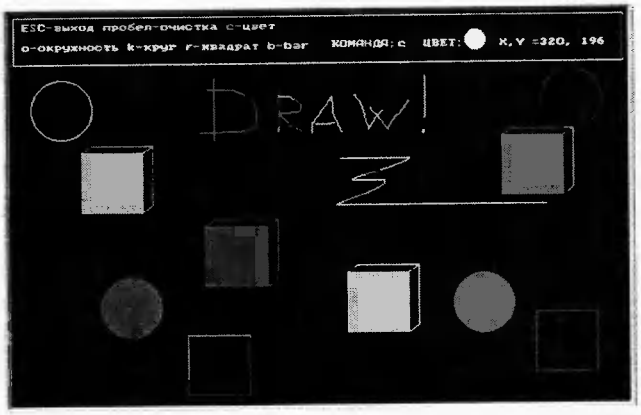
```
//============================================================
// Программа демонстрации работы в графическом режиме
//
// Файл:draw.cpp
// Дата:05.04.2002
// Автор: Вальпа Олег
//
============================================================

// Подключаемые библиотеки
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>

// Координаты сообщений
#define   XKM   400     //Координаты команд
#define   YKM   30
#define   XCL   480     //Координаты окружностей
#define   YCL   30
#define   XKR   550     //Координаты курсора
#define   YKR   30

// Описатели функций
void dr_rect (int x, int y, int xm, int ym, int c);
void dis_cur (void);
void ena_cur (void);

// Константы
// Скан-коды клавиатуры
#define   ESC    0x011b   // Esc

// Начало программы
int main(void)
{
// Инициализация переменных
int gdriver = DETECT, gmode, errorcode, i,j;
int x, y, mx, my, sx, sy, c, ret, bt, klav, mklav;
int xmax, ymax;
int p[8];

char curandxor[32][16]={
// Маска "И" указателя мыши
"00000000 0000000",
"0000000 0 000000",
"000000 000 00000",
"00000 0 0 0 0000",
"000  0000000  00",
"00 0000 0 0000 0",
"000    000    00",
"00000 00000 0000",
"00000 00000 0000",
"00000 00000 0000",
"00000 00000 0000",
"000000 000 00000",
"0000000   000000",
"00000000 0000000",
"000000000 000000",
"0000000  0000000",

// Маска "Исключающее ИЛИ" указателя мыши
"        0       ",
"       0 0      ",
"      0   0     ",
"     0 0 0 0    ",
"   00       00  ",
"  0    0 0    0 ",
"   0000   0000  ",
"     0     0    ",
"     0     0    ",
"     0     0    ",
"     0     0    ",
"      0   0     ",
"       000      ",
"        0       ",
"         0      ",
"        00      "
};

unsigned int cur[32];
char msg[80];
union REGS regs;
struct SREGS segs;
struct REGPACK regpack;

// Инициализация графики
initgraph(&gdriver, &gmode, "");

// Чтение и обработка результата инициализации
errorcode = graphresult();
if (errorcode != grOk)
   {
   printf("Ошибка инициализации графики: %s\n",
grapherrormsg(errorcode));
   printf("Нажмите любую клавишу...");
   getch();
   exit(1);
   }

// Преобразовать данные для курсора
for(i=0;i<=31;i++)
   {
   cur[i]=0;
   for (j=0;j<=15;j++)
     if( curandxor[i][15-j]=='0') cur[i]=cur[i] | (1<<j);
   }

// Сброс драйвера мыши и запрос состояния
regs.x.ax = 0;
int86(0x33, &regs, &regs);

// Если мышь не инсталлирована - выход с сообщением
if(regs.x.ax == 0)
  {
  cleardevice();
  closegraph();
  puts("\nМышь не инсталлирована\n");
  puts("\nИнсталлируйте ее и вновь запустите программу\n");
  return(1);
  }

// Метка инициализации
metbegin:
ret=0; x=0; y=0; sx=0; sy=0;
```

```
// Очистка экрана
dis_cur ();         // Спрятать указатель
cleardevice();      // Очистить экран
c = WHITE;          // Исходный цвет рисования
setcolor(c);        // Установить этот цвет
xmax = getmaxx();   // Получить предельные
ymax = getmaxy();   // координаты экрана

dr_rect (0,0,xmax,ymax,WHITE);          // Рамка основная
dr_rect (5,5,xmax-10,ymax*0.1,WHITE);   // Рамка подсказки
circle(XCL,YCL,10);                     // Рамка цвета

ena_cur ();             // Показать указатель
// Установить тип и форму указателя мыши
regs.x.ax = 0x9;
regs.x.bx = 7;          // Столбец прицела
regs.x.cx = 0;          // Строка  прицела
regs.x.dx = FP_OFF(cur);
segs.es   = FP_SEG(cur);
int86x(0x33,&regs,&regs,&segs);

// Установить границы по горизонтали
regs.x.ax = 0x7;
regs.x.cx = 31; //min x
regs.x.dx = xmax-41;  //max x
int86(0x33,&regs,&regs);

// Установить границы по вертикали
regs.x.ax = 0x8;
regs.x.cx = 90;  //min y
regs.x.dx = ymax-31; //max y
int86(0x33,&regs,&regs);

// Отобразить указатель на экране
ena_cur ();

// Отобразить справку на экране
outtextxy(xmax/30, ymax/40, "ESC-выход пробел-очистка c-цвет f-красить");
outtextxy(xmax/30, ymax/40+12, "o-окружность k-круг r-квадрат b-bar");
outtextxy(xmax/30, ymax/40+24, "d-треуг. a-дуга e-эллипс p-сектор");
outtextxy(XKM-65,YKM,"КОМАНДА:");
outtextxy(XCL-50,YKM,"ЦВЕТ:");
outtextxy(XKR-40,YKR,"X,Y =");

// Метка главного цикла
metl:
   moveto(x,y);           // Установить указатель в позицию x,y
   if( bioskey(1) != 0 ) // Если нажата клавиша
     klav=bioskey(0);     // Читаем код клавиши
   else klav=0;           // Иначе - пустой код клавиши

// Вывод полученной команды
   if(mklav != klav && klav != 0)
    {
    mx=x; my=y;
    setcolor(BLACK);
    sprintf(msg,"%c",mklav);
    outtextxy(XKM,YKM,msg);
    mklav=klav;
    setcolor(getmaxcolor());
    sprintf(msg,"%c",mklav);
    outtextxy(XKM,YKM,msg);
    x=mx; y=my; moveto(x, y);
    setcolor(c);
    }

// Получить положение и состояние мыши
   regs.x.ax=3;
   int86(0x33, &regs, &regs);
   bt=regs.x.bx;
   x=regs.x.cx;
   y=regs.x.dx;

// Если нажата любая кнопка мыши
   if(bt != 0)
     {
     dis_cur ();
     lineto(x, y);  // Рисуем линию
     ena_cur ();
     }

// Вывод координат указателя мыши если они изменились
   if((sx != x) || (sy != y)) //getx(), gety());
     {
     mx=x; my=y;
     setcolor(BLACK);
     sprintf(msg,"%d, %d", sx, sy);
     outtextxy(XKR,YKR,msg);
     sx=x; sy=y;
     setcolor(getmaxcolor());
     sprintf(msg,"%d, %d", sx, sy);
     outtextxy(XKR,YKR,msg);
     x=mx; y=my; moveto(x, y);
     setcolor(c);
     }

// Обработка команд

  switch (klav & 0xff)
    {
    case 'c': dis_cur (); c++; setcolor(c);
              circle(XCL,YCL,10);
              setfillstyle(SOLID_FILL,c);
              floodfill(XCL,YCL,c);
              ena_cur ();break;
    case 'a': dis_cur (); arc(x, y, 0, 180, 30); ena_cur ();break;
    case 'p': dis_cur (); pieslice(x, y, 0, 180, 30); ena_cur ();break;
    case 'd': p[0]=x-30; p[1]=y-30; p[2]=x+30; p[3]=y-30;
              p[4]=x-0; p[5]=y+30; p[6]=x-30; p[7]=y-30;
              dis_cur (); drawpoly(4, p); ena_cur ();break;
    case 'e': dis_cur (); ellipse(x, y, 0, 360, 30, 15); ena_cur ();break;
    case 'r': dis_cur (); dr_rect(x-30,y-30,60,60,c); ena_cur ();break;
    case 'o': dis_cur (); circle(x,y,30); ena_cur ();break;
    case 'b': dis_cur (); bar3d(x-30, y-30, x+30, y+30,10,5);
              ena_cur ();break;
    case 'k': dis_cur (); circle(x,y,30); setfillstyle(SOLID_FILL,c);
              floodfill(x,y,c); ena_cur ();break;
    case 'f': dis_cur (); setfillstyle(SOLID_FILL,c);
             floodfill(x,y,c); ena_cur ();break;
    case ' ': ret=1; break;
  }

  if (ret==1) goto metbegin;  // Новый рисунок
  if(klav != ESC) goto metl;  // Продолжить цикл если не выходим

  dis_cur();        // Погасить указатель мыши
  closegraph();     // Выключить графику
  return 0;         // Выход из программы
}

// Подпрограмма рисования цветного прямоугольника
void dr_rect (int x, int y, int xm, int ym, int c)
{
    setcolor( c );
    moveto(x,y);
    lineto(x+xm,y);
    lineto(x+xm,y+ym);
    lineto(x,y+ym);
    lineto(x,y);
}

// Подпрограмма гашения указателя мыши
void dis_cur (void)
{
   union REGS regspp;
   regspp.x.ax=2;
   int86(0x33,&regspp,&regspp);
}

// Подпрограмма отображения курсора на экране
void ena_cur (void)
{
   union REGS regspp;
   regspp.x.ax = 1;
   int86(0x33, &regspp, &regspp);
}
```

После запуска программы на экране появится поле для рисования и справочная информация. В центре экрана будет отображаться указатель мыши в виде стилизованного изображения маленькой мышки. Нажав любую клавишу мыши, вы можете, перемещая указатель, нарисовать любую линию. Используя команды для рисования, можно выбирать цвет и рисовать различные фигуры. На рисунке приведена копия рабочего экрана программы.

Табл. 1

Функция	Описание
arc	Рисует дугу окружности
circle	Рисует окружность
drawpoly	Рисует контур многоугольника
eiiipse	Рисует эллиптическую дугу
iine	Рисует линию из точки (x0,y0) в (x1,y1)
iinerei	Рисует линию в точку, задаввемую относительным рвсстоянием от текущей позиции (CP)
iineto	Рисует линию из текущей позиции (CP) в точку (x,y)
moveto	Перемещает текущую позицию (CP) в точку (x,y)
moverei	Перемещает текущую позицию (CP) на относительное расстояние
rectangle	Рисует прямоугольник
setlinestyle	Устанавливает толщину и тип текущей линии
bar	Рисует и закрашивает столбец
bar3d	Рисует и звкрашивает трехмерный столбец
filleilipse	Рисует и закрашивает эллипс
filipoly	Рисует и закрашивает многоугольник
plesiice	Рисует и закрашивает сектор окружности
sector	Рисует и закрашивает эллиптический сектор
setfiiipattern	Выбирает шаблон закраски, определяемый пользователем
setfiilstyle	Устанавливает шаблон и цвет закраски
cleardevice	Очищает экран (активную страницу)
setactivepage	Устанавливает активную страницу для графического вывода
setvisualpage	Устанавливает номер видимой графической страницы
clearviewport	Очищает текущее графическое окно
getvlewsettings	Возвращает информацию о текущем графическом окне
setviewport	Устанавливает текущее графическое окно для направления на него графического вывода
getimage	Записывает битовый образ в заданный участок памяти
imagesize	Возвращает число байтов, требуемых для хранения некоторой прямоугольной области экрана
putimage	Помещает на экран ранее записанный в память битовый образ
getpixel	Получает цвет элемента изображения в точке (x,y)
putpixei	Помещает элемент изображения на экран в точку (x,y)
gettextsettings	Возвращает текущий текстовый шрифт, направление, размер и выравнивание
outtext	Посылает строку на экран в текущую позицию (CP)
outtextxy	Посылает текст на экран в заданную позицию
registerbgifont	Регистрирует компонуемый или определяемый пользователем шрифт
settextjustify	Устанавливает значения выравнивания текста, используемые функциями outtext и outtextxy
settextstyle	Устанавливает шрифт, тип и коэффициент увеличения текущего текста
setusercharsize	Устанавливает соотношение между высотой и шириной штриховых шрифтов
textheight	Возвращает высоту строки в элементах изображения
textwidth	Возвращает ширину строки в элементах изображения
getbcolor	Возвращает текущий цвет фона
getcolor	Возвращает текущий цвет вычерчивания
getdefaultpalette	Возвращает структуру определения палитры
getmaxcolor	Возвращает максимальное значение цвета, доступное в текущем графическом режиме
getpalette	Возвращает текущую палитру и ее размер
getpalettesize	Возвращает размер просмотровой таблицы палитры
setallpalette	Изменяет все цвета палитры, как задано
setbkcolor	Устанавливает текущий цвет фона
setcolor	Устанавливает текущий цвет вычерчивания
setpalette	Изменяет один из цветов палитры, как указано ее аргументами



Изменяя матрицы масок "И" и " Исключающее ИЛИ" в программе, вы после перекомпиляции можете измечнить форму указателя мышки. Таким образом можно превратить указатель мыши, например, в карандаш или в кисточку.

Рассмотрим более детально исходный текст программы. Прежде чем использовать в программе функции графики, необходимо подключить графическую библиотеку, которая входит в состав компилятора Borland C++, например, версии 4.0.

Это делается в строке:

```
#include <graphics.h>
```

Для запуска графической системы вызывается функция initgraph. Данная функция загружает графический драйвер и переводит систему в графический режим.

Графические функции находятся в библиотечном файле graphics.lib, а их прототипы — в файле заголовка graphics.h. Кроме этих двух файлов, в состав графического пакета входят драйверы графических устройств (файлы *.bgi) и символьные шрифты (файлы *.chr). Для запуска нашей программы достаточно будет иметь в каталоге, где находится программа, файл egavga.bgi.

Графические функции Borland C++ делятся на несколько категорий:

- функции управления графической системой;
- функции черчения и заполнения;
- функции манипулирования экранами и графическими окнами;
- функции вывода текстов;
- функции управления цветами;
- функции обработки ошибок;
- функции запроса состояния.

Для детального ознакомления со всеми перечисленными выше функциями можно воспользоваться литературой [1]. В табл.1 приводится краткое описание некоторых основных графических функций, в табл.2 — разрешенные для применения в графических программах названия цветов и соответствующие им числовые значения.

Табл. 2

Название цвета	Числовое значение
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGTHCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15

Надеюсь, эта программа послужит для начинающих программистов хорошим стартом в освоении графики.

От редакции: на сайт журнала http://radio-mir.com выложены исходный текст (draw.cpp) и исполняемый модуль (draw.exe) программы, а также файл egavga.bgi.

Литература.

1. М.Вахтеров, С.Орлов. Четвертый BORLAND C++ и его окружение. — М.: Мир, 1994.

Недокументированное сердце Windows

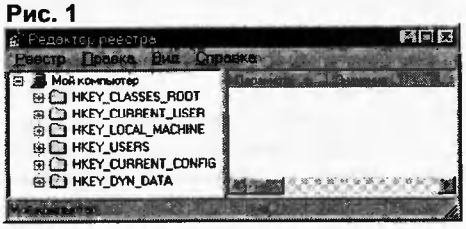
А.ГОРЯЧКИН,
г.Кыштым, Челябинской области.
E-mail: anatoliy_goryach@mail.ru

Что такое системный реестр? Системный реестр (Registry) является базой данных, используемой для хранения установочных параметров и конфигурационных настроек 32-разрядных версий операционных систем Windows 95, 98, Me и NT/2000. В нем содержится основная информация обо всех аппаратных средствах, программном обеспечении, пользователях и т.д. Всякий раз, когда пользователь устанавливает программное обеспечение или "железо", либо делает изменения в установках Панели управления или в файловых ассоциациях, все эти изменения отражаются и сохраняются в системном реестре.

Физически файлы, которые составляют реестр, хранятся на жестком диске в разных местах, в зависимости от версии Windows. В Windows 95, 98 и Me он содержится в двух файлах, которые называются **system.dat** и **user.dat**. Эти файлы, имеющие атрибуты "только для чтения" и "скрытый", размещены в системной директории Windows. В Windows NT/2000 эти файлы размещены отдельно, в директории %SystemRoot%\System32\Config. Эти файлы нельзя просмотреть и отредактировать непосредственно, как обычные файлы. Чтобы внести какие-либо изменения, нужно использовать программу, известную под названием "Редактор реестра".

Чтобы запустить эту программу, достаточно набрать команду **regedit** в меню "Пуск/Выполнить...". С помощью программы Regedit можно получить доступ к многочисленным скрытым возможностям операционных систем Windows 95, 98 и NT. Реестр имеет иерархическую структуру, и хотя она выглядит достаточно сложно, эту структуру можно уподобить структуре корневого директория на жестком диске. Реестр состоит из шести основных разделов или ключей (**рис.1**). Каждый раздел содержит в себе специфическую часть информации, сохраняемую в реестре.

Рис. 1



Сразу хочу обратить внимание пользователей на то, что системный реестр — вещь достаточно сложная, и его модификация требует определенного опыта и понимания. Неумелое и неосторожное обращение с системным реестром может привести к печальным последствиям. Модификация реестра может вызвать серьезные проблемы, которые способны потребовать от пользователя полной реинсталляции операционной системы. Автор статьи не может гарантировать, что проблемы, возникшие в результате модификаций в реестре, могут быть решены. Поэтому используйте информацию, приведенную в статье, на свой собственный страх и риск.

Я хочу привести несколько интересных и полезных, на мой взгляд, советов по использованию скрытых и недокументированных возможностей Windows, и рассказать, какие изменения для этого потребуется внести в системный реестр.

Оптимизация файловой системы (Windows 9x, NT) за счет конфигурации непрерывного размера размещения файлов. Эта установка оптимизирует непрерывный размер размещения файлов в файловой системе и способствует увеличению производительности компьютера. Это особенно полезно при использовании мультимедийных приложений, интенсивно совершающих дисковые операции. С помощью программы Regedit откройте ваш реестр и найдите ключ [HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ FileSystem].

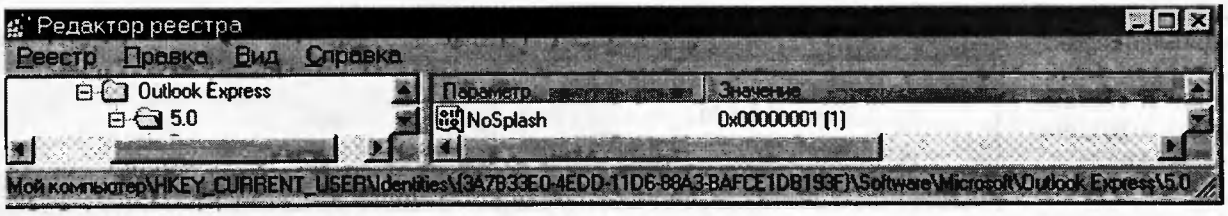
Создайте новый параметр DWORD, назовите его ContigFileAllocSize и установите его значение равным 200 (в шестнадцатеричной системе счисления), как показано на **рис.2**.

Запрет показа Splash Screen в Outlook Express (Windows 9x, NT). Каждый раз при запуске OutlookExpress на экране отображается Splash Screen. В этой заставке указывается название программы, ее версия, фамилия зарегистрированного пользователя и т.д. Можно запретить программе Outlook Express при старте показывать Splash Screen. Запрет заставки позволит ускорить загрузку Outlook Express, что особенно заметно на медленных машинах. Для этого нужно запустить Regedit, и выбрать раздел [HKEY_CURRENT_USER\Identities\{** Identity ID **}\Software\Microsoft\Outlook Express\5.0], где {** Identity ID **} — ваш системный идентификационный номер. Создайте новый параметр DWORD, назовите его NoSplash и установите величину его значения равным 1 (в шестнадцатеричной системе счисления!), как показано на **рис.3**.

Рис. 2

Обновление Windows без регистрации (Windows 98). При первом посещении узла обновления Windows (Windows Update), пользователю предлагается зарегистрировать свою копию операционной системы Windows. От-

Рис. 3



каз от регистрации делает функцию Windows Update недоступной. Но это вовсе не означает, что наличие регистрации проверяется по некой базе пользователей, хранящейся в Microsoft. На самом деле информация о том, зарегистрирована ли копия пользователя, хранится в реестре самого пользователя. Нужно запустить Regedit и выбрать раздел [HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion].

В этом разделе нужно найти параметр RegDone и изменить его значение, сделав его равным единице (**рис.4**). В результате этого пользователь сможет получить доступ к узлу обновления без регистрации.

Рис. 4

Изменение пути к дистрибутиву Windows (Windows 9x). Иногда возникает ситуация, когда нужно установить дополнительные компоненты Windows, а путь к инсталляционным файлам Windows изменился по каким-либо причинам. Чтобы операционная система "правильно" обращалась по нужному пути, нужно в реестре найти раздел [HKEY_LOCAL_MACHINE\ Software\Microsoft\Windows\CurrentVersion\Setup] и изменить параметр "SourcePath" с учетом произошедших изменений (**рис.5**).

Включение функции DoubleClick в двухкнопочной мыши (Windows 9x). Если вы пользуетесь двухкнопочной мышью Genius EasyMouse с интерфейсом PS/2, то можно добавить функцию DoubleClick, которая будет срабатывать при одновременном нажатии левой и правой кнопок мыши. Для этого нужно сменить стандартный майкрософтовский драйвер мыши, который устанавливается по умолчанию, на драйвер мыши Logitech с интерфейсом PS/2. После этого в системном реестре, в разделе [HKEY_LOCAL_MACHINE\ Software\Logitech\MouseWare\ CurrentVersion\ PS2\0000] в параметре DoubleClick прописать значение "0011", как показано на **рис.6**.

Выгрузка файлов библиотек DLL из памяти (Windows 9x, NT). Проводник Windows обычно кэширует файлы библиотек DLL, сохраняя их в оперативной памяти даже тогда, когда приложения, использующие их, уже закрыты. Это может снизить производительность и вызвать проблему нехватки "оперативки" в системах с небольшим объемом памяти. Можно запретить Windows удерживать в памяти неиспользуемые DLL-файлы. Для этого нужно в разделе [HKEY_LOCAL_MACHINE\ Software\Microsoft\Windows\CurrentVersion\Explorer] создать новый раздел, назвать его AlwaysUnloadDll и присвоить в этом разделе параметру по умолчанию значение, равное 1 (**рис.7**).

Примечание. При работе с программой "Редактор реестра" удобно пользоваться строкой состояния, расположенной по нижней границе рабочего окна программы. Если она отсутствует, ее можно вывести командой "строка состояния" в меню "Вид".

Нужно учесть, что после внесения любых изменений в системный реестр, необходимо каждый раз производить перезагрузку компьютера, для того чтобы эти изменения вступили в силу. Не стоит вносить в системный реестр несколько изменений за один раз, лучше всего производить изменения по одному.

Все советы, приведенные в этой статье, проверены и опробованы автором в среде Windows 98 SE.

Рис. 5

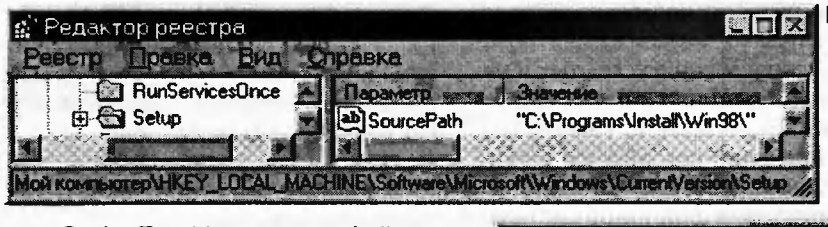


Рис. 6

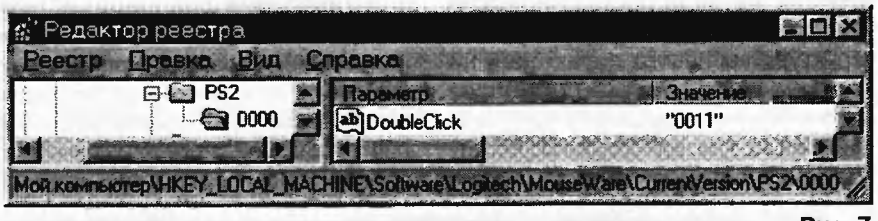
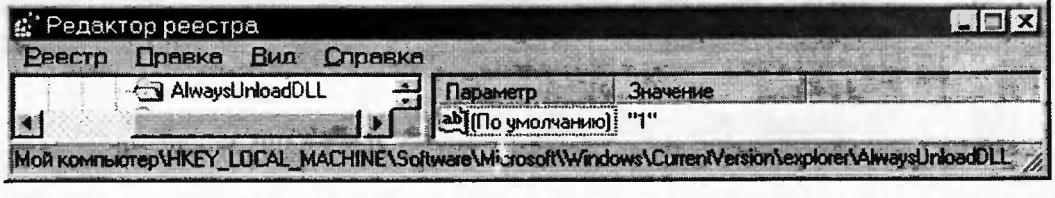


Рис. 7



Полезные Web-ресурсы

Исследованию системного реестра посвящены целые сайты, например, http://registry.winguides.com. На этом сайте можно найти программу Windows Registry Guide, которая после инсталляции представляет собой откомпилированный HTML-файл. В нем содержатся расширенные сведения о системном реестре Windows и разнообразные полезные советы по его модификации (правда, все на английском языке!). Также советую ознакомиться с документом "Руководство по использованию системного реестра Windows". Автор этого документа — Simon Clausen (http://www.regedit.com). Русский перевод сделал Андрей Зенченко (http://members.xoom.com/_vaz). Можно также ознакомиться со статьей "Тайны системного реестра" из журнала "Подводная лодка" (http://www.submarine.ru/article.cfm?id=408). Хороших книжек о системном реестре я встречал мало, да и стоят они сейчас недешево, проще всего "нарыть" интересующую информацию в Интернете.

Заключение

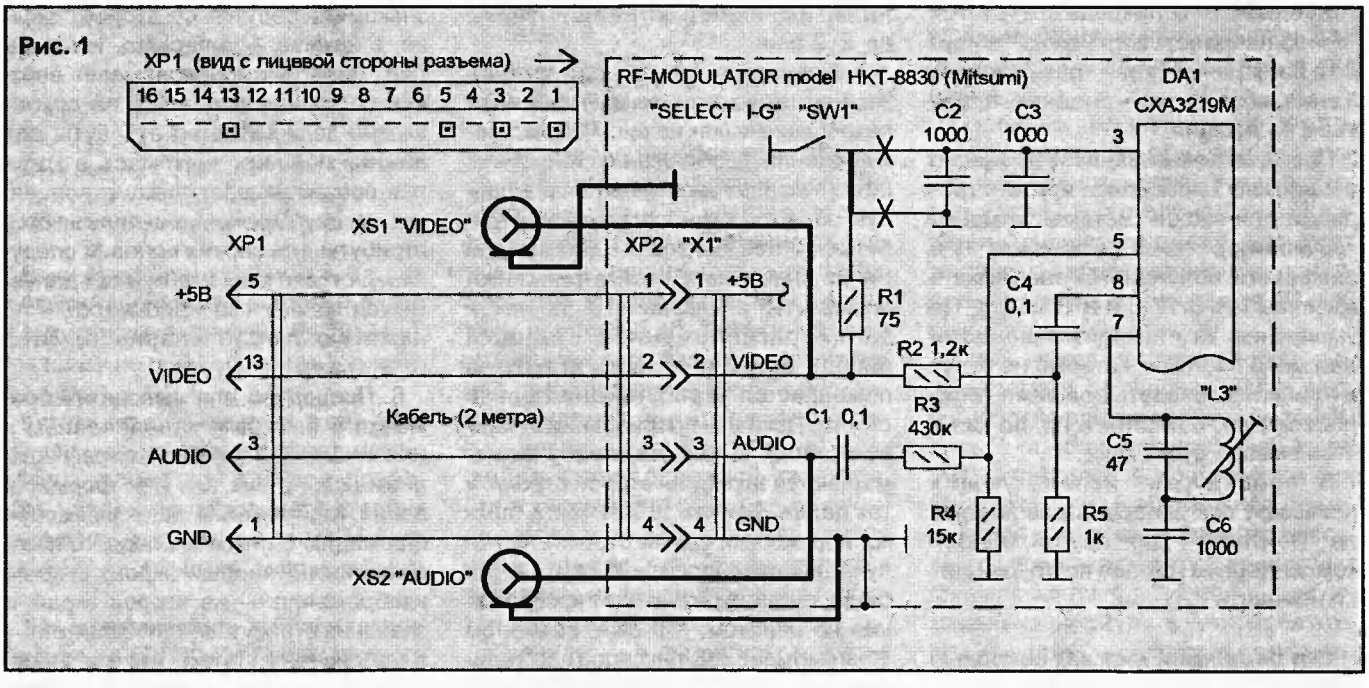
К сожалению, в этой статье рассказано лишь о малой части секретов системного реестра Windows. Более подробно о системном реестре при желании можно прочитать в книжках, найти полезные сведения в Интернете, и выяснить, где "прячутся" скрытые и недокументированные возможности Windows. И тогда можно узнать и открыть много нового и интересного для себя в одной из самых загадочных областей операционной системы Windows — системном реестре.

Доработка ВЧ-модулятора "Dreamcast"

С.РЮМИК,
г.Чернигов.

Игровые приставки "Dreamcast" (DC) при продаже комплектуются ВЧ-модулятором (High-Frequency Modulator). Он позволяет подключать DC к телевизору через высокочастотный тракт в дециметровом диапазоне длин волн. А что делать, если в телевизоре отсутствует или неисправен ДМВ-блок, либо качество изображения не удовлетворяет взыскательного зрителя? В этом случае я предлагаю использовать НЧ-тракт телевизора, разумеется, если в последнем имеются входные гнезда с маркировкой "VIDEO-IN" и "AUDIO-IN".

Рис. 1



На **рис.1** приведена электрическая схема входной части ВЧ-модулятора HKT-8830 фирмы Mitsumi (Тайвань), на которой утолщенными линиями показаны необходимые доработки. Вновь вводятся розетки XS1, XS2 ("тюльпан") и делаются два разрыва дорожек на печатной плате (отмечены "крестами").

Суть изменений

Основой ВЧ-модулятора является микросхема DA1 CXA3219M (Sony), которая формирует телевизионный сигнал ДМВ-диапазона в стандартах "I" и "G". Для справки, различие между стандартами заключается в разных значениях поднесущей звука: 5,5 МГц для "G" и 6 МГц для "I". В отечественном телевидении в ДМВ-диапазоне используется стандарт "K", имеющий частоту поднесущей звука 6,5 МГц. В связи с этим переключатель SW1 ставят в разомкнутое положение (стандарт "I"), а сердечник катушки с маркировкой "L3" на печатной плате подстраивают по достижению максимальной громкости звука в телевизоре.

Поскольку переключатель SW1 постоянно разомкнут, то его теперь можно использовать для других целей — для отключения резистора R1 от цепи "VIDEO". В новом варианте схемы при работе DC с телевизором по высокой частоте переключатель SW1 должен быть замкнут, а по низкой частоте — разомкнут. Дело в том, что при подключении НЧ-тракта телевизора к цепи "VIDEO" низкоомный резистор R1 является лишней нагрузкой, уменьшающей амплитуду видеосигнала. Его необходимо отключить от общего провода переключателем SW1.

Конструктивно розетки XS1, XS2 выводятся наружу ВЧ-модулятора тонкими проводами длиной около 10 см. К ним подключается стандартный кабель "AUDIO-VIDEO", имеющий с двух сторон вилки-"тюльпаны" RCA (**рис.2**).

Работа DC с телевизором по низкой частоте во многом улучшает качество изображения. Игровые картинки становятся более четкими, устраняются помехи в виде "муара" и дрожания растра. Предложенный способ доработки является наименее обременительным с финансовой точки зрения. Другое решение — приобрести в магазине специальный кабель, имеющий с одной стороны фирменную розетку "AV OUT" для DC, а с другой — "букет" из вилок-"тюльпанов" или разъем SCART.

Рис. 2

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

Вывод черно-белых изображений с градациями яркости

(Окончание. Начало в №№8-9/2002)

Приложение 2

Здесь приведены тексты процедур для вывода изображений шестью различными способами, рассмотренными в статье. Исходное изображение (размер изображения — 256*192, 256 градаций яркости) должно быть предварительно помещено в три банка ОЗУ, номера которых (точнее, числа, выводимые в порт #7FFD для их подключения) указываются в массиве N_BANK (по умолчанию там — #10, #11 и #13). Информация о пикселах хранится "слева направо, сверху вниз": в первом банке — верхняя треть изображения, во втором — средняя, в третьем — нижняя.

Перед использованием процедур вы должны определить яркости градаций для вашей системы "компьютер-монитор" (см. Приложение 1) и поместить полученные значения в массивы BRIGHT_0 и BRIGHT_1. По умолчанию там находятся значения для моей системы, которые не будут в точности совпадать с вашими. Можно, конечно, оставить и их, но качество вывода будет хуже.

В процедурах, использующих multicolor, все задержки рассчитаны на "Пентагон". Для другой модели компьютера их, скорее всего, придется изменить.

От редакции: *так как исходные тексты процедур имеют значительный объем, они в журнале не публикуются, приводится только их краткое описание. Полный листинг каждой процедуры, а также листинг общей части для каждой процедуры вынесены на Web-страницу журнала (http://radiomir.com). Они также могут быть получены из редакции по предварительному запросу.*

1. Процедура для вывода изображения в черно-белом режиме с 15 градациями яркости (как на рис.5). Формирует изображение в экранной области. Файл pr_2_1.asm

2. Процедура для вывода изображения с удвоенным атрибутным разрешением за счет использования multicolor'a (как на рис.6). Вначале формируются верхние половины знакомест со своими атрибутами на первом экране, а нижние половины со своими атрибутами — на втором. Затем, когда луч прорисовывает верхние половины знакомест, активным устанавливается первый экран, а когда нижние половины — второй. Выход из режима просмотра — по нажатию любой клавиши. Файл pr_2_2.asm

3. Процедура для вывода изображения с максимальным атрибутным разрешением (как на рис.7). При формировании изображения информация о пикселах записывается в экранную область, а информация об атрибутах (#1800 байтов — для каждой линии знакоместа свой атрибут) записывается в массив ATR (по умолчанию расположенный с адреса #8000). Затем в область атрибутов помещаются атрибуты для первой строки. Пока луч прорисовывает первую строку, в область атрибутов помещаются атрибуты второй строки, и так далее. Так как за 224 такта (время прорисовки одной строки) не получается перебросить 32 байта атрибутов, процедура выводит изображение не целиком, а в окне размером 11 знакомест (88 пикселов) по горизонтали. Клавишами "O"и "P" окно можно перемещать по изображению влево/вправо. Выход из режима просмотра — по нажатию пробела. Файл pr_2_3.asm

4. Процедура для вывода изображения с повышенным качеством за счет быстрого чередования двух изображений (как на рис.9). Она формирует одно изображение на первом экране, другое — на втором, и далее в каждом кадре меняет активный экран. Выход из режима просмотра — по нажатию любой клавиши. Файл pr_2_4.asm

5. Процедура для вывода изображения с быстрым чередованием и удвоенным атрибутным разрешением (как на рис.12). При формировании изображения записывает информацию о пикселах первого изображения на первый экран, а второго изображения — на второй экран. Атрибуты (#600 байтов) записываются в массив ATR. Далее, в первом кадре, когда луч прорисовывает верхние половины знакомест, активным устанавливается первый экран, а когда нижние половины — второй. В следующем кадре, наоборот, верхние половины знакомест берутся со второго экрана, а нижние — с первого, и так далее. Пока луч прорисовывает верхние половины знакомест, на другом экране записываются атрибуты для нижних половин знакомест, а когда луч прорисовывает нижние половины, на другом экране записываются атрибуты для верхих половин следующей строки знакомест, и так далее. Выход из режима просмотра — по нажатию любой клавиши. Файл pr_2_5.asm

6. Процедура для вывода изображения с быстрым чередованием и максимальным атрибутным разрешением (как на рис.13). При формировании изображения записывает информацию о пикселах первого изображения на первый экран, второго изображения — на второй экран, а значения атрибутов (#1800 байтов) — в массив ATR. Затем, как и в процедуре 3, область атрибутов динамически обновляется. В каждом кадре активный экран меняется. Изображение выводится в окне размером 11 знакомест (88 пикселов) по горизонтали. Клавишами "O" и "P" окно можно перемещать по изображению влево/вправо. Выход из режима просмотра — по нажатию пробела. Файл pr_2_6.asm

Общая часть процедур вывода — файл pr_2_gen.asm

Комментарии

Не все подпрограммы из общей части используются в каждой процедуре вывода. Для уменьшения объе-

ма неиспользуемые подпрограммы можно удалить. Сведения о том, какие именно подпрограммы не используются в каждой из шести процедур вывода, приведены в таблице:

Номер процедуры вывода	Подпрограммы, не используемые в этой процедуре
1	CLS_2, SET_TP_2, SET_P_2, ON_IM2, OFF_IM2, WAIT, PUT_DATA, A_FLICKER
2	PUT_DATA, A_FLICKER
3	GET_A_ATR, CLS_2, SET_TP_2, SET_P_2, PUT_DATA, A_FLICKER
4	ON_IM2, OFF_IM2, WAIT
5	GET_A_ATR
6	GET_A_ATR

Как выводить изображение с увеличением? Если требуется увеличение в два раза (размер выводимой области изображения тогда будет 128*96), то в начало процедуры GET_1R, после команд PUSH, добавьте такой фрагмент:

```
LD   A,D
SRL  A
ADD  A,координата X левого
       верхнего угла области
LD   D,A

LD   A,E
SRL  A
ADD  A,координата Y левого
       верхнего угла области
LD   E,A
```

Если нужно увеличение в 4, 8... раз (размер выводимой области тогда будет соответственно 64*48, 32*24...), то ставим не одну, а две, три... команды SRL A.

Кстати, для увеличения в два раза вышеприведенный фрагмент можно оптимизировать— вначале прибавить удвоенную координату, а затем разделить на два командой RRA вместо SRL A. Это будет короче и быстрее.

Как сформировать таблицу гамма-коррекции (GAMMA_T) для нужного значения гаммы? Ниже приведена соответствующая программа на Бейсике. Эта программа формирует таблицу в том же формате, в котором она хранится в общей части процедур вывода: значения — в формате 8.8, в первых 256 байтах — целые части, в следующих 256 — дробные. Общая длина таблицы, таким образом, составляет 512 байтов.

При запуске программы требуется указать нужное значение гаммы. Одновременно с вычислением таблицы, на экране строится график зависимости яркости пиксела от его значения в изображении для заданного значения гаммы. После окончания вычислений укажите имя файла, куда будет записана таблица, или просто нажмите ENTER для перезапуска программы.

```
 10 CLEAR 32767
 15 CLS
 20 INPUT "Gamma: ";G
 30 FOR i=0 TO 255
 40 LET y=(i/255)^G
 50 LET Word= INT (y*255*256+0.5)
 60 LET HighByte= INT (Word/256)
 70 LET LowByte= Word-HighByte*256
 80 POKE 32768+i, HighByte
 90 POKE 32768+256+i, LowByte
100 PLOT i*175/255,y*175
110 NEXT i
120 INPUT "File: ";a$
130 IF a$="" THEN GO TO 15
140 RANDOMIZE USR 15619: REM :SAVE a$
CODE 32768,512
```

При выводе изображения для получения псевдоградаций яркости используются 257 различных текстур 16*16. Простой подсчет показывает, что для их хранения требуется 257*16*16/8=8224 байта памяти, или более 8 Кб. Тратить столько памяти не хотелось, и я использовал такой прием — когда нужно узнать, включен или выключен пиксел с данными координатами, находящийся в определенной текстуре, ответ получается с помощью специального алгоритма, основанного на свойствах текстур. Представляется интересным рассмотреть этот алгоритм подробнее.

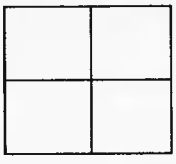
Сначала — о свойствах, которым должны удовлетворять текстуры, используемые для получения псевдоградаций яркости (в общем случае). Пусть L — длина стороны текстуры, причем L является числом вида 2^k (в данном случае $L=16=2^4$). Количество текстур N равно L^2+1 (в данном случае $N=16^2+1=257$).

Во-первых, в каждой следующей текстуре на один включенный пиксел больше, чем в предыдущей. Если пронумеровать текстуры от 0 до N-1, то номер текстуры будет равен числу включенных в ней пикселов.

Во-вторых, любые две соседние текстуры отличаются лишь одним пикселом. Учитывая предыдущее свойство, получаем, что каждая следующая текстура получается из предыдущей путем включения еще одного пиксела.

В-третьих, включенные и выключенные пикселы в каждой текстуре располагаются равномерно. Что это значит? Возьмем текстуру и разделим ее на четыре равные части (**рис.20**).

Рис. 20

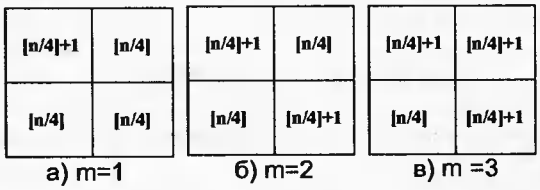


При этом в любых двух частях количество включенных (как и выключенных) пикселов будет различаться не более чем на один. Если каждую часть опять разделить на четыре, это условие будет выполняться и для полученных меньших частей, и так далее.

Очевидно, что если номер текстуры (обозначим его n) кратен 4 (а значит, и количество включенных пикселов кратно 4), то единственный вариант их расположения в четвертях текстуры, удовлетворяющий приведенному выше условию — по n/4 пикселов в каждой четверти.

Пусть номер текстуры не кратен 4, m — остаток от деления. Тогда, чтобы условие равномерности выполнялось, в m четвертях должно быть по [n/4]+1 пикселов, а в оставшихся 4-m четвертях — по [n/4] пикселов. Положим для определенности, что пикселы располагаются в трех возможных случаях (когда m = 1, 2 или 3), как показано на **рис.21а...в** (учитывая второе свойство — соседние текстуры различаются лишь одним пикселом!).

Рис. 21



а) m=1 б) m=2 в) m =3

Приведенной выше информации достаточно как для того, чтобы построить все n текстур, так и для того, чтобы определить, включен или выключен пиксел с данными координатами, находящийся в определенной текстуре.

Алгоритм очень простой. Пусть из-

вестны L — длина стороны текстуры, n — номер текстуры, x и y — координаты пиксела на экране. Нужно получить a — состояние пиксела (0 — выключен, 1 — включен).

1. Если размер текстуры — 1*1, то состояние пиксела равно номеру текстуры (т.е. если L=1, то a=n).

2. Иначе формулируем условия новой, более простой задачи для вдвое меньшей текстуры, являющейся четвертью исходной (т.е. получаем новые L и n). Ответ новой задачи будет ответом исходной задачи.

2.1. Если n кратно 4, то L:=L/2, n:=n/4, и решаем задачу с этими условиями.

2.2. Иначе определяем, в какой четверти текстуры оказывается пиксел. Если L and x = 0, то пиксел — в левой половине, иначе — в правой. Если L and y = 0, то пиксел — в верхней половине, иначе — в нижней.

2.3. По m (остатку от деления n на 4) определяем, в каких четвертях будет включено [n/4] пикселов, а в каких — [n/4]+1 (рис. 21).

2.4. Если в той четверти текстуры, где оказывается пиксел, включено [n/4] пикселов, то L:=L/2, n:=[n/4], и решаем задачу с этими условиями.

2.5. Иначе — L:=L/2, n:=[n/4]+1, и решаем задачу с этими условиями.

В программе этим занимается процедура TEXTURE, длина которой всего 46 байтов. Сравните это с первоначальными 8224 байтами!

Приложение 3

Здесь приведены тексты процедур для работы с pcx-файлами, содержащими черно-белые изображения с 256 градациями яркости. Кратко расскажу о формате таких файлов — это необходимо для понимания логики работы процедур. В таблице приведена структура формата ч/б pcx-файлов.

Часть файла	Длина
Заголовок	128
Упакованная информация о пикселах изображения	Переменная
Тип палитры	1
Палитра	768

Заголовок файла содержит информацию о размерах изображения, количестве цветов и т.п. В процедуре чтения (READ_PCX) заголовок игнорируется, так как эти данные предполагаются уже известными. В процедуре записи (MAKE_PCX) заголовок записывается всегда один и тот же.

Информация о пикселах изображения хранится в упакованном виде. При упаковке последовательность одинаковых байтов заменяется на два байта — счетчик и оригинал. Чтобы можно было распознавать байт-счетчик при распаковке, его старшие два бита устанавливаются в 1, а длина последовательности (от 1 до 63) хранится в младших шести битах. Если длина последовательности больше 63, записывается несколько пар "счетчик—значение". Если при упаковке попадается отдельный байт, старшие два бита которого равны 1, то, чтобы при распаковке он не был перепутан с байтом-счетчиком, перед ним ставится байт-счетчик с количеством повторений, равным 1.

Тип палитры имеет значение #0A, если значения байтов палитры находятся в диапазоне 0...63, и #0C — если в диапазоне 0...255. В процедуре чтения тип палитры игнорируется, а в процедуре записи записывается #0C.

Палитра — в ней содержится информация о компонентах R, G, B каждого из 256 цветов изображения. В процедуре чтения палитра игнорируется, а в процедуре записи записывается последовательность байтов (0,0,0, 1,1,1, ..., 255,255,255).

Исходя из вышесказанного, можно определить максимальную длину файла: 2 байта на пиксел — это 256*192*2 = #18000, еще #80 байтов — заголовок, 1 байт — тип палитры и #300 байтов — сама палитра, итого — #18381 байт. Но в TR-DOS длина файла не может превышать #FF00 байтов. Поэтому в процедуре MAKE_PCX, если длина файла получается больше #FF00, на диск записываются два файла: первый длиной #FF00, а во втором — остаток.

От редакции: *так как исходные тексты процедур имеют значительный объем, они в журнале не публикуются, приводится только их краткое описание. Полный листинг каждой процедуры, а также листинг общей части для каждой процедуры вынесены на Web-страницу журнала (http://radiomir.com). Они также могут быть получены из редакции по предварительному запросу.*

1. READ_PCX — чтение изображения из pcx-файла в три банка памяти.

Вход: имя и расширение файла указано в FILENAME, номера банков памяти — в N_BANK.

Выход: A=1 — Ok, A=0 — файл не найден.

Файл — read_pcx.asm

2. MAKE_PCX — запись изображения из трех банков памяти в pcx-файл.

Вход: имя и расширение файла указано в NAME_EXT, номера банков памяти — в N_BANK.

Выход: A=1 — Ok, A=0 — записать файл не удалось (нет места на диске или в каталоге диска).

Файл — make_pcx.asm

3. SCR2BANK — преобразование изображения в спектрумовском формате в черно-белое изображение с 256 градациями яркости, расположенное в трех банках памяти. С помощью предыдущей процедуры полученное изображение можно записать в pcx-файл.

Вход: исходное изображение загружено с #4000, номера банков памяти — в N_BANK, яркости градаций — в BRIGHT_0 и BRIGHT_1.

Файл — scr2bank.asm

Литература

1. А.Бордачев. О формате PCX. Библиотека информационной технологии, вып.8. — Москва: "Инфоарт", 1993.

2. Т.Кенцл. Форматы файлов Internet. — СПб.: "Питер", 1997.

3. К.Бочков. Сканирование — это так просто.... — Мир ПК, 2000, №11.

4. С.Кащавцев. Волшебная сила кривых — II. — Компьютерра, 1998, №№49, 50.

5. Е.Зарецкий. Быстрый вывод графики. — Радиолюбитель. Ваш компьютер, 2001, №№5, 6.

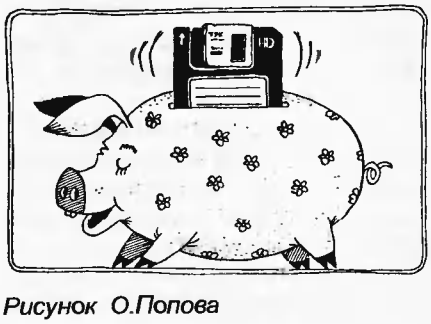


Рисунок О.Попова

Быстрая печать символа

А.СИЗЕНКО,
Украина, г.Луганск.
E-mail: pat_2000@mail.ru

Как максимально ускорить вывод символов на экран? В предлагаемой вниманию читателей статье рассматриваются некоторые методы решения этой задачи. Но предварительно я должен сделать два замечания:

1) входные данные ко всем программам:

- код символа — в регистре A (значения — от 0 до 127);
- видеоадрес — в регистровой паре DE.

2) в статье не рассматриваются вычисление и занесение атрибута печати, т.к. реализация этого сильно варьируется в зависимости от поставленной задачи.

Рассмотрим классический вариант печати символа для Speccy:

```
        ADD     A,A         ; *2, переполнение не грозит
        LD      L,A         ; помещаем код в регистровую пару HL
        LD      H,0
        ADD     HL,HL       ; *4
        ADD     HL,HL       ; *8, умножили код на восемь
        LD      BC,ADR_FONT ; (адрес начала знакогенератора в BC)
        ADD     HL,BC       ; сложили с BC и получили адрес символа
        LD      B,8         ; цикл по количеству линий
M1      LD      A,(HL)      ; берем очередной байт символа
        LD      (DE),A      ; и помещаем его на экран
        INC     HL          ; указываем на следующий байт
        INC     D           ; переходим к следующей горизонтальной черте
        DJNZ    M1          ; NEXT
        RET                 ; ушли
```

Есть резервы для ускорения? Есть! Но увеличение скорости достигается за счет увеличения размеров процедуры печати. Для этого раскрываем цикл (этот способ уже неоднократно описывался в печати):

```
                    Количество
                    тактов
    ADD A,A           ; 4; все то же самое
    LD L,A            ; 4
    LD H,0            ; 7
    ADD HL,HL         ;11
    ADD HL,HL         ;11
    LD BC,ADR_FONT    ;10
    ADD HL,BC         ;11

    LD A,(HL)         ; 7; только цикла нет
    LD (DE),A         ; 7; кидаем 1-й байт
    INC HL            ; 6
    INC D             ; 4

    LD A,(HL)         ; 7
    LD (DE),A         ; 7; 2-й байт
    INC HL            ; 6
    INC D             ; 4

    .........         ;24; 3-й (повторяем четыре команды)
    .........         ;24; 4-й
    .........         ;24; 5-й
    .........         ;24; 6-й
    .........         ;24; 7-й
    LD A,(HL)         ; 7
    LD (DE),A         ; 7; 8-й
    RET               ;10; все
```

Итого: 250 тактов.

Есть резервы для ускорения? Есть! "Хвост" программы печати символа будет таким:

```
                    Количество
                    тактов
    LD (HL),NN      ;10
    INC H           ; 4
    LD (HL),NN      ;10
    INC H           ; 4
    LD (HL),NN      ;10
    INC H           ; 4
    LD (HL),NN      ;10
    INC H           ; 4
    LD (HL),NN      ;10
    INC H           ; 4
    LD (HL),NN      ;10
    INC H           ; 4
    LD (HL),NN      ;10
    RET             ;10
```

Итого: 118 тактов.

Для ускорения сформируем из нашего фонта программы для печати каждого(!) символа (по 24 байта):

```
        LD BC,96        ; цикл по количеству символов
        LD DE,ADR_FONT; адрес знакогенератора
        LD HL,BUFER     ; адрес формируемых программ
M1      PUSH BC         ;
        LD B,8          ; делаем второй цикл,
                        ; чтобы "кидать" команду RET
M2      LD (HL),#36     ; это код команды LD (HL),NN
        INC HL
        LD A,(DE)       ; затем берем очередной байт фонта
        INC DE
        LD (HL),A       ; это и будет числом NN
        INC HL
        LD (HL),#24     ; код команды INC H
        INC HL
        DJNZ M2         ; NEXT B
        DEC HL          ; в конце "кинем" код команды RET
        LD (HL),#C9     ;
        INC HL          ;
        POP BC
        DJNZ M1         ; и так далее, пока не обработаем весь фонт
        RET
```

А также формируем таблицу адресов переходов (иначе умножение на 24 сводит на нет все наши старания):

```
        LD B,96         ; цикл по количеству символов
                        ; в знакогенераторе
M3      LD A,96         ; формируем последовательность от 0 до 95
        SUB B           ;
        LD L,A          ; умножаем на 24
        LD H,0          ;
        ADD HL,HL       ;
        ADD HL,HL       ;
        ADD HL,HL       ;
        LD E,L          ;
        LD D,H          ;
        ADD HL,HL       ;
        ADD HL,DE       ;
        LD DE,BUFER     ; начальный адрес "хвостов" программ
        ADD HL,DE       ; здесь получили адрес для печати N-го символа
        EX DE,HL        ; "кинули" его в DE
        LD HL,(NEXT)    ; NEXT указывает на очередной адрес в таблице
        LD (HL),E       ; "кидаем" сначала младший
        INC HL
        LD (HL),D       ; а потом старший байт
        INC HL
        LD (NEXT),HL
        DJNZ M3
        RET
NEXT    DEFW BUFER+24*96; таблицу будем располагать в конце блока
                        ; программ (назовем этот адрес TABL)
```

Ну а теперь непосредственно "голова" программы:

```
                    Количество
                    тактов
        ADD A,A         ; 4; код символа умножили на два
        LD L,A          ; 4; поместили в HL
        LD H,0          ; 7
        LD BC,TABL-32*2 ;10; коррекция начального адреса
                        ;    таблицы на 32*2
        ADD HL,BC       ;11; получаем адрес в таблице
        LD C,(HL)       ; 7; берем из нее младший байт
        INC HL          ; 6
        LD B,(HL)       ; 7; затем старший
        EX DE,HL        ; 4; "кидаем" видеоадрес в HL
        PUSH BC         ;11; эмулируем команду JP (BC)
        RET             ;10
```

Итого: 81 такт. Вместе с "хвостом" — 199. Экономия по сравнению со вторым вариантом — 51 такт, или повышение скорости на 20,4%.

Есть резервы для ускорения? Есть!!! Если использовать "круглый" адрес в таблице (например, #5B00), отпадает необходимость формировать старший байт (он будет постоянным).

```
                    Количество
                    тактов
    ADD A,A         ; 4
    LD L,A          ; 4; теперь младший байт уже сформирован
    LD H,#5B        ; 7; а старший всегда равен константе
    LD C,(HL)       ; 7; аналогично предыдущей программе
    INC L           ; 4; нет необходимости делать INC HL
    LD B,(HL)       ; 7
    EX DE,HL        ; 4
    PUSH BC         ;11
    RET             ;10
```

Итого: 58 тактов. Вместе с "хвостом" — 176. Экономия по сравнению со вторым вариантом — 74 такта, или повышение скорости на 29,6%. Замечу, что при формировании таблицы адресов ее начальный адрес должен быть равен #5b40 (чтобы не выполнять команду SUB 32 в начале программы).

Есть резервы для увеличения скорости? Честно говоря, на этом месте я хотел закончить статью, но что-то не давало мне покоя. Наконец, вечером мне пришла в голову следующая идея:

```
                    Количество
                    тактов
    ADD A,A         ; 4; вычисляем младший адрес в таблице
    LD (BODY+1),A   ;13; и "кидаем" его непосредственно в тело
                    ;    программы
BODY LD HL,(#5B00)  ;16; сразу грузим два байта адреса в HL
                    ;   (напомню: #5B40 - адрес начала таблицы)
    EX DE,HL        ; 4
    PUSH DE         ;11; JP (DE)
    RET             ;10
```

Итого: 58 тактов.

Видно, что увеличения скорости мы не получили. Зато у нас освободилась пара BC. Так есть ли еще резервы для увеличения скорости?

В последней программе адрес перехода находится в HL. В этом случае очень соблазнительно использовать команду JP (HL), которая выполняется всего за четыре такта. Но для этого придется вновь идти на жертвы — "хвосты" программ увеличатся на один байт и будут иметь такой вид:

```
    EXX DE,HL
    LD (HL),NN
    INC H
    ..........
    RET
```

Соответственно, придется переделать программы, генерирующие код и таблицу адресов (*25). А наша модернизированная "голова" будет иметь следующий вид:

```
                    Количество
                     тактов
FAST ADD A,A         ; 4
     LD (BODY+1),A   ;13
BODY LD HL,(#5B00)   ;16
     JP (HL)         ; 4
```

Итого: 37 тактов. Вместе с хвостом — 37+4+118=159 тактов.

Экономия по сравнению со вторым вариантом — 91 такт, или повышение скорости на 36,4%!

В заключение хочется показать, как использовать данные программы наиболее рационально в скоростном плане.

Печать статической надписи

Если необходимо быстро "выкинуть" на экран заголовок, то лучше всего это делать группой команд LD (HL),NN. Например, печать четырех символов в левом верхнем углу:

```
    LD HL,#4000     ; видеоадрес
    LD A,L          ; запомним младший байт для восстановления
    LD (HL),NN      ; "кидаем" первую строку первого символа
    INC L           ; следующий символ
    LD (HL),NN      ; первую строку второго символа
    INC L           ; следующий символ
    LD (HL),NN      ; первую строку третьего символа
    INC L           ; следующий символ
    LD (HL),NN      ; первую строку четвертого символа
    INC H           ; следующая строка
    LD L,A          ; начинаем сначала
                    ; ну и т.д.
```

Печать строки определенной длины

Делаем все как можно быстрее:

```
    LD BC,LINE   ; LINE - адрес строки текста символов
                 ; (лучше "круглый" на 256 -
                 ; как раз треть экрана)
    LD DE,#4000 ; видеоадрес в DE
    LD A,(BC)    ; грузим код символа
    CALL FAST    ; печатаем
    INC C        ; если адрес хранения не "круглый", то INC BC
    LD A,(BC)    ; код следующего символа

    LD DE,#4001 ; печатаем следующий символ
    CALL FAST    ;
    INC C        ;
    LD A,(BC)    ;
    ..........  ; и т.д.
```

Печать строки неопределенной длины

Здесь сначала рассмотрим вариант печати с маркером конца (0) в одной трети экрана, например, вверху:

```
    LD DE,#4000  ; в верхний левый угол экрана
    LD BC,LINE   ; адрес хранения начала текста делаем круглым,
                 ; ставя в соответствие с младшим видеоадресом
M1  LD A,(BC)    ; грузим код
    OR A         ; устанавливаем флаги
    RET Z        ; если это ноль, то ушли
    CALL FAST    ; печатаем
    INC C        ; следующий символ
    LD E,C       ;
    LD D,#40     ; старший всегда постоянен
    JR M1        ; ноль должен быть обязательно!
```

И второй вариант — с применением счетчика. Заодно проиллюстрируем занесение атрибута печати. Адрес атрибута расположен в альтернативных регистрах, там же будет цвет в регистре C и длина в регистре B. В основном наборе назначения регистров те же:

```
    LD HL,#5800  ; вычислять атрибутный адрес "некогда"
    LD C,#47     ; цвет
    LD B,10      ; длина строки
    EXX          ; все это останется в альтернативе
    LD DE,#4000
    LD BC,LINE
    LD A,(BC)
    CALL FAST
    EXX          ;
    LD (HL),C    ; "кидаем" цвет
    INC L        ; следующее знакоместо
    DEC B        ; уменьшаем счетчик
    RET Z        ; если все, то ушли
    EXX          ; и т.д. до предельной длины
    ............
```

Boot в одном секторе

Р.ФАСИХОВ,
г.Казань.
E-mail: foton2@mail.ru

Предлагаю программу-загрузчик — boot, которая занимает всего один сектор.

Кнопки управления загрузчика:

- 8, 9 — выбор файла для запуска;
- 7 — reload (перечитать) диска;
- 0 — запуск файла.

Рассмотрим процесс создания загрузчика. Сначала набираем в BASIC 48 следующие строки и сохраняем их в файле:

```
10 RANDOMIZE USR VAL "23916"
20 RANDOMIZE USR VAL "15619":REM:RUN"........"
30 REM ......................
```

Между кавычками после RUN должно быть 8 пробелов — это зарезервированное место для имени файла. После REM (строка 30) должно быть около 200 пробелов — это зарезервированное место для кодового блока. Более точно количество пробелов определяется опытным путем по длине файла, она должна быть равна 252 байта.

Следующий этап — создание кодового блока. Набираем в ассемблере следующий код:

```
;XAS 9.08
;MICRO-boot
;(C) Ru 2002
              ORG     #5D6C
POINT         EQU     #D400   ;Адрес указателя на имя файла

;Считывание загрузочной дорожки и открытие канала "S"
BEGIN  LD     HL,#C000
              PUSH    HL
              LD      D,L
              LD      E,L
              LD      BC,#0905
              CALL    #3D13
              LD      A,#02
              CALL    #1601
              XOR     A
              OUT     (#FE),A
              LD      (BAS_CNT),A   ;Количество BASIC-файлов

;Заполняем буфер именами     BASIC-файлов
              POP     HL
              LD      DE,#D000
              LD      A,(#C8E4)     ;Количество файлов на диске
LOOP0         EX      AF,AF'
              LD      BC,#0008
              ADD     HL,BC
              LD      A,(HL)
              CP      "B"
              JR      NZ,NO_FILL
              PUSH    HL
              LD      HL,BAS_CNT
              INC     (HL)
              POP     HL
              PUSH    BC
              SBC     HL,BC
              LDIR
              POP     BC
NO_FILL       ADD     HL,BC
              EX      AF,AF'
              DEC     A
              JR      NZ,LOOP0

;Главный цикл
;Очистка экрана и вывод имен файлов

LABEL         LD      HL,BAS_CNT
              CP      (HL)
              JR      NC,LABEL2
              LD      (POINT),A
              LD      A,#04
              LD      (AT_Y),A
              LD      HL,#4000
              LD      DE,#4001
              LD      BC,#1AFF
              LD      (HL),L
              LDIR
              LD      A,(POINT)     ;Номер файла в буфере
LABEL1        CP      #00
BAS_CNT       EQU     $-1
              JR      NC,LABEL2
              PUSH    AF
              CALL    M000
              LD      A,#16
              RST     #10
              LD      A,#00

AT_Y          EQU     $-1
              RST     #10
              LD      A,#0C
              RST     #10
              EX      DE,HL
              LD      BC,#0008
              CALL    #203C         ;Печать имени файла на экране
              HALT
              LD      HL,AT_Y
              LD      A,#20
              CP      (HL)
              JR      Z,LABEL0
              INC     (HL)
              POP     AF
              INC     A
              JR      LABEL1
LABEL0        POP     AF            ;Балансируем стек

;Рисуем курсор и опрашиваем клавиатуру

LABEL2        LD      HL,#588C
              LD      B,#08
LABEL5        LD      (HL),%01010110
              INC     L
              DJNZ    LABEL5

              LD      A,(POINT)
              LD      BC,#EFFE
              IN      B,(C)
              BIT     0,B
              JR      Z,EXIT
              BIT     3,B
              JP      Z,BEGIN
              BIT     2,B
              JR      NZ,LABEL3
              INC     A
              JR      LABEL
LABEL3        BIT     1,B
              JR      NZ,LABEL2
              DEC     A
              JR      LABEL

;Пишем имя файла после "RUN"
;и выход в BASIC

EXIT          CALL    M000
              LD      DE,#5D5D
              LD      BC,#0008
              LDIR
              RET

;Вычисление адреса имени файла в буфере
;A=номер имени файла
;HL=адрес имени

M000          LD      L,A
              LD      H,#00
              LD      DE,#D000
              ADD     HL,HL
              ADD     HL,HL
              ADD     HL,HL
              ADD     HL,DE
              RET
```

После компиляции программа должна занимать 189 байтов. Затем загружаем STS 6.2. С помощью функции LOAD SECTOR загружаем ранее полученный BASIC-файл. Вставляем на место пробелов после "REM" кодовый блок. Адрес нужно уточнить в STS. Например, если загрузить BASIC-файл по адресу #5D3B, то кодовый блок должен начинаться с адреса #5D6C. Записываем с помощью SAVE SECTOR программу на диск. Boot готов!!!.

И напоследок надо заметить, что в нем есть еще 11 байтов для реализации какой-нибудь функции.

ELROND,
E-mail: elrond@land.ru

Демиурги

Тандем компаний **Nival Interactive** и **1С** не раз радовал нас замечательными играми, ставшими уже классическими — это и "Аллоды", и "Проклятые Земли" ("Аллоды III"), и многие другие. Кстати говоря, на Западе успехом пользовались первая и вторая части "Аллодов", известной там под названием "Rage of Mages". Какая еще отечественная игра может похвастаться тем, что она была переведена на английский язык?

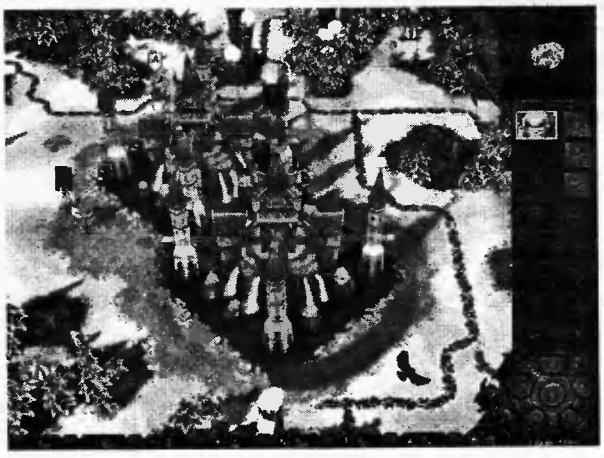
К ряду этих знаменитых игр не так давно была добавлена еще одна — "Демиурги", пошаговая стратегия, которая сразу же стала одной из лучших отечественных игр.

Действие ее происходит в мире, где всем правит Эфир — что-то наподобие манны. Вы — один из четырех Лордов Эфира, повелевающий целой расой фантастических существ. Виталы и Кинеты объединились в борьбе против союза Синтетов и Хаотов, и за одну из этих рас вам придется выступать. Ваша главная цель — первым достичь ступеней Храма Времени, чтобы стать верховным повелителем этого эфирного мира — Белым Лордом. Естественно, не один вы хотите сделать это, поэтому ваш путь к храму долог, опасен и полон неожиданностей. Но отступать уже поздно... Победа не дается легко, и чтобы достичь своей цели, вам предстоит пройти через целый ряд миссий, где каждая новая задача труднее предыдущих.

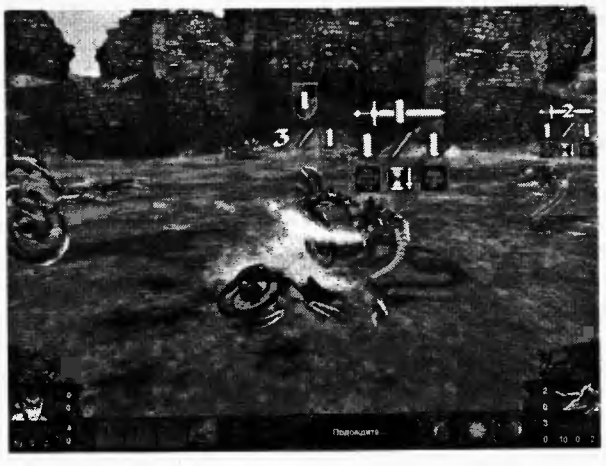
Так как источником силы каждого Лорда служит его Замок, вам предстоит уничтожить Замки своих противников, но при этом уберечь свой.

Поклонники "Героев Меча и Магии" не увидят здесь коней, которые были там средством передвижения героев. В "Демиургах" герои оседлали каких-то чудных существ, некоторые из которых похожи не то на черепах, не то на тараканов... Хотя это, конечно, не единственное отличие "Демиургов" от "Героев".

Разительное отличие в том, что в этой игре нельзя нанимать войска. Звучит странновато, не правда ли?



Дело в том, что процесс битвы напоминает карточную игру (в данном случае — Magic:The Gathering). Вместо карт здесь используются заклинания. В самом начале боя герою дается случайным образом пять заклинаний из его арсенала.



Через каждый ход одно (или более —в зависимости от навыков героя) заклинание заменяется другим. Или, если у него есть свободные ячейки, то заклинания просто добавляются.

Заклинания бывают разные. Черных, белых и красных, конечно, нет, но ассортимент все равно велик. И, естественно, чем выше уровень героя, тем более сильные заклинания он может использовать.

Главными стоит считать заклинания призыва существ. Это объясняется тем, что герои не могут атаковать друг друга непосредственно — только с помощью магии. Поэтому героям и необходимы существа-помощники для атаки и защиты.

Собственно, битва и состоит из двух фаз — фазы атаки и фазы защиты. В фазе атаки вы можете распределить своих существ по целям, которые они должны атаковать. А в фазе защиты вы, соответственно, распределяете, кто и от кого должен своей грудью прикрыть хозяина.

Для исполнения заклинаний, как вы, наверное, догадались, необходим эфир. При каждом ходе его количество восполняется — в зависимости от количества открытых эфирных каналов. Так что эфир, в принципе, заканчивается не скоро. Значит, некоторые битвы могут стать бесконечными — что, разумеется, не очень приятно. Поэтому в игре введено такое понятие как "эфирное возмущение". Это явление наступает при длительном бое и проявляется в виде молнии, которая с каждым ходом попадает в вашего героя или героя противника, и тогда наступает черед "русской рулетки" — кому повезет больше, кто переживет противника?

Стоит сказать несколько слов о ресурсах. Понятно, что если мир — эфирный, то главный ресурс в нем — эфир. Золота здесь нет и подавно, как нет камней и дерева. Зато, кроме эфира, присутствует немалое количества различных магических ингредиентов — Корень Мандрагоры, Черный Лотос, Кровавый Рубин, Ядовитый Изумруд, Звездный Сапфир, Дымчатый Алмаз, Замерзшее Пламя.

Порадовал юмор разработчиков. При выборе уровня сложности игры необходимо переключаться между изображениями доисторического человека. И если первый таковым и выглядит — доисторическим, то последний с видом знатока восседает за... компьютером. Кажется, что это — мелочь, но все же приятно лишний раз улыбнуться.

В "Демиургах" поддерживаются различные виды сетевой игры, в том числе — игры через Интернет и используя клиентскую программу GameSpy Arcade, которая находится на лицензионном компакт-диске игры.

Разочаровывает, однако, отсутствие в начальной версии поддержки "горячего кресла", т.е. игры нескольких человек за одним компьютером. Но с официального сайта "Демиургов" — www.etherlords.com — всегда можно скачать свежий патч, который добавляет эту возможность и исправляет некоторые ошибки, возникающие в процессе игры.

Звуковое сопровождение всегда было одной из сильных сторон игр от Nival. Исключением не стали и "Демиурги". Однако стоит сказать об одном непонятном явлении. Хотя я и несведущ в магии эфирного мира, мне кажется странным, что в процессе боя одни и те же заклинания герой произносит по-разному. Это, конечно, можно объяснить тем, что через определенное время одни и те же звуки начинают порядком надоедать. С этой точки зрения различные звуки выглядят вполне оправданно — хотя хотелось бы, чтобы все было как можно ближе к реальности. Впрочем, о какой реальности можно говорить в фантастическом мире?

Много говорить о графике, я думаю, не стоит, — она просто великолепна. Все сделано в полном 3D и динамике. В процессе боя "разумная" камера автоматически изменяет угол и дальность обзора, чтобы лучше показать участки поля, на которых в данное время происходит действие. И, конечно, за все это надо платить не очень скромными системными требованиями:

Минимальная конфигурация

Операционная система Windows 98/ME/2000, DirectX версии 7.0 или выше

Процессор Pentium II 300 МГц

Оперативная память 64 Мб

Видеоадаптер AGP с 3D-ускорителем и 8 Мб видеопамяти

Монитор с поддержкой разрешения 800x600

Привод CD-ROM 4x

Звуковая карта

Рекомендуемая конфигурация

Операционная система Windows 98/ME/2000, DirectX версии 7.0 или выше

Процессор Pentium III 550 МГц

Оперативная память 128 Мб

Видеоадаптер AGP 4x с 3D-ускорителем и 32 Мб видеопамяти

Монитор с поддержкой разрешения 1024x768

Привод CD-ROM 32x

Звуковая карта

Для установки игры необходимо иметь не менее 1,4 Гб свободного места на жестком диске. Рекомендуется дополнительно иметь 500 Мб свободного места для сохранения игр и файла подкачки.

Для игры через Интернет необходимо модемное или выделенное Интернет-соединение со скоростью не ниже 19,2 К. Для запуска собственного сервера вы должны иметь прямое подключение, так как ваш IP-адрес должен быть доступным из внешней сети.

P. S. По завершении написания этой статьи стало известно, что Nival Interactive анонсирует игру "Демиурги II", которая включает в себя лучшие идеи игроков. Права на издание и распространение в странах СНГ и Балтии, как и первой части, принадлежат компании 1С. Планируемая дата выхода "Демиургов II" — первый квартал 2003 года. Более подробную информацию можно почерпнуть на сайте "Демиургов".

К.КЛИЩЕНКО,
г.Минск.

Veni, Vidi, Vici

Help me out, or let me in
Watch me die for someone's sin.
Royal Hunt.

Here we are,
We're the princes of the Universe.
Queen.

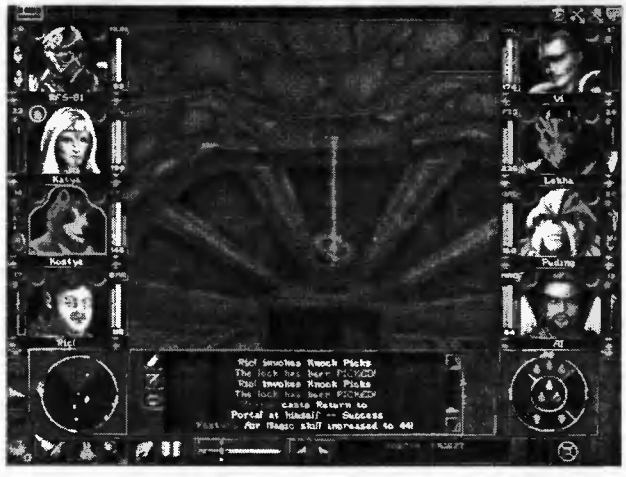
Эта игра посвящается любителям бесконечных дорог, сражений и пронизывающих душу взглядов вдаль.

Есть Земля, и есть Доминус. Тут — жизнь, там — Wizardry 8. И невозможно даже после долгих размышлений решить, что же тебе нужнее. У игры есть только один недостаток — она имеет предел. Но не более того.

Я уверен, что Wizardry 8 стоила девяти лет и немыслимого труда, затраченных на нее. Редкому адепту удается избежать ее очарования и отринуть Доминус. Кое-кто считает, что РПГ, завязанные

на сюжет или персонажи, уступают играм, акцентирующим внимание на мире. Не хочу спорить (да, в общем-то, и не с кем), но эта игра нарушает устоявшийся взгляд. А так как я вижу Wizardry гимном своей команды, то позвольте с нее и начать (в порядке построения в бою).

На острие атаки было бы неплохо иметь сотню-другую хип-хоп-маньяков — во-первых, много, во-вторых, не жалко. Но в моей команде вперед выдвинута валькирия (близкие виды — гарпия, фурия). Кроме нордического характера, умения громко визжать (известная ария из оперы Вагнера) и женского пола, она умеет потрясающе владеть копьем и обманывать смерть (аналог — Pkunk'a из Star Control). Набрала больше всех грехов (убийство).



Вторым ортодоксальным воителем в моей организации является дракон-боец. Он хороший малый, несмотря на то что змей. Впадет в берсеркирство — бойся, враг. А вот истинно драконий орган плевания слабоват. Зато он очень хитовый (почти как Бритни Спирс). Жизни в нем в полтора раза больше, чем в ближайшем преследователе.

Остальные мои "сраженцы" были NPC, о них и разговор будет отдельный.

А теперь следующий номер программы — раувульф-жрец (в просторечьи — болонка-проповедник, хотя я предпочитаю — Волк-Несущий-Истину). Избранник богов, ниспосылающий свет создателя на друзей и карающий гневом небес врагов (лечебно-защитная приставка к команде, под конец игры эти функции наполовину перешли к барду, но зато стали шире возможности сделать кому-нибудь больно). Могучий воитель (изредка попадает по врагу). Просто душка (если кому-нибудь может понравиться запах псины).

Самый новаторский персонаж в игре — мук-изобретатель. Честно говоря, я не могу представить йети за компьютером, к тому же, в ретроспективе — идеального монаха (в игре расы имеют профессиональные предпочтения). Муку выданы патенты на частое исчезновение из команды, меткую стрельбу подручными предметами из ружья (с ростом уровня он приделывает приспособления, расширяющие возможности "пушки"), выделывание из бесполезных предметов практически незаменимых (удушающих, пугающих, землетрясущих), работу мишенью для пинков в ближнем бою.

AAAND NOOOOW THEEEE MVP (most valuable person) of game — халфинг-бард (видимо, в угоду буму на "Властелина Колец" и на хоббитов). Бренчит, стучит, дудит, валяется полбоя в обмороке, машет ножичком, курит. Волшебная сила музыки делает невозможное элементарным. По сравнению с седьмой частью, бардов подняли на заоблачные высоты. А количество инструментов заставляет задуматься о создании сводного Доминусского струнно-духового оркестра (а может, это опять подлизывание под Толкиена с "музыкой Айнуров").

А где же шестой? А вот и наша бабочка! Фаери-маг, из-за моих ошибок — самый бесполезный персонаж в игре. Он настолько привык умирать, что воскрешение воспринимает как обыденность. Ну, кое-что он кастовать умеет, да и есть у меня страсть к полным партиям. А потому, хай жыве.

Теперь о NPC, которые провели со мной почти всю игру, и я уже не могу представить жизнь без них.

RFS-81 — робот-монах (для полноты картины не хватает только счетов-ниндзя или калькулятора-алхимика). Я его починил, и эта машина смерти (до 8 ударов за ход при большой загрузке партии) более не служит Dark Savant'у. Шао-линь удивляется, Жан-Клод Ван Дамм вешается на собственных носках, лишь только RFS принимает боевую стойку. Кроме рук, он может и палочкой лицо пощекотать.

Виталия — валькирия, которая так сильно участвовала в нашей судьбе в седьмой части. Здесь она лишь член команды, причем не самый полезный. Кастует, но неважно. Сражается, но хуже, чем любой другой воин. Так что в команде она лишь в память о Крестоносцах Дарк Саванта.

Даже обидно сознавать, что так превозносимый в седьмой части Астрал Домина в данной игре оказывается лишь одним из трех великих артефактов. Но ничего не поделаешь, именно сбору трех безделушек и посвящена основная часть игры, вернее, основная сюжетная часть.

Ибо основное содержание игры состоит в долгих прогулках по местности с занесением всего нападающего в красную книгу. Все эти,

бесцельные на первый взгляд, шатания совершаются не только для исполнения таинства сюжета, но в первую очередь для нахождения себя в этом мире. Чужаки на Доминусе (у игры четыре варианта начала, и по одному из них вы падаете на планету, подбитые Дарк Савантом), ваши подопечные пытаются обрести понимание своего места на этой прекрасной планете. Доминусу волей судеб предстоит стать ареной свершений вселенской значимости. Местные племена и такие же чужаки, как и вы, пытаются внести свою лепту в вашу и в свои судьбы. Эта игра стала первой, в которой приходится иметь дело со столькими организациями одновременно. А с кем из них сотрудничать — это уже ваше личное дело. От этих контактов сильно зависит и ваш путь к познанию себя.

Раз уж зашла речь об отношениях с персонажами, то не забудем и о NPC как о личностях (не всем же являться винтиками могущественных объединений).

Большинство из встречаемых вами дружественных персонажей немногословны. По-видимому, это из-за тягот суровой (солдатской, монашеской, жреческой, воровской) жизни. Те же, кто хочет поговорить, предоставляют вам возможность расспросить их о чем угодно. Но, вопреки задумке авторов, мне так и не удалось вести долгие беседы о судьбе мира или снадобьях в лавке. Большинство персонажей не настроены на длинные диалоги, и сильно ограничены в темах (хотя, это кому как). Некоторые из них могут присоединиться к вашей компании (правда, для каждого есть такие места, в которые они не хотят соваться, зато их вполне можно обмануть).

При восприятии игры важная роль отводится графике (удивительно, но это правда). Очень приятный для РПГ движок (хотя, возможно, кому-нибудь милей спрайты BG2). Он сумел взять на себя труды по передаче совершенно разных мест. Ему одинаково хорошо удается справляться и с гигантским деревом Тринтоном, и с болотами, и с подводным миром, да и на побережье никто не жаловался. Не 3D Action конечно, но неплохо. Вот только с монстрами у авторов прокол. Редкий монстр не похож на беглеца из кунсткамеры. Но, видимо, авторы догадывались о том, что же они написали, и потому очень часто попадаются насекомые (можно подумать, что здесь уже прошла ядерная война, и более совершенные существа захватывают мир). Но оставим это на совести ребят, сотворивших такое чудо.

Кстати, о птичках. Большую часть игры ваш экран будут занимать враждебно настроенные личности. Особенность игры подбрасывать монстров в просто неимоверных количествах, отвратила многих, кто начал играть. Согласен, что бой в этой игре даст фору большинству файтинговых систем. Пофазовая система боя дает простор для творчества и порождает комбинации вроде такой — "ты защищаешь мага, ты играешь на лютне, чтобы свести врага с ума, ты бьешь вот эту образину, ты колдани защитное заклинание, а ты накрой фаэрболом задний ряд". Причем победная тактика имеет много вариантов, и в следующий раз вы вполне можете кардинально ее изменить. Привычка думать о воинах как об ущербных личностях не поможет. Все амбразуры ими не закроешь, а вот поотшибать тянущиеся к закромам родины руки можно с огромным успехом. Инженеры и барды становятся в один ряд с магами и жрецами, а часто и делают три шага вперед по сравнению с оккультистами и чернокнижниками. Ведь манна — это хрупкая материя, которая заканчивается с катастрофической скоростью, а выносливость, на расходовании которой построены действия инженеров и менестрелей — штука менее хрупкая, к тому же, быстро восстанавливающаяся. Поэтому в плане ВБП (валового боевого продукта) создание манны сильно проигрывает товарищам из дружественной предметно-использующей компании. К монстрам, между прочим, понятие манны неприменимо, так как колдуют они на полную катушку при любом раскладе.

Так вот, если вернуться к бою как к части игры, а не как к самостоятельному продукту, то у него есть слабости. Ну почему (слово "почему" было предложено моим спонсором для прохождения цензуры) сражения занимают до 80% игрового времени? Прелести данного времяпрепровождения вполне можно осознать и за меньший срок. Именно это и является главным минусом игры. Ведь что же еще? Графика погружает, общение занимает, бой (в разумных пределах) вызывает. Сюжет же подобен дуновению. Вот вы сидите в лодочке, плывущей по тихой узкой речке, вы засыпаете, не беспокоясь за свою судьбу. А пробуждение застает вас уже в предштормовом море, и вы наедине с ним...

P.S. Засим позвольте откланяться и оставить вас с этим великолепием, играть в которое можно практически бесконечно.

Рисунок О.Попова

КУПЛЮ, ПРОДАМ, ОБМЕНЯЮ

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

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

передавать по тел. в Минске ***(017) 221-01-10,***
либо через E-mail:

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

Продаю видеокарту S3 Savage3D (8Mb, AGP 2x, TV-out — композит и S-video), поддержка оверлея, поддержка 3D, диск с последними дрейверами.
Тел. в Минске 258-36-33.

Куплю б/у CD-ROM.
E-mail: taras000@atlasua.net

Бесплатно **вышлем** каталоги радиосхем, радиодеталей, книг, радиоконструкторов, видео- и аудиокассет, дискет с подборкой полезных программ и каталог на CD по радиолюбительству. К заявке на каталог необходимо прикладывать оплаченный конверт с обратным адресом.
422981, Татарстан, г.Чистополь, а/я 248,
Романов А.А. (B.R.S.)

Куплю контроллер жесткого диска для ПК "Истра 4816.03", программное обеспечение для ПК "Искра-1030М".
453500, Башкортостан, г.Белорецк, а/я 21,
Сафонов С.Н.
Тел. (34792) 4-45-67.

Ищу схему и описание EPSON-совместимого принтера EC-7245 (применялся в КУВТ "КОРВЕТ").
E-mail: zgo@mail.chita.ru

Куплю: игровую приставку "Panasonic 3DO" модели FZ-10; видео-CD-адаптер (68 pin x 1); игровые компакт-диски системы 3DO; джойстики; пистолеты; мышь, можно от других приставок (GoldStar 3DO, Sonyo 3DO и т.д.) с логотипом 3DO; карту Creative Labs 3DO Blaster и другие устройства для "Panasonic 3DO".
681000, Хабаровский край,
г.Комсомольск-на-Амуре,
ул.Краснофлотская, 26 — 42, Решетник К.П.

Куплю для ПК "Scorpion ZX-256 turbo" плату GMX, контроллер SMUS, контроллер IBM-клавиатуры и Kempston-mouse с документацией, а также ПО.
681090, Хабаровский край,
Комсомольский р-н, ст.Гурское,
ул.Владивостокская, 4 — 1, Курбатов Н.В.
Тел. 3-91.

Ищу программу или описание способа установки и изменения элементной базы для программы-симулятора "Electronics Workbench EDA".
191123, г.С.-Петербург, а/я 12, Ильин А.

Приму в дар любой компьютер и комплектующие.
456430, Челябинская обл., Уйский р-н,
п.Вишневка. Бичан С.

Школьник с большой благодарностью **примет в дар** любые комплектующие к IBM-совместимым компьютерам и цифровым видеокамерам.
Беларусь, г. Барановичи,
ул. Фабричная, 14 — 84.

Приму в дар компьютер или комплектующие.
E-mail: zkr@freemail.ru

Приму в подарок б/у компьютер без монитора.
40034, Украина, г. Сумы, ул.Коротченко, 15/311.
В.Стороженко.

Бесплатно высылаю схемы на любые темы и любой сложности, с подробным описанием и рисунком печатной платы. Для получения каталога схем высылайте конверт с обратным адресом.
357081, Ставропольский кр., Андроповский р-н,
ст.Воровсколесская, ул.Центральная, 91.
Ширяев А.

Куплю контроллер дисковода "Электроника МС8414" для ПК "Электроника МС1502".
E-mail: mishsv@om.msi.ru

Требуется информационная поддержка по ремонту CD-ROM, CD-RW, DVD-ROM (только от жителей г. Минска).
Тел. 8-029-660-58-11,
Сергей.

Ищу инструкцию по переделке игровой приставки Nintendo в генератор телевизионных испытательных сигналов.
Тел. в г.Бобруйске (8-0225) 55-41-64.
E-mail: ox68@rambler.ru

Уважаемые читатели!

Подписаться на наши журналы (индексы: **48996** —"Радиомир", **48924** — "Радиомир. КВ и УКВ", **48925** — "Радиомир. Ваш компьютер") во II полугодии 2002 г. можно по каталогу Агентства "Роспечать", **стр. 235**, или по каталогу РО "Белпочта" "Газеты и журналы Республики Беларусь", **стр. 124** (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	Имеющиеся в наличии номера			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	3, 5, 8	1 — 5, 8 — 12	1, 6, 7, 11, 12	25	800	6
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	1 — 3, 5, 6	30	1000	7
2000	2 — 5, 7 — 12	1 — 8, 10 — 12	4 — 7, 9 — 12	35	1200	8
2001	2 — 6	1 — 6	2 — 6	40	1400	8,5
	Радиомир	Радиомир. Ваш компьютер	Радиомир. КВ и УКВ	По России и СНГ (рос. руб)	По Беларуси (бел. руб.)	По Украине (гривны)
2001	7 — 12	7 — 12	7 — 12	40	1400	9
2002	1 — 12	1 — 12	1 — 12	45	1500	10

Наши платежные реквизиты:
- для жителей России и стран СНГ (кроме Беларуси) —
получатель: ООО "Радиомир Пресс", ИНН 7729404741,
р/с 40702810900010000084 в ООО КБ "Агропромкредит", ф-л Центральный, г.Москва,
корр. счет 30101810500000000109, БИК 044525109
Адрес банка: 113054, г.Москва, ул.Зацепа, 43Б;
- для жителей Беларуси —
получатель: УП "РЛД", УНН 190218688
р/с 3012000004882 в ф-ле №524 АСБ "Беларусбанк" в г.Минске код 121.
Адрес банка: 220028, г.Минск, ул.Физкультурная, 31.

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 платежным поручением нужно предварительно выписать счет.

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

Приобретение отдельных номеров журналов

В РОССИИ:

В **ЗАО "Интерпресс — Экспресс"**: тел. в Москве (095) 208-85-55, 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
Радиорынки: Царицынский (место 13/А), Митинский (ряд 8, контейнер 12, место Z25).

В **"Фонде содействия радиолюбителям-радистам"**:
г.Москва, 4-й Войковский проезд, 10, тел. (095) 150-09-72, 150-04-16 (ст. метро "Войковская").

На УКРАИНЕ:

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

В БЕЛАРУСИ:

В Минске в магазине **"Книга XXI век"**, пр.Ф.Скорины, д.92, тел. (017) 264-27-97 (ст.метро "Московская").


VENI,
VIDI,
VICI
VENI,
VIDI,
VICI