9•2002

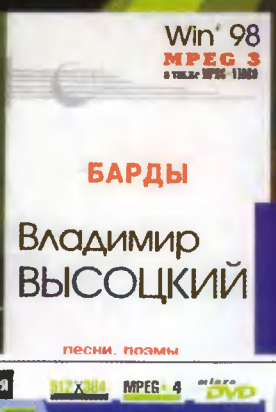
http://radio-mir.com

радиомир

Ваш компьютер

Индексы: 48925, 45995

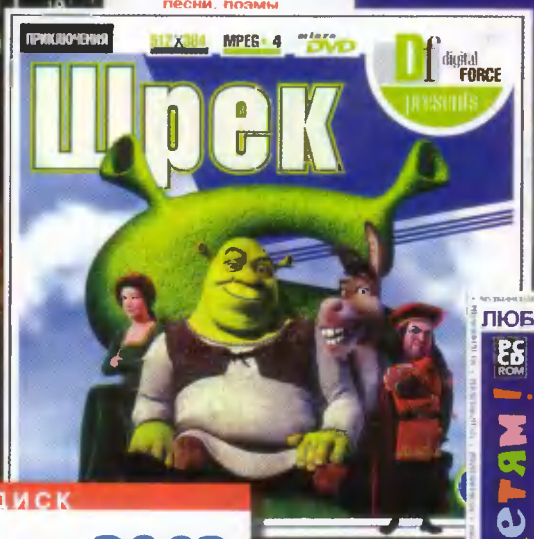
В НОВЫЙ ВЕК —
С НОВЫМ НАЗВАНИЕМ!



SONAR

ABBYY LINGVO Multilingual Edition
FINEREADER Professional

macromedia
FREEHAND 10



TeachPro Internet
ОБУЧАЮЩАЯ СИСТЕМА ПО
INTERNET

+ Подарок в каж

ЗАГРУЗОЧНЫЙ ДИСК
Система 2002
русская версия
Microsoft Windows XP Professional
+
Windows 98 Second Edition русская версия
Windows Millennium Edition русская версия
DirectX 8.1 Final для Win9x/98/Me русская версия

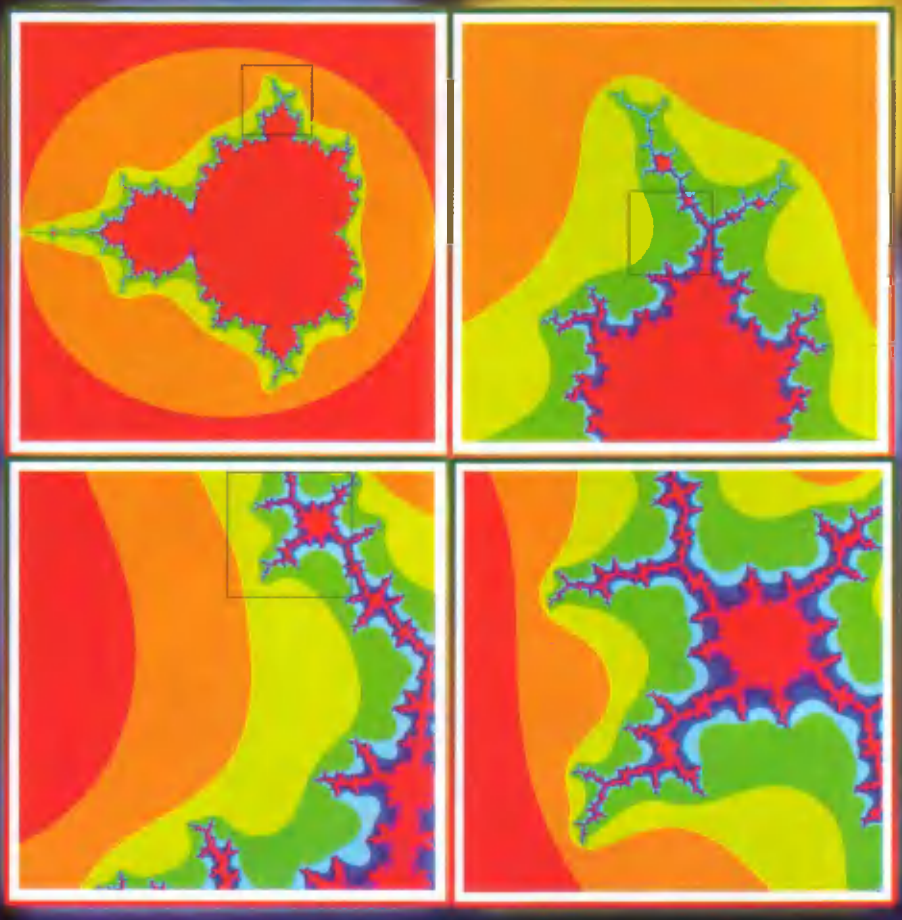
SQL ДЛЯ ПРОФЕССИОНАЛОВ

Все, что вам может потребоваться для изучения, разработки и отладки баз данных на SQL на ОДНОМ CD.

Microsoft SQL Server 2000 Developer Edition
Microsoft SQL Server and MSDE Deployment Toolkit v1.1
Sybase SQL Server Professional
Access 97 + SQL Server 7.0 + Crystal Reports 7.0
4 книги для изучения языка
7 компьютерных изучающих курсов
5 тренировочных экзаменов на получение сертификатов MSSQL
+ утилиты (включая отладочные от SYSInternals: FileMon, RegMon, etc)

А.Гринчук.

Работа в системе Mathematica 3.0/4.0. Примеры применения



Фрактал множества Мандельброта при различной степени детализации

Два варианта фрактала Джулиа

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

Главный редактор
Ольга СТРЫЖАНКОВА

Зам. гл. редактора
Иван БЕЛЬСКИЙ

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

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

Компьютерная верстка —
Янина БЕЛЬСКАЯ

Техническая графика —
Неталья 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: rl@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 мвтериалам 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 26.07.2002 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная.

Отпечатано в типографии
ЗАО "Красногорская типография".
Заказ 1951.

радиомир
Ваш компьютер

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

№9/сентябрь/2002

ЧИТАЙТЕ В НОМЕРЕ

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

С микропроцессором по жизни

Через двадцать лет компьютеры могут сравняться по мощности с человеческим интеллектом.

Развитие информационных технологий будоражит человеческий ум уже добрых полвека. Компьютеры прочно вошли в нашу повседневную жизнь. Работа в современном офисе немыслима без Интернета, электронной почты, а заслуженный отдых для многих начинается только тогда, когда включается игровая приставка. Мобильные телефоны третьего поколения теперь не только передают голос, но и легко заменяют практически любое офисное оборудование. Появились даже автомобили с бортовыми компьютерами, которые могут составить маршрут вашей поездки и доставить вас до точки назначения.

А началось все в далеком 1943 г., когда в Великобритании была создана вычислительная машина Colossus, предназначенная для раскрытия военных шифров. Год спустя в США был создан Mark-1 — первый программно-управляемый компьютер, который стал доступен широкой общественности. А еще через два года американские инженеры представили электронно-цифровой компьютер "Эниак" (ENIAC — Electronic Numerical Integrator and Calculator), в 1000 раз более быстродействующий, чем Mark-1. В "Эниаке" было 18000 электронных ламп, и теснились они в шкафах общим объемом около 100 м^3. Устройство такой же вычислительной мощности, но значительно меньших размеров удалось создать лишь спустя 30 лет — 11 ноября 1971 г. компания Intel предложила процессор, в котором 2300 транзисторов уместились на схеме размером с ноготь. Выпущенный компанией Intel первый в мире микрочип выполнял 60000 операций в секунду — ничто по современным меркам, но тогда это был серьезный прорыв. Вскоре после процессора 4004 корпорация Intel представила микропроцессор 8008, который обрабатывал за один цикл 8 бит информации, т.е вдвое больше, чем первая микросхема.

С тех пор вычислительные технологии шагнули далеко вперед. Например, подсчитано, что за 30 лет существования микропроцессоров минимальный размер элементов процессора уменьшился в 17 раз, тогда как количество транзисторов увеличилось в 18000 раз, а тактовая частота возросла в 14000 раз. Нынешняя технология производства процессоров, применяемая корпорацией Intel, настолько совершенна, что размеры транзистора сопоставимы с молекулой, а в будущем будут иметь ширину нескольких атомных слоев.

Выпущенный в конце 2000 г. процессор Intel Pentium 4 не только имеет в своей основе революционную микроархитектуру Intel Netburst (набор уникальных технологий обработки видео- и аудиоматериалов, благодаря которому самые современные средства Интернета и отображения трехмерной графики достигают невиданного уровня быстродействия), но и содержит рекордное (в последних моделях — 55 миллионов) число транзисторов. Появление этого процессора, как, впрочем, и всех остальных процессоров Intel, согласуется с законом, который вывел в 1965 г. Гордон Мур — один из основателей компании Intel. Этот закон гласит, что новые модели микросхем появляются каждые полтора-два года, а плотность размещенных на них транзисторов и, соответственно, быстродействие при этом возрастают примерно вдвое. Выведенный первоначально как чисто эмпирическое правило, закон Мура вскоре стал для всей отрасли персональных компьютеров руководящим принципом в создании все более и более мощных полупроводниковых микросхем по все меньшей стоимости.

Индустрия информационных технологий — одна из наиболее динамично развивающихся сфер жизни. В соответствии с законом Мура, в 2020 г. компьютеры достигнут мощности человеческого мозга, т.к. смогут выполнять 20 квадриллионов (т.е. 20000000 миллиардов) операций в секунду, а к 2060 г., как считают некоторые футурологи, компьютер сравняется по силе разума со всем человечеством. Впрочем, еще в 1994 г. ПК на базе процессора Intel Pentium со смехотворной по нынешним временам частотой 90 МГц обыграл в серии турниров по шахматам нескольких сильнейших гроссмейстеров мира, включая действующего чемпиона планеты — Гарри Каспарова.

Тот же Гордон Мур в середине 90-х годов так сравнивал темпы развития микропроцессорных технологий и автомобильной промышленности: "Если бы автомобильная промышленность развивалась с той же скоростью, что индустрия полупроводников, то "Роллс-Ройс" смог бы сегодня преодолеть расстояние в полмиллиона миль на одном галлоне бензина, причем его было бы дешевле каждый раз выбрасывать, чем парковать".

Продолжая автомобильную тематику и сравнив быстродействие микропроцессоров с болидом "Формула-1", можно прийти к следующим результатам. Если приравнять тактовую частоту процессора Intel Pentium 4 (для примера возьмем выпущенную в январе 2002 г. модель с частотой 2,20 ГГц) к оборотам в минуту двигателя болида "Формула-1", то при 2200000000 оборотах в минуту за одну секунду поршни мотора прошли бы расстояние в 3055 км; при этом за ту же секунду в мотор поступило бы 55 млн. литров воздуха, а колеса вращались бы со скоростью более 6 миллионов оборотов в секунду. Сам болид при таких оборотах смог бы развить скорость 44 миллиона километров в час, что в 147 раз превосходит скорость света. Если бы существовала возможность развить и поддерживать такую скорость, то до ближайшей от Солнца звездной системы Альфа Центавра (находящейся на расстоянии 4 световых лет) можно было бы добраться всего за 10 дней.

Конечно, информационные техно-

логии пока не способны даже приблизить автомобиль к таким уровням скоростей. Тем не менее, они уже сегодня способны на многое. В последнее время активно развивается разработка телематических терминалов (бортовых систем управления) для автомобилей. По данным аналитической компании Forrester Research, к 2006 г. телематическими терминалами для обработки и передачи информации будет оборудовано около 80 процентов от общего числа новых машин. "Автолюбители уже успели оценить все преимущества таких телематических средств первого поколения как скорая помощь в чрезвычайных ситуациях на дорогах и базовые навигационные системы, и это только начало, — говорит Патрик Керриган (Patrick Kerrigan), директор по маркетингу подразделения Intel Telematics Operation. — Новые процессоры Intel на базе технологии Intel XScale обеспечивают наращивание производительности и масштабируемость вычислительных средств, применяемых в широком спектре продукции — от встроенных гарнитур hands-free для сотовых телефонов до высококлассных мультимедийных информационно-развлекательных систем".

Уже сегодня существуют реальные возможности применения такого рода технологий практически в любом автомобиле. Например, телефонная гарнитура BlueConnect производства компании Johnson Controls, Inc. — интегрированный автомодуль hands-free на базе процессоров Intel PXA250 и Intel PXA210 — позволяет водителю выполнять самые разнообразные действия, активизируемые голосом, с помощью сотового телефона и технологии Bluetooth.

Еще одним устройством, в котором применены новые процессоры, является мультимедийная автомобильная платформа, которая предоставляет пассажирам автономный доступ к таким ресурсам как видео в формате DVD и аудиозаписям в формате MP3, транслируемым по сети Media Oriented System Transport (MOST).

Автомобилестроение — только одна из многих сфер жизни, где микропроцессоры занимают все большее место. Очевидно, что с каждым годом все более мощные микропроцессоры будут применяться во все большем количестве различных бытовых устройств. Недавно специалистами Intel были разработаны транзисторы, скорость действия которых превышает скорость самого быстрого современного процессора Pentium 4 почти на 1000%. Тем самым доказано, что нет никаких фундаментальных препятствий для продолжения развития микропроцессоров в соответствии с законом Мура до конца текущего десятилетия.

Такие транзисторы, имеющие размер всего 20 нанометров, позволят компании Intel к 2007 г. создать процессоры с миллиардом транзисторов, работающие на частоте до 20 ГГц при напряжении питания около 1 В. А руководство компании уже вслух говорит о грядущих процессорах с тактовой частотой до 30 ГГц. Предпосылки для производства таких микропроцессоров в Intel уже созданы.

По материалам www.intel.ru

Возвращаясь к напечатанному

("РМ.ВК", №4/2002, С.2)

В апрельском номере журнала редакция предложила читателям найти неточность в статье Н.Лутковской и В.Пономарева "Поколения ЭВМ: четыре, пять или шесть?".

Неточность состояла в том, что в статье годом выпуска первой мини-ЭВМ PDP-8 считается 1960 г., как указано в одном из интернетовских источников. Но в статье Прохорова "Итоги тысячелетия, столетия, года" (Компьютер, 2000, №1, С.9) указан 1965 г., что более вероятно. На рис.2 это событие отмечено где-то между 1960 и 1965 годами. В литературе также есть разные мнения относительно года появления ЭНИАК (от 1943 до 1946 гг.) и МЭСМ (1950 и 1951 гг.).

Читателям, указавшим на эту неточность были обещаны призы. Сегодня мы подводим итоги.

Победителем этого своеобразного конкурса признан Андрей Поплетеев из Минска (E-mail: Popleteev@tut.by), обнаруживший помимо упомянутой еще три неточности:

1. Авторы пишут: *"В действительности поколение персональных компьютеров с обычной архитектурой фон Неймана оказалось живучим, не торопилось уступать дорогу новому и продолжало развиваться. Доказательством тому служит компьютер с процессором Pentium, на котором редактировался этот текст."*

В 60...70-х годах фирмой NEC была создана ЭВМ PDP-11, ключевой особенностью которой являлось использование магистральной архитектуры. Ввиду ее явного преимущества перед неймановской, все последующие разработки используют именно магистральную архитектуру [1, С.57]. В том числе и современные компьютеры с процессором Pentium :).

2. Авторы утверждают, что ЭВМ MARK-1 появилась не раньше 1948 г. Однако в соответствии с [2, С.32], эта ЭВМ была сконструирована значительно раньше — в 1944 г, т.е. раньше, чем ENIAC.

3. В архитектуре фон Неймана центральный процессор напрямую не связан с устройствами ввода-вывода. [1, С.33].

Литература

1. А.Н.Жигарев, Н.В. Макарова, М.А.Путинцева. Основы компьютерной грамоты. — Л.: Машиностроение, 1987, 255 с..

2. Г.А.Бордовский, В.А.Извозчиков, Ю.В. Исаев, В.В.Морозов. Информатика в понятиях и терминах. — М.: Просвещение, 1991, 208 с.

Таким образом, самым внимательным читателем мы признали А. Поплетеева. В качестве приза он получит компакт-диск "Искусственный интеллект на твоем компьютере ".

Ко всем читателям журнала мы обращаемся с вопросом: "Какой приз вы хотели бы получить в качестве самого внимательного читателя?"

Мы планируем объявить новый конкурс, когда получим достаточное количество предложений.

Пишите нам!

А.ГРИНЧУК,
г.Минск, РИВШ БГУ,
E-mail: gav@nihe.niks.by

Работа в системе Mathematica 3.0/4.0.
Примеры применения

Этой статьей мы завершаем наше краткое знакомство с системой компьютерной алгебры Mathematica 3.0/4.0. Поэтому сейчас перейдем от изучения каких-то частных возможностей системы к испытанию ее на ряде достаточно известных задач. Эти задачи возникают при исследовании сложных систем, поведение которых характеризуется сложностью и хаотичностью. Это сравнительно новая дисциплина, окончательное формирование ее относят обычно к 80-м годам прошлого столетия.

Начало систематических исследований в области "хаоса" обычно связывают с именем Эдварда Лоренца. В 60-х годах XX века он занимался метеорологическими расчетами на одном из самых мощных на то время компьютеров. В одной из серий вычислений Лоренц, для экономии времени, решил не производить все вычисления заново, а часть данных ввести с распечаток предыдущих вычислений. К своему удивлению, он заметил, что на этот раз компьютер выдал совершенно другой прогноз погоды. Выяснилось, что вычисления велись с 8-ю значащими цифрами, а при печати указывалось 5. И эта, небольшая на первый взгляд разница, привела к значительному конечному расхождению. Земная атмосфера проявила себя в высшей степени нестабильной системой — небольшие изменения приводят к большим последствиям. В результате были похоронены всякие надежды на долгосрочные прогнозы погоды, так как в любых данных содержатся ошибки, любой компьютер обладает ограниченной точностью вычислений и т.д. И до сих пор метеорологи не в состоянии составить надежный прогноз погоды более чем на несколько дней.

Однако сюрпризы на этом не закончились — долгое время считалось, что нестабильное поведение характерно только для больших и достаточно сложных систем, таких как земная атмосфера или океанические течения. Лоренцу удалось найти достаточно простую систему из трех дифференциальных уравнений, обладающую довольно необычными свойствами. На **рис.1** вы можете увидеть, как выглядит эта система, и график ее решения. Существует две точки неустойчивого равновесия. Система после нескольких оборотов по расходящейся спирали "прыгает" к другой точке, и этот процесс ("раскручивание" спирали и прыжок) повторяется. Но количество оборотов и время прыжка очень чувствительны к начальным условиям, поэтому на практике можно предсказать только общую картину поведения, но не детали.

Встроенные в систему Mathematica численные методы справляются с решением данной системы. Однако используемое число шагов при решении дифференциального уравнения по умолчанию не превышает 1000. Для увеличения этого параметра используется опция **MaxSteps→ 8000**, а для более гладкого изображения кривой используется 2000 точек — **PlotPoints→2000**.

Модель Лоренца оказалась достаточно "плодотворной" для описания реально существующих в природе систем. Самопроизвольные и практически непредсказуемые переходы между состояниями неустойчивого равновесия обнаружились как в вытекающей из крана струе воды, так и в изменении биржевых котировок. Вполне возможно, что наступление и ледниковых периодов, и периодов глобального потепления не требуют для своего объяснения каких-либо дополнительных предположений — просто это типичное поведение сложной системы.

Однако остался открытым еще один вопрос. В природе можно найти примеры как предсказуемых систем, так и таких, к которым понятие "определенность" просто неприменимо. Так как же "спокойная" линия поведения превращается в хаотическую и случайную? И здесь приняли эстафету Роберт Мэй и Митчелл Фейгенбаум. Мэй исследовал биологические системы. Для описания роста биологических популяций часто применяют простую модель — $x_{n+1}=\lambda x_n$, т.е. прирост (а следовательно, и число животных на следующий год — x_{n+1}) прямо пропорциональны численности популяции. Мэй использовал немного более сложный вариант, учитывающий ограниченность природных ресурсов — $x_{n+1}=\lambda x_n(1 - x_n)$. Далее исследовалось, что же произойдет при изменении параметра λ. При малых значениях λ не было никаких сюрпризов — численность популяции быстро стабилизировалась со временем и оставалась постоянной. Однако при значении λ больше 3 поведение стало более сложным. Например, при λ=3.1 переменная x, по прошествии некоторого времени, совершает прыжки между двумя значениями: 0.558014, 0.764567; 0.558014, 0.764567; 0.558014, 0.764567...

Попробуем изучить этот вопрос немного подробнее с помощью Mathematica. В первую очередь, определим функцию **f[x_]:= λ x (1-x)**. Далее необходимо подождать, пока система стабилизируется. Для этого используется команда **Nest**, которая многократно применяет функцию к аргументу: **Nest[f, 0.5, 1]** —

Рис. 1

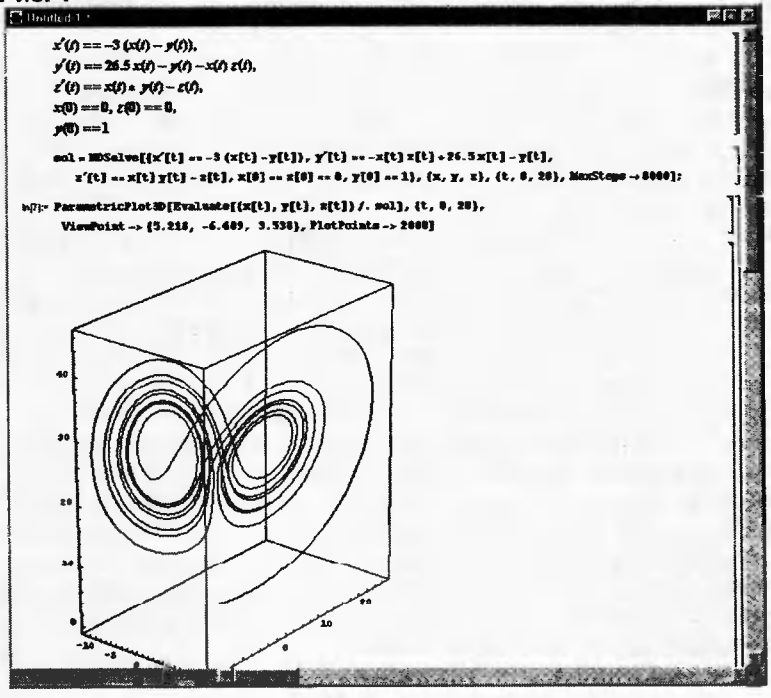


Рис. 2

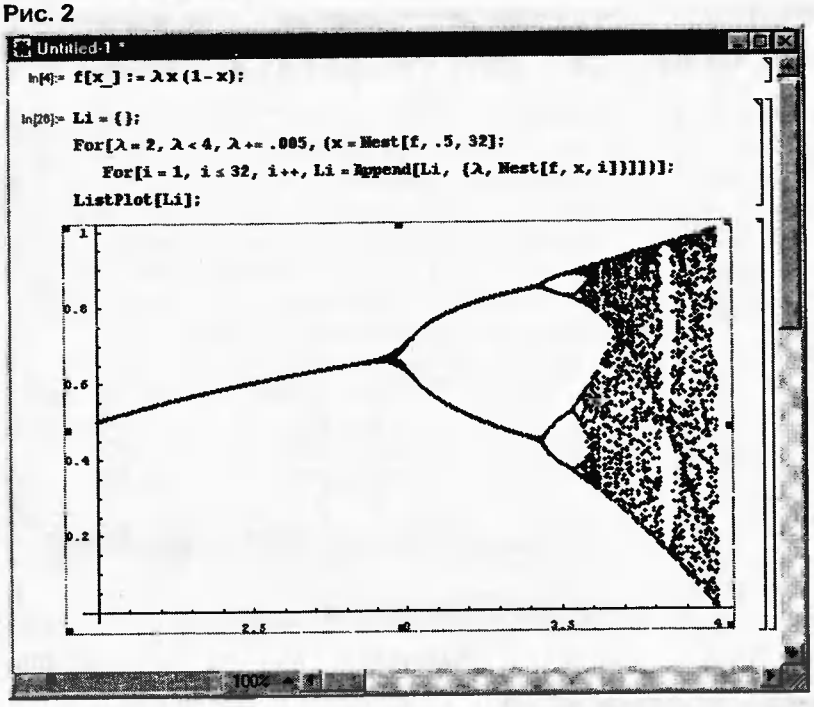
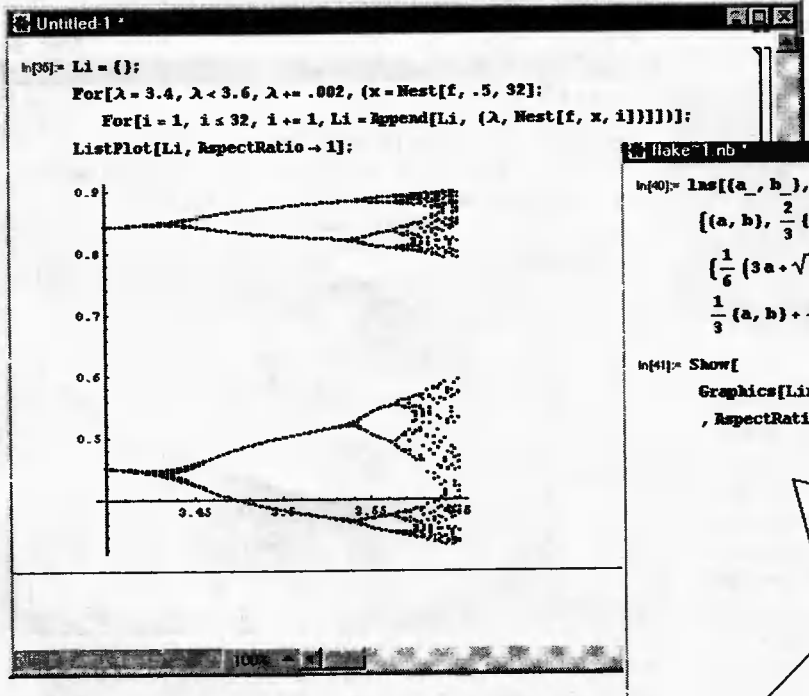


Рис. 3

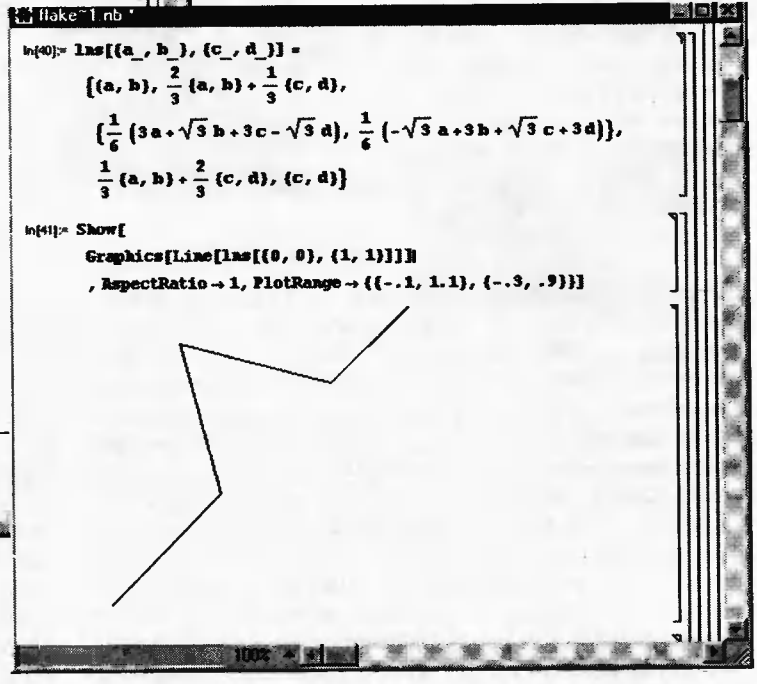


Фейгенбаум открыл эту закономерность, пользуясь программируемым калькулятором.

При более подробном рассмотрении (**рис.3**) можно заметить еще одну закономерность — участки ветвящейся линии повторяют в какой-то мере всю картину целиком. И с развитием компьютерной техники и компьютерной графики в руках ученых появился мощный инструмент исследования таких самоподобных структур. Наиболее известной из них является так называемая снежинка Коха. Ее построение начинается с равностороннего треугольника. На каждой стороне берется средняя треть, на которой строится равнобедренный треугольник. Затем этот отрезок удаляется и все повторяется многократно. Для осуществления этой операции на языке системы Mathematica нам в первую очередь понадобится команда рисования ломаной по заданному набору точек — **Line**. Основой построения является "наращивание" треугольной части на отрезке. Для этих целей можно создать функцию (у нас она названа **ins**), которая переводит отрезок с координатами начальной и конечной точек {a, d} и {c, d} в ломаную (**рис.4**). А для рисования линии применяется команда **Show[Graphics[Line[...]]]**.

Команда **ins** является основой процедуры **doub**, которая работает следующим образом. Внутри цикла **For** каждый отрезок заменяется ломаной, после чего команда **Partition** позволя-

Рис. 4



это f[0.5], **Nest[**f, 0.5, 2**]** соответственно — f[f[0.5]] и т.д. Командой **Nest[**f, 0.5, 32**]** мы отводим 32 шага на стабилизацию (**рис.2**), а затем добавляем к списку 32 точки For[i=1, i≤32, i+=1, Li=Append[Li, { λ, Nest[f,x,i]}]]. Все эти операции вкладываются в цикл For[λ=2, λ<4, λ +=.005, ...], благодаря которому мы изменяем значение параметра λ от 2 до 4 с шагом 0.005.

Из рис.2 видно, что "раздвоения" повторяются, и в конце концов наступает хаотический режим. Фейгенбаум выяснил, что расстояние между соседними критическими точками убывает по одной и той же геометрической прогрессии вне зависимости от конкретной функции f[x]. Теоретически это утверждение до сих пор не доказано, в его правоте пока убеждают только результаты компьютерного моделирования. Интересно, что сам

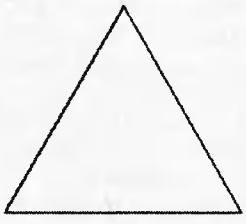
ет избавиться от лишних фигурных скобок внутри полученного списка (lst). Многократное применение команды tb=doub[tb] (**рис.5**) и позволяет получить все более и более сложную фигуру. На **рис.6** показан набор из четырех первых фигур.

Сама кривая получила название в честь шведского математика Хвльга фон Коха, который еще в 1904 г. описал ее свойства. Он показал, что длина кривой стремится к бесконечности, в то время как площадь внутренней части "снежинки" остается конечной.

Рис. 5

```
In[46]:= tb = {{0, 0}, {1/2, Sqrt[3]/2}, {1, 0}, {0, 0}};

In[48]:= Show[
           Graphics[Line[tb]]
           , AspectRatio → 1, PlotRange → {{-.1, 1.1}, {-.3, .9}}]
```

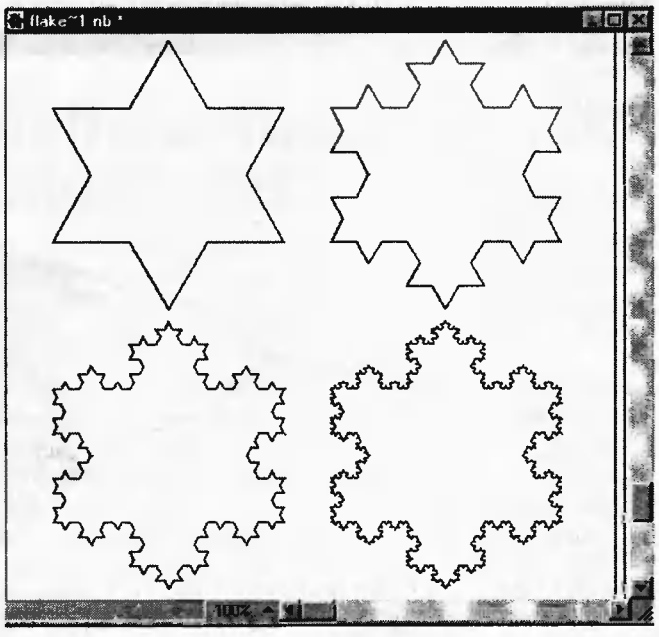


```
Out[48]= - Graphics -

         doub[tb_] :=
          Block[{lst = {}}, For[i = 1, i < Length[tb],
            lst = Append[lst, Line[tb[[i]], tb[[i + 1]]] ]; i += 1];
           lst = Partition[Flatten[lst], 2]; lst]

         tb = doub[tb];
         Show[Graphics[Line[tb]], AspectRatio → 1,
          PlotRange → {{-.1, 1.1}, {-.3, .9}}]
```

Рис. 6

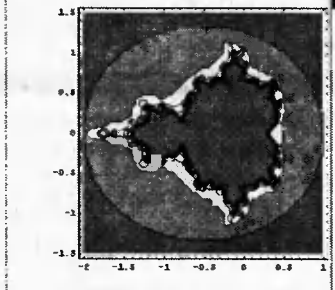


Без использования компьютера можно рассмотреть только простейшие фигуры с такими необычными свойствами. Пионером компьютерного моделирования "нестандартных" объектов является французский математик польского происхождения Бенуа Мандельброт. Именно он ввел в оборот термин "фрактал". Естественно, список самых известных фракталов открывает множество Мандельброта. Задать его можно следующим образом. На комплексной плоскости ($z = x + iy$) выбирается точка z_0. Затем строится последовательность $z_{n+1}=z_n^2+z_0$. Если при больших n точка не выходит за пределы круга радиуса 2 ($|z|=2$), то считается, что точка принадлежит множеству. Еще раз обратимся к программе Mathematica. Для подсчета количества шагов, нужных для выхода за пределы круга, можно воспользоваться циклом For[m=1, Abs[z] ≤ 2, m+=1, z=z^2+c];m]. Во избежание зацикливания ограничимся 30-ю шагами, для этого условие Abs[z] ≤ 2 заменим на And[m≤30, Abs[z]≤2] (**рис.7**). Остается только графически отобразить результаты вычислений с помощью команды **ContourPiot** (контурный график), которая за счет опции **ColorFunction → Hue** разными цветами выделяет точки с различным значением числа шагов m. При последовательном увеличении частей графика (это достигается за счет уменьшения отображаемой области: {x,-.4,0.1},{y,.7,1.2} и т.д.) можно увидеть, что граница при любом увеличении остается сложной, а более мелкие детали в какой-то мере повторяют крупные (см.2-ю страницу обложки).

Рис. 7

```
In[66]:= mnd[x_, y_] := Block[{c, z, m}, c = x + i y; z = c;
           For[m = 1, And[m ≤ 30, Abs[z] ≤ 2], m += 1, z = z^2 + c]; m]

In[67]:= ContourPlot[mnd[x, y], {x, -2, 1}, {y, -1.5, 1.5},
           PlotPoints → 72, ColorFunction → Hue]
```



```
Out[67]= - ContourGraphics -
```

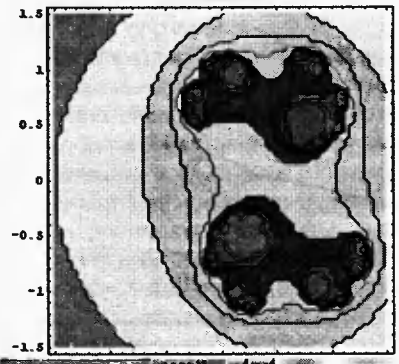
Рис. 8

```
In[71]:= c = 0.5 + 0.2 i

Out[71]= 0.5 + 0.2 i

In[72]:= mnd2[x_, y_] := Block[{z, m}, z = N[x + i y];
           For[m = 1, And[m ≤ 30, Abs[z] ≤ 2], m += 1, z = z^2 + c]; m]

In[74]:= ContourPlot[mnd2[x, y], {x, -2, 1}, {y, -1.5, 1.5},
           PlotPoints → 72, ColorFunction → Hue]
```



Если же при построении последовательности к z_n^2 прибавлять постоянное комплексное число с — $z_{n+1}=z_n^2+c$, то мы получим фрактал, названный в честь французского математика Гастона Джулиа (**рис.8**). В этой ситуации появляется большое количество вариантов за счет возможности изменения числа с (см.2-ю страницу обложки).

Однако для рисования достаточно детальных графиков потребуется не только увеличить количество точек (опция PlotPoints), но и запастись терпением. Рисование с числом точек, равным 288 (точнее 288 на 288), уже занимает примерно 3 минуты (Pentium 233). Программам, написанным на C/C++, для такого построения требуется всего несколько секунд. Так что приходится решать обычную дилемму — затратить несколько дней на написание программы, которая исполняется за несколько секунд, или же в течение получаса написать и отработать программу в системе Mathematica, которой требуется больше времени для вычислений.

В заключение нельзя не сказать несколько слов в защиту системы Mathematica. Во-первых, появление новых и более мощных процессоров и оптимизация алгоритмов наверняка уменьшат разрыв в скорости работы. Во-вторых, Mathematica обладает огромной гибкостью и набором готовых команд для математических расчетов. Все это дает большой простор для экспериментов, освобождая от рутинной работы. Конечно, каждый в этой ситуации будет сам делать свой выбор, однако попытаться поработать с системой Mathematica все-таки стоит.

ANSI-графика в архивах

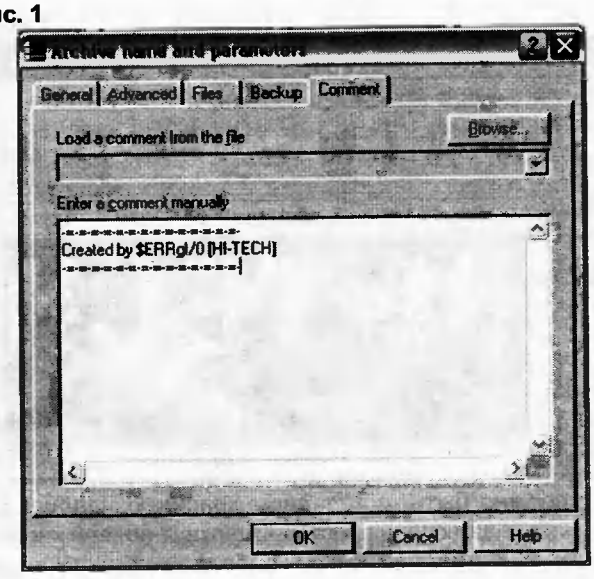
Serrgio /HI-TECH,
E-mail: serrgio@gorki.unibel.by
http://hi-tech.nsys.by/

Наверняка все вы умеете архивировать. Самые "продвинутые" даже знают, что на архив можно навесить **комментарий**. А вот о том, что этот комментарий можно сделать цветным, знает лишь старое поколение хакеров. Давайте исправим это досадное недоразумение и научимся делать **красивые комментарии** ;).

Все, описанное ниже, применимо для большинства архиваторов, однако, во избежание различного рода непоняток, уточню — в статье речь идет о WinRAR 2.80 (Evaluation Copy), текст файла комментария набивается в Volcov Commander при помощи текстового редактора F4 ;).

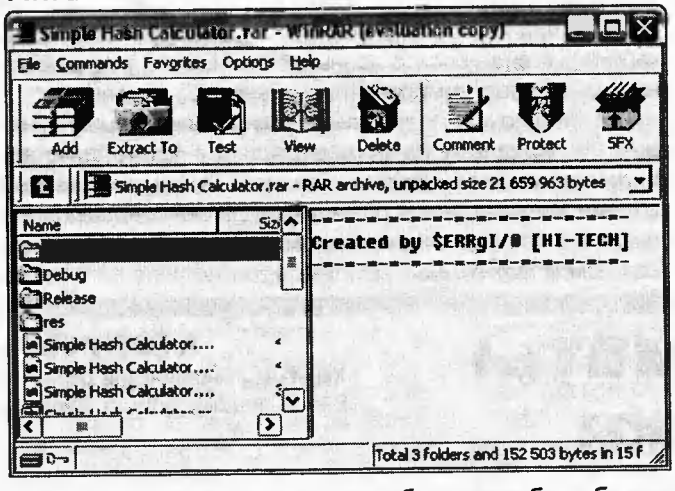
Итак, обычный комментарий можно "впаять" в соответствующем диалоговом окне архиватора (**рис.1**).

Рис. 1



При открытии архива WinRAR'ом он будет отображаться следующим образом (**рис.2**):

Рис. 2



Первое, что мы сделаем, это облегчим себе работу и вынесем текст комментария в отдельный файл *info.ans*. А в диалоговом окне в поле *Load a comment from the file* пропишем полный путь к этому файлу. Переархивируем наши файлы. Должна получиться та же самая картинка, что и в первый раз.

Теперь начинается самое интересное. Открываем файл с комментарием в текстовом редакторе F4, и сотворим с ним "страшное". А именно — добавим в начало текста "страшный" непечатный символ "стрелка влево", затем напишем "открывающую квадратную скобку", потом — число "36" и, наконец, букву "m" (**рис.3**).

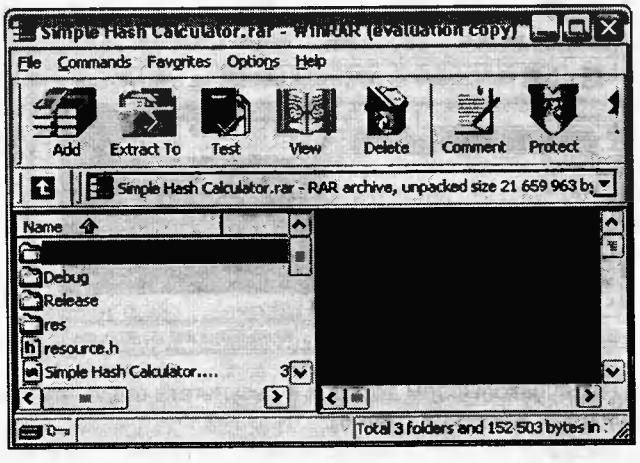
Рис. 3

Для тех, кто после долгих и упорных поисков так и не нашел на своей клавиатуре кнопку с такой стрелкой, подскажу, что ASCII-код этого символа — 27. А любые непечатные символы в большинстве встроенных текстовых редакторов набираются при помощи комбинации ALT+{код символа}. В нашем случае для написания этой "стрелки" необходимо:

- нажать на ALT;
- набрать "27";
- отпустить ALT.

После переупаковки архива и открытия его WinRAR'ом, мы увидим следующее (**рис.4**).

Рис. 4



То есть появился черный фон, помельчал размер символов и изменился их цвет. Получившаяся "картинка", безусловно, безобразна и намного хуже своего первоначального варианта, однако не спешите ругаться. Мы все поправим :)).

Вот табличка возможных атрибутов символов, которые мы можем устанавливать при помощи неудобноваримой "команды" <[<атрибут>m:

Атрибут символа		Атрибут фона	
Код	Цвет	Код	Цвет
30	Черный	40	Черный
31	Красный	41	Красный
32	Зеленый	42	Зеленый
33	Желтый	43	Желтый
34	Синий	44	Синий
35	Вишневый	45	Вишневый
36	Голубой	46	Голубой
37	Белый	47	Белый

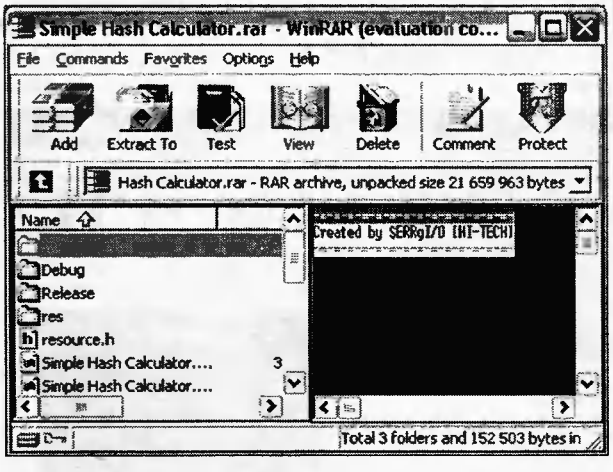
Первое, на что следует обратить внимание, это то, что при использовании подобного рода команд фон всего текста устанавливается черным! Это потому что, во-первых, во времена царствия DOS'а (и драйвера ANSI.SYS иже с ним) белые символы на черном фоне были стандартом. Во-вторых, шрифт стал моноширинным, то есть и символ "i", и символ "m" имеют одинаковую ширину в пикселах, в отличие от, скажем, того же Word. В-третьих, действие "команды установки атрибутов" распространяется на все символы, которые находятся от нее справа.

Как видно из таблицы, мы можем "управлять" не только цветом символа, но и цветом фона, на котором этот символ печатается. То есть в нашей команде между квадратной скобкой и символом m можно указывать два параметра, разделенные точкой с запятой — цвет символа и цвет фона соответственно.

Давайте попробуем. Изменим наш *info.ans* следующим образом (**рис.5**):

Рис. 5

Рис. 6



Переупаковываем архив, и в результате получаем следующую "картинку" (**рис.6**):

Расшифровываю. Первую строчку мы напечатали с атрибутами 37 и 45, то есть, согласно вышеприведенной таблице, белыми символами на вишневом фоне. Вторую, с атрибутами 30 и 46 — черными на голубом. Конечно же, зеленые на желтом (32, 43) символы в последней строчке — это не есть гуд, но я и не претендую на наличие художественного вкуса ;). Однако посмотрите на следующий рисунок — я сделал его всего лишь за три минуты (**рис.7**).

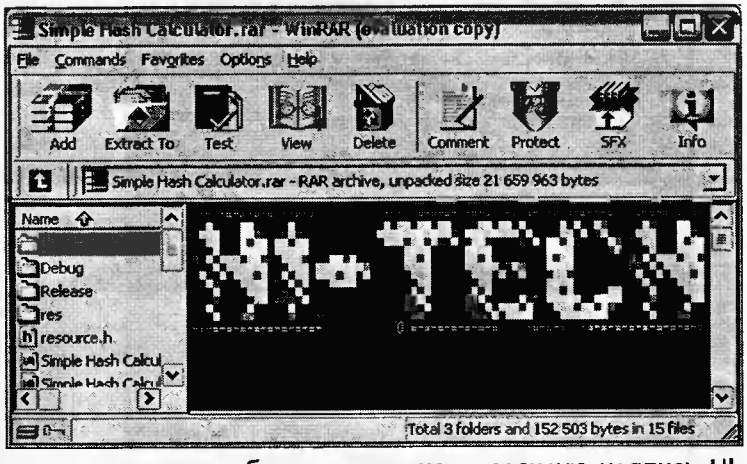
Разве он не восхитителен? А теперь прикиньте, что ваш приятель, которому вы отослали по почте свой архив, откроет его Winrar'ом, а там ТАКАЯ красотища!

Как я уже неоднократно говорил в своих статьях, программеры — народ ленивый, и набирать длинные простыни "стрелок" им, определенно, "не катит". Поэтому выдаю маленький секрет — в природе существует целая пачка так называемых ANSI-редакторов. Найти их можно на сайте http://thuglife.org/, а вот прямая ссылка на download самого известного из них (The Draw, v4.63) — http://www.thuglife.org/cgi/load/load.cgi?/_thug/apps/tdraw463.zip

Скачиваем, устанавливаем, запускаем... Вначале видим черное поле со статусной строкой внизу. А разве вы ожидали другого интерфейса от DOS'овской программулины?

Делаем щелчок правой кнопкой мыши, в результате чего вывалится целая куча всевозможных менюшек. Как вы

Рис. 7



уже, должно быть, догадались, красивую надпись HI-TECH я сделал очень просто — щелкнул правой кнопкой мышки, зашел в меню "foNts" и выбрал там "ColorRounded C". А потом набрал нужное мне слово на клавиатуре ;).

Сохранить полученную ANSI-картинку можно не только в формате ANS, но и в виде досовской com-программы, которая при запуске будет рисовать вам ANSI-картинку. Но наибольший интерес, по-моему, представляет сохранение картинок в виде блоков данных, "заточенных" под различные языки программирования — Pascal, C, Assembler...

Тех, кто хочет получить дополнительную информацию, по большому приколу отсылаю в ближайшую антикварную лавку искать книгу В.Л.Григорьева под странным названием "Самоучитель по операционной системе для персональных компьютеров". — М.: Энергоатомиздат, 1992.

Файловые форматы точечной графики

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

В настоящее время известны два основополагающих принципа представления графических изображений в оцифрованном виде — векторная графика и точечная графика. Точечную графику еще называют растровой или битовой. При использовании графических программных средств важно знать о свойствах, особенностях и областях применения файловых форматов точечной графики, понимать различия между ними. В данной статье сделан краткий обзор распространенных графических растровых файловых форматов.

В основе точечной графики лежит математическая модель — матрица чисел, описывающая цветовые пара-

метры каждой точки. Все точечные изображения представляют собой мозаику из элементов — пикселов. Пиксел (pixel — Picture Element) — это минимальный отображаемый элемент растра экрана монитора. Каждый пиксел имеет определенное местоположение в сетке битовой карты и цветовое значение.

Принцип кодирования графической информации в точечной графике был изобретен задолго до появления компьютерной техники. Это и рисование "по клеточкам" — продуктивный способ переноса изображения, это и такие направления в искусстве как мозаика, витраж и вышивка. В любом из этих случаев изображение строится из дискретных цветных элементов.

Для редактирования цифровых фотоизображений применяют графические редакторы. Их разнообразие среди современного программного обеспечения огромно — от простого редактора Paint (штатной программы Windows 9x/Me) до профессионального редактора Adobe Photoshop, являющегося мировым стандартом и самым популярным средством для редактирования изображений.

Основными достоинствами точечной графики являются фотореалистичность и простота реализации процесса ввода (оцифровки) изображений, полученных с помощью электронного фотоаппарата или сканера. К недостаткам точечной графики относятся сложности в трансформациях формы объектов, плохое сочетание изображений с текстами.

Рассмотрим наиболее распространенные файловые форматы точечной графики.

GIF (Graphics Interchange Format) — формат обмена графикой. Этот формат был разработан компанией CompuServe для обеспечения сжатия растровых файлов, загружаемых по телефонной линии и в компьютерных сетях. Этот файловый формат обычно используется для сохранения графических файлов перед помещением их в Интернет. При использовании формата GIF необходимо помнить, что данный формат не поддерживает более 256 цветов или 256 оттенков серого (8-битный цвет). GIF-файлы могут быть импортированы во многие графические программы PC. На Macintosh очень многие программы обработки и редактирования изображений имеют возможность сохранять и импортировать файлы в формате GIF.

TIFF (Tag Image File Format) — графический формат с метками. Большинство программных пакетов рисования, редактирования изображений и сканирования дают возможность сохранять данные в общедоступном файловом формате TIFF. Файлы этого формата имеют расширение ***.tif**, реже ***.tiff**. У этого универсального файлового формата имеется множество достоинств. Во-первых, он свободно переносится на разные компьютерные платформы. Это означает, что при сохранении файла в формате TIFF он становится доступен как для IBM-совместимых компьютеров, так и для Macintosh. Во-вторых, в одном файле может храниться несколько независимых рисунков (многостраничный рисунок). В-третьих, для уменьшения размера графических файлов можно производить сжатие без потери качества изображения, используя алгоритм LZW (Lempel-Ziv-Welch) Compression, который является безубыточной схемой сжатия. Это означает, что при сжатии данным способом не происходит ни малейшего ухудшения качества при существенном уменьшении размера файлов, и обеспечивается точное воспроизведение оригинала. Кроме этого, формат TIFF имеет открытую архитектуру и обладает широким диапазоном передачи цветов.

JPEG (Joint Photographic Experts Group) — объединенная группа экспертов по фотографии. Формат файлов JPEG (файловые расширения ***.jpg**, ***.jpeg**, ***.jpe**, ***.jif**, ***.jfif**) является широко используемым графическим форматом, особенно в сети Интернет. Как и формат GIF, формат JPEG отображается практически всеми Web-браузерами. Файловый формат JPEG вполне пригоден для сохранения полноцветных изображений. Сохранение в этом формате происходит со сжатием, однако, в отличие от формата TIFF, формат JPEG является убыточным файловым форматом. Чем выше степень сжатия, тем больше происходит потеря качества изображения.

BMP (bitmap) — битовая карта. Широко применяемый графический файловый формат для DOS и Windows. Для операционной системы OS/2 существует аналогичный формат OS/2 Bitmap. Поддерживает разное количество цветов, позволяет сохранять 24-разрядный, 256-цветный, 16-цветный и монохромный типы файлов. Графические файлы в формате BMP характеризуются значительными размерами, особенно при 24-битном цвете. Поэтому формат BMP не получил распространения в Интернете, т.к. на передачу файлов через медленные телефонные соединения будет затрачиваться много времени.

PCX — файловый формат, который в течение многих лет использовался для сохранения файлов в графических программах для PC. Он был разработан фирмой ZSoft для использования в операционной системе MS-DOS программой PC Paintbrush. Большинство программ для PC поддерживают 5-ю версию формата PCX. Файлы 3-й версии не поддерживают заказные палитры цветов. По этой причине при открытии PCX-файла 3-й версии палитра будет проигнорирована, и вместо нее будет использована стандартная палитра цветов VGA. Формат PCX впоследствии был вытеснен форматом TIFF, потому что первые версии формата PCX не поддерживали 16- и 24-битный цвета.

PSD — Photoshop Document. Собственный формат редактора Adobe Photoshop. Изображения в формате PSD состоят из нескольких слоев. Каждый слой может быть прозрачным относительно других слоев. Изображение в каждом слое редактируется отдельно от остальных.

TGA — Targa. Формат TGA очень широко используется в компьютерах, поддерживающих систему MS-DOS. Этот формат был изначально разработан для IBM-совместимых компьютеров.

Кроме вышеперечисленных файловых форматов, существуют еще множество других форматов точечной графики. Вот некоторые из них — PNG (Portable Network Graphics), IFF (Interchange File Format), PIC (SoftImage's PIC Format). Практически все растровые форматы могут быть конвертированы в любой из них. Пользователи Windows 9x/Me могут реализовать это как в графических редакторах, используя в меню **Файл** команду **Сохранить как ...**, так и в графических просмотрщиках, например, ACDSee 32. Пользователям MS-DOS для просмотра и конвертирования графических файлов можно порекомендовать программу Sea v1.3 фирмы Photodex Corporation (http://www.photodex.com).

КОМПЬЮТЕРНЫЕ ХРОНИКИ

Microsoft хочет отказаться от традиционной файловой системы

Можно ли хранить всю информацию в одной громадной базе данных? Microsoft заявляет: "Можно, и даже нужно". Компания считает, что настало время отказаться от традиционной файловой системы и заменить ее базой данных, в которой могла бы храниться информация всех типов. Это значительно облегчило бы задачу поиска необходимых данных и увеличило бы надежность всей системы в целом.

Правда, намерения эти далеко не новы. Первый раз подобная крамольная идея прозвучала из уст Джима Оллчина в 1992 г., но внедрение единой БД с тех пор постоянно откладывалось, а элементы новой ОС (под кодовым именем Cairo) внедрялись в другие версии Windows. Теперь о необходимости кардинального преобразования принципов хранения данных заговорил и CEO Microsoft Стив Баллмер. По его словам, дебют новой технологии (по крайней мере, ее части) состоится в очередной версии Windows под кодовым именем Longhorn, тестовая версия которой должна выйти в следующем году.

Главная проблема, встающая перед разработчиками — это необходимость коренной переработки всех ныне существующих приложений для Windows, чтобы они могли работать с новой файловой системой. Но если ее удастся решить в приемлемые сроки, то Microsoft сможет записать еще одно очко в свой актив, заставив при этом поморщиться Ларри Эллисона. Oracle также продвигает подобную технологию, но до сих пор никогда еще не предпринималось попыток полностью заменить файловую систему базой данных в массовой операционной системе.

Большинство домашних компьютеров заражены вирусами

Сотрудники издания Consumer Report в ходе проведенного среди своих подписчиков опроса пришли к выводу, что владельцы домашних компьютеров в США уделяют недостаточное внимание защите от вирусов и хакерских атак. Подробные результаты исследования опубликованы в июньском номере Consumer Report, а о его основных выводах сообщил сайт Newsbytes.

Оказалось, что в течение последних двух лет свыше 60% опрошенных обнаруживали в своих ПК хотя бы один вирус. Около 20% сталкивались с вирусами, по крайней мере, четырежды, и почти треть из их числа теряли свои файлы вследствие заражения.

Особо отмечается, что при использовании антивирусных программ информационный ущерб вирусы наносили лишь в 7% случаев, тогда как без антивируса ущерб проявлялся на каждом третьем компьютере.

Главным источником вирусов является, безусловно, Интернет. При этом доля заражений по электронной почте достигает 62%, а в 13% случаев заразу несли в себе файлы, загружаемые из Интернета.

Крайне уязвимы домашние компьютеры и для хакеров. По данным опроса, 40% пользователей широкополосного Интернета не пользуются брандмауэрами, т.е. их компьютеры практически открыты для проникновения хакеров и не защищены от троянских программ.

Intel — роадмап на год

Компания Intel представила свой официальный роадмап по выпуску новых моделей процессоров семейства P4 на период с первого квартала 2002 г. до первого квартала 2003 года включительно.

Во втором квартале 2002 г. выпущены процессоры P4 Northwood 533 МГц с частотами 2,53 ГГц, 2,4 ГГц и 2,26 ГГц; P4 Northwood 400 МГц с частотой 2,2 ГГц; P4 Willamette Celeron 400 МГц с частотой 1,8 ГГц.

В третьем квартале текущего года картина будет следующей — P4 Northwood 533 МГц с частотами 2,66 ГГц и 2,6 ГГц; P4 Northwood 400 МГц с частотой 2,5 ГГц; P4 Willamette Celeron 400 МГц с частотой 1,9 ГГц.

В четвертом квартале появятся: P4 Northwood 533 МГц с частотой 2,8 ГГц; P4 Northwood 400 МГц с частотой 2,6 ГГц; P4 Willamette Celeron 400 МГц с частотой 2,0 ГГц.

Ну и, наконец, в первом квартале будущего года выйдут: P4 Northwood 533 МГц с частотами 3,06 ГГц и выше; P4 Northwood 400 МГц с частотой 2,6 ГГц; P4 Willamette Celeron 400 МГц с частотами выше 2,0 ГГц.

Сколько будут стоить новые процессоры, зависит от хода "ценовой" войны с AMD.

Компьютер грязнее туалета

Казалось бы, что может быть общего между настольным компьютером и бактериологией? Оказывается, много, и даже слишком. Американские исследователи задались целью выяснить, насколько опасен персональный компьютер с точки зрения наличия на нем вредоносных бактерий. Результаты ошеломили самих ученых, так как оказалось, что некоторые клавиатуры примерно в 400 раз грязнее среднего туалета.

Относится это, в основном, к офисным ПК, так как именно там к одному компьютеру имеют доступ многие люди. Каждый из них, печатая что-то на клавиатуре, или просто держа в руке мышь, оставляет после себя миллионы бактерий, которые прекрасно чувствуют себя в питательной среде из кожных выделений, покрывающих любую клавиатуру.

А чисткой клавиатур озабочена лишь незначительная часть офисных работников, чаще всего клавиатура вообще ни разу не чистится за все время ее эксплуатации. Вот и получается, что компьютер в сотни раз грязнее туалета, так как всякий туалет регулярно моют дезинфицирующими средствами.

Старые документы MS Office опасны и без вирусов

Дыра в системе безопасности, присущая документам некоторых старых офисных приложений Microsoft, все еще может хранить государственные тайны. Поисковая система Google сообщает, что около полумиллиона документов с расширением ".doc" доступны для скачивания с различных Web-сайтов. Из них небольшой, но значимый процент документов создан старыми версиями "офисных" приложений Microsoft.

Ошибка в MS Office впервые

была обнаружена в 1998 г. Тогда выяснилось, что внутри DOC-файлов могут сохраняться случайные части данных из ранее удаленных файлов. Эти самые случайные данные и могут нести практически любую информацию, которая ранее хранилась на ПК создателя doc-файла, а затем была удалена.

Таким образом, кто угодно может скачать доступные файлы и просмотреть их в HEX-коде, тем самым обнаружив, что именно лишнее хранится внутри файла. Список приложений, которые могли создавать неправильные файлы, не столь велик, но они используются в работе намного чаще остальных. К ним относится Microsoft Word 6.0 и 7.0, PowerPoint 7.0 и Excel 7.0.

Компанией Microsoft был выпущен patch, который исправлял ошибки в программе. Документы, созданные до применения этой заплатки, все еще остаются "дырявыми" и могут содержать любую информацию. Стоит добавить, что на сайтах американского правительства находится около 240 тысяч документов Word и около 32 тысяч PowerPoint, и примерно 5% из них были созданы именно старыми версиями офисных программ.

IBM создала высокопроизводительный транзистор на основе углеродных нанотрубок

Компания IBM объявила о создании транзистора, превосходящего по производительности транзисторы на основе кремния. Новая разработка может стать ключом к созданию еще более миниатюрных и быстрых компьютеров.

Как сообщили представители IBM агентству Reuters, для изготовления новых транзисторов использовались углеродные нанотрубки — тонкие цилиндрические структуры, которые примерно в 100000 раз тоньше человеческого волоса, состоящие из атомов углерода. По утверждению разработчика, транзистор на основе углеродных нанотрубок превзошел по производительности самый быстрый транзистор на основе кремния.

Компания IBM объявила о создании первого миниатюрного транзистора из нанотрубок еще в прошлом году, однако опытный экземпляр не мог работать с силой тока, более чем в два раза превышающей силу тока, подводимого к обычным транзисторам на основе нанотрубок. По словам Фэйдона Эйвориса, руководителя подразделения IBM, занимающегося исследованиями в области нанотехнологий, миниатюрность имеет важное значение, однако цель ученых — добиться еще более высокой производительности.

Ученые ищут замену кремнию, поскольку в течение ближайших 10...15 лет потенциальные возможности совершенствования микросхем на основе кремния будут исчерпаны, что не позволит и дальше повышать производительность микросхем и уменьшать их размеры. Помимо работы с углеродными нанотрубками, которые являются самыми прочными волокнами в природе (их прочность в 10 раз выше прочности стали), ученые изучают возможность создания квантовых вычислительных машин.

По материалам ZDNet, CNet, Cnews, 4User, Компьюлента.

Н.МАРТЫНЮК
E-mail:n_mar@tut.by

Программы для GSM-телефонов

Обрушившаяся волна сотовых телефонов застала врасплох многих людей, желающих приобрести это чудо техники. Потенциальные покупатели не знают, какую модель лучше выбрать, а пользователи — как расширить функции. Современные "трубочки" стандарта GSM, помимо самых необходимых функций, наделяются большим количеством дополнительных. Это и "продвинутые" телефонные книги с сохранением E-mail и голосовым управлением, и программируемые мелодии, и логотипы, и диктофоны, и календари. Довершают картину сменные цветные панельки. Но возникает вопрос, как со всем этим разобраться? Оказывается, все не так уж и сложно. Единственное, что требуется — компьютер и необходимое программное обеспечение. Рассмотрим некоторые программы для подключения телефона к персональному компьютеру и передачи данных через GSM.

Все огромное количество программ, которые запускаются на ПК и работают с GSM-телефоном, делятся на две большие группы:

- программы для сервис-центров, для ремонта и программирования;
- программы для пользователей, то есть для нас с вами.

Программы для сервис-центров позволяют "настраивать" телефон, изменять параметры его работы и производить смену программного обеспечения. Да-да, представьте себе, телефон — это своего рода компьютер, и "живет" он в соответствии с определенной программой, которая со временем устаревает, как, например, Windows 3.11. Появление новых версий софта для телефонов Siemens, Nokia или Sony, конечно, не сопровождается рекламными кампаниями, как обновление Microsoft Office. Тем не менее, новая версия способна избавить наш аппаратик от небольших "глюков" или добавить пару-тройку часов работы на одной зарядке аккумулятора. Однако самостоятельно использовать подобные программы не стоит, поскольку, кроме самих программ, требуется специальное оборудование для их установки, а главное — знание того, как они работают. Все это представлено в сервис-центрах соответствующих компаний-производителей мобильных телефонов, куда можно обращаться для ремонта и замены программы.

Другая группа программ предназначена как раз для владельцев мобильных телефонов и позволяет редактировать телефонную книгу, синхронизировать данные органайзера с настольным компьютером, устанавливать в телефон новую мелодию звонка или логотип на экране, осуществлять удобное управление режимами переадресации в сложных моделях и многое другое.

Возможность передачи данных зависит от способностей конкретной модели телефона. Например, если ваш аппарат оснащен встроенным модемом, все, что необходимо — это специальный драйвер, благодаря которому компьютер поймет, что вы приобрели новый модем. После установки драйвера, в списке модемов и сетевых карт вы получите соответствующее коммуникационное устройство, работа с которым будет мало отличаться от работы через обычный модем и телефонную линию. Драйверы под Windows для большинства телефонов с аппаратным модемом можно получить непосредственно с сайта производителя оборудования. Кто является производителем кабеля для соединения трубки с компьютером, уже не столь важно, хотя производители телефонов рекомендуют использовать исключительно оригинальное оборудование. Их можно понять — в этом случае за $40...60 вы покупаете вещь, себестоимость которой составляет $3...4. Кабели третьих фирм обойдутся вам в $15...20 (при покупке через Интернет-магазины).

Возможность работать на компьютере с телефонной книжкой, SMS, логотипами, мелодиями и другими сервисными функциями вашей "трубочки" фиксируется в соответствующей документации. Вам останется лишь обзавестись кабелем и специальной программой, чаще всего бесплатной, которую можно скачать через Интернет. Подобные программы позволяют с большим удобством редактировать и классифицировать записи в телефонной книге и календаре, синхронизируя их с привычным Outlook'ом. С их помощью можно менять мелодии и логотипы экрана, устанавливать различные режимы переадресации и др., если это, конечно, поддерживается аппаратом. Возможности и особенности работы программ зависят от конкретной модели телефона, но общая картина соответствует нарисованной выше. Доступность таких программ и простота в использовании позволяют без особого труда и затрат существенно расширить функции телефона и облегчить работу с ним.

Одна из самых популярных программ — это Logo Manager. Это отличная программа для различных моделей телефонов Nokia, предназначенная для работы с графикой и настройки аппарата (**рис.1**). Она позволяет редактировать, сохранять и загружать в трубку: приветствие; картинку, которая будет появляться при загрузке аппарата; логотип оператора — картинку, которую вы будете наблюдать на экране телефона после регистрации в сети; групповые логотипы — картинки, которые будут появляться во время звонков тех людей, чьи номера есть в вашей записной книжке.

Рис. 1

Кроме того, Logo Manager позволяет работать с записной книжкой, отсылать SMS, а также различные картинки и мелодии звонка в виде SMS, редактировать мелодии звонка и активизировать Netmonitor — дополнительный пункт меню телефона, содержащий массу технической информации о модели телефона.

К сожалению, эта программа, как и большинство аналогичных, относится к числу коммерческих. Иными словами, вам придется купить ее в магазине или получить ключ для регистрации, заплатив автору, например, по кредитной карте. Однако если воспользоваться услугами основных поисковых систем типа www.altavista.com, озадачив их словосочетанием "gsm software", можно найти то же самое бесплатно. Естественно, придется смириться с тем, что, установив скаченную "на халяву" программу, вы лишаетесь технической поддержки и бесплатного обновления.

Еще одной интересной программой является SMS-pager (**рис.2**). Зачем нужен SMS-пейджер, если с мобильного телефона можно просто позвонить? Во-первых, так дешевле. Во-вторых, так можно передать информацию абоненту, если дозвониться к нему в настоящее время не удается. В-третьих, справочные сведения, которые могут понадобиться (место и время встречи, номер телефона и т.д.), удобнее послать по SMS, а не запоминать или записывать. В-четвертых, с помощью SMS легко наладить общение между владельцем "мобильника" и пользователями Internet (через E-mail, www).

Рис. 2

В заключение, для всех пользователей GSM, желающих самостоятельно научиться пользоваться другими программами, могу предложить компакт-диски "Секреты GSM" и "Самоучитель по GSM". Тел. для справок в Кобрине, Брестской области: (01642) 22-4-88, E-mail: n_mar@tut.by.

ЛИСТАЯ СТРАНИЦЫ

О новом пакете, предназначенном для разработки программ, функционирующих в среде Windows, рассказывает *Ю.Анищенко* в статье "Visual Studio .NET — новая парадигма программирования" **(ПЛ: компьютеры, №5/2002, С.44...45)**.

Выпуская новую версию пакета Visual Studio, компания Microsoft открывает для программистов новые горизонты творчества, а для простых пользователей — небывало высокую степень использования информационных технологий в повседневной жизни, считает автор. Ничуть не преувеличивая, Visual Studio .NET можно охарактеризовать как новаторский прорыв не только в области средств программирования, но и на всем рынке прикладного прогрвммного обеспечения. Ведь на этот раз Microsoft в корне меняет свой подход к архитектуре ПО.

Речь идет о платформе .NET, которую компания ускоренными темпами внедряет в жизнь. Вкратце суть ее заключается в следующем. До сих пор программное обеспечение создавалось на базе обособленных приложений, предназначенных для выполнения жестко очерченного круга задач. После появления технологий компонентного моделирования (COM, CORBA и т. п.) жизнь программистов несколько облегчилась — в работе стало возможным использовать готовые компоненты со своей собственной функциональностью, не вдаваясь в подробности их реализации. Теперь же в качестве основы построения будущих информационных систем выдвигаются так называемые сервисы, умеющие не только общаться с пользователем, но и успешно взаимодействовать друг с другом, соединяясь в структуры различного масштаба — от простенькой телефонной книжки в карманном компьютере до корпоративного механизма обмена и хранения данных.

Платформа .NET активным образом использует Интернет как основную информационную магистраль. Экспоненциальный рост числа пользователей Всемирной паутины, многократное увеличение скорости доступа и возможность подключения к ней различных устройств — от микроволновой печи до наручных часов — превратили Сеть в оптимальный канал управления и контроля на транспорте, в бизнесе, образовании и многих других отраслях экономики. Поэтому Web-сервисы — одни из "столпов" .NET — подразумевают Интернет как "среду обитания" программного обеспечения нового уровня. Для разработчиков и администраторов компьютерных систем внедрение этой платформы означает ускорение процесса компоновки сколь угодно сложных решений, а для простых пользователей — окончательный переход множества проблем, мешающих в повседневной жизни, в виртуальную область.

В состав Microsoft .NET входит несколько базовых компонентов. В первую очередь это .NET Framework, включающий набор инструментов для разработки приложений и сервисов, компиляторы языков программирования, а также новая среда времени выполнения (Common Language Runtime, CLR). CLR — настоящее достижение компьютерной отрасли. Эта среда призвана обеспечивать полную независимость результатов труда программиста от выбора языка, на котором он работает. Те, кто привык писать на Visual Basic, C++, Java, Delphi или даже Perl, могут и дальше продолжать использовать свой любимый инструмент. Незаметно для них .NET Framework преобразует высокоуровневый код в низкоуровневый, устраняя тем самым традиционные преграды на пути интеграции модулей, созданных в различных системах.

Следующий компонент — это целый набор .NET-серверов корпоративного уровня, в т.ч. Windows 2000 Advanced Server, SQL Server 2000. Commerce Server 2000, Exchange Server 2000, ISA Server 2000, BizTalk 2000 и др.

Третий компонент — набор стандартных служб, которые можно использовать при разработке приложений с той же легкостью, с какой программисты сегодня прибегают к помощи COM-объектов. Приобретая коробку с Visual Studio .NET, пользователь получает в свое распоряжение все упомянутые средства.

Работа в Visual Studio .NET проходит в визуальном режиме, который претерпел по сравнению с предыдущей версией значительные изменения, еще более упростив процесс создания приложений. От пользователя требуется, конечно же, написание определенного программного кода вручную, однако в пакет заложено большое количество стандартных шаблонов на все случаи жизни. При его запуске достаточно выбрать, какого рода приложение вы собираетесь создать, и компьютер сделает за вас всю черновую работу по подготовке каркаса будущей программы. Нечто подобное уже встречалось в прошлых версиях Visual Studio, однако на этот раз набор шаблонов значительно увеличился.

Visual Studio .NET поставляется в трех базовых комплектациях, ориентированных на различный масштаб приложений, которые можно создавать с ее помощью. Минимальный набор — Professional — включает в себя собственно графическую оболочку, CLR, средства создания Web-сервисов и приложений для стандартной среды Windows, а также мобильных устройств, и генератор Web-форм. Сюда же включены компиляторы Visual Basic .NET, C++ .NET, C# .NET, J# .NET, а также поддержка других языков программирования для .NET-платформы, появление которых ожидается в ближайшее время. Из вспомогательных средств стоит отметить "Мастера публикаций", оригинальный отладчик .NET, мощную справочную систему и графический конструктор для HTML- и XML-страниц. Более продвинутая версия — Enterprise Developer — содержит SQL Server 2000 и ядро MSDE, сервер контроля версий Source Safe, Windows 2000 Advanced Server и ряд других серверных продуктов бизнес-класса. Наконец, самый дорогой пакет, Enterprise Architect, добавляет к этому набору средства визуального моделирования баз данных и сервер BizTalk, используемый в электронной коммерции.

Оценку Visual Studio .NET дать трудно, отмечает автор. По большому счету, оценивать следует те продукты, которые будут созданы на ее основе. Потенциал самого пакета поражает и, в то же время, настораживает — велика вероятность, что даже самый серьезный программист сумеет использовать его только процентов на 20...30, т.е. настолько же, насколько обычная секретарша задействует Word. Кроме того, при сравнительно высокой стоимости Visual Studio .NET (от 550 до 2500 USD), не каждая компания в состоянии приобрести по экземпляру пакета для всех своих разработчиков.

Системные требования:

- процессор Pentium II 450 МГц и выше;
- оперативная память от 64 Мб и выше (в зависимости от операционной системы);
- жесткий диск 2,5...3,5 Гб, в т.ч. не менее 500 Мб на системном диске;
- операционная система Windows NT 4.0 Workstation, NT 4.0 Server, 2000 Professional, 2000 Server или XP Professional.

О перспективах использования порта USB 2.0 рассказывает *Том Мейнелли* в статье "USB 2.0: какова реальная скорость" **(Мир ПК, №5/2002, С.28...32)**

Пользователи ПК оценили по достоинству действительно универсальные USB-порты своих машин, отмечает автор. Однако высокоскоростные периферийные устройства, типа внешних жестких дисков, вполне могут превратить обеспечиваемую USB скорость передачи данных, равную 12 Мбит/с, в узкое место, снижающее общую производительность системы. В этом случае можно использовать порт USB 2.0, называемый также Hi-Speed USB (высокоскоростной USB-порт). По заявлению разработчиков, в нем сочетаются универсальность предшественника, совместимость с нынешними USB-устройствами и скорость обмена данными, достигающая 480 Мбит/с. Это в 40 раз быстрее, чем позволяли компьютеры с USB 1.1.

Для проверки этого утверждения были проведены несколько тестов. Отмечается хорошая совместимости между различными устройствами с портами Hi-Speed USB. Быстродействие, однако, не соответствует широко разрекламированным обещаниям. Максимальное ускорение переноса данных, достигнутое при обмене информацией между ПК и внешним жестким диском, увеличилось всего в 12,6 раза. При тестировании других устройств с USB 2.0, дисковода CD-RW и сканера, прирост производительности оказался существенно меньшим по причине более низкой пиковой производительности самих этих аппаратов.

И все же Hi-Speed USB — это отнюдь не пустышка. По сравнению с любыми другими видами модернизации, кроме, разве что, установки интерфейсв IEEE 1394 — это самый эффективный способ вложения денег. Пользователь получает реальный шанс реализовать дополнительные преимущества современных внешних устройств.

Анализировалась работа пяти плат PCI с портами Hi-Speed USB: USB2connect 3100LP корпорации Adaptec, USB 2.0 F5U220 фир-

мы Belkin, USB 2.0 U2PCI-5 компании Keyspan, OrangeUSB 2/0 Hi-Speed PCI производства Orange Micro и USB 2.0 5-Port PCI компании SIIG. Для каждой из них использовался драйвер, предоставляемый производителем.

Все пять перечисленных USB 2.0 PCI-плат тестироаались со следующими устройствами: внешним жестким диском Personal Storage 3000LE фирмы Maxtor емкостью 40 Гб и скоростью вращения 5400 об/мин, планшетным фотосканером Perfection 2450 компании Epson и внешним дисководом CD-RW 241040UE VeloCD производства TDK.

Тест проводился на ПК IBM NetVista в следующей конфигурации: процессор Intel Pentium 4 с тактовой частотой 1,4 ГГц, оперативная память объемом 256 Мб, внутренний жесткий диск на 60 Гб и операционная система Windows XP Professional. Чтобы результаты были сопоставимы, тесты для USB 1.1 проводились с использованием интегрированных портов USB 1.1 того же самого ПК **(см. табл.)**

	Тесты для жесткого диска		Тесты для дисковода CD-RW		Тесты для сканера	
	Копирование файлов и папок	Photoshop 6.0.1	Цифровое аудио	Запись на CD-RW	Разрешение 1600 DPI	Разрешение 400 DPI
Средний результат для пяти плат USB 2.0, мин:сек	0:58	4:24	1:38	4:03	6:44	0:15
Показатель встроенных портов USB 1.1, мин:сек	12:13	37:19	6:22	20:10	13:42	0:26
Увеличение быстродействия	12,6	8,5	4	5	2	1,7

Из трех периферийных устройств, которые тестировались, наибольший прирост скорости передачи данных показал внешний жесткий диск компании Maxtor. Среднее время, потребовавшееся пяти платам на завершение копирования файлов на этот диск, составило 58 с — в 12,6 раза быстрее, чем 12 мин 13 с, затраченные при использовании порта USB 1.1. Средний показатель для пяти адаптеров в тесте на работу с Photoshop составил 4 мин 24 с — в 8,5 раза быстрее, чем 37 мин 19 с для порта USB 1.1. Но при всем этом простенький анализ показал, что внутренний жесткий диск со скоростью вращения 5400 об/мин и стандартной шиной UDMA/100 работает значительно быстрее протестированного внешнего с портом USB 2.0.

Однако обвинять диск компании Maxtor не стоит — он-то как раз способен на устойчивую передачу данных со скоростью до 374 Мбит/с. Это медленнее, чем теоретический максимум, достижимый Hi-Speed USB, но все же много быстрее, чем фактически показанная им скорость в 90 Мбит/с. Причина состоит в том, что по меньшей мере от 10 до 15% всей заявленной скорости в 480 Мбит/с уходит на протокольные (служебные) данные, т.е. на поддержание коммуникационного протокола между платой и периферийным устройством. Кроме того, за более низкие, чем ожидалось, показатели несут ответственность ОС и контроллер, пока еще не вполне оптимизированные для обеспечения максимального быстродействия.

Тесты с использованием внешнего дисковода CD-RW компании TDK также выявили значительное приращение быстродействия, обеспечиваемое Hi-Speed USB, хотя ограничения, присущие даже очень быстрому накопителю CD-RW, не позволяют получить прирост производительности, подобный тому, который показал жесткий диск. В тесте на получение цифровых аудиоданных пять плат Hi-Speed USB завершили задачу со средним результатом в 98 с, что означало четырехкратное ускорение по сравнению с 6 мин 32 с для порта USB 1.1. В тесте с записью на CD-RW интерфейс Hi-Speed USB дал пятикратное приращение быстродействия — скорость передачи составила около 21 Мбит/с. Заявленная для дисковода производительность 24X теоретически должна быть в состоянии обеспечить максимальную скорость передачи данных, равную 28,8 Мбит/с, а для чтения — 40X, т. е. 48 Мбит/с.

В случае сканера ограничения на скорость передачи данных еще более заметны. Сканер ф.Epson в тесте на передачу изображения с разрешением 300 DPI работал с портом USB 2.0 в 1,7 раза быстрее, чем с портом USB 1.1, а при разрешении 1600 DPI приращение скорости было двукратным. По словам инженеров ф.Epson, это соответствует их ожиданиям, поскольку буферы памяти большинства сканеров попросту слишком малы, для того чтобы в полной мере реализовать преимущества Hi-Speed USB. Но даже с учетом этого, сканирование изображения высокого разрешения было завершено через Hi-Speed USB всего лишь за 6 мин 44 с — почти на 7 мин быстрее, чем при использовании USB 1.1.

Велика вероятность того, что скоро каждый ПК будет укомплектован Hi-Speed USB. В январе 2002 г. фирма Gateway стала первым крупным поставщиком ПК, предлагающим системы с Hi-Speed USB на системной плате. Можно ожидать, что и другие производители компьютерных систем последуют этому примеру уже в ближайшие месяцы.

А пока что, если вы намереваетесь приобретать новый сканер, внешний дисковод CD-RW, внешний жесткий диск или любое другое периферийное устройство, которое способно хотя бы частично использовать более высокую скорость передачи данных, обеспечиваемую USB 2.0, автор рекомендует оснастить компьютер платой расширения. В конце концов, любая модернизация, которая позволяет безболезненно увеличить быстродействие ПК в два...пять раз (или даже больше), заслуживает внимания.

Кулер (охладитель) — недорогое устройство, основное назначение которого поддерживать нормальный температурный режим процессора ПК. Насколько хорошо разные кулеры справляются с этой задачей, проверено в редакции журнала **"ПЛ: компьютеры"**, и результаты этой проверки привел А.Кузнецов в статье "Ураган над кристаллом" **(№5/2002, С.64...69).**

Стабильная работа процессора зависит от эффективности системы охлаждения, основное назначение которой — поддерживать рабочую температуру процессора (40...50°C). В противном случае компьютер будет периодически "виснуть", или вовсе откажется загружаться. Проблема охлаждения современных процессоров решается за счет применения кулеров. Особенно актуально использование хороших кулеров для процессоров AMD. По сравнению с процессорами Intel, они выделяют больше тепла. Кроме того, у них отсутствует встроенный датчик термозащиты. Поэтому в случае выхода вентилятора из строя процессор AMD может попросту сгореть.

В испытаниях участвовали кулеры, предназначенные для использования в самых актуальных на сегодняшний день процессорных разъемах SocketA/370/478. Тестировались Buran, Storm 1 ST-1B и Storm 2 ST-2B ф.Arctic; DI4-7H53D и DP5-6131C от Cooler Master; GlacialTech Igloo 4300; Intel Box Cooler A80856-001; Chrom Orb, Dragon Orb 1 A1137, Dragon Orb 3 A1135, Mini Copper Orb DU0462-9, Volcano 6Cu A1139, Volcano 6Cu+ A1138 и Volcano P4 A1119 от Thermaltake; Majesty Superstar Cooler TTC-MS2AB и Majesty Twins Cooler TTC-MT1AB ф.Titan; Zalman CNPS3100plus. Их параметры, заявленные производителями, приведены в **табл.1**, результаты тестирования — в **табл.2**.

При тестировании сначала с помощью программ аппаратного мониторинга определялись температура нагрева процессора и скорость вращения вентилятора. Затем аудиоанализатором Euroaudio PRO 600 измерялся создаваемый вентилятором кулера уровень шума в диапазоне от 20 Гц до 20 кГц.

Все кулеры были условно разбиты на две группы. В одну из них вошли модели, используемые в системных платах с разъемами Socket370 и SocketA (испытания проводились на системной плате Abit KT7A-RAID с установленным процессором AMD Athlon 1,2 ГГц), а в другую — кулеры, работающие с процессорами Intel Pentium 4 в исполнении под Socket478 (для испытаний применялась материнская плата Gigabyte GA-8IRM с процессором Pentium 4 1,6 ГГц).

Кулеры тестировались с использованием штатного термоматериала, поставляемого в комплекте. Для более быстрого разогрева процессоров AMD 1,2 ГГц и Intel Pentium 4 1,6 ГГц применялась программа CPUBurn 4.0. Температура процессоров фиксировалась после 10 мин работы программы. Степень нагрева процессора AMD и скорость вращения вентилятора определялись с помощью программы MotherBoard Monitor v.5.1.0.7, а в случае процессора Intel Pentium 4 — с помощью фирменной утилиты аппаратного мониторинга SIV от компании Gigabyte.

Табл. 1

Модель	Размеры вентилятора, мм	Процессорный разъем	Размеры кулера, мм	Скорость вращения вентилятора, об/мин	Уровень шума, дБ	Воздушный поток, куб. футов/мин	Тип подшипника	Цена, USD
Buran	60x60x15	Socket478	83x62,5x70	5000	30	20,83	Качения	17
Storm 1 ST-1B	70x70x15	Socket478	83x69x55	4500	32	25,73	Качения	11
Storm 2 ST-2B	70x70x15	Socket478	83x75x60	4500	32	25,73	Качения	12
DI4-7H53D	70x70x15	Socket478	83x69x37	3800	34,5	33,2	Качения	11
DP5-6131C	60x60x13	SocketA/370	80x60x41	5400	38	27,72	Качения	8,4
Igloo 4300	70x70x15	Socket478	83x69x53	5000	38	42	Качения	10
Box Cooler A80856-001	-	Socket478	-	-	-	-	Качения	10
Chrom Orb	50x50x25	SocketA	69x45	5500	29	23,1	Качения	6
Dragon Orb 1 A1137	60x60x25	SocketA/370	69x79	7000	37	38	Качения	12
Dragon Orb 3 A1135	60x60x25	SocketA/370	69x79	7000	37	38	Качения	14
Mini Copper Orb DU0462-9	50x50x25	SocketA/370	65x48	5500	29	23,1	Качения	10
Volcano 6Cu A1139	60x60x25	SocketA/370	80x60x65	4550	31	32	Качения	8,8
Volcano 6Cu+ A1138	60x60x25	SocketA/370	80x60x65	7000	39	38	Качения	9
Volcano P4 A1119	70x70x15	Socket478	88x68x61	4800	37	30	Качения	8,2
Majesty Superstar Cooler TTC-MS2AB	60x60x25	SocketA/370	69,5x54,5x71	4000	30	25,15	Качения	11,6
Majesty Twins Cooler TTC-MT1AB	50x50x15	SocketA/370	60x60x60,7	5500	29	21,4	Качения	8,8
CNPS3100plus	92x92x25	SocketA/370	110x52x65	1600/2800	20/31,1	-	Скольжения	33

Табл. 2

Модель	Уровень шума, дБ	Температура, °C	Скорость вращения вентилятора, об/мин
Buran	90	45	5113
Storm 1 ST-1B	91	47	4115
Storm 2 ST-2B	91	44	4017
DI4-7H53D	92	44	4115
DP5-6131C	87	51	5720
Igloo 4300	91	46	4560
Box Cooler A80856-001	86	49	2343
Chrom Orb	88	56	5921
Dragon Orb 1 A1137	96	46	7031
Dragon Orb 3 A1135	97	45	7031
Mini Copper Orb DU0462-9	89	49	6136
Volcano 6Cu A1139	89	52	5357
Volcano 6Cu+ A1138	94	45	7180
Volcano P4 A1119	93	53	4560
Majesty Superstar Cooler TTC-MS2AB	89	50	4440
Majesty Twins Cooler TTC-MT1AB	88	51	5273
CNPS3100plus	91	47	2657

По результатам тестирования среди кулеров, устанавливаемых в разъем SocketA/370, лучше всего охлаждают процессоры Thermaltake Dragon Orb 3 A1135 и Volcano 6Cu+ A1138, в основном, за счет использования высокоскоростного вентилятора (7 тыс. об/мин) и медной вставки в радиаторе. При этом Volcano 6Cu+ A1138 отличается более простым дизайном и меньшим уровнем шума по сравнению с Thermaltake Dragon Orb 3 A1135. К самым тихим из кулеров этой группы можно отнести Thermaltake Chrom Orb и Titan Majesty Twins Cooler TTC-MT1AB. Правда, охлаждающие способности модели Chrom Orb — наиболее слабые.

Из кулеров, предназначенных для использования с процессорами Intel Pentium 4 в исполнении под Socket478, самый эффективный отвод тепла обеспечивают модели Arctic Storm 2 ST-2B и Cooler Master DI4-7H53D. Больше всего процессор нагревается при работе с кулером Thermaltake Volcano P4 All9, в то время как тише всех функционирует фирменный кулер Intel — очевидно, сказывается сравнительно низкая скорость вращения вентилятора.

Подводя итоги, автор отмечает, что среди моделей, предназначенных для использования в процессорных разъемах SocketA/370, "Выбором редакции" становится Thermaltake Volcano 6Cu+ A1138 благодаря применению медной вставки и вентилятору со скоростью вращения 7 тыс. об/мин. Кулер обладает сравнительно низкой ценой, не сильно шумит, к тому же, он компактен и легко устанавливается. "Лучшей покупкой" в этой группе может стать Thermaltake Chrom Orb. Он не отличается сверхъестественными теплорассеивающими свойствами, но имеет самую низкую цену, а главное — создает минимальный уровень шума во время работы.

Среди кулеров, предназначенных для работы с процессорами Pentium 4, "Выбором редакции" стала модель Arctic Storm 2 ST-2B. Кулер обладает хорошими охлаждающими способностями и несколько лучшими акустическими характеристиками по сравнению с ближайшим конкурентом — моделью Cooler Master DI4-7H53D. "Лучшей покупкой" может стать фирменный кулер Intel A80856-001, входящий в коробочную версию процессора Pentium 4, обеспечивающий надежное охлаждение и имеющий очень низкий уровень шума. Его качество гарантировано репутацией этого известного производителя.

Сбой компьютера — крайне неприятная вещь. Он чреват потерей большого объема ценной информации. Если же это происходит из-за пропадания или значительного снижения питающего напряжения, возможны и более серьезные последствия, например, повреждение винчестера. Поэтому во многих случаях источник бесперебойного питания (ИБП) совершенно необходим. Но какой ИБП выбрать? Определенную помощь в решении этого вопроса может оказать статья *С.Пахомова и С.Самохина* **"Тестирование источников бесперебойного питания" (КомпьютерПресс, №5/2002, С.95...109)**.

Совместно с компьютерами используются несколько типов ИБП. Off-line ИБП состоят из фильтра, инвертора, устройства контроля и аккумулятора. Фильтр служит для ослабления высокочастотных помех в сетевом напряжении. В недорогих ИБП он состоит из одного варистора, защищающего от выбросов напряжения выше 400 В, иногда дополненного LC-фильтром. Устройство контроля следит за степенью заряда аккумулятора и величиной сетевого напряжения, включая при его исчезновении инвертор. Время перехода на работу от батарей является важной характеристикой ИБП и, как правило, не превышает 20 мс. Инвертор преобразует низкое постоянное напряжение аккумулятора в переменное напряжение с частотой около 50 Гц и формой, как правило, близкой к прямоугольной. Во время работы от аккумулятора устройство контроля следит за степенью заряда аккумулятора и, при ее снижении ниже определенной величины, отключает инвертор. Устройство контроля управляет и работой динамика, который звуковым сигналом (попискиванием) оповещает пользователя о том, что сетевое напряжение отключено. У большинства ИБП характер попискивания меняется в процессе разряда аккумулятора — ста-

новится более частым по мере приближения к моменту отключения инвертора.

ИБП Line-Interactive (взаимодействующие с линией) включают те же узлы, что и предыдущие, но к ним добавляется трансформатор. Функции устройства контроля дополняются отслеживанием сетевого напряжения и переключения при помощи реле отводов обмотки трансформатора для поддержания выходного напряжения в разумных пределах. При снижении сетевого напряжения ниже критического, ИБП этого класса также переходят в режим преобразования энергии, запасенной в аккумуляторе. Некоторые ИБП отключаются от сети и переходят на работу от аккумулятора при превышении сетевым напряжением заданного предела — обычно 260 В и более.

Близко к Line-Interactive находятся феррорезонансные ИБП, которые стабилизируют выходное напряжение за счет встроенного феррорезонансного стабилизатора. При выходе сетевого напряжения за рабочие границы они также переключаются на работу от аккумулятора.

ИБП On-Line всегда работают от аккумулятора, который, при наличии сетевого напряжения, постоянно подзаряжается. Благодаря этому отсутствует провал выходного напряжения, сопровождающий переход на работу от аккумулятора остальных классов. Кроме того, для поддержания имиджа ИБП высокого класса, они, как правило, имеют выходное напряжение, близкое по форме к синусоидальному.

Гибрид On-Line и Line-Interactive называется ИБП с дельта-преобразованием. Эти ИБП производят стабилизацию выходного напряжения посредством преобразования небольшой части входного напряжения. В результате алгебраического суммирования этой части с нестабильным входным напряжением получается стабилизированное выходное напряжение.

Как и любые другие аккумуляторы, те, что применяются в ИБП, имеют две основные характеристики — номинальное напряжение, которое зависит от типа аккумулятора и количества банок в батарее, и емкость, выражаемую в ампер-часах. Следует иметь в виду, предупреждают авторы, что емкость зависит от режима разряда. Например, аккумулятор HR 1224WF2F1 производства CSB Battery при разряде до напряжения 10,8 В током 16,1 А имеет емкость 2,7 А·ч, а при разряде током 4,1 А — 4,1 А·ч. На большинстве же аккумуляторов указывается емкость, получаемая при 20-часовом разряде, что совершенно не характерно для их применения в ИБП, но позволяет изготовителям гордо заявлять о семи и даже девяти ампер-часах.

Еще один нюанс — это напряжение, до которого осуществляется разряд. Чем глубже разряд, тем быстрее происходит деградация аккумуляторов и тем меньше циклов "заряд-разряд" он выдержит. Например, если вышеупомянутый аккумулятор разряжать не до 10,8 В, а до 9,6 В, то емкость повышается до 3,2 А·ч, то есть почти на 20%.

И, конечно, значительное влияние на длительность работы ИБП от аккумулятора оказывает КПД инвертора: чем он выше, тем дольше период автономной работы ИБП при прочих равных условиях.

Для испытаний были выбраны ИБП диапазона мощностей от 300 до 600 В·А стоимостью до 150 USD, как наиболее массовые. Тестировались 17 источников бесперебойного питания: APC Back-UPS CS 350, ARC Back-UPS Pro 420, APC Back-UPS CS 500, Best Power Patriot Pro II 400 VA, Eline Power ELP-350, Eline Power ELP-500, Liebert PowerSure ProActive PSA 350-230, MGE UPS Pulsar Ellipse 300, MGE UPS Pulsar Ellipse Premium 500, NeuHaus SmartLine 450, Powercom BNT 400 VA, Powerman BackPro 500, Powerman BackPro 500 Plus, Powerware 3110 425i, Powerware 3110 600 VA, Powerware 5115 VA, Tripp-Lite OmniSmart 500. Их основные параметры и результаты тестирования приведены в **таблицах**.

В качестве основного критерия оценки ИБП было выбрано максимальное время работы от аккумуляторной батареи. Для этого использовался компьютер с процессором Pentium III 800 МГц и диском емкостью 40 Гб. Все возможности по управлению потребляемой мощностью были отключены. От того же ИБП был запитан монитор LG Flatron 795FT Plus (диагональ — 17"). Общее энергопотребление составляло около 200 Вт — оно является типичным и лежит в верхней части диапазона, обычного для рабочих станций среднего класса. Перед началом измерения аккумулятор заряжался от сети в течение не менее 12 ч. Отсчет времени проводился при помощи секундомера, работающего на другом компьютере.

Подводя итоги, авторы отмечают, что наличие функции управления повышает как цену, так и ценность ИБП. Поэтому определение выбора редакции журнала "Компьютер-Пресс" производилось в двух категориях ИБП — управляемые и неуправляемые. Нвличие технологии Line-Interactive не учитывалось, так как современные компьютеры обеспечивают работу в весьма широком диапазоне напряжений питания, а мониторы способны работать при напряжении питания до 90 В.

Среди управляемых ИБП лучшими в номинации "Самый качественный ИБП" были признаны Powerware 3110 600 VA, а среди неуправляемых — MGE UPS Pulsar Ellipse 300. В номинации "Оптимальная покупка" среди управляемых ИБП выбор редакции получил Powerman BackPro 500 Plus, а среди неуправляемых — Powerman BackPro 500.

Табл. 1

Название ИБП	APC Back-UPS CS 350	ARC Back-UPS Pro 420	APC Back-UPS CS 500	Best Power Patriot Pro II 400 VA	Eline Power ELP-350	Eline Power ELP-500	Liebert PowerSure ProActive PSA 350-230	MGE UPS Pulsar Ellipse 300	MGE UPS Pulsar Ellipse Premium 500
Тип ИБП	Off-Line	Line-Interactive	Off-Line	Line-Interactive	Line-Interactive	Line-Interactive	Line-Interactive	Off-Line	Line-Interactive
Мощность, В·А	350	420	500	400	350	500	230	300	480
Наличие функции управления	Есть	Есть	Есть	Есть	Нет	Нет	Есть	Нет	Есть
Время до срабатывания сигнала предупредительной тревоги, мин:сек	12:55	10:20	6:17	18:10	3:10	3:45	8:30	4:00	8:00
Время полной разрядки батареи, мин:сек	16:05	14:05	13:30	21:24	3:33	4:01	9:50	13:05	11:02
Напряжение переключения на батарею, В	181	195	165	190	163	160	167	185	167
Напряжение отключения от батареи, В	191	205	170	200	180	179	175	197	175
Цена, USD	64	92,5	154	116,5	47	58	135,5	79	165

Табл. 2

Название ИБП	NeuHaus SmartLine 450	Powercom BNT 400 VA	Powerman BackPro 500	Powerman BackPro 500 Plus	Powerware 3110 425I	Powerware 3110 600 VA	Powerware 5115 VA	Tripp-Lite OmniSmart 500
Тип ИБП	Line-Interactive	Line-Interactive	Line-Interactive	Line-Interactive	Off-Line	Off-Line	Line-Interactive	Line-Interactive
Мощность, В·А	450	400	500	500	425	600	500	500
Наличие функции управления	Есть	Нет	Нет	Есть	Есть	Есть	Есть	Есть
Время до срабатывания сигнала предупредительной тревоги, мин:сек	12:37	4:40	9:00	9:00	7:55	21:40	19:00	16:00
Врамя полной разрядки батареи, мин:сек	16:00	5:55	12:07	11:25	13:25	24:00	21:05	18:05
Напряжение переключения на батарею, В	178	175	165	165	180	180	185	149
Непряжение отключения от батареи, В	191	182	175	175	185	189	191	158
Цена, USD	105	43,5	45	49	89,5	139	182,5	152

М.ДОЛИНСКИЙ,
г.Гомель.

Решение задач на графах

(Продолжение. Начало в №№7-8/2002)

3.2. Задача "Перекрестки"

(Республиканская олимпиада по информатике 1998 г., автор — Котов В.М.)

Даны декартовы координаты N перекрестков города, которые пронумерованы от 1 до N. На каждом перекрестке имеется светофор. Некоторые из перекрестков соединены дорогами с двусторонним (правосторонним) движением, которые пересекаются только на перекрестках. Для каждой дороги известно время, которое требуется для проезда по ней от одного перекрестка до другого.

Необходимо проехать от перекрестка с номером A до перекрестка с номером B за минимальное время.

Время проезда зависит от набора проезжаемых дорог и от времени ожидания на перекрестках. Так, если вы подъехали от перекрестка X к перекрестку C по дороге X—C и хотите ехать дальше по дороге C—У, то время ожидания на перекрестке C зависит от того, поворачиваете ли вы налево, или нет. Если вы поворачиваете налево, то время ожидания равно D*K, где D равно количеству дорог, пересекающихся на перекрестке C, а K — некоторая константа. Если вы не поворачиваете налево, то время ожидания равно нулю.

Необходимо написать программу, которая определяет самый быстрый маршрут.

Спецификация входных данных

Входные данные находятся в текстовом файле с именем PER.IN и имеют следующую структуру:

- в первой строке находится число N (натуральное, ≤1000);
- во второй строке — количество дорог M (натуральное, ≤1000);
- в третьей строке — константа K (натуральное число, ≤1000);
- в каждой из N следующих строк находится пара чисел x и y, разделенных пробелом, где x, y — координаты перекрестка (целые числа, не превышающие по модулю 1000);
- в каждой из M следующих строк находится 3 числа p, r, t, разделенные пробелом, где p и r — номера перекрестков, которые соединяет дорога, а t (натуральное, ≤1000) — время проезда по ней;
- в последней строке находятся номера начального A и конечного B перекрестков.

Спецификация выходных данных

Выходные данные должны быть записаны в текстовый файл с именем PER.OUT и иметь следующий формат:

- в первой строке находится натуральное число T — время проезда по самому быстрому маршруту;
- в каждой из следующих строк находится одно число — номер очередного перекрестка в маршруте (начиная с перекрестка с номером A и кончая B).

Кратко идея решения может быть изложена следующим образом.

Алгоритм Дейкстра напрямую не применим, поскольку:

а) оптимальное решение в следующей вершине не является суммой оптимального решения для предыдущей вершины и веса текущего ребра;

б) вес текущего ребра — непостоянная величина, зависящая от того, каким поворотом мы вышли из вершины.

Поэтому предлагается решение поиском в глубину. При этом разрешается использовать каждую вершину столько раз, сколько у нее есть ребер. Отсекаются решения хуже текущего оптимального.

Можно еще ускорить решение, если обеспечивать перебор, начиная с правых поворотов (отсечения работали бы более эффективно).

Замечание: в тексте задачи явно не указано, но авторы оригинальных тестов к задаче полагали, что поворот на 180° — это левый поворот (как в армии: поворот на 180° — "через левое плечо").

Рассмотрим решение более подробно.

Ввод исходных данных осуществляется следующим образом:

```
read(N,M,K);
for i:=1 to N do readln(x[i],y[i]);
for i:=1 to N do begin kG[i]:=0; D[i]:=0; end;
for i:=1 to M do
   begin
      readln(p,r,t); inc(kG[p]); inc(kG[r]);
      G[p,kG[p],1]:=r; G[p,kG[p],2]:=t; inc(D[p]);
      G[r,kG[r],1]:=p; G[r,kG[r],2]:=t; Inc(D[r]);
   end;
readln(vA,vB);
```

Здесь:

- kG[i] — количество дуг из вершины i;
- G[i,j,1] — вершина, с которой соединена вершина i дугой с номером j в списке всех дуг из вершины i;
- G[i,j,2] — время проезда от вершины i до вершины, с которой она соединена дугой j в списке всех дуг из вершины i;
- D[i] — количество дорог, пересекающихся на перекрестке i (т.е. суммарное количество дуг, исходящих из вершины i и входящих в нее);
- vA и vB указывают, откуда и куда нужно добираться.

Поскольку по условиям задачи дороги двусторонние, то для каждой введенной дороги мы добавляем в граф две дуги.

Основной алгоритм выглядит следующим образом:

```
for i:=1 to N do
  begin
    for j:=1 to kG[i] do Color[i,j]:=WHITE; {все дуги свободны}
    Time[i]:=maxlongint;   {время маршрута до I — максимально}
  end;

OptT:=Maxlongint;              {Оптимальное время — максимально}
Sled[1]:=vA;                      {Заносим в маршрут вершину vA}
kv:=1;                         {Количество вершин в маршруте = 1}
Time[vA]:=0;                 {Оптимальное время до вершины vA =0}

DFS(vA);                                {Поиск в глубину от вершины vA}
... вывод ответа ...
```

Рекурсивная процедура DFS(i) обеспечивает выполнение следующей работы:

```
Procedure DFS(i)
  ...
   begin
      for j:=1 to kG[i] do
         if G[i,j,1]<>vB
           then
              begin
                 if color[i,j]=FREE
                   then
                      begin
                        ...  продолжение пути с вызовом DFS
                      end
              end
           else
              begin
                 ... сравнение пути с текущим оптимальным
              end;
   end;
```

Если текущая вершина — конечная (vB), то выполняем " ... сравнение пути с оптимальным".

Иначе, проверяем, если текущая дуга еще не использовась (color[i,j]=FREE), то выполняем "... продолжение пути с вызовом DFS". При этом перед входом в DFS помечаем дугу как использованную (color[i,j]:=DONE), а после выхода — как вновь свободную (color[i,j]:=FREE);

"... продолжение пути с вызовом DFS" включает в себя следующие операторы:

```
inc(kv); sled[kv]:=G[i,j,1];    {помещаем в путь новую вершину}
NewTime:=Time[i]+G[i,j,2]+CountTime;    {вычисляем новое время}
if NewTime<OptT          {если новое время меньше оптимального}
  then                          {то продолжаем,иначе - отсекаем}
    begin
      color[i,j]:=DONE;         {Помечаем - ребро использовано}
      RetTime:=Time[G[i,j,1]];        {Сохраняем старое время}
      Time[G[i,j,1]]:=NewTime;      {Устанавливаем новое время}
      DFS(G[i,j,1]);              {Вызываем поиск от G[i,j,1]}
      Time[G[i,j,1]]:=RetTime;  {Восстанавливаем старое время}
      color[i,j]:=FREE;           {Помечаем - ребро свободно}
    end;
Sled[kv]:=0; dec(kv);                   {Удаляем вершину из пути}
```

Для вычисления нового значения времени здесь используется функция CounTime с параметром kv — номер последней вершины в стеке. Эта функция восстанавливает номера вершин для прохода через перекресток "из i1 через i2 в i3":

```
if kv=2 then begin CountTime:=0; exit; end;
i1 := sled[kv-2];
i2 := sled[kv-1];
i3 := sled[kv];
if i3=i1 then begin CountTime:=K*D[i2]; exit; end;
```

Попутно, выясняется, что:

а) если вершин в пути всего 2, т.е. не было никакого поворота — это шаг из начальной вершины, и CountTime=0.

б) если i1=i3, то это поворот на 180°, и авторы задачи считают, что это тоже левый поворот. CountTime=K*D[i2]; где K — коэффициент, который вводится; i2 — перекресток; D[i2] — количество дорог, входящих в этот перекресток.

Затем из массивов координат перекрестков выбираются координаты текущих перекрестков:

```
(x1,y1) (откуда); (x2,y2) (через какой); (x3,y3) (куда).
      x1 := x[i1]; x2:=x[i2]; x3:=x[i3];
      y1 := y[i1]; y2:=y[i2]; y3:=y[i3];
```

Вычисляется уравнение прямой по точкам (x1,y1) и (x2,y2):

```
A := y2-y1;
B := x1-x2;
C := y1*(x2-x1)-x1*(y2-y1);
```

Подстановкой координат (x3,y3) в уравнение прямой Ax+By+C=0, построенной по первым двум точкам (x1,y1) и (x2,y2) и с учетом крайних случаев, когда прямая параллельна осям координат, вычисляем, был ли поворот левым или правым, и, соответственно, устанавливаем значение функции CountTime:

```
Left :=   (((x2>x1) and ((A*x3+B*y3+C)<0))) or
          (((x2<x1) and ((A*x3+B*y3+C)<0))) or
          (((y2=y1) and (x1>x2) and (y3<y1))) or
          (((y2=y1) and (x1<x2) and (y3>y1))) or
          (((x2=x1) and (y1>y2) and (x3>x1))) or
          (((x2=x1) and (y1<y2) and (x3<x1))) ;
if Left
   then CountTime:=K*D[i2]
   else CountTime:=0;
```

"Сравнение пути с оптимальным" выполняется следующим образом:

```
inc(kv); sled[kv]:=G[i,j,1];
T := Time[i]+G[i,j,2] + CountTime;
if T < OptT
   then begin
          OptT:=T;
          OptKv:=kv; OptSled:=Sled;
        end;
Sled[kv]:=0; dec(kv);
```

Таким образом, оптимальное время хранится в переменной OptT, а оптимальный путь храниться в массиве OptSled с количеством элементов OptKv. Поэтому вывод результатов выглядит так:

```
writeln(OptT);
for i:=1 to OptKv do writeln(OptSled[i]);
```

Ниже приводится полный текст программы:

```
program by98d2s2;
Const
    FREE  = 1;
    DONE  = 2;
Type
    int1000 = 0..1000;
    int3    = 1..3;
var
   G                  : array[1..1000,1..10,1..2] of int1000;
   Time,D             : array[1..1000] of longint;
   x,y,kG,Sled,OptSled,pred  : array[1..1000] of int1000;
   Color              : array[1..100,1..10] of int3;
   N,M,K,i,p,r,t,vA,vB,v,kv,OptKv,j : int1000;
   OptT               : longint;

  function CountTime(i:int1000):longint;
    var
       Left     : boolean;
       i1,i2,i3 : int1000;
       x1,x2,x3,y1,y2,y3,A,B,C      : longint;
    begin
      if kv=2 then begin CountTime:=0; exit; end;
      i1 := sled[kv-2];
      i2 := sled[kv-1];
      i3 := sled[kv];
      if i3=i1 then begin CountTime:=K*D[i2]; exit; end;
      x1 := x[i1]; x2:=x[i2]; x3:=x[i3];
      y1 := y[i1]; y2:=y[i2]; y3:=y[i3];
      A  := y2-y1;
      B  := x1-x2;
      C  := y1*(x2-x1)-x1*(y2-y1);
      Left := (((x2>x1) and ((A*x3+B*y3+C)<0))) or
              (((x2<x1) and ((A*x3+B*y3+C)<0))) or
              (((y2=y1) and (x1>x2) and (y3<y1))) or
              (((y2=y1) and (x1<x2) and (y3>y1))) or
              (((x2=x1) and (y1>y2) and (x3>x1))) or
              (((x2=x1) and (y1<y2) and (x3<x1))) ;
     if Left
       then CountTime:=K*D[i2]
       else CountTime:=0;
    end;

Procedure DFS(i:int1000);
    var
         j : byte;
       T : longint;
       RetSled,RetPred,RetTime :longint;
    begin
      for j:=1 to kG[i] do
        if G[i,j,1]<>vB
          then
            begin
              if color[i,j]=FREE
                 then
                    begin
                      inc(kv); sled[kv]:=G[i,j,1];
                      NewTime:=Time[i]+G[i,j,2]+CountTime;
                      if NewTime<OptT
                         then
                           begin
                             color[i,j]:=DONE;
                             RetTime:=Time[G[i,j,1]];
                             Time[G[i,j,1]]:=NewTime;
                             DFS(G[i,j,1]);
                             Time[G[i,j,1]]:=RetTime;
                             color[i,j]:=FREE;
                           end;
                      Sled[kv]:=0; dec(kv);
                    end
            end
          else
             begin
                 inc(kv); sled[kv]:=G[i,j,1];
                 T := Time[i]+G[i,j,2] + CountTime;
                 if T < OptT
```

```
                then begin
                               OptT:=T;
                               OptKv:=kv; OptSled:=Sled;
                             end;
                Sled[kv]:=0; dec(kv);
            end;
    end;

begin
    assign(input,'PER.in'); reset(input);
    assign(output,'PER.out'); rewrite(output);
    read(N,M,K);
    for i:=1 to N do readln(x[i],y[i]);
    for i:=1 to N do begin kG[i]:=0; D[i]:=0; end;
    for i:=1 to M do
       begin
            readln(p,r,t); inc(kG[p]); inc(kG[r]);
            G[p,kG[p],1]:=r; G[p,kG[p],2]:=t; inc(D[p]);
            G[r,kG[r],1]:=p; G[r,kG[r],2]:=t; Inc(D[r]);
       end;
    readln(vA,vB);

    for i:=1 to N do
       begin
           for j:=1 to kG[i] do Color[i,j]:=FREE;
           Time[i]:=maxlongint;
       end;
    Time[vA]:=0; kv:=1; Sled[1]:=vA; OptT:=Maxlongint;

    DFS(vA);

    writeln(OptT);
    for i:=1 to OptKv do writeln(OptSled[i]);
    close(input); close(output);
end.
```

3.3. Задача "Скрудж Мак-Дак"

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

Скрудж Мак-Дак решил сделать прибор для управления самолетом. Как известно, положение штурвала зависит от состояния входных датчиков, но эта функция довольно сложная.

Его механик Р.Винт сделал устройство, вычисляющее эту функцию в несколько этапов с использованием промежуточной памяти и вспомогательных функций. Для вычисления каждой из функций требуется, чтобы в ячейках памяти уже находились вычисленные параметры (которые являются значениями вычисленных функций), необходимые для ее вычисления. Вычисление функции без параметров может производиться в любое время. После вычисления функции, ячейки могут быть использованы повторно (например, для записи значения вычисленной функции). Структура вызова функций такова, что каждая функция вычисляется не более одного раза, и любой параметр используется не более одного раза. Любой параметр есть имя функции. Так как Скрудж не хочет тратить лишних денег на микросхемы, он поставил задачу минимизировать память прибора. По заданной структуре вызовов функций необходимо определить минимально возможный размер памяти прибора и указать последовательность вычисления функций.

Входной файл INPUT.TXT

Формат ввода:

1 строка — <общее количество функций N>

2 строка — <имя функции, которую небходимо вычислить>

3 строка — <имя функции> <кол-во параметров>[<список имен параметров>]

...

N+2 строка — <имя функции> <кол-во параметров>[<список имен параметров>]

Выходной файл OUTPUT.TXT

Формат вывода:

Размер памяти (в ячейках)

Имя 1-й вычисляемой функции

Имя 2-й вычисляемой функции

....

Имя функции, которую необходимо вычислить

Примечание: имя функции есть натуральное число от 1 до N.

Пример:

Ввод	Вывод
5	Размер памяти: 2 ячейки
1	Порядок:
1 2 2 3	Функция 4
2 0	Функция 5
3 2 4 5	Функция 3
4 0	Функция 2
5 0	Функция 1

В кратком изложении в данной задаче требуется сделать следующее:

- найти S — максимальную степень среди тех вершин, что подлежат обходу (поиск в глубину DFS1);
- вычислить необходимую память:

$$MaxS = S + k - 1,$$

где S найдено на предыдущем шаге (DFS1), k — максимальное количество вершин одного уровня вложенности с такой же степенью S;

- для вычисления k совершается обход повторным поиском в глубину DFS2;
- третьим поиском в глубину (DFS3) найти и поместить в очередь V все вершины, степень которых равна S.
- последним, четвертым поиском в глубину (DFS4) обходим данное дерево в порядке вершин в очереди V. Т.е. в первую очередь вычисляем те функции, для которых требуется максимальная память.

Рассмотрим реализацию алгоритма более подробно. Ввод исходных данных осуществляется следующим образом:

```
read(N,FC);
for i:=1 to N do
    begin
        read(f,kG[f]);
        for j:=1 to kG[f] do read(G[f,j]);
    end;
```

Здесь:

- N — исходное количество вершин;
- FC — номер функции, которую нужно вычислить;
- kG[i] — количество параметров для вычисления функции i;
- G[i,j] — j-й параметр, требуемый для вычисления функции i.

Тело главной программы выглядит так :

```
for i:=1 to N do color[i]:=WHITE;          {Пометить свободными}
                                                {все вершины}
DFS1(FC);              {Находим S - максимальную степень вершин,}
                         {используемых для вычисления функции FC}
MaxS:=S;
DFS2(FC);     {Вычисляем K - количество вершин со степенью S в}
                {текущем слое и MaxS - максимальная из значений}
                                        {по слоям величина S+K-1}
kv:=0;
DFS3(FC);  {Поместить в массив V все вершины, степень которых}
          {равна S, количество таких вершин - в переменной kv}

kr:=0;
for i:=1 to kv do DFS4(v[i]);  {Обход графа поиском в глубину}
                                 {начиная с вершин с максимальной}
                                 {степенью из массива V. Найденные}
                                    {вершины заносить в массив r}
```

```
writeln(MaxS);                    {Вывод количества ячеек памяти}

for i:=1 to kr do writeln(r[i]);                {Вывод порядка}
                                          {вычисления функций}
```

Ниже приводится полное решение задачи:

```
program by95d2t1;
const
    WHITE = 1;
    GRAY  = 2;
    BLACK = 3;
var
   G                        : array [1..100,1..100] of
longint;
   kG,v,color,r             : array [1..100] of longint;
   N,FC,i,j,s,f,kv,MaxS,kr : longint;

procedure DFS1(u:longint);
  var
     i,k : longint;
  begin
     if s<kG[u] then s:=kG[u];
     for i:=1 to kG[u] do DFS1(G[u,i]);
  end;

procedure DFS2(u:longint);
   var
      i,k : longint;
   begin
      k:=0;
      for i:=1 to kG[u] do
          if kG[G[i,j]]=s then inc(k);
      if MaxS<s+k-1 then MaxS:=s+k-1;
      for i:=1 to kG[u] do DFS2(G[u,i]);
  end;

procedure DFS3(u:longint);
   var
      i,k : longint;
   begin
      if kG[u]=s
         then
            begin
                for i:=1 to kG[u] do DFS3(G[u,i]);
                inc(kv); v[kv]:=u;
            end;
   end;

procedure DFS4(u:longint);
   var
      i : longint;
  begin
     color[u]:=GRAY;
     if kG[u]<>0
       then
          for i:=1 to kG[u] do
              if color[G[u,i]]<>GRAY
                 then DFS4(G[u,i]);
     inc(kr); r[kr]:=u;
  end;

begin
   assign(input,'input.txt'); reset(input);
   assign(output,'output.txt'); rewrite(output);
   read(N,FC);
   for i:=1 to N do
       begin
           read(f,kG[f]);
           for j:=1 to kG[f] do read(G[f,j]);
       end;
  for i:=1 to N do color[i]:=WHITE;
  DFS1(FC);
  MaxS:=s; DFS2(FC);
  kv:=0;       DFS3(FC);
  kr:=0;       for i:=1 to kv do DFS4(v[i]);
  writeln(MaxS);
  for i:=1 to kr do writeln(r[i]);
  close(input); close(output);
end.
```

(Продолжение следует)

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

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

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

Программирование под WIN32

5. Знакомство с SoftIce'ом

#1. Сначала был 8086-й процессор, операционная система DOS и отладчик *debug* фирмы Microsoft. Отладчик неудобный, со скромными возможностями — он был (однако, есть и будет!) пригоден разве что для разного рода низкоуровневых забав и изучения ассемблера ;). В то время отладчики росли подобно грибам после мягкого кислотного дождика. И с такой же скоростью уходили в забвение — ибо от своего мелкомягкого прототипа отличались лишь интерфейсом. Это было золотое время для разработчиков всемозможных защит, так как достаточно было "запереть" клавиатуру, запретить прерывания, сбросить флаг трассировки, и у незадачливого хакера надолго отпадала охота копаться в чужом исполняемом коде...

Потом был 80286-й, такие шедевры программирования как отладчики *AFD Pro* и *Turbo Debugger*... Золотое время разработчиков защит плавно перетекло в серебряное. А затем, с выходом на рынок фирмы NuMega и ее отладчика *SoftIce*, их жизнь вовсе превратилась в адский, неблагодарный, и, что самое важное, малоэффективный труд... Было вот как — разработчики защит были отдельными личностями либо небольшими фирмами, а за разработчиками *SoftIce* стояла намного более многочисленная группа людей, да не абы каких, а хакеров ;). Притом хакеров не в (увы!) современном понимании этого слова (т.е. прыщавых компутерных вандалов, постоянных покупателей клирасила), а в единственно верном его толковании, т.е. людей в первую очередь высокоинтеллектуальных и стремящихся во всем разобраться до мелочей... Поэтому нисколько не удивительно, что на появление 80386-го, который имел специальные механизмы, обеспечивающие контроль за исполнением кода на аппаратном уровне, NuMega отреагировала намного быстрее, чем противоположный лагерь, который "разобрался" с этими новыми возможностями только годы спустя; и, как очевидно, при этом безнадежно проиграл "хакерским средствам".

А потом был Прозорливый Билли и его "принципиально новая архитектура", перед которой пасовали все существующие отладчики. Взамен им мелкософт предлагал свои, но ориентированы эти неповоротливые монстры были на программиста, отлаживающего собственный код, но никак не исполняемый файл, избавленный от отладочной информации. При этом документацию, необходимую для написания отладчиков под Windows, мелкомягкие не распространяли... некоторое время. И несмотря на то что конкурирующие фирмы все-таки вытянули ее посредством каленых клещей US'овской судебной системы, легче им от этого не стало — их отладчики все равно не превосходили мелкософтовский. Ибо это были отладчики под Windows. Неповоротливые отладчики под такой же неповоротливый виндовс.

NuMega же пошла своим путем, и повторить ее шедевр до сих пор никто не решается. Если операционная система не позволяет отлаживать программы — "забиваем" на опе-

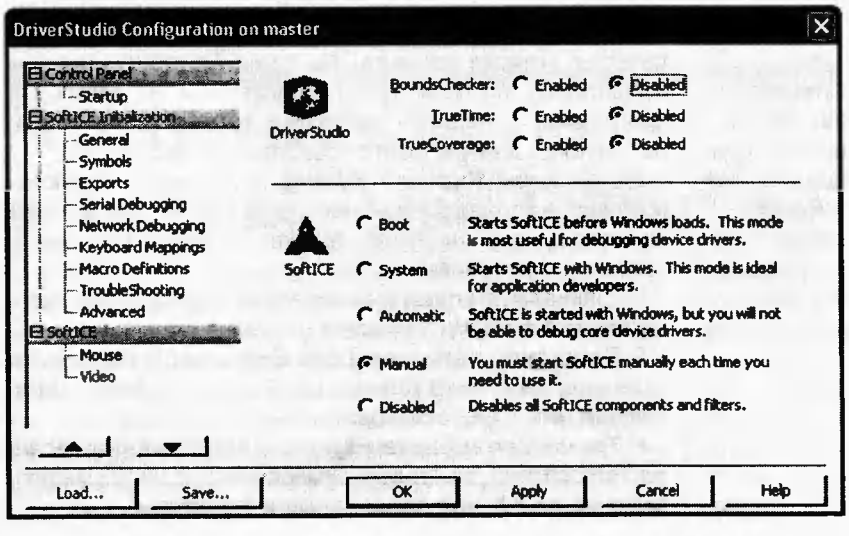
рационную систему! Если стену не перепрыгнуть и не обойти, то почему бы ее не подкопать? Результат такого подхода оказался выше всяких похвал. "Виндозный" отладчик от NuMega ни в чем не полагался на операционную систему, а опирался исключительно на аппаратное обеспечение и, вследствие этого, позволял отлаживать практически **любую** программу, в том числе и ядро операционной системы...

Что мы имеем в результате всего этого? Фирму NuMega — единственного "поставщика" высококачественных хакер-ориентированных иструментов под Windows, оперативно реагирующего на все изменения операционной системы от мелкософта...

Подробнее обо всем этом вы можете прочитать в книге К.Касперски "Техника и философия хакерских атак" — настоятельно рекомендую (при всем своем уважении к автору, не могу не отметить, что второе издание этой книги, выпущенное издательством Солон, имеет отвратительный переплет — при первом же прочтении книга буквально развалилась на отдельные страницы).

#2. Как вы, должно быть, уже поняли, способ запуска сайса (именно так мы будем назвать SoftIce) несколько сложнее, чем всенародно любимого пасьянса "косынка". В Windows 9X для этого нужно было редактировать *autoexec.bat* и делать мультиконфигурацию, но в NT с этим дела обстоят немного проще. Всю последовательность действий я привожу из предположения, что у читателя установлена операционная система Windows 2000 (которая, как известно, "build on NT technology"), английская, а сайс берется из пакета *DriverStudio* версии 2.5. При этом желающие "сходу" произвести полную установку этого пакета должны иметь в виду, что программа инсталляции попросит, помимо всего прочего, указать пути к *Driver Development Kit* ;).

Итак, после установки (и появления на панели задач соответствующей ветки меню), первое, что мы должны сделать, это выбрать тип его загрузки. Для этого запускаем программу конфигурирования (***Start > NuMega DriverStudio > SoftIce > Settings***) и видим следующее окошко:



Честно говоря, меня вполне устраивает ручной режим загрузки, то есть *Manual* — наверное, это потому что я еще не сталкивался с отладкой "core device driver" ;). Поэтому насчет остальных режимов ничего сказать не могу. Итак, ставим *Manual* ;).

Чтобы запустить сайс в этом режиме, необходимо ввести команду:

```
NET START NTICE
```

либо запустить "Start SoftICE" из менюшки (pif на обыкновенный батник, где эта команда написана).

Все! Отладчик запущен. Только вот не увидите вы его ни в *Applications*, ни в *Processes*. Ни даже на экране — пока не нажмете на волшебную комбинацию клавиш Ctrl+D ;).

Жмем на Ctrl+D! Если отладчик установился успешно, то всплывет приблизительно следующая "картинка":

```
-------------------------------------------------------------------------------
EAX=00005305   EBX=C4920074   ECX=C14698E4   EDX=00000000   ESI=C1476EC0
EDI=C49202B0   EBP=67890000   ESP=C4687E2C   EIP=000080D2   o d I s z a P c
CS=0128   DS=0030   SS=0030   ES=0030   FS=0078   GS=0030
-------------------------------------------------byte----------------PROT16-
0030:00000000 9E 0F C9 D8 65 04 70 00-16 00 C9 09 65 04 70 00 ....e.p.....e.p.
0030:00000010 65 04 70 00 54 00 FF F0-58 7F 00 F0 FF E7 00 F0 e.p.T...........
0030:00000020 00 00 00 C9 D2 08 A3 0A-6F EF 00 F0 6F 00 F0 00 ........o...o..
0030:00000030 6F EF 00 F0 6F EF 00 F0-9A 00 C9 09 65 04 70 00 o...o.......e.p.
---------------------Cancel_Call_When_Idle+002C-----------------------PROT16-
0128:80D1  POPF
0128:80D2  CLS
0128:80D3  RETF
0128:80D4  POPF
0128:80D5  STC
0128:80D6  RETF
0128:80D7  CMP     AL,13
0128:80D9  NOP
0128:80DA  NOP
0128:80DB  JBE     80E1
-------------------------------------------------------------------------------
:rs
:g
WINICE: Free32  Obj=01 Mod=NOTEPAD
WINICE: Free32  Obj=02 Mod=NOTEPAD
WINICE: Free32  Obj=03 Mod=NOTEPAD
WINICE: Free32  Obj=04 Mod=NOTEPAD
WINICE: Free32  Obj=05 Mod=NOTEPAD
WINICE: Free16  Sel=351F
:X
-------------------------------------------------------------------------------
X, XFRAME, XG, XP, XRSET, XT                                        KERNEL32
-------------------------------------------------------------------------------
```

Сверху, как вы уже, должно быть, догадались, регистры. Чуть ниже — дамп памяти. Еще ниже — дизассемблированные команды процессора. Далее следуют окно диспетчера, в котором мы можем вводить команды и читать различные сообщения, и контекстная подсказка.

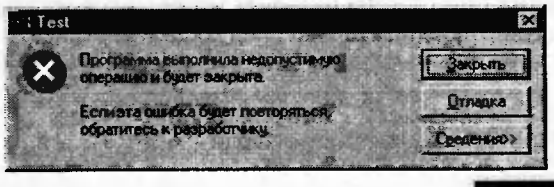
#3. Не знаю, как на вашем дисплее, но на моем 17-дюймовом с разрешением 1024x768 окошко отладчика получилось больно уж маленьким. Чтобы не напрягать глаза, его можно немножко "под себя" настроить.

Ок! Давайте попробуем ввести несколько команд, которые позволяют выполнить подобную настройку. Для этого пишем следующие команды:

- **SET FONT 2** — и шрифт в окошке немного увеличивается, как и сам размер окна;

- **LINES 60** — увеличиваем число строк в окне отладчика;
- **WD 22** — задаем число строк под дамп;
- **WC 25** — задаем число строк под код;
- **CODE ON** — разрешить отображать байты инструкций;
- **COLOR A A 20 20 2** — устанавливаем "извращенную" цветовую схему (подробнее о параметрах смотрите в *SoftIce Command Reference*, про битовое кодирование цвета см. 1.6. #3).

Еще одна команда (я настоятельно рекомендую ее использовать, особенно пользователям W9X) — это **FAULTS OFF**, которая предотвращает "всплытие" отладчика при возникновении *GPF — General Protection Fault*, в просторечии также известную как "ваши ручки выполнили недопустимую операцию и будут ампутированы":



Теперь, когда мы настроили "под себя" внешний вид отладчика, давайте позаботимся о том, чтобы нам не нужно было при его запуске каждый раз вводить эти семь команд. Т.е. пропишем все эти команды в строку инициализации — "простыню" команд, которые автоматически будут выполняться при загрузке отладчика.

Для этого ищем конфигурационный файл *winice.dat* (в 2000 я его нашел в Windows\system32\drivers\; в 9X, насколько я помню, он находился в том же каталоге, что и проинсталлированный SoftICE) и дополняем строчку ***INIT="X;"*** нашими командами ("X;" в самом конце — это выход из окна сайса):

```
INIT="SET FONT 2; LINES 60; WD 22; WC 25; CODE ON; COLOR A A 20 20 2;
FAULTS OFF; X;"
```

Далее раскомментируйте в winice.dat все строки наподобие

```
EXP=\SystemRoot\System32\kernel32.dll
```

Это необходимо, для того чтобы сайс загрузил имена экспортируемых функций, находящихся в этих библиотеках. Иначе вместо понятных команд, наподобие

```
call USER32!MessageBoxA
```

мы увидим безобразие типа

```
call 0044F2A1
```

Теперь нам нужно перезапустить отладчик, чтобы проверить, подхватывает ли сайс наши настройки. Для этого нужно... В общем, как я уже говорил, сайс — прога весьма специфическая, и выгрузить ее "из компьютера" можно только одним способом — перезагрузкой ;). ***Start > Shutdow > Restart...***

#4. Существуют две области применения сайса — отладка собственных программ и исследование чужих, так называемый *reverse ingeneering*. Для начала давайте научимся использовать сайс в качестве инструмента для отладки и изучения собственных приложений.

Что мы имеем в этом случае? Исходные тексты программы, и как следствие — возможность откомпилировать ее отладочную версию, т.е. тот же экзешник, но содержащий, помимо всего прочего, еще и кучу дополнительной информации. Как в самом исполняемом файле, так и в специально для этой цели сгенерированных дополнительных файлах.

Итак, мы должны:

1. Проассемблировать исходник с ключем Zi:

```
ml /c /coff /Zi src.asm
```

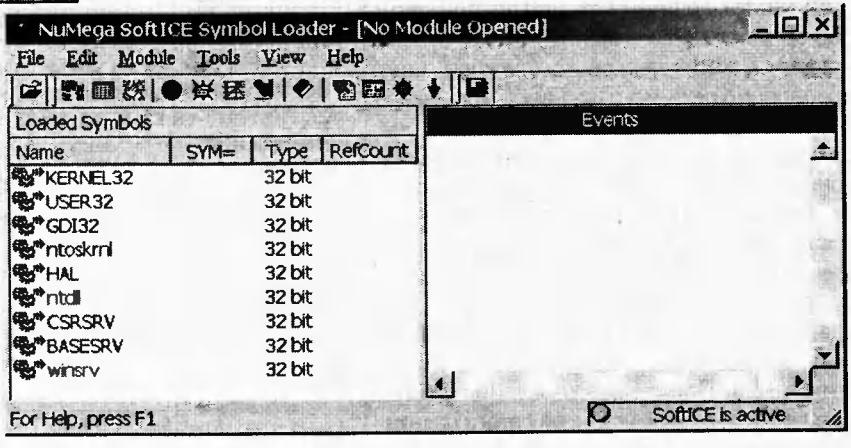
В результате этого мы получим объектный файл, содержащий отладочную информацию для отладчика CodeView. Легко заметить, что размер этого файла намного больше, чем у его "нормального" аналога.

2. Слинковать объектный файл с ключами /DEBUG и /DEBUGTYPE:CV

```
link.exe /SUBSYSTEM:CONSOLE /DEBUG /DEBUGTYPE:CV src.obj
```

После этого, помимо экзешника, мы получим отладочные файлы *src.ilk* и *simple.pdb*, *Microsoft Linker Database* и *Microsoft C/C++ program database* соответственно.

Теперь загружаем нашу **отладочную версию** программы в отладчик. Для этого запускаем *Symbol Loader*:



В левом окне мы видим, из каких файлов подгружена символическая информация. Зеленая лампочка в строке статуса свидетельствует о том, что отладчик подгружен (конечно, если вы это сделали после перезагрузки). Правое окно — это окно отчета, там появляется информация о выполненных действиях, ошибках и т.д. Ах да..., еще есть заголовок окна, там, помимо названия программы, есть еще и надпись в скобках **[No Module Opened]**, то есть "не открыт модуль". Не правда ли, не очень тонкий намек?

Жмем ***File > Open*** и открываем наш *src.exe* (отладочные файлы, как, желательно, и исходник, должны находиться в этом же каталоге). На первый взгляд ничего не изменилось, но посмотрите внимательно на заголовок программы — надпись "не открыт модуль" заменилась на "src.exe". Значит, что-то все-таки произошло ;).

Далее жмем ***Module > Setting***, и всплывает диалоговое окно, в котором можно настроить все, что мы желаем сделать с модулем, перед тем как он будет загружен в отладчик. Настраиваем:

Из закладки **General** нам ничего не нужно — все поля оставляем пустыми, никаких галочек не ставим.

В **Debugging** выбираем ***Load Executeable*** ("загрузить экзешник") и ставим галочку на ***Stop at WinMain, Main, DllMain, etc***... т.е. "остановиться на точке входа".

В ***Translation*** выбираем ***Symbols only (included locals and structures)***, т.е. "только символическую информацию, включая локальные переменные и структуры".

Теперь мы готовы свершить самое главное действо — загрузить программу под отладчик. Жмем ***Module > Load***, и вот оно, долгожданное! У нас всплывает окно сайса с указателем, установленным на точку входа в нашу программу:

```
Main
001B:00401010  55                PUSH EBP
```

#5. Для начала рассмотрим команду **display memory** (отобразить память). Ее синтаксис:

```
D[size] [address [l length]]
```

Если ее использовать без параметров, то просто будет отображаться следующая страница дампа.

b	Byte
w	Word
d	Double Word
s	Short Real
l	Long Real
t	10-Byte Real

Необязательный параметр *size* — это размерность, в которой выводится дамп. В таблице расшифровка его значений:

Размерности *byte*, *word* и *double word* вам, конечно же, хорошо знакомы. А вот загадочные *short real* и *10-byte real* мы рассмотрим немного позже.

Обратите внимание на то, что первый параметр (размерность) мы должны писать "в одно слово" с, собственно, самой командой "d", т.е. — "db", "dw", "dd" и т.д.

Необязательный параметр address — это адрес памяти, дамп которого вы хотите получить. Притом вовсе не обязательно писать адрес цифрами — "составные" адреса сайс также принимает "за милую душу".

Таким образом, если мы введем команду:

```
DD EIP
```

то увидим дамп той области памяти, в которой в настоящее время располагается секция кода нашей программы (конечно, если после всплытия отладчика на точке входа вы ничего не успели напортачить).

Если же мы начнем использовать параметр length, то данные, запрашиваемые с использованием этого параметра, начнут отображаться в командной строке. Например, команда

```
DW EIP L 1000
```

выведет в командную строку 1000 байтов памяти, начиная с текущего значения регистра EIP, сгруппированных пословно, и вы будете вынуждены несколько раз нажать на any key, прежде чем все это просмотрите. По большому секрету скажу, что нажатие на ESC сразу же прервет просмотр.

#6. Пожалуй, одна из главных возможностей любого отладчика — это трассировка, которая позволяет выполнять программу пошагово. Итак, загрузим нашу программу в SymbolLoader, нажмем на Load, а в появившемся окне отладчика введем команду "p" (она же — клавиша F10). В результате этого выполнится один логический шаг нашей программы (program step). Под "логическим шагом" подразумевается своего рода "поверхностная" трассировка, без входа в процедуры, циклы и т.д.

Команда "t" (она же — клавиша F8) выполняет трассировку одной инструкции (trace one instruction), с "заходом" во все функции, в том числе и апишные. Подобная трассировка — это очень длинный путь, поэтому у данной команды существует еще и два параметра:

```
T [=start-address] [count]
```

Первый — это адрес, с которого вы желаете начать трассировку, а *count* — это число инструкций, которое сайс протрассирует, прежде чем остановиться (и не забывайте о том, что он "заходит" в апишные функции!).

Конечно же, трассировочные возможности сайса намного больше, чем те немногие, которые мы рассмотрели. Впоследствии мы обязательно ознакомимся со всем их многообразием — по мере необходимости ;).

#7. Еще одна полезная команда — это rs (она же клавиша F4), restore the program screen (восстановление экрана программы). А как же без этого? Ведь внешне работа программы заключается вовсе не в выполнении команд процессора и пересылках данных между регистрами ;).

Например, в нашем случае программа печатает приглашение ввести строку символов. Мы можем оттрассировать программу до (включительно) строки:

```
001B:00401046   E8BD000000 CALL _WriteConsoleA
```

и затем нажать на F4, для того чтобы подсмотреть, действительно ли в консольном окне появилась строчка "Type something >". А затем нажать на any key и снова очутиться в отладчике.

Или же, например, выполнив апишную функцию

```
001B: 00401070 E881000000 CALL _ReadConsoleA,
```

которая "просит" ввести строку символов, нажать на F4 и ввести необходимую строку символов, а по нажатию на Enter (типа "ввод закончен") снова оказаться в отладчике.

#8. Теперь, когда мы поверхностно ознакомились с "хакерскими" инструментами, пришло время вспомнить о цели, которую мы преследовали, начав знакомство с идой и сайсом. Напоминаю — мы хотели исследовать, каким образом происходит декларирование локальных переменных в стеке, и как происходит обращение к ним. Что же, у вас есть целый месяц, чтобы исследовать это самостоятельно ;). Свою версию исследования я предоставлю в следующем номере журнала.

И да пребудет с вами сила!

(Продолжение следует)

Трактать о Dynamic Link Libraries

А.ИВАНЧИКОВ, И.ИВАНЧИКОВА,
г.Минск.

Краткие теоретические сведения

Все Windows-приложения используют специальный формат исполняемых файлов — формат PE (Portable Executable). Такие файлы начинаются как обычные EXE-файлы старого образца (их также называют MZ по первым двум символам заголовка), но после обычного MZ-заголовка файла следует PE-заголовок. Если такой файл попытаться выполнить из DOS, он выдаст на консоль сообщение об ошибке — "This program cannot be run in DOS mode." и корректно завершится.

Кроме обычных приложений (EXE-файлов), в Windows появился специальный тип исполняемого файла — динамические библиотеки (DLL — Dynamic Link Library). DLL — это файл, содержащий коды функций и данные, которые доступны программам, обращающимся к нему. Динамические библиотеки позволяют уменьшить использование памяти и размер файлов для тех случаев, когда несколько программ (или несколько копий одной программы) используют один и тот же функционал. Можно считать, что DLL — это

аналог пассивной резидентной программы, с тем лишь отличием, что DLL не находится в памяти, если ни одна программа, его использующая, не загружена.

Можно выделить следующие преимущества организации программного кода в виде DLL:

- методы, экспортируемые из DLL, могут одновременно использоваться несколькими приложениями. При этом сами методы находятся в ОЗУ в единственной копии. Если вызываемый код достаточно велик, и имеется несколько приложений, которые вызывают данный код, то использование DLL позволяет достичь существенной экономии системных ресурсов;
- возможность хранения общих ресурсов. Если несколько приложений работают с одними и теми же ресурсами (к примеру, с растровыми изображениями), то при сохранении их в DLL можно иметь эти ресурсы в одном экземпляре;
- поддержка новых версий приложений. В случае внесения изменений в реализацию методов, определенных в DLL, достаточно передать конечному пользователю новую версию DLL. При этом клиентский код остается без изменений (при условии неизменности интерфейса);
- независимость языков разработки DLL и клиентских приложений, эти DLL использующих.

Первый опыт (не "Hello, World!")

В качестве примера рассмотрим процесс создания DLL, предоставляющей одну функцию QuickSort, которая реализует алгоритм быстрой сортировки. Сразу оговоримся, что данная реализация не может претендовать на "промышленное" использование из-за наличия рекурсивных вызовов. При достаточной размерности сортируемого массива обязательно возникнет ошибка, связанная с переполнением стека. Однако в качестве примера данная реализация выглядит привлекательно, тем более, что скорость работы функции QuickSort не ниже скорости работы библиотечной C-функции qsort.

Коротко алгоритм быстрой сортировки можно описать следующей последовательностью двух шагов:

- массив A[n] делится функцией SplitArray на два массива: A[0], ... A[q] и A[q+1], ... A[n-1] с соблюдением условия — каждый из элементов массива A[0], ... A[q] меньше, чем каждый из элементов массива A[q+1], ... A[n-1];
- функция SplitArray рекурсивно вызывается для двух полученных массивов.

Ниже представлен исходный текст файла QuickSort.cpp.

```
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  QuickSort.cpp
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
// Смена значений *pcElem1 и *pcElem2
//-------------------------------------------------
static void swapElem(
    char* pcElem1,
    char* pcElem2,
    unsigned unWidth )
{
    char cTmp;

    if ( pcElem1 != pcElem2 )
        while( unWidth- )
        [
            cTmp = *pcElem1;
            *pcElem1++ = *pcElem2;
            *pcElem2++ = cTmp;
        }
}

// Разделение одного массива на два
//-------------------------------------------------
static char* SplitArray(
    char* pcFirstElem,
    char* pcLastElem,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) )
{
    char* pcFirst = pcFirstElem;

    pcLastElem += unWidth;
    pcFirstElem -= unWidth;

    while ( pcLastElem > pcFirstElem )
    {
        do
            pcLastElem -= unWidth;
        while ( compF(pcLastElem, pcFirst) > 0 );
        do
            pcFirstElem += unWidth;
        while ( compF(pcFirstElem, pcFirst) < 0 );
        if ( pcLastElem > pcFirstElem )
            swapElem(   pcFirstElem, pcLastElem,
                        unWidth);
    }
    return pcLastElem;
}

// Управление процессом сортировки
//-------------------------------------------------
static void RecSort(
    char* pcFirstElem,
    char* pcLastElem,
    unsigned unWidth,
    int (*compF)(   const void *pvElem1,
                    const void *pvElem2) )
{
    if ( pcFirstElem < pcLastElem )
    {
        char* pcMediumElem = SplitArray(
                pcFirstElem, pcLastElem,
                unWidth, compF);
        RecSort(pcFirstElem, pcMediumElem,
                unWidth, compF);
        RecSort(pcMediumElem + unWidth,
                pcLastElem, unWidth, compF);
    }
}

// Начало процесса сортировки
//-------------------------------------------------
extern "C" 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);
}
```

Ключевым моментом в создании DLL является организация экспорта функций, т.к. именно в этом случае они становятся доступны внешним приложениям. Для этого необходимо:

- обеспечить C-компоновку для экспортируемых функций пометкой функции как *extern "C"*. Данное выражение в описании функции нужно, чтобы компилятор C++ не кодировал имя функции информацией о типе. Без *extern "C"* MS Visual C++ 6.0 преобразует, например, имя функции QuickSort в ?QuickSort@@YAXPAXIIP6AHPBX1@Z@Z. Другие компиляторы используют иные схемы дополнения имени информацией о типе. На дополненные имена нет стандарта, поэтому они не переносимы;
- сообщить компоновщику, что функция экспортируется. Для этого создается .DEF-файл. В разделе *EXPORTS* пе-

речисляются имена экспортируемых функций. При желании можно назначить каждой функции порядковый номер. В строке *LIBRARY* указывается фактическое имя DLL.

В нашем случае предполагается экспортировать одну функцию QuickSort. Поэтому файл QuickSort.def будет иметь вид:

```
;------------------------------------------------
;  QuickSort.def
;------------------------------------------------
LIBRARY QuickSort.dll
EXPORTS
        QuickSort @1 PRIVATE
```

Собрать DLL, используя компилятор/компоновщик командной строки MS Visual C++, можно либо выполнив последовательно две команды (при желании их можно объединить в пакетном .BAT-файле):

```
cl /c /O2 QuickSort.cpp
link /DLL QuickSort.obj /DEF:QuickSort.def,
```

либо выполнить makefile следующего содержания (командной строкой nmake –f makefile):

```
#------------------------------------------------
#  makefile for QuickSort.dll building
#------------------------------------------------
QuickSort.dll:
        cl /c /O2 QuickSort.cpp
        link /DLL QuickSort.obj /DEF:QuickSort.def
```

Две строки компоновки можно заменить одной —

```
cl /LD /O2 QuickSort.cpp QuickSort.def.
```

Однако в этом случае компоновщик встроит в результирующую DLL пустую функцию DllMain, просто возвращающую значение TRUE. Это незначительно увеличит размер DLL. Функция DllMain является стандартной точкой входа DLL (аналогично функции main для консольных приложений), и в нашем случае она не нужна.

При помощи утилиты Dumpbin.exe из пакета поставки MS Visual C++ можно получить список имен, экспортируемых из DLL. Команда

```
dumpbin -exports QuickSort.dll
```

сгенерирует для QuickSort.dll такой результат:

```
Dump of file QuickSort.dll

File Type: DLL

Section contains the following exports for QuickSort.dll
          0 characteristics
    3C593307 time date stamp Thu Jan 31 14:05:27 2002
        0.00 version
           1 ordinal base
           1 number of functions
           1 number of names

    ordinal hint RVA      name
          1    0 00001000 QuickSort

  Summary
        3000 .data
        1000 .rdata
        1000 .reloc
        4000 .text
```

Для проверки работоспособности созданной DLL воспользуемся специально написанной консолью TestSort.exe. Ниже приведен исходный текст файла TestSort.cpp.

```
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
//  TestSort.cpp
//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//~\\~//
#include <stdio.h>
#include <stdlib.h>
#include <wtypes.h>
// размерность массива
#define SIZE_TETS_ARRAY 10
// Новый тип – указатель на функцию QuickSort
//-----------------------
typedef void(*QUICKSORTPTR)(
    void* pvArray,
    unsigned unCount,
    unsigned unWidth,
    int (*compF)(   const void* pvElem1,
                    const void* pvElem2) );

// Функция сравнения для QuickSort
//-----------------------
int _cmp(
    const void* ppvElem1,
    const void* ppvElem2 )
{
    return *(int*)ppvElem1 - *(int*)ppvElem2;
}

//-------------------------------------------------------
void main(void)
{
    // загрузка DLL
    HMODULE hModl =
                ::LoadLibrary("QuickSort.dll");

    // если загрузить не удалось – выход
    if ( !hModl )
        return;

    // получение адреса функции QuickSort
    QUICKSORTPTR pQuickSort =
        (QUICKSORTPTR)::GetProcAddress(hModl, "QuickSort");

    // если возникла ошибка – выход
    if ( !pQuickSort )
    {
        ::FreeLibrary(hModl);
        return;
    }

    // демонстрация работы QuickSort
    int nArray[SIZE_TETS_ARRAY];

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

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

    // выгрузка DLL
    ::FreeLibrary(hModl);
}
```

Собрать консольное приложение TestSort можно, выполнив командную строку

```
cl /O2 TestSort.cpp
```

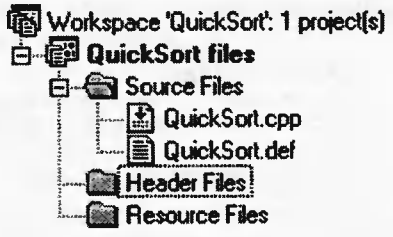
Излишне говорить, что при выполнении приложения TestSort.exe библиотека QuickSort.dll должна находиться в том каталоге, который был указан при вызове LoadLibrary.

Разумеется, наиболее удобную работу над проектом обеспечивает интегрированная среда MS Visual Studio. Для построения проекта, генерирующего DLL, после загрузки MS Visual C++ следует выбрать пункт *File→New...* главного меню. В открывшемся диалоге выбрать пункт *Win32 Dynamic-Link Library*; ввести в окне *Project Name* строку QuickSort; в окне *Location* выбрать путь, по которому данный проект будет располагаться, и щелкнуть по кнопке *Ok*. На следующем шаге согласиться с пунктом, выбранным по умолчанию (*An empty DLL project.*), щелкнув по кнопке *Finish*.

Теперь, когда пустой проект создан, необходимо добавить в него файл исходного кода QuickSort.cpp и файл

определения модуля QuickSort.def. Для этого следует перейти на закладку *FileView* окна окружения проекта, щелкнуть правой кнопкой мыши на строке *Source Files* и выбрать пункт контекстного меню *Add Files to Folder...* В открывшемся диалоге необходимо выбрать два указанных выше файла и щелкнуть по кнопке *Ok*. Теперь окно проекта имеет вид, показанный на **рис.1**.

Рис. 1



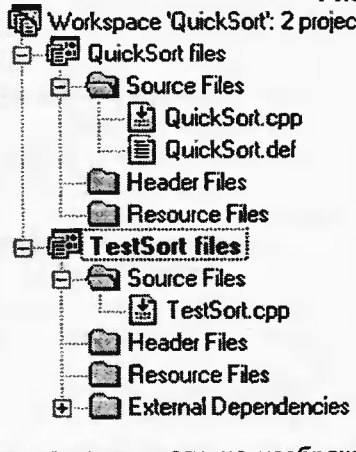
Собрать QuickSort.dll Debug и Release конфигурации можно, выбрав пункт главного меню *Build→Batch Build...* и щелкнув по кнопке *Build*.

После завершения процесса компиляции и компоновки две одноименные DLL будут находиться на диске в соответствующих каталогах Debug и Release в каталоге проекта.

Для упрощения процесса отладки QuickSort.dll и для демонстрации ее возможностей полезно добавить в окружение проектов еще один проект — TestSort.

Это можно легко сделать, щелкнув на строке *Workspace 'QuickSort': 1 project(s)* (в закладке *FileView* окна окружения проекта) правой кнопкой мыши и выбрав из контекстного меню пункт *Add New Project to Workspace...* В открывшемся диалоге выбрать пункт *Win32 Console Application*, в поле *Project Name:* ввести строку TestSort и щелкнуть по кнопке *Ok*. На следующем шаге выбрать *An empty project* и щелкнуть по кнопке *Finish*. Остается добавить в созданный проект файл исходного текста TestSort.cpp. После означенных процедур окно окружения проектов будет выглядеть аналогично изображенному на **рис.2**.

Рис. 2



Для того чтобы результирующие файлы TestSort.exe обеих конфигураций сборки помещались в каталоги основного проекта Release/Debug, т.е. туда, куда будут сложены и построенные динамические библиотеки, необходимо выбрать пукт гдавного меню *Project→Settigs...*, в открывшемся диалоге *Project Settigs* перейти в закладку *General*, и для проекта TestSort конфигураций Release и Debug ввести в окне *Output files:* строки "../Release" и "../Debug" соответственно.

Теперь можно собрать любую конфигурацию работающих тестового примера и библиотеки "одним щелчком мыши". Построение проекта в интегрированной среде предоставляет полный сервис по его сопровождению, что в значительной степени означает удобную работу с отладчиком в реальном времени.

(Продолжение следует)

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

Физические адреса в Win 95 (98)

О чем пойдет речь

Вы никогда не задумывались над тем, в каком именно мегабайте вашего компа выполняется ваша программа? А в каком уютно разместился кернел? Нет? А мне вот стало интересно, и я решил узнать...

Страничная адресация

Для начала немного теории. Несколько слов о том, как процессоры 386+ осуществляют страничную адресацию. Информация взята из [1].

Пара определений. Физическим адресом назовем реальный номер байта в памяти. Линейными адресами назовем адреса, которые используют выполняющиеся программы.

Допустим, для доступа к сегменту данных в win32-приложении может встретиться такая инструкция:

mov byte ptr ds:[eax],2h

Как же процессор может вычислять физический адрес требуемой ячейки памяти?

Если не используется страничная адресация, то процессор обратится к дескриптору памяти, задаваемому значением селектора в регистре ds, возьмет из него базовый адрес сегмента памяти, являющийся именно физическим адресом, прибавит к нему смещение, задаваемое регистром eax:

Физический адрес = база из дескриптора + eax.

Если включено страничное преобразование, то все происходит иначе. Рассмотрим случай размера страницы в 4 Кб. Процессор выделяет из значения смещения в eax три части:

Номера битов смещения	Directory
22...31	Index in Page Directory
12...21	Index in Page Table
0...11	Index in Page

Взяв "Index in Page Directory", процессор обращается к так называемой "Page Directory" — каталогу страниц. Это область памяти с физическими адресами таблиц страниц. Физический адрес этого каталога находится в регистре CR3 (только старшие 20 битов CR3!), а число элементов нетрудно получить из количества битов, отводимых под индекс в каталоге — 10 битов дают 1024 элемента. Таким образом, сначала процессор извлекает элемент из каталога страниц:

*Элемент каталога = [CR3 + Index in Page Directory*4],*

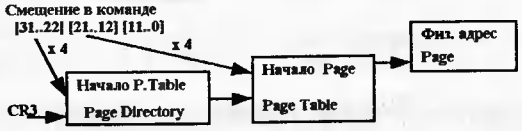
который является (не совсем весь, только старшие 20 битов) физическим адресом начала одной из таблиц страниц. Таблицы страниц (Page Table) являются, в свою очередь, набором физических адресов (опять только старшие 20 битов) начала самих страниц в памяти. Для выборки конкретного элемента из Page Table процессор использует адрес ее начала (Элемент каталога) и "Index in Page Table" из смещения команды:

*Адрес начала страницы = [Элемент каталога + Index in Page Table*4]*

Окончательный физический адрес элемента памяти вычисляется по адресу начала страницы и индексу элемента страницы из смещения в команде:

Физический адрес = Адрес начала страницы + Index in Page.

Схематично это можно представить таким образом:



Управление страничной адресацией

Теперь подробнее о том, как задается страничная адресация.

За включение страничной адресации ответственен бит PG (Paging Flag) — бит 31 регистра CR0 процессора. Если он установлен в логическую "1", то страничное преобразование разрешено.

Тип страничной адресации задается битами PAE (Physical Address Extention) и PSE (Page Size Extention, бит 4) — соответственно биты 5 и 4 регистра CR4 процессора, а также битом PS (Page Size) — бит 7 в выбранном элементе Page Directory.

Следует отметить, что регистр CR4 доступен только в процессорах Pentium, и в общем случае необходимо проверять тип процессора командой CPUID и только затем — биты в регистре CR4.

Если бит PAE=1, то разрешен 36-разрядный физический адрес, иначе — "обычный" 32-разрядный. Если бит PSE=0, то размер страницы — 4 Кб, иначе — может быть 2 или 4 Мб.

Экспериментально тип страничного преобразования можно проверить, например, так:

```
.586p ; Pentium Processor
PG equ 1 shl 31
PAE equ 1 shl 5
PSE equ 1 shl 4
; ...
   mov   eax,CR0
   test  eax,PG
   jz    @@NoPageRegim
   mov   eax,CR4
   test  eax,PAE
   jnz   @@PhysAddr_36bit
; ...
@@PhysAddr_36bit:
; ...
@@NoPageRegim:
```

Итак, если бит PG=1, а биты PSE=PAE=0, то у нас "обычное" страничное преобразование с 4-килобайтным размером страницы и 32-битным адресом.

Значения этих битов в Windows 98 показывают, что эта ОС использует как раз именно такой тип страничного преобразования.

Как получить физический адрес по линейному адресу

Решим небольшую задачку — напишем процедуру, которая будет возвращать win32-приложению по заданному линейному адресу физический адрес.

Код процедуры разместим в динамическом VxD, поскольку для его реализации необходимо использование привилегированных инструкций типа "mov eax,CR0", недопустимых в win32-коде. Подробнее о написании динамических VxD можно прочесть, например, в туториалах Iczelion'a, размещенных на сайте HI-TECH.

Условимся также пропустить проверки на тип страничного преобразования, считая, что Windows 9x использует 4-килобайтные страницы и 32-разрядный физический адрес.

Итак, начнем. Сначала — стандартное начало динамического VxD:

```
; Программа чтения физического адреса по линейному.
; Этот динамический VxD можно загружать через DeviceIOControl
; и получать по указателю физический адрес по линейному адресу.
; Coded by Chingachguk. 2002.
.386p
include vmm.inc
include vwin32.inc
DECLARE_VIRTUAL_DEVICE PHY,1,0, PHY_Control,\
     UNDEFINED_DEVICE_ID, UNDEFINED_INIT_ORDER
Begin_control_dispatch PHY
       Control_Dispatch w32_DeviceIoControl, OnDeviceIoControl
End_control_dispatch PHY
```

Определимся с передачей параметров и получением результатов нашего кода. Будем передавать VxD адрес следующей структуры:

```
; Структура параметров при вызове
CallParams struc
LinearAddr dd ? ; Линейный адрес
PhysAddr   dd ? ; Сюда вернуть физический адрес
CallParams ends
```

Определим в сегменте данных одну переменную — флаг ошибки:

```
; Сегмент данных нашего VxD
VxD_PAGEABLE_DATA_SEG
Result dd ?
; Если будет ошибка, мы будем хранить тут 0FFFFFFFFh (-1).
VxD_PAGEABLE_DATA_ENDS
```

Алгоритм вычисления физического адреса понятен. Остается решить один важный вопрос — как же обращаться к памяти, если у нас есть физический адрес, а мы находимся в режиме страничного преобразования? Например, мы получили физический адрес элемента в Page Directory (допустим, сейчас он в eax), но команда вида:

mov eax,[eax]

совсем не приведет к чтению элемента каталога страниц, поскольку процессор будет трактовать значение в eax согласно страничному преобразованию!

Разумным решением было бы вызвать сервис другого VxD, например, VMM, с целью получить физический адрес по линейному (в этом случае нам вообще делать будет нечего, даже не надо читать никаких Page Directory, Page Table...). Однако такого сервиса не существует (DDK)! С другой стороны, существует сервис получения линейного адреса по физическому у VMM. Необходимость его существования следует из необходимости некоторым драйверам адресоваться к конкретной физической памяти, например, при работе с BIOS, и т.д.:

Get linear address by physical address(DDK)

VMMCall _MapPhysToLinear,

Таким образом, нам придется вызывать данный сервис всякий раз, когда потребуется по физическому адресу извлечь из памяти значение. А вот наш сегмент кода:

```
VxD_PAGEABLE_CODE_SEG
BeginProc OnDeviceIoControl
  assume esi:ptr DIOCParams
.if [esi].dwIoControlCode==DIOC_Open ; Контрольное сообщение?!
  xor  eax,eax ; Надо отвечать: eax=0.
```

А вот это уже серьезно. Это вызов из win-приложения с конкретным заданием. Что это за задание — знаем только мы и тот, кто нас вызвал.

```
.elseif [esi].dwIoControlCode==1
  mov dword ptr Result,0FFFFFFFFh ; Установим флаг ошибки
  pushad ; На всякий случай сохраним все регистры, кроме сегментных
  pushfd ; Сохраним флаг направления. Видимо, это перестраховка.
  mov  edi,[esi].lpvInBuffer
  mov  edi,[edi] ; Указатель на буфер, который нам передал win32-код
```

Теперь получаем начальный физический адрес Page Directory:

```
  mov eax,CR3
  and eax,11111111111111111111000000000000b ;Выделить биты 31..12
```

Получаем индекс в Page Directory (биты 22..31):

```
  mov  ecx,[edi].LinearAddr
  shr  ecx,22
  shl  ecx,2
```

Получаем физический адрес элемента в Page Directory:

```
  add  eax,ecx
```

Получаем линейный адрес по физическому адресу от VMM.vxd. Физический адрес элемента сейчас в eax:

```
  call GetLinearAddr_Memory
  jz   @@PageDirectoryErr
```

Прочтем по линейному адресу, выданному нам системой, содержимое ячейки Page Directory:

```
mov  eax,[eax]
and  eax,11111111111111111111000000000000b
```

И теперь у нас eax=физический адрес Page Table. Получаем индекс в Page Table (биты 21...12):

```
mov  ecx,[edi].LinearAddr
shr  ecx,12
and  ecx,1111111111b
shl  ecx,2
```

Получаем физический адрес элемента в Page Table:

```
add  eax,ecx
```

Получаем линейный адрес по физическому адресу от VMM.vxd (аналогично Page Directory):

```
call GetLinearAddr_Memory
jz   @@PageDirectoryErr
```

Читаем элемент Page Table:

```
mov  eax,[eax]
and  eax,11111111111111111111000000000000b
```

Теперь eax=физический адрес Page. Получаем индекс в Page (биты 11...0):

```
mov  ecx,[edi].LinearAddr
and  ecx,111111111111b
```

И, наконец, получить физический адрес:

```
add  eax,ecx
```

Вернем его вызвавшей программе:

```
mov [edi].PhysAddr,eax
; Сбросим флаг ошибки - все прошло нормально.
mov  dword ptr Result,0h
```

Обычное окончание подпрограммы — восстановление регистров, возврат результата выполнения:

```
@@PageDirectoryErr:
  popfd ; Восстановим флаги направления и т.д. - перестраховка ?!
  popad
  mov  eax,dword ptr Result ; Вернем в eax флаг ошибки
.endif
  ret
EndProc OnDeviceIoControl
```

Ниже приведен текст служебной подпрограммы — для оформления вызова сервиса VMM. Она получает физический адрес в eax и в нем же возвращает линейный:

```
GetLinearAddr_Memory proc
; Input: eax=phys addr
; Result: eax=linear addr or ZF is set
; Get linear address by physical address(DDK)
; VMMCall _MapPhysToLinear,
  push 0h ; flags
  push 4h ; 4 bytes
  push eax ; PhysAddr
  int  20h      ; Call VxD
  dw   006Ch    ; 006Ch map physical address to linear address
  dw   0001h    ; ID VMM
  add  esp,3*4   ; C-call function
  cmp  eax,0FFFFFFFFh ; 0FFFFFFFFh if not addressable
; eax = address of first byte
; Returns the linear address of the first byte in the specified
; range of physical addresses. Uses EAX, ECX, EDX and Flags.
  ret
GetLinearAddr_Memory endp
VxD_PAGEABLE_CODE_ENDS
end
```

Для примера ниже приведен фрагмент вызова такого VxD из win32-кода. Та же структура для передачи параметров:

```
.data
CallParams struc
LinearAddr dd ?
PhysAddr dd ?
CallParams ends
```

Имя загружаемого VxD, которое передается CreateFile-у:

```
VxDName     db "\\.\PHY.VXD",0
```

Структура, которой передаются параметры коду VxD:

```
InBuffer    dd offset MyMem ; Указатель на буфер
                            ; для чтения параметров
```

Тут будет храниться хэндл открытого VxD:

```
.data?
hVxD        dd ?
MyMem CallParams <>
```

Начало исполняемого кода:

```
.code
start:
```

Загрузим динамический VxD через CreateFile:

```
invoke CreateFile,addr \ VxDName,0,0,0,0,FILE_FLAG_DELETE_ON_CLOSE,0
.if eax!=INVALID_HANDLE_VALUE ; VxD Успешно загружена ?
  mov hVxD,eax
```

Получить физический адрес какого-нибудь линейного, например, текущего указателя EIP:

```
  call @@GetOfs
@@GetOfs:
  pop eax
```

Передать наш линейный адрес VxD-коду:

```
  mov dword ptr MyMem.LinearAddr,eax
 invoke DeviceIoControl,hVxD,1,addr InBuffer,sizeof \
        InBuffer,NULL,NULL,NULL,NULL
```

Проверка на ошибку. Если ошибка, то eax = 0:

```
  test eax,eax
  jz @@ErrorReadPhys
```

Ошибки нет. Покажем физический адрес:

```
  call Print_PhysAddr
; ...
@@ErrorReadPhys:
```

Пример трансляции линейных адресов

Для примера в таблице приведены трансляции нескольких характерных линейных адресов в физические под Windows 98 на компьютере с 16 Мб памяти:

	Линейный адрес	Физический адрес
Сегмент кода win32-приложения (EIP)	0040103Bh	0072103Bh (~7 Мб)
Kernel32.dll, функция CreateFile	BFF77ADFh	00345ADFh (~3 Мб)
Стек win32—приложения (ESP)	0063FE3Ch	00635E3Ch (~6 Мб)
Сегмент кода, загруженного VxD	C188A4E6h	00E3C4E6h (~14 Мб)

Подведение итогов

Таким образом, единственный необходимый сервис ОС для получения физического адреса по линейному — это сервис "Получить линейный адрес по физическому". Очевидно, аналогичные сервисы существуют не только в Windows 98 (95), а и в Windows NT и ее наследниках — Windows 2000 и т.д, что позволит переносить приведенный выше код без особых изменений на эти платформы, используя соответствующие модели драйверов (*.wdm).

Знание действительного положения программ в памяти, на мой взгляд, не только любопытно, но и позволяет глубже понять стратегию размещения программ ОС, а также делать грубые оценки ее работы и эффективности. Например, оказывается, что Windows 98 использует наиболее простой тип страничного преобразования в случае размера памяти компьютера 16 Мб, в то время как технологии позволяют использовать еще несколько режимов с большими размерами страниц или же смешанным размером страниц. Было бы интересно оценить поведение этой ОС в случае существенного увеличения размера памяти, ведь в наше время не такая уж экзотика компьютер с размером памяти 128 Мб и более, а также используемые типы страничного преобразования в новых версиях Windows.

От редакции: *на сайте журнала (http://radio-mir.com) выложены исходные тексты программ определения физических адресов — phy.asm и readphys.asm*

Литература

1. Е.Бердышев. Технология MMX. Новые возможности процессоров P5 и P6. — Москва, "Диалог-МИФИ", 1998.

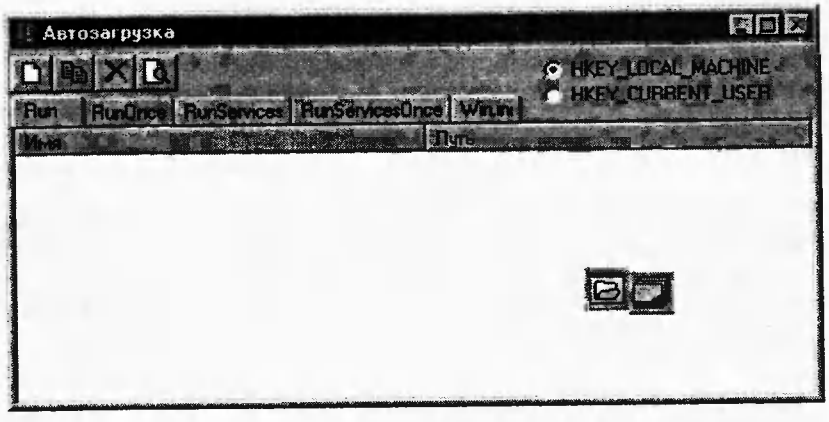
Сделай сам: Управление автозагрузкой

И.ШИРКО,
E-mail: FDC@tut.by

Продолжаем серию "Сделай сам", посвященную практическому применению Delphi. На этот раз мы поговорим о такой важной составляющей жизни нашего компьютера как автозагрузка. При загрузке Windows могут запускаться самые разные приложения — от системной панели (вы ее можете узнать по характерным для нее часам) до программы оптимизации реестра.

Все мы любим устанавливать новые программы. А некоторые из них, желая, чтобы мы с ними никогда не расставались, втихую прописывают себя в автозагрузку. Вот и получается, что со временем Windows загружается все дольше и дольше. А все потому, что с ней за компанию запускаются различные "удобняшки", антивирусы (а иногда и их анти-сородичи) и т.п.

Рис. 1



Вывод из всего вышесказанного — время от времени нужно проводить "зачистку" в автозагрузке. "Продвинутые" пользователи это делают при помощи редактора реестра и Блокнота, а все остальные используют специальные программы. Давайте посмотрим, что нам предлагает Windows для управления автозагрузкой. После недолгих поисков была найдена программа MsConfig (лежит она в каталоге ...\Windows\System). MsConfig позволяет просматривать список файлов, запускающихся вместе с Windows, и удалять элементы из автозагрузки с возможностью восстановления. Программа обрабатывает файл Win.ini, папку "Автозагрузка" и следующие ключи реестра:

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run,

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run и

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices.

Теперь, когда есть с чем сравнить, сделаем свою программу для управления автозагрузкой. Вот что наше творение должно будет уметь делать к концу статьи:

- все, что умеет MsConfig, за исключением восстановления удаленных из автозагрузки файлов;
- добавлять новые файлы в автозагрузку и изменять пути к существующим;
- обрабатывать большее количество ключей реестра, чем MsConfig.

Теперь немного теоретических сведений.

В файле Win.ini есть два параметра — Load и Run. Все файлы, имена которых записаны в этих параметрах через пробел, будут загружаться при старте Windows. А значит, в именах файлов не должно быть пробелов. Этого легко добиться при помощи функции sysutils.ExtractShortPathName (filename).

В реестре есть "автозагрузочные" ключи с одинаковыми названиями — Run и RunOnce. Одна пара находится в разделе HKEY_LOCAL_MACHINE, а другая — в HKEY_CURRENT_USER. В связи с этим, в нашей программе будем осуществлять переход между разделами при помощи двух компонентов TradioButton (**рис.1**).

Создайте в Delphi новый проект и поместите на форму PageControl1:TpageControl. Затем на этот компонент поместите ListView1:TlistView. После этого на форму поместите следующие компоненты — ToolBar1:TtoolBar, ImageList1:TimageList, OpenDialog1:TopenDialog, RadioButton1:TradioButton и RadioButton2:TradioButton.

Теперь выделите PageControl1 и нажмите правую клавишу мыши. Из появившегося меню выберите пункт *New Page*. После этого на компоненте должна появиться новая страница. Создайте, таким образом, еще четыре страницы и дайте каждой название (свойство Caption): 1 — Run, 2 — RunOnce, 3 — RunServices, 4 — RunServicesOnce, 5 — Win.ini. Каждая страница будет отвечать за определенный ключ реестра, либо за файл Win.ini.

Теперь выделите ListView1 и при помощи свойства Columns создайте две колонки с названиями (свойство Caption) Имя и Путь соответственно. Также для каждой колонки установите свойству Autosize значение True. После этого измените значения свойства ViewStyle на vsReport. Затем сделайте активным компонент ToolBar1 и создайте на нем четыре кнопки. Для каждой кнопки свойству ShowHint (показывать подсказку) установите значение true (истина), а в свойство Hint запишите эти самые подсказки, которые будут появляться при наведении указателя мыши на ту или иную кнопку: 1 — Добавить файл, 2 — Изменить путь, 3 — Удалить файл, 4 — Автозагрузка.

На кнопки, при помощи ImageList1, можно поместить иконки (рис.1). Чтобы сделать это, выполните следующие действия — кликните двойным щелчком по компоненту ImageList1 (должно появиться окно, как на **рис.2**), при помощи кнопки "Add..." добавьте в компонент нужные значки (каждый значок будет помещен под своим номером), выделите компонент ToolBar1, и в свойстве Images выберите компонент ImageList1. После этого нужно в свойстве ImageIndex

Рис. 2

каждой кнопки выбрать номер понравившегося значка. Теперь приведите вашу форму к виду, подобному форме, изображенной на рис.1.

На этом внешнее оформление программы завершено, а значит, пора заняться непосредственно программированием. Для начала в разделе Uses подключите два модуля — Registry (для работы с реестром) и IniFiles (для работы с Ini-файлами). После этого нужно объявить несколько глобальных переменных:

```
var
 ...
 ActIndex:integer; {индекс активной страницы}
 Appini:TiniFile; {для работы с Ini-файлами}
 key:cardinal; {имя текущего ключа реестра}
 path:string; {путь к файлу}
 reg:TRegistry; {для работы с реестром}
 i:integer; {используется в циклах}
 pathfile:String;
 Value, Load, Run:TStringList; {списки файлов автозагрузки}
 ListItem:TListItem; {для работы с компонентом ListView1}
```

Теперь запишите самую первую процедуру, которая будет из строки извлекать пути к файлам, разделенные пробелами, и помещать их в список. Позже эта процедура будет востребована:

```
procedure
 ExtractFilenames(s:String;FileNames:String;qw:TStrings);
Begin
{если найдены два пробела, то удаляем один из них}
 i:=pos(' ',filenames);
 while i<>0 do
  begin
   delete(filenames,i,1);
   i:=pos(' ',filenames);
  end;
 i:=pos(' ',filenames);
{добавляем в список qw все файлы из строки}
 while i<>0 do
  begin
   qw.Add(copy(Filenames,1,i-1));
   delete(filenames,1,i);
   i:=pos(' ',filenames);
  end;
 qw.Add(Filenames);
end;
```

Для обработки события OnShow компонента TabSheet1 (мы ее назвали "Run") запишите следующую процедуру:

```
procedure TForm1.TabSheet1Show(Sender: TObject);
begin
{очищаем ListView1}
 ListView1.Items.Clear;
{создаем объект для работы с реестром}
 reg:=Tregistry.Create;
{будем работать в ключе хранящемся в переменной Key}
 reg.RootKey:=Key;
{открываем ключ, в котором содержатся имена файлов загружаемых
при каждом старте Windows и записываем эти в ListBox1}
 reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',False);
 reg.GetValueNames(value);
 for i:=0 to value.Count-1 do
  with ListView1 do
   begin
    ListItem := Items.Add;
    ListItem.Caption :=Value.Strings[i];
    ListItem.SubItems.Add(reg.ReadString(Value.Strings[i]));
   end;
{освобождаем память, выделенную под объект reg}
 reg.Free;
end;
```

Запишем процедуру события OnShow компонента TabSheet2. Делаем все то же самое, только открываем другой ключ реестра:

```
...
reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunOnce', False);
...
```

Обработка события OnShow компонента TabSheet3 — то же, что и в предыдущих процедурах, но открываем ключ реестра, в котором хранятся имена сервисов, которые запускаются вместе с Windows:

```
...
reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServices', False);
...
```

Обработка события OnShow компонента TabSheet4 — открываем ключ реестра, в котором хранятся имена сервисов, они запустятся только один раз при ближайшей перезагрузке Windows:

```
...
reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServicesOnce', False);
...
```

Процедура обработки события OnShow компонента TabSheet5:

```
procedure TForm1.TabSheet5Show(Sender: TObject);
begin
 ListView1.Items.Clear;
 load.Clear;
 run.Clear;
{получаем доступ к файлу Win.ini}
 AppIni:=TIniFile.Create('Win.INI');
{читаем пути к файлам из строки Load}
 path:=appini.ReadString('windows', 'Load', ' ');
{удаляем все пробелы}
 while path[1]= ' ' do
  delete(path, 1, 1);
{записываем пути к файлам в список Load}
 if path<>' ' then
  ExtractFileNames('Load', path, Load);
{теперь все это повторяем со строкой "Run" и записываем их в список Run}
 path:=appini.ReadString('windows','Run', 'error');
 while pos(' ',path)=1 do
  delete(path,1,1);
 if path<>' ' then
  ExtractFileNames('Run',path,Run);
 AppIni.Free;
{отображаем пути ко всем файлам из списков Run и Load в ListView1}
 for i:=0 to Load.Count-1 do
  with ListView1 do
   begin
    ListItem := Items.Add;
    ListItem.Caption :='Load';
    ListItem.SubItems.Add(Load.Strings[i]);
   end;
 for i:=0 to Run.Count-1 do
  with ListView1 do
   begin
    ListItem := Items.Add;
    ListItem.Caption :='Run';
    ListItem.SubItems.Add(Run.Strings[i]);
   end;
end;
```

Обработка события OnClick компонента RadioButton1:

```
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
{если текущий ключ не HKEY_LOCAL_MACHINE, то делаем его
активным и, если нужно, обновляем информацию в ListBox1}
if Key<>windows.HKEY_LOCAL_MACHINE then
  begin
    key:=Windows.HKEY_LOCAL_MACHINE;
    if PageControl1.ActivePageIndex<2 then
    PageControl1.ActivePage.OnShow(self);
```

```
  end;
end;
```

Обработка события OnClick компонента RadioButton1:

```
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
{если текущий ключ не HKEY_CURRENT_USER, то делаем его
активным; и, если нужно, обновляем информацию в ListBox1}
if Key<>windows.HKEY_CURRENT_USER then
  begin
    key:=Windows.HKEY_CURRENT_USER;
    if PageControl1.ActivePageIndex<2 then
     PageControl1.ActivePage.OnShow(self);
  end;
end;
```

После этого создайте новую форму и поместите на нее кнопку Tbutton и два компонента TradioButton (рис.2). Запишите обработку события нажатия на кнопку:

```
procedure TForm2.Button1Click(Sender: TObject);
var
 FName:string;
Begin
{преобразуем имя выбранного файла в краткий формат}
 FName:=sysutils.ExtractShortPathName(form1.opendialog1.filename);
{записываем в выбранную константу имя файла после пробела, также
отображаем добавленный файл в окне нашей программы}
 if form2.RadioButton1.Checked then
  path:='Load'
 else path:='Run';
 AppIni := TIniFile.Create('Win.INI');
 appini.writeString('windows', path, appini.readstring('windows',
 path, ' ')+' '+FName);
 AppIni.Free;
 with form1.ListView1 do
  begin
   ListItem:=Items.Add;
   ListItem.Caption:=path;
   ListItem.SubItems.Add(FName);
  end;
 form2.hide;
 form1.show;
end;
```

Теперь запишите процедуру обработки нажатия кнопки New, при помощи которой можно добавить новый файл в автозагрузку:

```
procedure TForm1.ToolButton1Click(Sender: TObject);
begin
{если выбран файл, который нужно добавить, то продолжаем}
 if OpenDialog1.Execute then
{если активна страница с ключами реестра, то добавляем файл в
реестр, иначе - в Win.ini}
  if PageControl1.ActivePageIndex<4 then begin
   reg:=Tregistry.Create;
   if PageControl1.ActivePageIndex<2 then
    reg.RootKey:=key else
   reg.RootKey:=windows.HKEY_LOCAL_MACHINE;
   case PageControl1.ActivePageIndex of
   0:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',False);
   1:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunOnce',False);
   2:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServices',False);
   3:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServicesOnce',
                 False);
   end;
{просим ввести имя константы для нового файла}
   path:=inputbox('Введите имя константы', ' ', Extractfilename
                  OpenDialog1.filename));
{если имя не введено, то вместо него записываем имя
добавляемого файла}
   if path=' ' then path:=Extractfilename(OpenDialog1.filename);
   pathfile:=ExtractShortPathName(OpenDialog1.filename);
   reg.WriteString(path,pathfile);
   reg.CloseKey;
   reg.Free;
   with ListView1 do
    begin
     ListItem := Items.Add;
     ListItem.Caption :=path;
     ListItem.SubItems.Add(pathfile);
    end;
  end else
{если нужно добавить файл в Win.ini, то вызываем форму 2}
   form2.show;
end;
```

Нажатие кнопки "Изменить":

```
procedure TForm1.ToolButton2Click(Sender: TObject);
var
 Temp:string;
 Results:string;
Begin
{если выбран новый файл для константы, то продолжаем}
 if ListView1.ItemFocused.Selected then
  if opendialog1.Execute then
   begin
{получаем индекс активной константы}
    ActIndex:=ListView1.ItemFocused.Index;
{преобразовываем имя выбранного файла в короткий формат}
    results:=ExtractShortPathName(OpenDialog1.filename);
{теперь записываем в константу новый файл}
    if PageControl1.ActivePageIndex<4 then
     begin
      reg:=Tregistry.Create;
      if PageControl1.ActivePageIndex<2 then
      reg.RootKey:=key else
      reg.RootKey:=windows.HKEY_LOCAL_MACHINE;
      case PageControl1.ActivePageIndex of
      0:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',False);
      1:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunOnce',False);
       2:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServices',False);
      3:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServicesOnce',False);
      end;
      reg.WriteString(listview1.Items.Item[ActIndex].Caption,results);
      reg.CloseKey;
      reg.Free;
     end else begin
      AppIni:=TIniFile.Create('Win.INI');
     path:=appini.readstring('windows',listview1.Items.Item[ActIndex].Caption,'error');
      temp:=ListView1.Items.Item[actindex].SubItems.Strings[0];
      i:=pos(temp,path);
      if i<>0 then begin
       delete(path, i, length(temp));
       path:=path+' '+results;
       appini.writeString('windows',listview1.Items.Item[ActIndex].Caption, path);
       AppIni.Free;
      end;
     end;
   ListView1.Items.Item[actindex].SubItems.Strings[0]:=Results;
 end;
end;
```

Нажатие кнопки "Удалить":

```
procedure TForm1.ToolButton3Click(Sender: TObject);
var
 temp:string;
begin
{если пользователь уверен в своем ремении, то удаляем
выбранный файл из автозагрузки}
 if ListView1.ItemFocused.Selected then
  if messagedlg('Вы уверены?',mtConfirmation,[mbYes,mbNo],0)=mryes then
   begin
    ActIndex :=ListView1.ItemFocused.Index;
    if PageControl1.ActivePageIndex<4 then
     begin
      reg:=Tregistry.Create;
      if PageControl1.ActivePageIndex<2 then
       reg.RootKey:=key else
```

```
      reg.RootKey:=windows.HKEY_LOCAL_MACHINE;
     case PageControll.ActivePageIndex of
     0:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',False);
    1:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunOnce',False);
    2:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServices',False);
    3:reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\RunServicesOnce',False);
     end;
     reg.DeleteValue(listview1.Items.Item[ActIndex].Caption);
     reg.CloseKey;
     reg.Free;
    end else
    begin
     AppIni:=TIniFile.Create('Win.INI');
   path:=appini.readstring('windows',listview1.Items.Item[ActIndex].Caption,'error');
     temp:=ListView1.Items.Item[actindex].SubItems.Strings[0];
     i:=pos(temp,path);
      if i<>0 then
       begin
        delete(path,i,length(temp));
          appini.writeString('windows',listview1.Items.Item[ActIndex].Caption, path);
        AppIni.Free;
       end;
    end;
   ListView1.Items.Item[actindex].Delete;
  end;
end;
```

Нажатие на кнопку "Автозагрузка":

```
procedure TForm1.ToolButton4Click(Sender: TObject);
var
 Auto:string;
begin
 reg:=Tregistry.Create(windows.key_Read);
 reg.RootKey:=HKEY_USERS;
reg.OpenKey('.DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\
          Shell Folders', False);
{читаем из реестра путь к папке "Автозагрузка" и открываем ее}
 Auto:='Explorer '+reg.ReadString('Startup');
 reg.Free;
 winexec(pchar(Auto),SW_ShowNormal);
end;
```

Теперь добавим в программу пару штрихов.

Выделите компонент ListView1, и для события OnDblClick (при двойном щелчке мышью) выберите процедуру ToolButton2Click, а для события OnKeyDown (при нажатии клавиши) запишите следующее:

```
procedure TForm1.ListView1KeyDown(Sender: TObject; var
Key: Word;
  Shift: TShiftState);
begin
if key=VK_DELETE then form1.ToolButton3.OnClick(self);
end;
```

В итоге, при нажатии кнопки Delete, выбранный файл будет удален, а при двойном щелчке мыши будет предложено заменить этот файл другим.

Вот и все. Предложения по поводу тем следующих статей этого цикла присылайте мне на E-mail.

Утилита бинарного сравнения файлов

С.РЮМИК,
г.Чернигов,
E-mail: 1972r@mail.ru

Слово "утилита" происходит от латинского "utilitas" — польза, выгода. Действительно, программисты часто пишут небольшие, но весьма полезные программы и традиционно называют их утилитами.

Своим появлением утилиты обязаны развитию операционных систем (ОС). В частности, в середине 70-х годов Г.Килдалом была разработана ОС CP/M для 8-битных компьютеров. В ее состав входили три программных уровня — операционное ядро, внешние (транзитные) программы-утилиты, а также сопутствующие прикладные программы текстовых и графических редакторов. Программы-утилиты загружались с дисковода и позволяли выполнять такие важные процедуры как форматирование дискеты, копирование файлов, ассемблирование и отладку программ в машинных кодах.

В середине 80-х годов, в эпоху расцвета ОС MS-DOS, парк утилит значительно расширился. Их наборы стали включаться прямо в комплект поставки ОС, а затем стали выпускаться в виде приложений к разнообразным файловым оболочкам.

Многие из утилит рассчитаны на работу с файлами. В качестве примера рассмотрим практическую задачу — как определить идентичность двух файлов на бинарном уровне, по принципу "байт в байт"? С подобной проблемой сталкиваются обладатели программаторов, занимающиеся прошивкой микросхем ПЗУ, контроллеров, ПЛИС. Длина и название файлов прошивок могут быть одинаковыми, а внутреннее содержание — отличаться в одном или нескольких байтах, что принципиально.

Другая категория пользователей — это геймеры, работающие с эмуляторами игровых компьютеров и приставок. Бинарные прошивки образов ПЗУ картриджей обычно располагаются на разных Интернет-сайтах, часто имеют одинаковую длину, но разные названия, хотя по внутренней структуре они абсолютно одинаковы.

Файловые оболочки, подобные Norton Commander, DOS Navigator, Windows Commander, позволяют сравнивать внешние атрибуты файлов на уровне содержимого каталогов, не опускаясь до их внутренней структуры. А жаль, ведь эта функция всегда должна быть под рукой.

В MS-DOS задачей сравнения содержимого файлов занимается небольшая утилита "fc.exe". Поискать ее следует в директории "\DOS" или "\WINDOWS\COMMAND" IBM-совместимого компьютера. Утилита позволяет сравнивать файлы построчно и побайтно, используя различные ключи [1].

Предположим, требуется сравнить файлы "100.bin" и "step.hex" на бинарном уровне, а результат сравнения поместить в файл-протокол "diff.rpt". Для этого следует предварительно скопировать сравниваемые файлы в один каталог и в командной строке ввести — **fc 100.bin step.hex /b > diff.rpt**.

Кажется, все просто, но дело осложняется при сравнении файлов большой длины. Например, образы ПЗУ картриджей приставки "Sega Mega Drive-II" имеют длину до 4 Мб. При сравнении двух таких разнородных ПЗУ

объем файла-протокола ошибок достигает 52(!) Мб, и на это требуется 200 с (компьютер AMD-K6-II-500). Если отказаться от файла-протокола, то вся информация об ошибках будет выводиться на экран монитора со значительным увеличением общего времени работы...

Разрешить проблему поможет использование собственной программы-утилиты:

```
/*  УТИЛИТА compare, БИНАРНОЕ СРАВНЕНИЕ ДВУХ ФАЙЛОВ  */
# include <stdio.h>          /* Версия 1.0, 30.09.01 */
# include <string.h>  /* Рюмик С.М., Журнал РАДИОМИР */
# include <stdlib.h>    /* TURBO C-2.0, память Small */
# define BEG 32768       /* Задержка "бегущих точек" */

main (int argc,  char **argv)     /* Командная строка */
{ FILE *in1, *in2;       /* Указатели на файлы чтения */
  int ch1, ch2;        /* Символы, вводимые из файлов */
  static char n[20], p[20];           /* Имена файлов */
  unsigned long count = 0; /* Счетчик символов файла */
  unsigned long error = 0;  /* Число разных символов */
  unsigned int len = 0; /* Неодинаковая длина файлов */
  if (argc != 3)         /* Проверка числа аргументов */
  { printf ("Синтаксическая ошибка!\nОбразец ");
    puts ("команды: compare <file1.*> <file2.*>");
    exit (1);         /* Аварийный выход из программы */
  }
  strcpy (n, argv[1]); strcpy (p, argv[2]); /* Копии */
  if ((in1 = fopen (argv[1], "rb")) == NULL)
  { printf ("Нельзя открыть файл %s!", n);
    exit (1); /* Выход, если не читается первый файл */
  }
  if ((in2 = fopen (argv[2], "rb")) == NULL)
  { printf ("Нельзя открыть файл %s!", p);
    exit (1); /* Выход, если не читается второй файл */
  }
  while ((ch1 = getc(in1)) != EOF) /* Чтение символа */
  { if (count++ %BEG == BEG-1)
       printf (".");     /* "Бегущая строка" из точек */
    if ((ch2 = getc(in2)) == EOF) break;      /* Стоп */
    if (ch1 != ch2) error++;         /* Разные символы */
  }
  if (ch2 == EOF) /* Признак окончания второго файла */
    { printf ("\nФайл <%s> длиннее, чем <%s>.",n, p);
      len++;                    /* Разная длина файлов */
    }
  else            /* Признак окончания первого файла */
  ( if ((ch2 = getc(in2)) != EOF)
    { printf ("\nФайл <%s> длиннее, чем <%s>.",p, n);
      len++;                    /* Разная длина файлов */
    }
    count++;  /* Прибавление к счетчику одного байта */
  }
  fclose (in1); fclose (in2);      /* Закрытие файлов */
  printf ("\nФайлы <%s> и <%s> идентичны!",n,p);
  if (len > 0 || error > 0)       /* Добавление фразы */
  {  printf ("\b в %lu байтах.\nЧисло ",count-error-1);
     printf ("различий в %lu байтах.\n", error);
  }
}                      /* Завершение работы программы */
```

После компиляции программы в среде TURBO C v2.0, должен получиться исполняемый файл "compare.exe". Его, а также сравниваемые файлы следует поместить в один общий каталог и в командной строке ввести — **compare <имя первого файла с расширением> <имя второго файла с расширением>**.

Для удобства в программе имеется таймер в виде бегущей строки из точек, что полезно при сравнении файлов большой длины.

При использовании описываемой программы время сравнения уменьшилось в 40 раз, вместо файла-протокола выводится информация об общем количестве совпадающих и несовпадающих байтов, а также указание о разной или одинаковой длине файлов.

Кстати, расширение у файлов может быть любым, будь-то *bin, *.doc или *.txt. В случае полного совпадения всех байтов при одинаковой длине сравниваемых файлов выводится сообщение: "Файлы <...> и <...> идентичны!"

От редакции: *на сайте журнала (http://radio-mir.com) выложены листинг программы (файл compare.c) и исполняемый модуль (compare.exe).*

Литература

1. Фигурнов В.Э. IBM PC для пользователя. — М.: Финансы и статистика, 1991.

Хранитель экрана

М.ВАЛЬПА,
E-mail: vmax@74.ru

Как известно, продолжительность "жизни" монитора напрямую зависит от яркости и времени свечения экрана. Поэтому во время простоя компьютера рекомендуется запускать специальную программу "хранитель экрана" (screensaver), которая выводит на монитор затемненную, меняющуюся картинку, экономя таким образом ресурсы люминофора и распределяя излучение равномерно по всему экрану. О том, как написать такую программу самому, я и хочу рассказать.

Моя программа называется "scrsave" и написана на языке программирования C++. Эта программа после запуска переключает монитор в графический режим работы и отображает на экране картину звездного неба. Картина постоянно меняется, т.к. все время зажигаются новые цветные звездочки и гаснут старые. При нажатии на любую клавишу клавиатуры картина изменяется. Так возникает эффект движения по звездному небу, но это уже дополнительный и необязательный эффект. Для выхода из программы нужно нажать клавишу Esc.

Листинг программы "scrsave"

```
//////////////////////////////////////////////////////////
// Программа хранителя экрана
//
// Файл:scrsave.cpp
// Дата:01.04.2002
// Автор: Вальпа Максим
//////////////////////////////////////////////////////////
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
```

```
#define PIX     100          // Количество точек на экране
#define DMS      50          // Задержка в миллисекундах
#define ESC     0x1b         // Код клавиши ESC

int main(void)
{
int gdriver = DETECT, gmode, errorcode;
                             // Переменные инициализации графики
int i, j, x, y, c, maxx, maxy, maxc;  // Прочие переменные
int nx[PIX], ny[PIX];        // Массивы координат точек
char v;                      // Переменная опроса клавиатуры

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

// Чтение результата инициализации
errorcode = graphresult();

// Обработка ошибок
if (errorcode != grOk)
   {
   printf("Ошибка инициализации графики: %s\n",
            grapherrormsg(errorcode));
   printf("Нажмите любую клавишу...");
   getch();
   exit(1); // Завершить работу с кодом ошибки
   }

// Запустить генератор случайных чисел
randomize();
// Получить максимальное значение координаты X
maxx = getmaxx();
// Получить максимальное значение координаты Y
maxy = getmaxy();

maxc = getmaxcolor()+1;    // Получить ЧЕРНЫЙ ЦВЕТ

met:
 // Вычислить массив значений точек и отобразить на экране
   for(j=0;j<PIX;j++)
      {
      nx[j] = random(maxx);
      ny[j] = random(maxy);
      c     = random(maxc);
      putpixel(nx[j], ny[j], c);
      }

   i=0;              // Обнулить переменную цикла
   while (!kbhit()) // Делать пока не нажата любая клавиша
   {
      putpixel(nx[i], ny[i], maxc); // Погасить звездочку

      nx[i]=random(maxx);           // Вычислить координаты
      ny[i]=random(maxy);           // новой звездочки
      c = random(maxc);             // Цвет новой звездочки
      putpixel(nx[i], ny[i], c); // Зажечь новую звездочку
      delay(DMS);                   // Задержка

      i++;                          // Новое значение
      if(i==PIX) i=0;               // Зациклить
   }

   cleardevice();                   // Очистить экран

   v=getch();                    // Прочесть нажатую клавишу
   if(v != ESC) goto met;     // Если не 'ESC' - продолжить

   closegraph();                // Иначе - выключить графику
   return 0;                     // и выход
}
```

В этой программе все команды прокомментированы, поэтому разобраться в ней несложно. Хочу только добавить, что ее можно развить. Например, можно периодически плавно увеличивать, а затем гасить некоторые звездочки. Можно добавить и другие эффекты. Для запуска программы необходимо, чтобы в каталоге, где хранится этот файл, находился еще и стандартный файл драйвера графики egavga.bgi.

Из избранного

Мысли, пришедшие в голову однажды ночью (ближе к утру).

Что такое полное одиночество?
Это когда с Новым Годом тебя не поздравляют даже спаммеры.

Можно ли загадать желание, если сидишь между двумя программистами?
Можно! Только глючить будет.

Программист отличается от обычного смертного тем, что он в состоянии ответить на вопрос, в котором уже заключен ответ.
Например. Сколько будет 2x2=4? TRUE!

В программировании трудно найти правильную единицу времени для измерения прогресса. Некоторые соборы строились веками. Можно ли вообразить грандиозность и размер программы, на которую затратили столько времени?

Длина популярной программы прямо пропорциональна квадрату ее версии, умноженной на десять.

Каждая домохозяйка может программировать... А может не программировать... Результат не изменится!

Каждый человек сам выбирает свою судьбу из той единственной, что ему досталась.

Переустановка Windows — как разморозка холодильника. Помогает, но ненадолго...

Проще написать неправильную программу, чем понять правильную.

Нажмите кнопку "Start".
В меню "Find" выберите "Files or Folders...".
В маске файлов напишите — *.exe.
Ищите файлы по всему компьютеру.
В списке, где появятся найденные файлы, выберите все (Ctrl+A).
Нажмите Enter и наслаждайтесь... :))).

Рисунок О.Попова

Еще раз о приятных мелочах

А.ЧЕХУНОВ,
г.Белгород.

Думаю, ни у кого не вызывает сомнений, что наиболее распространенная сейчас операционная система — Windows 98. Безусловно, сегодня лучший выбор — это Windows Millennium, но тут есть две проблемы. Первая (большая) — отсутствие драйверов под некоторое "железо", вторая (маленькая) — отсутствие "чистого" DOS. Windows 2000, по моему мнению, крайне громоздкая, все еще глючная и недоработанная операционка (это мое личное впечатление). К сожалению. Windows 98 намного медленнее и Windows 95, и Windows Millennium, а также достаточно ненадежна. Но когда приходится решать разноплановые задачи при ограниченных ресурсах, то от нее никуда не денешься.

Из своего опыта знаю, что комфортность работы с тем или иным программным обеспечением зависит не только от возможностей продукта "по существу", но и от удобного и легко перенастраиваемого интерфейса. Поскольку эта статья предназначена для так называемых "чайников" (коим в своей жизни был каждый user), то речь пойдет о вещах элементарных (но полезных).

Вначале я хочу рассмотреть настройку системы с помощью файла MSDOS.SYS (c:\ MSDOS.SYS). Для внесения поправок в этот файл измените его атрибуты так, чтобы он не имел атрибута "только для чтения", и откройте его в Блокноте.

Помню, как меня раздражала анимированная картинка, появляющаяся при загрузке Windows 98. Решается эта проблема просто — изменением параметра "Logo=1" на "Logo=0". Тем самым вы блокируете отображение экранной заставки (если в файле MSDOS.SYS у вас нет какого-либо из описанных здесь параметров — это равносильно тому, что они включены, т. е. "...=1"). Также вы можете поменять эту картинку на любую другую, например, на фотографию любимой девушки.

Делается это так. В Paint открывается любой рисунок в формате *.bmp, изменяется его размер — он должен быть 320x400, палитра — 256 цветов. Страшно, конечно, но... Далее он переименовывается в "logo.sys" и помещается в корневой каталог. После этого перезагружаетесь и наслаждаетесь полученным эффектом. То же самое можно сделать с сообщениями "Теперь питание компьютера..." и "Подождите, идет..." (соответственно файлы "logos.sys" и "logow.sys"), но находятся они в папке Windows. Формат у них такой же, как и у "logo.sys".

Windows 98 по умолчанию разрешает использование функциональных клавиш при начальной загрузке. Т.е. при нажатии "F5" Windows загружается в безопасном режиме, "F6" — в режиме защиты от сбоев с поддержкой сети. Для отключения функциональных клавиш необходимо изменить параметр "BootKeys=" на "BootKeys=0". Соответственно, для их использования — "BootKeys=1".

По умолчанию Windows 98 не показывает меню начальной загрузки, но если вы им пользуетесь, то нужно значение параметра "BootMenu=" изменить на "BootMenu=1" для его отображения, и на "BootMenu=0" для блокировки.

Если у вас включено меню начальной загрузки, то можно управлять временем ожидания Windows дальнейших команд. Это устанавливается параметром "BootMenuDelay=x", где x — время задержки в секундах.

Следующие два параметра, скорее всего, никому не пригодятся, но описать их необходимо — для "общего развития".

"BootMulti=1" — загружается предыдущая операционная система, если она у вас была, "BootMulti=0" — блокируется этот режим.

"BootGUI=1" — автоматически загружается рабочий стол Windows (по умолчанию), "BootGUI=0" — отображается только командная строка MS-DOS.

Достаточно часто приходится работать на чужих, либо "общественных" компьютерах. В таких случаях, чтобы не тратить каждый раз 10...15 мин на настройку Windows "под себя", проще один раз написать собственный файл записей реестра, в котором отразить все необходимые настройки — разрешение монитора, скорость "выпадения" менюшек, перемещения курсора, цветовые настройки и т. п. Для этого в текстовом редакторе создается файл с расширением *.reg, который в общем виде выглядит следующим образом:

```
REGEDIT4
[раздел реестра]
"параметр"="строковое значение"
```

Если параметр имеет тип dword, то последняя строка принимает вид:

```
"параметр"=dword:00000000
```

Вместо "00000000" ставится необходимое значение. В конце файла должна быть пустая строка. Для полного удаления ключа из реестра необходимо перед его именем в reg-файле поставить "-", например:

```
REGEDIT4
[-HKEY_CURRENT_USER\Software\Microsoft\Wmdows\
CurrentVersion\ Policies\System]
```

Еще один совет. Перед запуском созданного файла на выполнение (если это ваш первый опыт) создайте, например, с помощью Norton Utilities, копию реестра. В том случае, если что-то пойдет не так, не нужно будет копаться в реестре вручную и "гасить" все, что вы там натворили. Желательно, правда, чтобы между сохранением и экспортом реестра вы не устанавливали и не деинсталлировали программы, не изменяли конфигурацию системы — иначе обязательно возникнут проблемы. В принципе, я бы всем рекомендовал почаще использовать возможности реестра в плане настройки Windows — жизнь станет приятней и проще. Поэтому далее будут рассмотрены несколько популярных исправлений, вносимых в реестр.

Первое, что мне захотелось сделать, когда я задумался о внешнем виде своего рабочего стола — это

избавиться от ленточек (стрелочек) на ярлыках. Честно говоря, в таком случае становится невозможным отличить EXE-файл от его ярлыка, и по ошибке он частенько "летит" в корзину. Но, как известно, красота требует жертв. Для этого открываем реестр и в разделе HKEY_CLASSES_ROOT в папках PIFFILE и LNKFILE удаляем параметр IsShortcut. Перезагружаемся.

Здесь же можно сказать, что переименовав *.bmp-файл в *.ico- или в *.cur-, вы можете получить иконку и изменять вид курсора (соответственно). Но при этом размер файла не меняется, и возможно обратное преобразование.

Далеко не всем нравятся стандартные системные иконки и названия: "Корзина", "Мой компьютер" и т.д. Для преодоления этой проблемы войдите в соответствующий раздел реестра (наименования конкретных подразделов приведены ниже) и смените значение соответствующего параметра "По умолчанию" на желаемое название иконки, параметра "InfoTip" — на соответствующую фразу. Чтобы сменить сам значок, необходимо войти в подраздел DefaultIcon и изменить значения параметра (в котором присутствует название либо путь к библиотеке системных иконок Shell32.dll) на путь к вашему файлу *.ico.

Раздел реестра — HKEY_CLASSES_ROOT\CLSID.

Подразделы для различных системных пиктограмм:

{20D04FE0-3AEA-1069-A2D8-08002B30309D} — "Мой компьютер";

{645FF040-5081-101B-9F08-00AA002F954E} — "Корзина";

{FBF23B42-E3F0-101B-8488-00AA003E56F8} —"Проводник";

{208D2C60-3AEA-1069-A2D7-08002B30309D} — "Сетевое окружение";

{00028B00-0000-0000-C000-000000000046} — "The Microsoft Network";

{00020D75-0000-0000-C000-000000000046} — "Входящие".

Иногда, при удалении, после удаления программы она остается, тогда необходимо в разделе реестра HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall найти папку программы и удалить ее.

Очень неудобно, создавая ярлык для какого-либо файла, избавляться потом от приставки "Ярлык для...". Удаляется это следующим образом — в разделе реестра HKEY_USERS\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer изменяется значение параметра "Link" на "00 00 00 00". После этого необходимо перезагрузиться.

Далее я хотел бы привести список наиболее популярных опций, позволяющих запретить некоторые функции в Windows. Они могут быть полезны системным администраторам, а также всем тем, кто хочет ограничить возможности посторонних лиц, использующих ваш компьютер (за счет установки параметров типа DWORD:. Значение "1" включает ограничение, "0" — снимает). Большинство описанных параметров необходимо будет создать, например:

"DisableRegistryTools"=dword:1.

Ключ — HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System

Опции:

- "DisableRegistryTools" — не позволяет пользователю запустить Regedit.exe или Regedt32.exe для изменения системного реестра;
- "NoConfigPage" — скрывает вкладку "Профили оборудования" в окне свойств системы;
- "NoDevMgrPage" — скрывает вкладку "Устройства" в окне свойств;
- "NoFileSysPage" — скрывает кнопку "Файловая система..." на вкладке "Быстродействие" в окне свойств системы;
- "NoVirtMemPage" — скрывает кнопку "Виртуальная память..." на вкладке "Быстродействие" в окне свойств системы;
- "NoPwdPage" — скрывает вкладку "Смена паролей";
- "NoDispCPL" — отключает доступ к значку "Экран" в Панели управления и не позволяет пользователям менять параметры дисплея;
- "NoDispAppearancePage" — скрывает вкладку "Оформление" в окне свойств экрана;
- "NoDispBackgroundPage" — скрывает вкладку "Фон" в окне свойств экрана;
- "NoDispScrSavPage" — скрывает вкладку "Заставка" в окне свойств экрана;
- "NoDispSettingsPage" — скрывает вкладку "Настройка" в окне свойств экрана.

Ключ — HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\ExplorerОпции:

-"NoInternetIcon" — скрывает значок Интернет на Рабочем столе Windows;

- "NoNetHood" — скрывает значок "Сетевое окружение" на Рабочем столе Windows;
- "NoRun" — скрывает команду "Выполнить" в меню "Пуск";
- "NoFind" — скрывает команду "Найти" в меню "Пуск";
- "NoCommonGroups" — скрывает группу "Стандартные" в меню "Пуск — Программы";
- "NoFavoritesMenu" — скрывает группу "Избранное" в меню "Пуск";
- "NoRecentDocsMenu" — скрывает группу "Документы" в меню "Пуск";
- "NoSetFolders" — скрывает пункты "Панель управления" и "Принтеры" в меню "Пуск — Настройка";
- "NoSetTaskbar" — скрывает пункт "Панель задач" в меню "Пуск — Настройка";
- "NoClose" — отключает команду "Выключить компьютер";
- "NoSaveSettings" — отключает сохранение изменений параметров настройки настольной конфигурации при выходе из Windows;
- "NoDesktop" — скрывает все элементы и программы на Рабочем столе Windows.

Хотелось бы уточнить, что хотя реестры Windows 95, Windows 98 и Windows Millennium во многом похожи (фактически, все это развитие Windows 95), но все вышесказанное в полной мере относится только к Windows 95 и Windows 98 (не Second Edition).

От редакции: *настоятельно рекомендуем перед внесением каких-либо изменений в реестр делать копию реестра, чтобы иметь возможность для "отката".*

Возвращаясь к напечатанному

(РМ. ВК, №5/2002, С.38)

Хочу откликнуться на статью А.Горячкина **"Девять причин не ставить Windows Millenium"**. Судя по ней, автор, на мой взгляд, недостаточно хорошо знаком с этой ОС.

С момента выпуска фирмой Microsoft своего очередного продукта — Windows Millenium Edition — прошло не так уж много времени, как пишет А.Горячкин. На самом деле, по меркам жизни компьютерного железа и программного обеспечения прошло целое поколение. Многие пользователи, чтобы стать "крутыми" и "идти в ногу со временем" установили на свои машины Windows XP. Автор указал девять "глюков", с которыми он столкнулся при работе Windows ME. На самом деле их может быть гораздо больше. Все зависит от конкретного пользователя, его машины, задач, решаемых на этом компьютере, и др. Кстати, занимаемое место после установки ME на чистый винчестер в общем у меня составило около 600 Мб. Но можно добиться и уменьшения до примерно 200 Мб. Впрочем, сейчас не об этом. Я хочу хотя бы частично опровергнуть доводы А.Горячкина. Итак, начну по порядку.

1. "Пожирающая" свободное место папка "_RESTORE".

Как пишет автор, можно ограничить размер папки восстановления 200 Мб, но ведь можно и вообще отключить восстановление системы. Для этого нужно открыть закладку *Пуск/Настройка/Панель управления/Система/Быстродействие/Файловая система/Устранение неполадок* и поставить галочку на опции восстановления системы. Затем перезагрузиться, и папка "_RESTORE" уже не растет.

2. Да, привычный "Лазерный" проигрыватель исчез. И отменить установку этого монстра под названием "Проигрыватель Windows Media" нельзя. Но воспроизводить музыкальные и видеофайлы можно на старом проигрывателе, который находится по пути *C:\Program files\Windows Media Player\mplayer2.exe*. Нужно только отменить загрузку файлов в громоздком "Проигрывателе Windows Media" по умолчанию. Это можно сделать так — *Открыть программу*, затем *Сервис./Параметры.../Форматы* и снять галочки с форматов файлов, которые нужно запускать на альтернативных проигрывателях.

Единственный формат, который я не смог запустить на mplayer2.exe — это Audio CD, не считая таких экзотических форматов как *.ra.

3. Размер файла подкачки в системе Windows ME можно изменять по своему желанию, по необходимости или не использовать вовсе.

4. Насчет "звонилок". При работе со всеми ОС, начиная с Windows 95 и кончая Windows XP, лично я не заметил особой разницы, используя и внутренний PCI-модем, и внешний. Но, по моему субъективному мнению, связь в Windows XP лучше и быстрее. Конечно, все возникающие сложности зависят от причин, которых до неприличия много.

5. Теперь насчет режима эмуляции MS-DOS. Обновить прошивку BIOS'а материнской платы, во-первых, можно из-под Windows. А во-вторых, разве сделать это с загрузочной дискеты религия не позволяет:)? И еще, существуют патчи для загрузки эмуляции MS-DOS.

6. Программа msconfig (входящая в состав Windows Me) позволяет настроить все конфигурационные файлы под свои нужды.

7. Для нормальной работы звуковой карты, как и для другого оборудования, нужны нормальные драйвера.

8. И проблемы с Word я не обнаружил, хотя открывал документы разной длины и насыщенности разнообразными вставными объектами. Я использовал Office 2000 с настройками по умолчанию. Но, возможно, у меня просто достаточно быстрый компьютер. Конечно, все относительно.

9. В своих архивах я специально нашел программу ACDSee 32 v2.4 (сейчас уже вышла v4.0) и, установив ее, не смог добиться того, чтобы на моей машине возникла такая же проблема, как у автора статьи.

И.КОСАРЕВ,
E-mail: igor@hotbox.kaluga.net

Взрыв CD-ROM'а

С. ШИРКО,
E-mail: S_Shirko@tut.by

До чего все-таки докатился прогресс. Если первые приводы CD-ROM "скромно" читали данные со скоростью 172 Кб/с (односкоростные), то сейчас некоторые особо "продвинутые" представители вполне могут похвастаться скоростью 50х и более. При этом количество оборотов при вращении диска достигает 10000 в минуту. Глядя на эту цифру, уже не удивляешься, почему во время работы привод CD-ROM ревет как реактивный самолет на взлете. Но повышенные шумы — это еще полбеды. Основное "веселье" начинается, когда в высокоскоростной привод попадает низкокачественный пиратский диск. Вот тогда уже действительно может случиться беда, если не сказать больше.

Как известно из курса физики и из опыта повседневной жизни, на любое вращающееся тело действует так называемая центробежная сила. При этом модуль этой силы пропорционален квадрату угловой скорости вращения тела (диска в нашем случае). Подсчитано, что для 50-скоростных приводов эта сила достигает 30...33 Н, что эквивалентно ситуации, когда на диск подвешен груз массой 3 кг. Конечно же, это не говорит о том, что если вы возьмете дома соответствующую гирю и подвесите ее к диску, с последним что-нибудь случится. Тут дело в том, что из-за неизбежных отклонений геометрических размеров диска относительно осевой линии симметрии, к статической центробежной силе добавляется еще и периодическая вибрация. И здесь достаточно даже небольшой трещинки на диске, чтобы вызвать его разрушение, что называется, на полном ходу. Последствия этого процесса приводят как к потере самого носителя информации, так и, в большинстве случаев, к повреждению внутренних элементов CD-привода. Вполне вероятно, что и сам пользователь ПК может стать мишенью для разлетающихся осколков диска. И что самое обидное — фирма-производитель дисковода CD-ROM не несет никакой ответственности в случае разрушения диска внутри CD-привода и снимает с себя всякие гарантийные обязательства в подобных ситуациях.

"А справедливость где?" — спросите вы. С ней-то как раз все нормально. Стремление производителей приводов к высоким скоростям вполне объяснимо — чем больше оборотов, тем быстрее происходит чтение информации. Нетрудно догадаться, что перед выпуском в массовое производство высокоскоростные приводы тестируются, и тестируются на качественных дисках, которые вполне выдерживают прила-

гаемые к ним нагрузки. А тот факт, что в нашей стране большинство CD-продукции — пиратского производства, вовсе нельзя ставить в вину изготовителям приводов CD-ROM.

Таким образом, пользователь сам должен обезопасить себя от подобных неприятностей. Сделать это можно, руководствуясь следующими правилами:

1. Не нужно стремиться к высоким оборотам при выборе CD-привода. Абсолютное большинство пиратских дисков никогда не смогут читаться со скоростью выше 24...26x. В то же время, многие модели дисководов "любят" сначала раскрутить CD до максимальной скорости, а затем плавно уменьшать ее до тех пор, пока данные не начнут более или менее устойчиво считываться. И чем выше эта максимальная скорость, тем хуже и для некачественного диска, и для привода.

2. Будьте внимательны при покупке CD-ROM-дисков. Недопустимы трещины, сколы, заметные отклонения геометрии "компакта". Кроме того, в процессе эксплуатации периодически осматривайте поверхность диска на наличие повреждений. Распространен случай, когда CD очень плотно закрепляется в футляре, и при неосторожном извлечении на его внутреннем ободке появляются трещины и сколы. Впоследствии они разрастаются, и в один "непрекрасный" день такой диск разнесет вам половину системного блока (стучу по дереву).

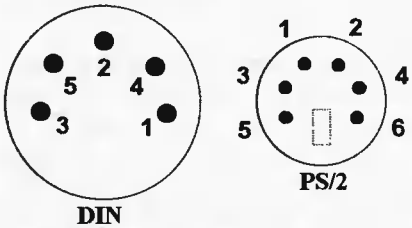
3. В случае необходимости пользуйтесь специальными программами-замедлителями скорости вращения приводов CD-ROM. Первой такую программу предложила фирма ASUSTeK, поместившая на свой сайт соответствующее решение. Еще одно программное средство под названием Cd-Bremse находится по адресу www.cd-bremse.de. У отечественных пользователей ПК популярна небольшая утилита CDSlow от российского автора Вадима Дружина — http://vdruzhin.chat.ru. Стоит отметить, что различные модели имеют разные возможности в плане выбора скорости вращения. Особо выделяются приводы от ASUSTeK, которые позволяют выбирать любую скорость с шагом, равным 1. С другой стороны, некоторые приводы вообще отказываются поддерживать программное изменение скорости вращения (например, мой LG).

На этом можно завершить это небольшое повествование. Несмотря на то что разрывы дисков в приводах еще не приняли массового характера, забывать о такой проблеме не стоит. Единичные случаи этой досадной неприятности все равно случаются, поэтому будьте бдительны при покупке как приводов, так и дисков CD-ROM.

ПЕРЕХОДНИК ДЛЯ КЛАВИАТУРЫ

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

Тем, кто в ближайшее время собирается сделать апгрейд своего компьютера и перейти с платформы AT на платформу ATX, необходимо учесть, что клавиатура с 5-контактным разъемом DIN не подойдет к разъему клавиатуры материнской платы форм-фактора ATX, так как там используется 6-контактный малогабаритный разъем PS/2. Поэтому, скорее всего, придется покупать новую клавиатуру с разъемом PS/2.



Но есть и другое решение этого вопроса. Можно использовать прежнюю клавиатуру, но для этого необходим соответствующий переходник. Такие переходники имеются в продаже под названием "Переходник для клавиатуры DIN—PS/2". Пользователи, умеющие держать паяльник в руках, могут изготовить подобный переходник своими руками. Для этого потребуется:

- 5-контактный DIN разъем "мама";
- 6-контактный PS/2 разъем "папа";
- 4-жильный экранированный провод длиной 10...15 см с внешней изоляцией.

Предвижу закономерный вопрос — где взять разъемы? Разъемы DIN широко применяются в звуковоспроизводящей аппаратуре, а PS/2 можно использовать от неисправных клавиатур или поискать на радиорынках. Данный переходник можно также использовать и в качестве удлинителя клавиатуры, применив более длинный провод. Оплетку (экран) провода необходимо заземлить на корпус системного блока для предотвращения помех. На рисунке приведена нумерация контактов на разъемах DIN и PS/2 (показан сзади вид, со стороны пайки). В таблице приведено соответствие контактов разъемов DIN и PS/2. Аналогичным методом можно изготовить переходник для клавиатуры с интерфейсом PS/2, используя для этого соответствующие разъемы.

DIN	Сигнал	PS/2
1	Синхросигнал	5
2	Данные	1
3	Не задействован	2
4	Корпус	3
5	+ 5 В	4
–	Не задействоаан	6

Screen Saver как защита "от дурака"

Н.АНИСИМЕНКО,
г. Ярославль.

Предохранение экрана от выгорания (это главная функция любой экранной заставки) можно совместить с охраной машины от не в меру любопытных коллег — защитить заставку паролем. Правда, чтобы в Windows 95/98 добраться до вашего жесткого диска, злоумышленнику достаточно будет перезагрузиться, но он, по крайней мере, не сможет войти в сеть под вашим именем. К тому же, часто нужна защита, что называется, "от дурака" — кто-то шел мимо вашего стола, сел за компьютер на минутку... Заставки, защищенной паролем, будет достаточно, чтобы "случайный прохожий" шел дальше мимо вашего компьютера.

В Windows NT/2000/XP пароль необходим и для загрузки. В этом случае заставка с паролем обеспечит достаточно надежную защиту вашей системы.

Чтобы установить пароль, кликните правой кнопкой мыши по рабочему столу, выберите пункт меню "Свойства" (Properties). Затем перейдите на закладку "Заставка" (Screen Saver), выберите любую заставку, установите флажок "Защита паролем" (Password protected), перейдите на закладку "Смена пароля" (Change Password), введите желаемую комбинацию символов, подтвердите пароль и выйдите с сохранением изменений. Несмотря на очевидность такой защиты, ею редко кто пользуется. А зря.

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

(Продолжение. Начало в №8/2002)

Гамма-коррекция

Градации яркости в исходном изображении обозначены числами от 0 до 255. Эти числа мы, собственно, и считали яркостью пикселов (имея в виду определение яркости, данное в Приложении 1). То есть мы считали, что яркость пиксела линейно зависит от числа, соответствующего этому пикселу в изображении, т.е если, например, одному пикселу соответствует число 10, а другому — 20, то второй пиксел в два раза ярче первого.

Однако в общем случае это не так. Зависимость здесь не линейная, а степенная (показатель степени называют "гамма"). Линейная зависимость — лишь частный случай, когда гамма равна 1. На **рис.17а...в** приведены графики для трех возможных случаев (по оси x откладывается значение, соответствующее пикселу в изображении, а по оси y — фактическая яркость пиксела).

Рис. 17а

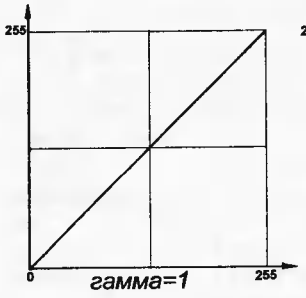


Рис. 17б

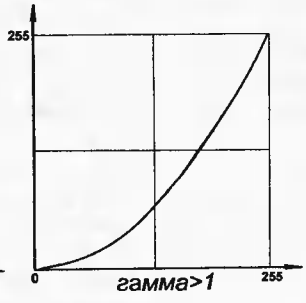
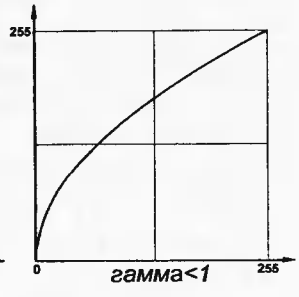


Рис. 17в



Как видим, если гамма больше 1, то в изображении с большей точностью представлена информация о темных областях, и с меньшей точностью — о светлых, а если гамма меньше 1 — наоборот.

Чтобы правильно выводить изображения с гаммой, отличной от единицы, и нужна гамма-коррекция. Из значения пиксела (обозначим его x) мы должны получить его яркость (обозначим ее y). Если и значение, и яркость — числа от 0 до 255, то формула такова:

$$y=255\cdot(x/255)^{Gamma}$$

Удобно сначала вычислить y для всех x от 0 до 255, помещая значения в таблицу, а при выводе просто брать из таблицы уже готовое значение.

Приведенные в Приложении 2 процедуры вывода могут выполнять гамма-коррекцию, если установить в 1 флаг условной компиляции GAMMA_CORR. При этом значением гаммы по умолчанию считается 1,5. Для другого значения надо будет пересчитать таблицу коррекции с помощью программы на Бейсике, приведенной там же. Кстати, эту программу можно запустить и просто так, чтобы посмотреть, какие графики получаются при разных значениях гаммы.

А как узнать, какое значение гаммы у выводимого изображения? Если вы берете это изображение из файла в формате png (или в каком-либо другом формате, где значение гаммы указывается в самом файле), то сложностей не возникает. А если вы (как и я) будете брать изображение из pcx-файла, где таких сведений не содержится? Тогда придется подбирать этот коэффициент опытным путем. Попробуйте сначала вывести изображение без гамма-коррекции. Если оно выглядит естественно, то больше ничего и не надо. Если оно слишком темное — значит, гамма больше 1; если слишком светлое — гамма меньше 1.

Кстати, изображения с гаммой больше 1 будут выводиться более точно за счет того, что они содержат больше информации о темных областях, а распределение спектрумовских градаций яркости (см. рис.19 в Приложении 1) как раз таково (по крайней мере, у меня), что темные области воспроизводятся гораздо точнее светлых.

Работа с pcx-файлами

Исходные изображения надо откуда-то брать. Я считывал их из файлов в формате pcx. В Приложении 3 приведены две процедуры для работы с такими файлами. Первая процедура читает данные из pcx-файла и формирует в памяти массив данных о пикселах. Вторая процедура выполняет обратную задачу — записывает pcx-файл по находящимся в памяти данным о пикселах (именно с ее помощью я подготовил все pcx-файлы с рисунками для этой статьи).

Еще одна приведенная там процедура непосредственно с pcx-файлами не работает, но окажется полезной, если вам нужно преобразовать изображение в спектрумовском формате (6912 байтов) в черно-белый pcx-файл. Она формирует в памяти массив данных о пикселах такого изображения по заданным значениям градаций яркости, а на основе этих данных уже можно создать pcx-файл.

И еще одно небольшое дополнение

Если исходное изображение — цветное, и вы хотите вывести его в черно-белом виде, то для получения яркости пиксела по известным значениям компонентов R, G, B воспользуйтесь следующей формулой:

$$Y = 0{,}299R + 0{,}587G + 0{,}114B.$$

Так как вычислять яркость нужно для каждого пиксела, а даже изображение размером со спектрумовский экран — 256x192 — содержит 49152 пиксела, то понятно, что эти вычисления должны быть как можно более быстрыми. Удобно применить табличный способ.

Пусть имеется таблица длиной 768 (т.е. 3x256) байтов, начинающаяся с адреса, кратного 256 (обозначим этот адрес TAB_RGB). Пусть в первых 256 байтах этой таблицы содержатся значения функции y=0,299x для x от 0 до 255, округленные до целых (для практического применения такая точность вполне достаточна), во вторых 256 байтах — значения функции y=0,587x, а в третьих 256 байтах — значения функции y=0,114x (также для значений x от 0 до 255).

Вот 32-байтная процедура построения такой таблицы:

```
MK_T_RGB    LD   BC,TAB_RGB+#2FF ;Адрес
                 ;последнего байта таблицы.
            LD   H,29            ;0,114*256
            CALL MK_T_RGB_1

            LD   H,150           ;0,587*256
            CALL MK_T_RGB_1

            LD   H,77            ;0,299*256

MK_T_RGB_1  XOR  A
            LD   L,A
            LD   D,A
            LD   E,H
```

```
MK_T_RGB_2  SBC  HL,DE
            RET  C

;Запись округленного результата:

            LD   A,H
            BIT  7,L
            JR   Z,MK_T_RGB_3
            INC  A
MK_T_RGB_3  LD   (BC),A

            DEC  BC
            JR   MK_T_RGB_2
```

С помощью этой таблицы можно очень быстро вычислить яркость пиксела — например, используя следующий фрагмент программы:

```
;Вход:  значения компонентов R,G,B -
;в регистрах L,D,E соответственно.
;Выход: вычисленное значение Y - в аккумуляторе.
;Длина: 9 байтов.
;Время выполнения: 44 такта.

        LD   H,TAB_RGB/256
        LD   A,(HL)       ;A=0,299R
        INC  H
        LD   L,D
        ADD  A,(HL)       ;A=0,299R+0,587G
        INC  H
        LD   L,E
        ADD  A,(HL) ;A=0,299R+0,587G+0,114B
```

Если надо обработать сразу группу пикселов, выгоднее будет не загружать каждый раз регистр H, а загрузить его только один раз, а затем только изменять командами INC и DEC (обрабатывая по два пиксела в цикле), как в нижеприведенной процедуре:

```
;Вход: IX - адрес начала исходных данных
;      (R1,G1,B1,R2,G2,B2...);
;      DE - куда помещать результат (Y1,Y2...);
;      BC - количество обрабатываемых
;      пикселов (должно быть четным),
;      деленное на 2.
;Длина: 49 байтов.

CALC_Y   EXX
         LD   DE,6
         EXX
         LD   H,TAB_RGB/256

CALC_Y_1 LD   L,(IX)      ;L=R1
         LD   A,(HL)      ;A=0,299R1

         INC  H
         LD   L,(IX+1)    ;L=G1
         ADD  A,(HL)      ;A=0,299R1+0,587G1

         INC  H
         LD   L,(IX+2)    ;L=B1
         ADD  A,(HL) ;A=0,299R1+0,587G1+0,114B1

         LD   (DE),A
         INC  DE

         LD   L,(IX+5)    ;L=B2
         LD   A,(HL)      ;A=0,114B2

         DEC  H
         LD   L,(IX+4)    ;L=G2
         ADD  A,(HL)     ;A=0,114B2+0,587G2

         DEC  H
         LD   L,(IX+3)    ;L=R2
         ADD  A,(HL) ;A=0,114B2+0,587G2+0,299R2

         LD   (DE),A
         INC  DE

         EXX
         ADD  IX,DE       ;IX:=IX+6
         EXX

         DEC  BC
         LD   A,B
         OR   C
         JR   NZ,CALC_Y_1

         RET
```

Если заранее известно, что DE на входе будет четным (или нечетным), то первую (или, соответственно, вторую) команду INC DE можно заменить на более быструю INC E.

Если обрабатывается не более 512 пикселов, то в качестве счетчика можно использовать не регистровую пару BC, а только один регистр B, используя для организации цикла команду DJNZ.

Приложения

Приложение 1. Методика определения яркости цветов

Как определить величины a0...a7 (яркость цветов 0...7 при bright=0) и b0...b7 (яркость тех же цветов при bright=1), когда изображение выводится в черно-белом виде?

Очевидно, что эти величины могут различаться для разных моделей "ZX-Spectrum", для разных преобразователей цветного сигнала в черно-белый, а также для разных мониторов и телевизоров. Поэтому, вместо того чтобы привести готовый набор чисел, я расскажу о том, как вы сможете определить их именно для своей системы "компьютер—монитор", причем для этого вам не понадобятся какие-либо специальные приборы.

Что можно сказать сразу?

1. Самый темный цвет — черный:

$$a_0 = 0,\ b_0 = 0.$$

2. Самый яркий цвет — белый при bright=1:

$$b_7 = 255.$$

3. Яркость всех остальных цветов принимает промежуточные значения:

$$0 < (a_1...a_7,\ b_1...b_6) < 255.$$

4. С увеличением номера цвета яркость возрастает:

$$a_1 > a_0 \qquad b_1 > b_0$$
$$a_2 > a_1 \qquad b_2 > b_1$$
$$\ldots\ldots\ldots\ldots \qquad \ldots\ldots\ldots\ldots$$
$$a_7 > a_6 \qquad b_7 > b_6.$$

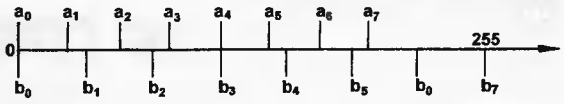
5. Для каждого цвета, кроме черного, яркость при bright=1 больше, чем при bright=0

$$b_1 > a_1$$
$$b_2 > a_2$$
$$\ldots\ldots\ldots$$
$$b_7 > a_7.$$

Итак, кое-что мы уже знаем. Основываясь на этой информации, можно предположить, что распределение яркостей будет примерно таким, как показано на **рис.18**.

Мы ранее уже употребляли термин "яркость", а каков физический смысл этого понятия? Определим его так: яркость некоторого цвета (точнее, яркость градации, соответствующей этому цвету при черно-белом режиме отображения) — это величина, пропорциональная количеству фотонов, испускаемых единичной областью данного цвета за единицу времени. Например, если яркость одного цвета вдвое больше яркости другого, это значит, что некоторая область этого цвета испускает за единицу времени вдвое больше фотонов, чем такая же область другого цвета.

Рис. 18



Если вывести в одной половине экрана какой-либо цвет (далее — исходный цвет), а в другой половине — смесь пикселов двух цветов, темнее и светлее исходного (далее, соответственно — темный цвет и светлый цвет), то, подбирая соотношение числа пикселов этих двух цветов, можно добиться равенства яркостей обеих половин экрана. Равенство яркостей означает, что обе половины экрана за единицу времени испускают одинаковое количество фотонов. Тогда справедлива следующая формула:

$$x = k \cdot y + (1-k) \cdot z, \qquad (1)$$

где:

- x — яркость исходного цвета;
- y — яркость темного цвета;
- z — яркость светлого цвета;
- k — доля пикселов темного цвета;
- $1-k$ — доля пикселов светлого цвета.

Например, если яркости уравнялись при заполнении второй половины экрана текстурой 4x4, в которой 3 пиксела темного цвета и 13 пикселов светлого цвета, это будет записано так:

$$x = 3/16 \cdot y + 13/16 \cdot z.$$

Используем изложенное выше для вычисления искомых яркостей a0...a7 и b0...b7. Сначала выразим яркость каждого из цветов b1...b6 через яркость соседних с ним цветов:

$$b_1 = k_1 \cdot b_0 + (1-k_1) \cdot b_2 \qquad (2)$$
$$b_2 = k_2 \cdot b_1 + (1-k_2) \cdot b_3$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$b_6 = k_6 \cdot b_5 + (1-k_6) \cdot b_7.$$

Из (2) следует:

$$b_2 = \frac{b_1 - k_1 \cdot b_0}{1 - k_1}.$$

И далее:

$$b_3 = \frac{b_2 - k_2 \cdot b_1}{1 - k_2}$$
$$b_4 = \frac{b_3 - k_3 \cdot b_2}{1 - k_3}$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$b_7 = \frac{b_6 - k_6 \cdot b_5}{1 - k_6}$$

Таким образом, зная b_0 и b_1, а также $k_1...k_6$ (найденные опытным путем), можно последовательно вычислить $b_2...b_7$.

Значение b_0 нам известно — это 0. Значение b_1 — неизвестно, зато мы знаем, что b_7=255. Можно выразить b_7 через b_1, затем определить b_1 и далее вычислить $b_2...b_6$. Но этот путь связан с громоздкими преобразованиями.

Поступим проще. Заметим, что $b_2...b_7$ линейно зависят от b_1. В этом нетрудно убедиться из приведенных выше формул (не забывая, что b_0=0). Примем b_1=1 и вычислим $b_2...b_7$. Тогда достаточно определить, во сколько раз получившееся значение b_7 оказалось меньше 255, и затем умножить каждое из вычисленных значений $b_1...b_7$ на эту величину, чтобы получить их истинные значения.

После того как яркости цветов с bright=1 известны, остается определить яркости цветов с bright=0 ($a_1...a_7$). Для этого достаточно опытным путем найти для каждого из этих цаетов эквивалентную по яркости смесь двух цветов с bright=1, после чего воспользоваться формулой (1).

Это была теория, а теперь перейдем к практике. Рассмотрим, как я определял уровни яркости на своем компьютере и что для этого использовал.

Сначала я написал процедуру, которая заполняет верхнюю часть экрана одним цаетом, а нижнюю — смесью пикселов двух других цветов (с помощью текстуры 4x4). Вот она:

```
        LD      H,1  ;исходный цвет
        LD      D,0  ;темный цвет
        LD      E,2  ;светлый цвет
        LD      L,4  ;номер текстуры
                     ;(0...16)

;Заполняем верхнюю половину экрана:

        LD      A,H
        ADD     A,A
        ADD     A,A
        ADD     A,A
        ADD     A,H

;Если исходный цвет - с повышенной
;яркостью (bright=1), то
;следует команда ADD A,#40,
;иначе она не нужна (можно заменить
;на ADD A,0):

        ADD     A,#40

        EXX
        LD      HL,#5800
        LD      DE,#5801
        LD      BC,#180
        LD      (HL),A
        LDIR
        EXX

;Заполняем нижнюю половину экрана:

        LD      A,D
        ADD     A,A
        ADD     A,A
        ADD     A,A
        ADD     A,E
        ADD     A,#40
        EXX
        LD      BC,#17F
        LD      (HL),A
        LDIR
        EXX

;Заполнение текстурой:

        LD      H,0
        ADD     HL,HL
        ADD     HL,HL;*4
        LD      DE,TEXTURES
        ADD     HL,DE
        EX      DE,HL

;DE указывает на первый байт текстуры.

        LD      B,24
        LD      HL,#4880

M2      PUSH    DE
        PUSH    BC
        LD      B,4
M1      LD      A,(DE)
        PUSH    BC
        PUSH    DE
        PUSH    HL
        LD      D,H
        LD      E,L
        LD      (HL),A
        INC     DE
        LD      BC,#1F
        LDIR
        POP     HL
        POP     DE
        POP     BC
        CALL    DOWN_HL
        INC     DE
        DJNZ    M1
        POP     BC
        POP     DE
        DJNZ    M2

        RET

DOWN_HL INC     H
        LD      A,H
        AND     7
        RET     NZ
        LD      A,L
        ADD     A,#20
        LD      L,A
        RET     C
        LD      A,H
        SUB     8
        LD      H,A
        RET

TEXTURES DB     %00000000
        DB      %00000000
        DB      %00000000
        DB      %00000000

        DB      %00000000
        DB      %01000100
        DB      %00000000
        DB      %00000000

        DB      %00000000
        DB      %01000100
        DB      %00000000
        DB      %00010001

        DB      %00000000
        DB      %01010101
        DB      %00000000
        DB      %00010001

        DB      %00000000
        DB      %01010101
        DB      %00000000
        DB      %01010101

        DB      %00000000
        DB      %01010101
        DB      %00100010
        DB      %01010101

        DB      %10001000
        DB      %01010101
        DB      %00100010
        DB      %01010101

        DB      %10101010
        DB      %01010101
        DB      %00100010
        DB      %01010101

        DB      %10101010
        DB      %01010101
        DB      %10101010
        DB      %01010101

        DB      %10101010
        DB      %01110111
        DB      %10101010
        DB      %01010101

        DB      %10101010
        DB      %01110111
        DB      %10101010
        DB      %11011101

        DB      %10101010
        DB      %01110111
        DB      %10101010
        DB      %11111111

        DB      %10101010
        DB      %11111111
        DB      %10101010
        DB      %11111111

        DB      %10101010
        DB      %11111111
        DB      %11101110
        DB      %11111111

        DB      %10111011
        DB      %11111111
        DB      %11101110
        DB      %11111111

        DB      %11111111
        DB      %11111111
        DB      %11101110
        DB      %11111111

        DB      %11111111
        DB      %11111111
        DB      %11111111
        DB      %11111111
```

Чтобы было удобнее сравнивать яркости половин экрана, когда на одной половине яркость равномерная, а на другой — текстура, я рассматривал изображение через полупрозрачную полиэтиленовую пленку (для этого годится обычный пакет).

С помощью вышеприведенной процедуры были получены следующие соотношения:

$$b_1 = 9/16\ b_0 + 7/16\ b_2$$
$$b_2 = 10/16\ b_1 + 6/16\ b_3$$
$$b_3 = 8/16\ b_2 + 8/16\ b_4$$
$$b_4 = 13/16\ b_3 + 3/16\ b_5$$
$$b_5 = 9/16\ b_4 + 7/16\ b_6$$
$$b_6 = 10/16\ b_5 + 6/16\ b_7$$

(т.е. k_1=9/16, k_2=10/16, k_3=8/16, k_4=13/16, k_5=9/16, k_6=10/16)

$a_1 = 3/16\ b_0 + 13/16\ b_1$
$a_2 = 9/16\ b_1 + 7/16\ b_2$
$a_3 = 9/16\ b_2 + 7/16\ b_3$
$a_4 = b_3$
$a_5 = 9/16\ b_4 + 7/16\ b_5$
$a_6 = 11/16\ b_5 + 5/16\ b_6$
$a_7 = 11/16\ b_6 + 5/16\ b_7$

Затем я написал программу, на основе этих данных вычисляющую искомые значения $a_0 \ldots a_7$ и $b_0 \ldots b_7$:

```
 10 DIM k(6): DIM a(7): DIM b(7)
 20 LET k(1)=9/16: LET k(2)=10/16:
    LET k(3)=8/16:
    LET k(4)=13/16: LET k(5)=9/16:
    LET k(6)=10/16
 30 REM CALCULATE b(1) - b(7)
 40 LET b(1)=1
 50 LET b(2)=1/(1-k(1))
 60 FOR i=3 TO 7
 70 LET b(i)=(b(i-1)-k(i-1)*b(i-2))/
       (1-k(i-1))
 80 NEXT i
 90 LET n=255/b(7)
100 FOR i=1 TO 7
110 LET b(i)=b(i)*n
120 NEXT i
130 REM CALCULATE a(1) - a(7)
140 LET a(1)=13/16*b(1)
150 LET a(2)=9/16*b(1)+7/16*b(2)
160 LET a(3)=9/16*b(2)+7/16*b(3)
170 LET a(4)=b(3)
180 LET a(5)=9/16*b(4)+7/16*b(5)
190 LET a(6)=11/16*b(5)+5/16*b(6)
200 LET a(7)=11/16*b(6)+5/16*b(7)
210 REM PRINT RESULT
220 PRINT "a0=0",0
230 FOR i=1 TO 7
240 PRINT "a";i;"=";a(i), INT (a(i)*256+0.5)
250 NEXT i
260 PRINT
270 PRINT "b0=0",0
280 FOR i=1 TO 7
290 PRINT "b";i;"=";b(i), INT (b(i)*256+0.5)
300 NEXT i
```

Ниже приведены результаты работы программы. В левом столбце — вычисленные значения яркостей, в правом — они же, но в 256-х долях (в таком формате они хранятся в процедурах вывода изображения, см. Приложение 2):

$a_0 = 0$	0
$a_1 = 4{,}344111$	1112
$a_2 = 8{,}3540597$	2139
$a_3 = 17{,}233232$	4412
$a_4 = 23{,}677792$	6062
$a_5 = 56{,}855343$	14555
$a_6 = 104{,}72922$	26811
$a_7 = 181{,}85936$	46556
$b_0 = 0$	0
$b_1 = 5{,}3465982$	1369
$b_2 = 12{,}220796$	3129
$b_3 = 23{,}677792$	6062
$b_4 = 35{,}134788$	8995
$b_5 = 84{,}781772$	21704
$b_6 = 148{,}61361$	38045
$b_7 = 255$	65280

На **рис.19** показано, как это выглядит на графике.

Как видите, это отличается от первоначальных предположений (рис.18). Значения располагаются неравномерно — чем меньше яркость, тем они ближе друг к другу. Чем это можно объяснить? Очевидно, при преобразовании цветного изображения в черно-белое

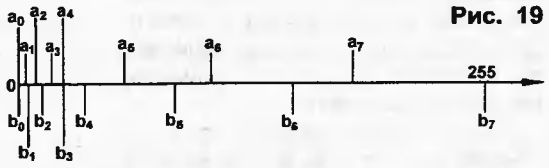


Рис. 19

стремятся к тому, чтобы различные цвета превращались в хорошо отличающиеся друг от друга градации серого. А для этого яркости соседних градаций должны отличаться друг от друга не на одну и ту же величину, а в одно и то же количество раз — таковы особенности человеческого зрения.

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.

(Окончание следует)

Какие бывают ZX (2)

Е.ЗАРЕЦКИЙ,
г.Чаусы, Могилевской обл.

COMPACT-128

Разработчик — фирма T.V.S., г.Санкт-Петербург. Год появления — начало 1993. Compact-128 — малоизвестная модификация "ZX-Spectrum". Это практичная Spectrum-совместимая машина, вобравшая в себя лучшие достоинства таких схем как "Pentagon" и "Composit". В то же время, она лишена некоторых существенных недостатков последних. За счет использования некоторых оригинальных схемотехнических решений разработчикам удалось создать легкую в сборке и надежную в эксплуатации машину.

"Compact-128" имеет:

- только одно напряжение питания — +5 В;
- контроллер НГМД с цифровой ФАПЧ и турбирование, что значительно повышает надежность чтения дискет и позволяет использовать более дешевые дисководы (MC 5305);
- интерфейс принтера Centronix;
- интерфейсы Kempston- и Sinclair-джойстиков;
- привязку к уровню черного;
- усиленные выходы RGB и VIDEO;
- встроенный усилитель мощности для внутреннего динамика;
- буферизированную клавиатуру.

Плата "Compact" имеет сравнительно малые размеры (280 x 130 мм), позволяющие устанавливать ее практически в любой корпус, предназначенный для сборки компьютера с дисководом. Микросхема контроллера дисководов КР1818ВГ93 питается напряжением +12 В, которое вырабатывается из напряжения +5 В преобразователем, собранным на этой же плате. Это обеспечивает ее надежную защиту от выхода из строя при пропадании одного из питающих напряжений. Хочется отметить, что "Compact" имеет высокую степень программной совместимости с фирменными компьютерами.

KAY-1024

Разработчик — (c) NEMO, г. Санкт-Петербург.

В KAY-1024 сигнал INT имеет произвольную длительность, зависящую от типа команд, выполняемых CPU на момент запроса прерывания. Дополнительным плюсом этой модели является отсутствие необходимости переключать длительность INT'а при переходе из TURBO в режим NORMAL и обратно.

На плате установлена розетка для упрощения стыковки по SCART и RGB. Меняя коммутацию линий в заглушке, можно подобрать необходимый набор сигналов как для SCART, так и для RGB (штатный режим). На розетку выведены все сигналы, необходимые для PAL/SECAM-кодера либо модулятора, включая питающее напряжение.

KAY-1024 снабжен 1 Мб ОЗУ. Это сделано для поддержки "увесистых" программных продуктов. В первую очередь это касается ОС IS-DOS. Дисковод из системного устройства промежуточного хранения данных превращается во внешнее устройство загрузки/выгрузки на внешний мобильный носитель — дискету.

Биты расширения: бит D7 — #1FFD; бит D6 — #7FFD. Линия D7 (7 бит) порта #1FFD, которая ранее обслуживала линию AUTO, переключена на 6 бит (D6) порта #7FFD. Использовать 6 бит (D6) порта #7FFD нельзя.

Внесенные изменения не затрагивают совместимость "сверху вниз", т.е. все старые периферийные устройства будут работать в новых компах с усовершенствованной шиной. Однако вноаь появляющиеся устройства могут отказаться работать на старых компах. Сигналы IORQ и IORQGE являются сигналами для географической адресации портов/карт. Если вектор адреса (не обязательно порта) опознается периферийной платой как свой, то дешифратор карты расширения включает схему, которая подтягивает соответствующий контакт IORQGE к уровню логической единицы. Импульс -IORQ дальше не проходит, и периферийная плата осуществляет обмен данными с процессором. Таким образом, исключаются конфликты между платами расширения.

Также под шину Nemo-bus разработаны контроллер дисководов BETA-Turbo, периферийная аудио-плата GENERAL SOUND, интерфейс винчестера IDE под управлением ОС IS-DOS и другие устройства.

Весьма интересная особенность компьютера KAY-1024 заключается в наличии драйвера RAM-диска, прошиваемого в ПЗУ KAY и эмулирующего работу реального дисковода, контроллера дисковода и, естественно, работу "мозга" дискового контроллера — м/с КР1818ВГ93 со всеми портами и командами.

Особенности драйвера:

- для работы драйвера не используется, не резервируется и вообще не требуется никаких системных переменных DOS;
- никакая программа не знает, что при работе с диском С работает эмулятор;
- TR-DOS тоже "не знает", что при работе дисковода (диска) С работает эмулятор;
- эмулируются все состояния всех портов ВГ93;
- выполняются абсолютно все известные команды ВГ93;
- RAM-диск "вращается" с определенной скоростью;
- имеются индексные импульсы;
- длина физической дорожки — около 63000 байтов;
- при чтении дорожки все считывается нормально (пробелы, метки, заголовки и пр.), но по окончании операции в регистре состояний (#1F) выставляется бит "потеря данных", как и должно быть на реальном контроллере;
- поддержка двух "виртуальных" МГ;
- поддержка 80 физических дорожек;
- проверка значения "Maximum Tracks" в "DCU", "ADS", "FUT", "RDS" и др. дает 80 треков;
- режим и статус прерываний на работу драйвера не влияют;
- не портится содержимое ни одного регистра;
- коды эмулятора располагаются в ранее не используемых областях ПЗУ DOS;
- область данных RAM-диска с 24 по 63 банк KAY-1024 составляет 640 Кб (2560 секторов TR-DOS);
- иногда (но не всегда!) при определении активности/неактивности RAM-диска драйвер использует стек для одноразового хранения AF;
- при работе RAM-диска организуются системные переменные в 23 банке ОЗУ KAY-1024 (к сведению, банки 0...15 занимают 256 Кб, как в "SCORPION-256", остальное — расширенная память KAY-1024), переменные и собственный стек располагаются с адреса #FE00 и занимают около 400 байтов (переменные и свой стек);
- при работе RAM-диска используются 4 байта из стека (для однократного сохранения AF, BC);
- при работе RAM-диска включается SCREEN 0 (#4000);
- форматирование RAM-диска как MS-DOS, IS-DOS,CP/M и еще бог весть как — возможно, но при чтении дорожки длина сектора будет 256 байтов, на дорожке —16 секторов и т.д. (формат дорожки системы TR-DOS). Это вызвано ограничением памяти RAM-диска размером в 640 Кб, что не позволяет хранить нестандартные дорожки целиком — с пробелами, метками и т.п.

На данный момент, на мой взгляд, именно эта модель Spectrum наиболее продвинута и реальна для пользователя, сделано все, что нужно для серьезной и надежной работы. Также хочется заметить, что все новшества данной модели Spectrum далеко не исчерпываются приведенным выше перечнем.

Расширение палитры Spectrum

А.ПЕХИМЕНКО,
Казахстан, Кустанайская обл.,
г.Рудный,
E-mail: paalex@hotmale.com

Работая на Spectrum, многие, особенно художники, столкнулись с такой особенностью цветной графики этого компьютера — яркость знакоместа задается одновременно и для INK, и для PAPER. Т.е. два соседних знакоместа с одним и тем же цветом PAPER, но с разной яркостью, будут выглядеть как два отличных друг от друга квадратика. Это создает некоторые трудности при рисовании и портит картинки, т.к. графика Spectrum в этом случае не является полноценной 16-цветной.

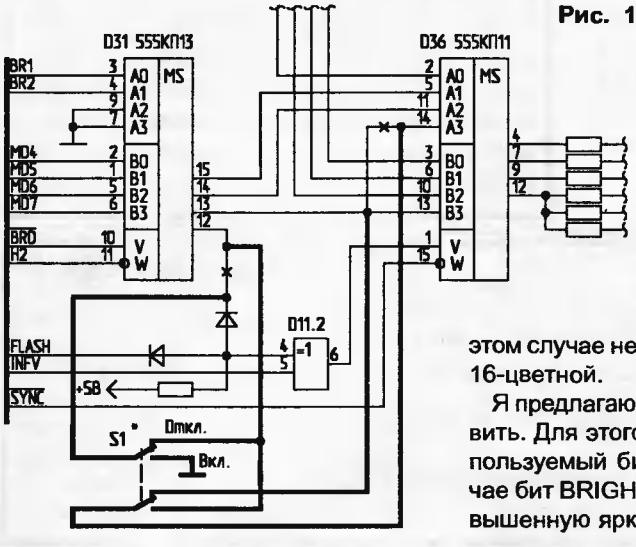


Рис. 1

Я предлагаю этот недостаток исправить. Для этого задействуем малоиспользуемый бит FLASH. В этом случае бит BRIGHT будет отвечать за повышенную яркость INK, а бит FLASH — за повышенную яркость PAPER. В тех программах, где FLASH используется для задания мерцания, а также в 48-м Бейсике, данное знакоместо будет отражаться как знакоместо повышенной яркости (т.е. будет все равно отличаться от обычного знакоместа).

Я предлагаю на ваше рассмотрение две очень простые схемы доработки. В соответствии с первой схемой (**рис.1**) необходимо на плате компьютера разорвать два печатных проводника и установить микропереключатель. Вторая схема (**рис.2**), в принципе — аналог первой. В ней микропереключатель заменен микросхемой 555КП11 (или аналогичной), которая управляется либо битом порта, либо кнопкой. На обоих рисунках места разрывов проводников платы обозначены крестиками, а новые соединения — жирными линиями.

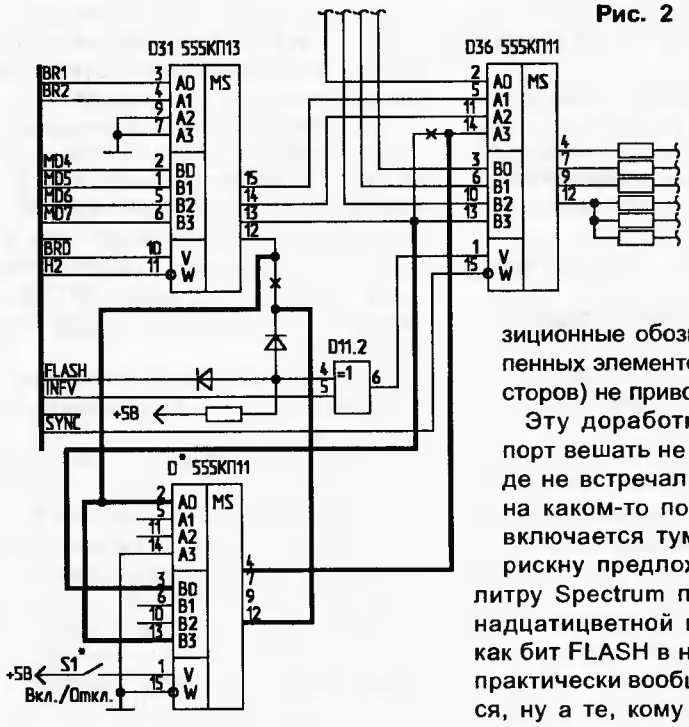


Рис. 2

Обе схемы доработки, повторяю, очень простые и, думаю, в пояснениях не нуждаются. Позиционные обозначения микросхем и разводка соответствуют схеме "Ленинграда", позиционные обозначения второстепенных элементов (диодов и резисторов) не приводятся.

Эту доработку я ни на какой порт вешать не стал, так как нигде не встречал, чтобы она была на каком-то порту — везде она включается тумблером. Я даже рискну предложить сделать палитру Spectrum полноценно шестнадцатицветной как стандарт, так как бит FLASH в новых программах практически вообще не используется, ну а те, кому очень хочется — пусть щелкают тумблером.

Для владельцев других компьютеров схема будет аналогичной. Могу посоветовать следующее:

1. Разделить сигнал BRIGHT, идущий и на INK, и на PAPER. Выяснить по схеме вашего компьютера, где бит данных D6 входит в видеоконтроллер, проследить его через мультиплексоры до последнего мультиплексора перед RGB-выходом, куда стекаются сигналы INK и PAPER. Сигнал D6 подать только к группе INK (D0, D1, D2), а от группы PAPER (D3, D4, D5) отключить.

2. Точно так же проследить, куда идет бит D7 (FLASH). Он должен поступить на некий логический элемент, где он сопрягается с сигналом, отвечающим за частоту мигания. Оторвать сигнал D7 от этого логического элемента.

3. Поставить тумблер или мультиплексор аналогично схемам для "Ленинграда". Теперь тумблером можно включать и отключать данную доработку.

Настройка цветов в STS 6.2

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

В широко используемом отладчике STS 6.2, к сожалению, не предусмотрена возможность настройки цвета панелей диалога. Иногда это доставляет неудобства — например, на моем черно-белом мониторе некоторые панели диалога выглядят слишком темными. Настроить цвета, однако, довольно легко — нужно только знать, в какой ячейке хранится значение, определяющее атрибуты экрана для конкретного диалога. В **табл.1** приведено соответствие диалогов настройки и адресов хранения конкретных атрибутов.

Напомню, что в байте атрибутов младшие три бита определяют цвет текста (ink), старшие три — цвет фона (paper), а 6-й бит — яркость.

Установка нужных значений производится с помощью самого же STS. Запускаем его, загружаем в память файл sts6.2 <C>, в котором и будут производиться изменения, устанавливаем по соответствующим адресам нужные значения и записываем файл — вот и все.

О некоторых возникающих в процессе работы ситуациях STS сигнализирует, изменяя цвет бордюра. Эти цвета также можно настроить, изменяя значения в соответствующих ячейках памяти (**табл.2**).

Еще одно замечание — при запуске STS, экран становится белым до нажатия любой клавиши. Но белый цвет можно заменить на любой другой, поменяв значение по адресу #D070. Учтите только, что в этом байте атрибутов ink и paper должны быть одинаковыми.

Табл. 1

Название диалога	Адрес атрибутов
Load file, Save file	#E701
Find	#E917
Resident	#E51F
Quit	#DECD
Trace	#E079
Fill Block	#E83F
Copy Block	#E8AB
Drive	#E469
Jump	#DF78
Call	#DF5F
Load sectors, Save sectors	#E31A
Get subroutine tacts	#D3E5
Help	#D56E

Табл. 2

Ситуация	Адрес цвета бордюра
При выполнении команды "A" — приглашение к вводу номера ячейки, где будет запомнен текущий адрес	#F6C7
Неправильно введена мнемоника ассемблера	#EC24
При поиске (Find) последовательность не найдена в 64 Кб памяти	#E419
Ошибка чтения/записи при работе с диском	
Нет места на диске	#E5B5

А.ВЕНДИЛОВСКИЙ,
г.Минск.

Operation "Blockade"

Вторая мировая война набирала обороты, войска фашистской Германии яростно атаковали войска Туманного Альбиона, заставляя их все чаще и чаще задумываться о том, что скоро наступит час, когда вся многотонная военная машина Гитлера безжалостно задушит оказавшийся в огненном кольце британский остров. Нужна была передышка, хоть на короткое время и любой ценой...

Пустынный островок, немного скал, немного песка — таких, наверное, сотни тысяч в океане. Взгляд скользнет мимо, и вновь будет поглощен безбрежным океаном. Но этот островок отличается от своих собратьев. Там, в глубине кажущейся хаотичности валунов, от непосвященного взора укрыт надежный бункер — бетон и гранит, готовые выдержать прямое попадание приличной авиационной бомбы. А рядом французский порт, откуда идут транспортные корабли с оружием и припасами для немецкой армии, и откуда начинают свой полет геринговские асы, чтобы бомбить английские города. Только в архивах можно узнать, когда, зачем и почему в глубокой тайне от всех англичане создали это укрепление. Но в настоящий момент его позиция с точки зрения стратегии была исключительна. Был шанс нанести удар как по транспортному снабжению, так и по ничего не ожидающим немецким эскадрильям. План безумен и прост, шансы на выживание бойцов мизерные — но это была единственная возможность хоть на мгновение перехватить у врага инициативу.

Небольшой пляж был наполнен кажущимся покоем. В этой тишине звук передергиваемого затвора и звяканье пулеметной ленты казались очень громкими и неуместными. Скоро, очень скоро воздух наполнится свинцом и гарью, от грохота взрывающихся бомб и снарядов ты оглохнешь, и даже не будешь слышать собственный срывающийся крик, когда очередной грузовой танкер погрузится в морскую пучину. Сквозь дым и огонь не будет видно неба, но даже ночь не принесет покоя — адские всполохи горящих самолетов в небе и пылающие танки у кромки воды. Волны, окрасившиеся в багровые цвета, и стоны раненых, которым уже никто не сможет помочь. И сквозь прицел ты снова и снова будешь искать глазами того, кто вызовет тебя на смертельный поединок, чтобы решить, кто останется на этом пляже навечно. Скоро...

Знаете, у каждого наступает такой период, когда просто уже не хочется играть в заумно-замороченные квесты или ролевухи с сюжетом, который потянет на добрую книгу. Не поднимается рука и на самые последние экшены, ну просто какая-то апатия наступает ко всем самым совершенным новинкам. И появляется стойкое желание найти что-нибудь простое, "спинномозговое", чтобы адреналин в крови кипел, а от вашего рычания за компьютером все тараканы от страха перебежали к соседям. И вспоминаешь со слезой умиления те старые добрые времена, когда компьютеры были большими, а игры маленькими, когда в каждой из них был хороший геймплей, когда (здесь можно продолжить, что, мол, мониторы были зеленее, а звуки, издаваемые спикером, были чище и приятнее, чем саунд самых последних навороченных колонок). И, конечно, вспоминается "Паратрупер" — на скольких платформах она появлялась, сколько безумных часов тогда утекло. Но рано хоронить жанр аркад — давайте поприветствуем "Operation "Blockade"!

Стандартный вид "из окопа", пять видов оружия, и очень много противников. Сюжет не играет никакой роли, хотя немецкие самолеты очень правильно и аккуратно прорисованы. Даже если бы это были летающие тарелки, игра не потеряла бы своей привлекательности.

Начнем с графики. Она приятно радует глаз — очень красивые самолетики и кораблики, с качественной детализацией. Движения пехоты плавны и не вызывают отвращения. Ваше оружие прорисовано на пять с плюсом. Попадание в песок или в воду поднимает султанчики пыли или воды, а каменные ограждения крошатся под ударами артиллерии. Впрочем, любоваться этой картинкой вы сможете лишь первые несколько минут, потом вам будет совсем не до этого.

Если говорить о геймплее, то самое первое, что приходит в голову — хардкорный. Все просто и элегантно. Ощущение азарта будет пьянить вас. После минутной тишины в начале уровня, первой в бой идет авиация. Можно в деталях рассмотреть каждый атакующий вас самолет, а вой пикирующего бомбардировщика точно ни с чем не спутаешь. Но прежде чем открывать огонь, нужно вспомнить, что разработчики подумали и о такой вещи как "упреждение". Придется учитывать и дальность цели, и скорость ее перемещения. А заодно можно полюбоваться системой поражения во всей красе. Времена, когда самолеты таяли в воздухе, канули в Лету. Удачное попадание в двигатель — и самолет осыплется на землю огненным дождем. Очередью ему можно оторвать крыло (которое рухнет отдельно от отчаянно кувыркающего самолета), или даже просто убить пилота. Но даже сильно подбитый самолет представляет опасность для вас. Некоторые в последний момент попытаются повторить подвиг Гастелло, чтобы лишить вас львиной доли жизни.

А пока вы разбирались с проблемами воздуха, на море тоже не дремлют. Под прикрытием катеров на берег пытается высадиться десант. Этого нельзя допустить! Часть платформ топим точными ударами артиллерии прямо в океане, ну а для тех, кто успел высадиться на берег — включаем увеличение и пулемет. Подождем, пока откроется кузов понтона, плавно нажимаем на гашетку — видели когда-нибудь фильм "Спасение рядового Райена"? Вот что-то похожее, только вы в роли пулеметчика, можно даже покричать что-нибудь в духе "Это вам за высадку в Нормандии!!!", наблюдая, как пулеметная очередь превращает в кашу немецкий десант. Избежавших общей судьбы нужно немедленно отправить к праотцам — эти безобразники, мало того что из "Шмайсеров" по вас стреляют, так еще и норовят гранату швырнуть! Самых нетерпеливых можно и ручными гранатами закидать — патроны ведь тоже нужно экономить. Бывает, что вместе с пехотой может высадиться и несколько танков, их тоже нужно быстро отправить в металлолом — для этого есть специальная пушка (самое главное — не нужно считать ворон).

Но бои идут не только на суше или в воздухе — на море очень даже неспокойно, медленно проплывают грузовые баржи, которые нужно в обязательном порядке отправить на дно. Крейсера, маломаневренные, но с пушками о-о-очень большого калибра; мелкие катера, снующие туда-сюда, меняя скорость и быстро маневрируя, не позволяют толком прицелиться. И все отчаянно желают побыстрее сравнять ваш дот с землей.

Вы скажите — простая игра... Да, в ней нет особых изысков, но есть в ней нечто притягательное, что заставляет сидеть за ней вечер за вечером, вечер за вечером...

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

Чужие против Хищника II (Aliens vs. Predator II)

ELROND,
E-mail: elrond@land.ru

Люди всегда все портят — это неоспоримая истина. Игра "Чужие против Хищника — 2" от компании **Monolith Productions** служит тому еще одним подтверждением.

В полной гармонии существовала некая далекая планета. Чужие, обитавшие на ней, спокойно жили и никого не трогали. Время от времени на эту планету прибывали Хищники — поохотиться на Чужих.

Земляне, прибывшие сюда и (о наглость!) основавшие свою колонию, одним видом своим нарушили равновесие. Кроме того, среди них, как всегда, оказался сумасшедший профессор, решивший заняться исследованием Чужих. Естественно, он пренебрег безопасностью, и Чужие спокойно вырвались на свободу. Остановить их оказалось не так уж просто, принимая во внимание и то, что земляне не могли разобраться даже между собой.

Очевидно, что на этой бедной, тихой и мирной планете завязалась настоящая кутерьма. И очередная группа Хищников, как раз в это время прибывших на сафари, не испортила вечеринки. Они даже были довольны увеличенным ассортиментом добычи.

Игра, как и следовало ожидать, стала одной из лучших в своем жанре. Играть предстоит за одного из трех персонажей: Десантника, Чужого или Хищника. Герои довольно своеобразные, поэтому разработчикам необходимо было побеспокоиться о максимальном разнообразии в процессе игры. Что, впрочем, вышло у них прекрасно — абсолютно разные персонажи, абсолютно разные миссии, в огромном количестве абсолютно разное оружие... Все это не скоро даст соскучиться, а великолепная графика позволит полностью погрузиться в мир далеких планет и цивилизаций.

Звуки, как и в первой части "Чужих", довольно часто заставляют вздрагивать. Игра, надо прямо сказать, не для слабонервных... Не каждый сможет спокойно продвигаться по темным исковерканным Чужими отсекам, время от времени слыша подозрительные шорохи где-то за стенкой. А когда Чужой с устрашающими звуками вдруг возникает из ниоткуда (по секрету скажу — до этого он затаился, вися на потолке) — холодный пот невольно прошибает даже самых закаленных в боях игроков. При этом вблизи можно рассмотреть каждую деталь уродливого тела. После игры в "Чужие", ночью руки дрожат еще долго.

Однако, несмотря на высокое качество графики, она меня немного разочаровала. И в первую очередь из-за ажиотажа, поднявшегося вокруг движка LithTech, который порой мало чем отличался от движка, использованного в первой части игры.

Добавление "устройства хакера", позволяющего "взламывать" электронные схемы для открытия ранее заблокированных дверей и др., сначала порадовало. Однако далее в ходе игры этот девайс порядком надоел. На мой взгляд, разработчики добавили его в игру лишь для того, чтобы добавить. Для квеста он, может быть, и подошел бы, но для шутера...

Таких мелких недостатков накапливается немало, но все же впечатление от игры у меня осталось довольно хорошее — вряд ли сейчас найдется что-либо подобное. Реалистичная графика, отличное звуковое сопровождение и продуманный сюжет поставили "Чужих" на высокий уровень и сделали их одним из лучших 3D-шутеров нашего времени. И, я думаю, не зря эта игра пользуется таким огромным успехом.

Системные требования

Windows 95/98/ME/2000/XP;
Pentium II 450 (рекомендуется Pentium III 500);
128 Мб ОЗУ;
видеокарта с 16 Мб ОЗУ (рекомендуется 32 Мб);
DirectX 8.0+.

К.КЛИЩЕНКО,
г.Минск.

ЧерноDiablo

Данная игра является эпитафией на могиле овечки Долли, умерщвленной в связи с подозрением на ящур.
Неизвестный автор

Ну, наконец, вот и русские добрались до клонирования великой аркады Diablo. К тому же, первой. Но, несмотря на постоянные сравнения с прародителем, постараюсь быть объективным.

Самый главный минус связан с опозданием этой игры на пару лет. Ведь когда есть вторая часть бессмертной "Дьяволиады", то клон первой части не может даже смутить умы праздных обывателей.

Изометрический вид и обилие серых тонов в спрайтовом движке полностью описывают изобразительный ряд игры. Но, несмотря на подобную неизощренность, картинка смотрится приятно-мрачновато. Интересно задумана графика для магии. Слабенькие персонажи, а тем более монстры (в прародителе монстры смотрелись намного лучше). Однообразие деревень лишь оттеняет интересные, но, на мой взгляд, немного затянутые подземелья. Затянутость проявляется не в величине — авторы слишком часто применяют удачные дизайнерские ходы и тем нивелируют их. Карта точь-в-точь скопирована с эталона. Неплохим решением, по моему мнению, стало очень маленькое поле зрения героя — очень, знаете ли, обостряет внимание и заставляет осторожничать.

По поводу звука не могу сказать ничего, то есть абсолютно. Я не слышал ничего особенно плохого, но и ожидать гитары от Diablo не приходится.

Сюжетик ничего — поинтересней, чем в двух Дьяволах вместе взятых. Правда, подача не лучшая. Однако чувствуется, что игра сделана русскими. Ну кто, не боясь резких снижений продаж, станет отдавать венец главного злодея Папе Римскому? Чтобы избежать конфликтных ситуаций, действие происходит не в нашем мире. Помимо сюжета, в игре есть и побочные задания — не особенно разнообразные, но зато и не навязанные насильно (реверанс в сторону второго Дьябло). Диалоги, конечно, не из Elder Scrolls (любой, кроме Battlespire). Но мы и не настаивали. Просто нормально, что не каждый хочет продать вам немного прекрасного хлама или дать задание.

В предметах наиболее смелое (в плане ухода от генетического кода Diablo) начинание. Теперь "всунуть" что-нибудь ценное можно в любой предмет, а не только в специальный. С помощью растворителя вы способны передать особенности многочисленных ингредиентов своему снаряжению. К сожалению, нужно следить, чтобы предмет не развалился, так как внедрение каждого нового свойства отражается не в лучшую сторону на "выносливости" предмета. И хотя он не может развалиться от ударов, однако следующее улучшение не только не добавит вам мощи, но и лишит былой. В связи с этим иногда появляются интересные коллизии — несильная изначально, но затем "прокачанная" экипировка иногда оказывается выгодней новой и слишком сырой. К тому же, на все нужны деньги, вот и ждешь момента, чтобы сделать качественный скачок в развитии.

Герои практически не отличаются от предоставленных Blizzard. Только "ближнебоевых" два вида — воин и рыцарь (у первого смещение в сторону защиты). У вора, кроме стрельбы из лука, есть реальная возможность взломать пару замков. Прогрессируют персонажи по привычной схеме. То есть после каждого уровня дается некоторое количество очков (10) на распределение их между характеристиками. Умения же растут по мере их применения (логичная для русских игр черта). Заклинания получаются путем нахождения или покупки.

Монстры, вопреки заявлениям на коробке, не слишком умны. Ничего они особо не патрулируют (в том смысле, что если они это и делают, то не вносят никаких отличий в процесс исследований). Ни за кем не следят (может, мне просто не повезло, и как раз в этот момент они следили не за мной). В их нападениях не проблескивают искры организованности и умелых тактических ходов. Правда, у монстров неплохое чувство опасности. У всех у них есть одна черта, которая позволит особенно осторожным игрокам при прохождении почти не получать повреждений — враги не могут выходить далеко из своего ареала обитания и перестают вас преследовать, если удастся оторваться.

Играть не слишком скучно (кроме приступов нетерпения в подземельях). Но и нет азарта — авторы не смогли увлечь (и не только меня) на полное прохождение игры. Может, хоть кто-нибудь пройдет ее и расскажет, как там в конце?

Общее мнение, в принципе, нейтрально. Игра неплохая. Но это клон устаревшей игры, к тому же, еще и не самой моей любимой. Возможно, игра даже лучше оригинала. Но теперь это уже не различить.

P.S. Извините за не слишком интересную и изящную статью. Но каково настроение, такова и статья.

P.P.S. Да, не пугайтесь, глядя на скриншоты — я играл в эту игру не пять минут.

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***

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

Куплю программное обеспечение: игры, графические и текстовые редакторы для компьютера "Искра-1030М" (СМ 5508, ДОС версии 3.30), инструкцию и описание принтера СМ 6337.
453500, Башкортостан, г.Белорецк, а/я 21.
Сафонов С.Н.
Тел. (34792) 4-45-67 (спросить Сергея).

Куплю компьютер "Радио-86 РК" с монитором и дисководом и документацию, программатор для PIC-контроллеров, литературу по микросхемам памяти.
Алексей.
E-mail: leoh1985@mail.ru

Недорого **продам или обменяю** струйный принтер EPSON COLOR STILUS-600 с засохшей головкой.
Тел. в Воронеже (0732) 72-46-62. Виталий.
E-mail: ra3qg@mail.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 **"Торм"**, 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").

Operation "Blockade"


ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Русский Проект
4X4 EVO 2
EVOLUTION
PC
ПОЛНАЯ РУССКАЯ ВЕРСИЯ
REAL POOL 2
Jane's
USAF
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Русский Проект
SID MEIER'S
CIVILIZATION III
200
ЛУЧШИХ ИГР ВСЕХ ЖАНРОВ
избранное
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
КРЕПКИЙ ОРЕШЕК
NAKATOMI PLAZA
RUS
PC CDROM
НА РУССКОМ ЯЗЫКЕ
ФАРГУС
НА РУССКОМ ЯЗЫКЕ
Might and Magic IX
2 IN 1
РУССКАЯ И АНГЛИЙСКАЯ ВЕРСИИ
ЗОЛОТАЯ КОЛЛЕКЦИЯ
ЧЕРВИ ПРОТИВ СВИНЕЙ
5в1
ЧЕРВИ ПРОТИВ СВИНЕЙ
ФАРГУС
НА РУССКОМ ЯЗЫКЕ
РУССКАЯ ВЕРСИЯ
OPERATION FLASHPOINT
COUNTER STRIKE 2
KILLER EDITION
РУССКАЯ И АНГЛИЙСКАЯ ВЕРСИИ