http://radio-mir.com
индексы: 48925, 45995

радиомир

Ваш компьютер

7•2003

ДОРАБОТКА ДИНАМИКОВ КОМПЬЮТЕРНЫХ КОЛОНОК

(СМ. СТАТЬЮ И.ДЗЮБЫ)

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

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

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

Компьютерная верстка —
Янина БЕЛЬСКАЯ

Техническая графика —
Наталья БЕЛЬСКАЯ,
Александр ГОРАЮТИН,
Мария КАЛАБУХОВА

Оформление обложки —
Наталья БЕЛЬСКАЯ,
Надежда БОГОМОЛОВА

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

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

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

Материалы для публикации принимаются в рукописном, печатном и электронном вариантах

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

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

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

Подписано к печати 27.05.2003 г.
Формат 60 x 84 1/8.
Печать офсетная, 6 печ. л.
Цена свободная.

Отпечатано в типографии ЗАО "Красногорская типография".
Заказ № 1429. Тираж 1800 экз.

радиомир
Ваш компьютер

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

№7/июль/2003

ЧИТАЙТЕ В НОМЕРЕ

Дорогие друзья!

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

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

Редакция.

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

Все они с одной планеты

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

Эта публикация посвящается всем тем, кто на протяжении долгих лет создавал и внедрял в повседневную жизнь новые компьютерные технологии. Тем, чьи имена, держу пари, помнят лишь немногие пользователи.

Может быть, рассказ о компьютерной индустрии периода 1971...1993 гг. кому-то покажется скучным. Но ведь именно в это время появились те базовые операционные системы и компьютеры, которые предрешили приход высоких технологий в наш повседневный быт. Теперь трудно себе представить офис или учебное заведение без PC. Так давайте же вместе постараемся вспомнить то время, когда компьютеры были такими большими, а мы — такими маленькими...

"Родители" персонального компьютера

По своей сути я все-таки "историк" — у меня дома до сих пор рядом с Pentium стоит старенький 286 IBM, на котором и я, и некоторые мои знакомые любим поиграть в старые игры. Мне кажется, я в этом не одинок. Ведь кто-то до сих пор работает на "Спектруме", "Байте", "Корвете". Многие делают это не от того, что не могут позволить себе новый компьютер, а просто из-за доброй ностальгии по бесследно прошедшей эпохе. Времени восьмиразрядных процессоров и пятидюймовых дискет. Вспомним основные вехи.

1971 — выпуск первого коммерчески доступного микропроцессора. Название — Intel 4004. Разработчик — корпорация Intel, разработка проводилась для компании Busicom.

1971 — начало регулярного использования 8-дюймовой гибкой дискеты. Разработчик — Алан Шугарт (Alan Shugart) из IBM.

1972 — первое клонирование компьютеров фирмы IBM. Название — ЕС ЭВМ. Разработчики —страны-участницы СЭВ (Совета экономической взаимопомощи) — Болгария, Венгрия, ГДР, Польша, СССР и Чехословакия. Приблизительный период разработки —1970...1987 гг.. Единая с американцами система электронных вычислительных машин (ЕС ЭВМ) базировалась на архитектуре IBM 360/370. Значительно позже, в 80-х годах, подобные копии с IBM PC/XT стали называться клонами или "аналогами" и потеснили "Голубого гиганта" на вторые роли. Возникло даже движение — "клономания", продолжающееся и по сей день.

1972 — первый цифровой микрокомпьютер, доступный для персонального использования.

Название — MITS 816. Разработчик — MITS (Micro Instrumentation and Telemetry System —англ., микроаппаратура и телеметрические системы).

1973 — первый полнофункциональный персональный компьютер, укомплектованный монитором. Название — Alto. Разработчик — фирма Xerox, лаборатория в Пало-Альто (Xerox PARK). Приблизительный период разработки — 1970...1973 гг.

1974 — первый выставленный на продажу комплект для сборки персонального компьютера. Название — Mark-8. Разработчик — Джонатан Титус (Jonatan Titus). Приблизительный период разработки — 1973...1974 гг.

А наш комплект для сборки компьютера состоял из паяльника, "мозгов" и кучи интегральных схем.

1975 — первый серийно произведенный и выставленный на продажу персональный компьютер (в комплекте для сборки и уже собранный). Название — Altair 8800. Разработчики — Эдвард Робертс (Edvard Roberts), Вильям Ятес (Wiliam Yates) и Джим Байби (Jim Bybee). Приблизительный период разработки — 1973...1974 гг. В первом Altair использовался процессор Intel 8080 и 4 Кб памяти. По заказу Эдварда Робертса из компании MITS, распространяющей компьютер, Бил Гейтс и Поль Ален написали интерпретатор языка Бейсик, втиснув его в имеющиеся 4 Кб (Б.Гейтс до сих этим пор гордится). Так начиналась софтверная компания Microsoft.

ALTAIR

1975 — первый персональный компьютер IBM— IBM 5100 Portable Computer. Приблизительный период разработки — 1973...1975 гг. IBM 5100 Portable Computer был первой (и неудачной) попыткой IBM в конце 1974 г. реализовать "в металле" концепцию персонального компьютера. Продажа и маркетинг этого устройства были неудачны.

IBM 5010

Этот компьютер имел встроенный ленточный накопитель, маленький экран и возможность управления программами на Бейсике или APL (языке программирования, созданном IBM). В отличие от следующих, более успешных моделей, в данном случае IBM не ориентировалась на микропроцессор Intel. Монитор IBM 5010 отображал 16 строк по 64 символа в каждой, память расширялась до 64 Кб, ленточный накопитель использовал стандартную аудиокассету, на которой можно было хранить приблизительно 200 Кб данных. Весила эта модель около 23 кг и стоила около 10000 USD. Данный компьютер разрабатывался для использования в малом бизнесе, но высокая стоимость, недостатки интерфейса и невозможность обмена данными с другими компьютерами ограничили его применение и не позволили стать широко используемым персональным компьютером.

1976 — первый чрезвычайно успешно продаваемый персональный компьютер. Название — Apple II. Разработчики — Стив Джобс (Steve Jobs) и Стив Возняк (Steve Wozniak). Приблизительный период разработки — 1974...1976. Первый компьютер Apple II, собранный буквально "на коленке", не очень отличался от своих собратьев (Altair и других). И только линия Apple II, изготовленная на коммерческой основе, стала чрезвычайно популярна. Немного позже появились Applle III и Lisa, а только затем — Macintosh, вышедший как Mac 128K (со всеми новинками, при-

писываемыми фирме Apple как первооткрывателю). Apple II имел 48 Кб памяти и операционную систему S.O.S. (Sophisticated Operating System — англ., замысловатая операционная система). Он положил начало тенденции всеобщей компьютеризации и породил фанатизм, который мешает адекватному восприятию компьютеров этой компании.

APPLE 2

1981 — первый успешно продаваемый персональный компьютер IBM. Название — IBM Personal Computer (IBM PC). Приблизительный период разработки — 1978...1981 гг. Оригинальный IBM PC — это модель 1983 г. с 640 Кб оперативной памяти, но самые ранние модели могли иметь только 64 Кб на материнской плате. Этот специфический PC имел два пятидюймовых дисковода для гибких дискет на 360 Кб (один из них — фирмы IBM, а другой — от некой третьей фирмы) и зеленый экран монитора (переключатель "Вкл./Выкл." отсутствовал) и "кликающую" клавиатуру с небольшими клавишами Shift и Return. Потребляемая мощность составляла всего 63,5 Вт. IBM представила Personal 12 августа 1981 г. В то время большинство компьютеров все еще были 8-разрядными и могли обрабатывать только 8 битов информации за такт. IBM революционизировала компьютерную индустрию, выйдя на рынок с персональным компьютером, базирующемся на процессоре Intel 8088 и совместимом с компьютерами на 8-разрядных процессорах, но обрабатывающем до 16 битов информации за такт. PC показал пример расширяемой архитектуры, известной как "открытая архитектура", которая дала возможность пользователям добавлять новые компоненты к их компьютерам без замены всего устройства. Первоначально IBM PC (модель 5150) выпускался снабженным 16 Кб стандартной оперативной памяти (микросхемы 9x16 Кбит), расширяемой до 64 Кб на материнской плате и до 540 Кб возможного общего количества памяти (не 640 Кб из-за аппаратной ошибки); имел одноцветный TTL-монитор (модель 5151). Там зеленым по черному отображалось 25 строк по 80 символов. Он подключался в AC-слот на блоке питания компьютера (63,5 Вт), так что не нуждался в собственном выключателе. Монохромный графический адаптер с параллельным портом для принтера, последовательный порт, два места полной высоты для внешних устройств, гибкий дисковод, способный использовать односторонние и двусторонние дискеты с одинарной и удвоенной плотностью записи (емкостью 80...360 Кб) — "джентльменский набор" первого IBM PC XT. Именно начиная с него в индустрии персональных компьютеров начался взрыв, который в значительной степени стал возможен благодаря открытой архитектуре IBM PC, ставшей промышленным стандартом.

1981 — первый успешно продаваемый переносной микрокомпьютер с экраном, дисководами и сумкой для переноски. Название — Osborne 1. Разработчик — Osborne Computer Corp. Приблизительный период разработки — 1980...81 гг. Краткое описание: дисковод для пятидюймовых дискет, крошечный экран (3,55 дюйма по горизонтали и 2,63 по вертикали), шаблон текстового процессора WordStar на клавиатуре, аккумуляторные батареи и сумка для переноски.

Какой была ваша первая операционная система? На этот вопрос каждый ответит по-разному. Для кого-то это был старик TR или CPM DOS, добрый продукт от MS, а для кого-то вообще восьмибитная приставка. Но суть вопроса не в этом, а в том, что для каждого из нас они были, прежде всего, близкими друзьями, под которыми работали наши любимые игры и программы. Синюшные клоны Norton, бесконечные лабиринты и уровни PackMan и Digger... А ведь все это было бы невозможно на компьютере, не будь на нем хоть какой-нибудь операционной системы...

DOS и ее "родственники"

Рассказ о DOS, наверное, стоило бы начать с чего-то другого. Однако, в силу "знаменитости" в бывшем Советском Союзе этой операционной системы, повествование начнется именно с нее. Отсчет DOS-истории начнем с 1973 г. — именно в столь теперь далеком от нас году Gary Kildall написал простую операционную систему на созданном им же языке PL/M (Programming Language/Microprocessor). Он назвал ее CP/M — Control Program/Monitor или Control Program for Microcomputer. К середине 70-х годов CP/M-80, права на которую принадлежали фирме Digital Research, стала наиболее популярной системой для компьютеров на базе процессоров Intel 8080 и Zilog Z80. Эта система обеспечивала доступ к разнообразным средствам прикладного программного обеспечения (текстовым процессорам, администраторам баз данных и т.д.). "Нормальные" же версии, дошедшие и до наших времен и пользовавшиеся относительной известностью, ведут свой счет с QDOS. Эта менее длинная история, нежели развитие UNIX, началась в 1980 г. в фирме Seattle Computer Products. Операционная система, первоначально названная QDOS, затем была модифицирована и, переименовавшись к концу года в 86-DOS, была продана Microsoft. В конце 1981 г., когда новый компьютер IBM PC приобрел широкую популярность (IBM 5150 PC: 4,77 МГц Intel 8088 CPU, 64 Кб RAM, 40 Кб ROM, 5.25-дймовый флоппи-дисковод — "всего" 3000 USD), его операционная система представляла собой модифицированную версию системы 86-DOS, названную PC-DOS 1.0.

IBM PC

Ее недостаток был лишь в том, что под каждую конкретную машину ее приходилось настраивать заново. Развитием PC-DOS занялась сама IBM, а MS досталась ее собственная модификация, именуемая MS-DOS. Вскоре после выпуска IBM PC, на рынке стали появляться персональные компьютеры, "схожие с PC" (PC-совместимые). Операционная система этих компьютеров называлась MS-DOS 1.0 — корпорация Microsoft предоставила в распоряжение фирмам, производящим эти машины, точную копию операционной системы PC. Версия MS-DOS 1.1 с некоторыми до-

бавленными и расширенными возможностями появилась в 1982 г. К 1983 г. была разработана версия 2.0. По сравнению с предыдущими, она давала возможность использовать жесткий диск и встроенные устройства для дискет, а также обеспечивала поддержку усложненного иерархического каталога диска и системы управления файлами. Третья версия MS-DOS, выпущенная в 1984 г., дала лишь некоторые улучшения. MS-DOS версии 5.0 (1991 г.) предоставила возможность использования памяти, расположенной выше 1 Мб. В нее была добавлена поддержка новых 2,88 Мб дискет, а также несколько утилит, в том числе UNDELETE — для восстановления удаленных по ошибке данных. В 1992 г. появляется версия 5.а — в ней были устранены грубые ошибки работы утилит UNDELETE и CHKDSK. В MS-DOS 6.0, вышедшей в 1993 г., расширились возможности использования памяти, расположенной выше 1 Мб, были добавлены утилита оптимизации использования памяти (Memmaker), средство увеличения эффективного дискового пространства (DoubleSpace, следует отметить, что после судебных разбирательств с компанией Stack по поводу авторского права на DoubleSpace, последний в версии MS DOS 6.22 был заменен утилитой DriveSpace), утилиты проверки и оптимизации жесткого диска (ScanDisk и Defrag). Для защиты была установлена антивирусная программа. После версии 6.22 появилась сильно урезанная 7.0, входящая в состав Windows 95 service relies. Больше Microsoft развитием DOS не занималась.

А тем временем PC DOS не умирала. Последняя ее версия включала в себя практически все, что могла MS DOS, плюс такие функции как средства резервного копирования и восстановления поврежденных данных, встроенные в систему средства антивирусного контроля, обеспечение синхронизации файлов на двух компьютерах и т.д. Еще одним ярким представителем данного класса была операционная система PTS DOS производства одной из российских физических лабораторий. В ней имелся свой встроенный драйвер кириллицы. Последняя ее версия была обозначена как 6.65. Однако самой необычной и малоизвестной является DR-Open DOS 7.02. Изначально эту ОС разрабатывала Digital Research, но потом по каким-то причинам от нее отказалась и продала компании Novell. Та встроила в нее свои сетевые утилиты, продав дальше — фирме CALDERA, которая дополнила DR-DOS средствами доступа в Internet и сейчас распространяет ее бесплатно. Единственным серьезным отличием всех вышеупомянутых систем от MS DOS было то, что называется "уровнем системы", т.е. для каждой машины необходимо было покупать свою операционную систему. Отличительные особенности каждой мог выявить только системный программист, в чьи обязанности входила работа по "подгонке" операционной системы к конкретной машине, при этом пользователь, работающий на разных машинах, не ощущал никакой разницы между ними.

UNIX и семейство

Считается, что в появлении UNIX повинна компьютерная игра. Дело в том, что Кен Томпсон из Bell Labs в 1969 г. на компьютере Honeywell 635, который использовался для разработки OS Multics, непонятно чего ради написал игрушку "Space Travel". "Фишка" в том, что ни вышеупомянутый Honeywell, ни имевшийся в лаборатории компьютер General Electric 645 не подходили для игрушки. И Кену пришлось найти другую ЭВМ — 18-разрядный компьютер PDP-7. В то время Кен участвовал в разработке новой операционной системы, и чтобы облегчить себе жизнь и работу, решил опробовать свои наработки на новенькой машине. Опробовал, результатам дружно радовался весь отдел патентов Bell Labs. Томпсону этого показалось мало, и он начал усовершенствовать новую ОС, включив в нее такие функции как inodes, подсистему управления процессами и памятью, обеспечивающую использование системы двумя пользователями в режиме Time Sharing (разделения времени), и простой командный интерпретатор. Кен даже разработал несколько утилит под эту систему. Сотрудники Bell Labs еще помнили, как они мучились с OS Multics, поэтому, в память старых заслуг, один из них, по имени Брайан Керниган решил назвать новую ОС похожим именем UNICS. Через некоторое время название сократили до UNIX (читается так же, просто писать лишнюю букву настоящим программистам во все времена было лень). Написана UNIX была на ассемблере. В ноябре 1971 г. был опубликован первый выпуск полноценной документации по этой системе. В соответствии с этим и ОС была названа "Первой редакцией UNIX". Довольно быстро увидела свет и вторая редакция. Последовавшая следом третья ничем особенным не отличалась, разве что, заставила Дениса Ритчи "засесть за словари", вследствие чего тот написал собственный язык, известный сейчас как язык С. И именно на нем в 1973 г. была написана 4-я редакция UNIX. В июле 1974 г. вышла 5-я версия UNIX. Шестая редакция (V6), выпущенная в 1975 г., стала первым коммерчески распространяемым Юниксом. Большая ее часть была написана на С. Позже была полностью переписана подсистема управления оперативной и виртуальной памятью, заодно изменили интерфейс драйверов внешних устройств. Все это позволило сделать систему легко переносимой на другие архитектуры и получило название "Седьмая редакция" (UNIX V7). Параллельно с улучшением Юникса шла разработка системы, известной нам как Free BSD. В 1976 г. в Университете Беркли появились свои местные юникс-гуру, одним из них был Бил Джой. Собрав своих друзей-программистов, он начал разработку собственной операционной системы на ядре UNIX. "Запихнув" в ОС, помимо основных функций, кучу своих разработок (в том числе компилятор Паскаля), он назвал всю эту "сборную солянку" Distribution (BSD 1.0). Вторая версия почти ничем не отличалась от первой. Третья редакция BSD основывалась на переносе UNIX V7 на компьютеры семейства VAX, что породило систему 32/V, легшую в основу BSD 3.x. Ну, и самое главное — при этом был разработан стек протоколов TCP/IP. Данная разработка финансировалась Министерством безопасности США. Первая коммерческая система называлась UNIX SYSTEM III, и вышла она в 1982 г. В этой OS сочетались все лучшие качества UNIX Version 7.

Далее все развивалось примерно так. Во-первых, появились компании, занимавшиеся коммерческим переносом UNIX на другие платформы. К этому приложила руку и компания Microsoft совместно с Santa Cruz Operation, создавшей UNIX-вариацию под названием XENIX. Во-вторых, Bell Labs создала группу по развитию UNIX и объявила о том, что все последующие коммерческие версии (начиная с System 5) будут совместимы с предыдущими. К 1984 г. был выпущен второй релиз UNIX 5, в котором появилась возможность блокировок файлов и записей, копирования совместно используемых страниц оперативной памяти при попытке записи (copy-on-write), странично-

го замещения оперативной памяти и т.д. К этому времени ОС UNIX была установлена уже на более чем 100 тыс. компьютеров. В 1987 г. был выпущен третий релиз UNIX 5 и было зарегистрировано четыре с половиной миллиона пользователей этой эпической операционной системы. Кстати, что касается Linux, то он возник лишь в 1990 г., а первая официальная версия этой ОС вышла лишь в октябре 1991 г. Как и BSD, Linux распространялся с исходниками, чтобы любой пользователь мог настроить ОС так, как ему хочется.

IBM и операционные системы

Все началось с ОС VM (Virtual Machine), которая вышла в 1972 г. под названием VM/370 и была предназначена для поддержки сервера для определенного количества пользователей. Эта ОС, давно отметившая свой 30-летний юбилей, по истории которой можно изучать развитие технологий IBM в области серверных операционных систем и сетевых решений, является надежной и мощной базой для организации корпоративной информационно-вычислительной системы, ориентированной на многопользовательскую среду крупной современной фирмы. Система VM/ESA очень эффективно использует возможности аппаратного обеспечения и несколько менее требовательна к вычислительным ресурсам компьютера по сравнению с OS/390, что делает ее хорошим вариантом для использования в качестве платформы для корпоративной системы, информационного сервера крупной организации или сервера в Internet. Позже IBM организовала совместный проект с компанией Microsoft, нацеленный на создание операционной системы, лишенной недостатков. Первая версия OS/2 вышла в конце 1987 г. Она была в состоянии использовать развитые вычислительные возможности процессора и обладала средствами обеспечения связи с большими машинами фирмы IBM. В 1993 г. увидела свет 0S/2 2.1 — полностью 32-разрядная система, обладавшая способностью выполнять приложения, созданные для Windows, имевшая высокую производительность и поддерживающая большое количество периферийных устройств. В 1994 г. вышла 0S/2 WARP 3. В этой реализации, помимо дальнейшего повышения производительности и снижения требований к аппаратным ресурсам, появилась поддержка работы в Internet. Сегодня же из последних версий следует отметить лишь 0S/2 WARP 4, способную работать с 64-разрядными процессорами. Кроме того, в ней достаточно полно представлены средства взаимодействия с Internet, позволяющие 0S/2 выполнять не только клиентские программы, но и выступать в качестве сервера. Начиная с третьей версии, фирмой IBM поставляются локализованные версии 0S/2 для России. Пройдя довольно большой и сложный путь, эта ОС для персональных компьютеров обладает сегодня такими особенностями как реальная многозадачность, продуманные и надежные подсистемы управления памятью и администрирования процессов, встроенная поддержка работы в сети и дополнительные функции сетевого сервера, мощный язык программирования REXX, предназначенный для решения задач системного администрирования. Перечисленные возможности позволяют использовать 0S/2 в качестве операционной системы для мощных рабочих станций или сетевых серверов.

И напоследок — один очень интересный факт. Среди изобретений человечества, компьютер поставили на второе место. Вы спросите, а что же на первом? Дорогие мои друзья, это всего лишь унитаз...

При написании статьи были использованы материалы сайтов http://www.edu.yar.ru, http://ol.holm.ru, http://www.astu.astranet.ru, а также публикации зарубежной прессы.

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

Создание файлов формата PDF

Многие из пользователей персональных компьютеров обращали внимание на файлы, имеющие расширение PDF, которые можно просмотреть, используя программу Adobe Acrobat Reader. Но наверняка не все пользователи знают, как в офисных или домашних условиях можно получить PDF-файлы, и какие программы для этого понадобятся. В статье описаны два несложных способа получения файлов формата PDF.

Что же такое PDF? Аббревиатура PDF расшифровывается как Portable Document Format — формат переносимых документов, который является открытым стандартом для повсеместного распространения и публикации электронных документов в Интернете. Systems Incorporated разработан фирмой Adobe. Это универсальный файловый формат, сохраняющий все шрифты, форматирование, цвета и графику любого исходного документа, независимо от использованных для его создания приложений и платформы. Файлы формата PDF могут быть отображены на различных платформах: MS-DOS, Windows, Unix и Macintosh. Для создания файлов формата PDF используется программа Adobe Acrobat, а для их оперативного просмотра и печати — Adobe Acrobat Reader. Файловый формат PDF специально разрабатывался для пересылки при помощи электронных средств связи документов, содержащих графические объекты. Для просмотра файлов формата PDF не нужно иметь те шрифты, которые использовались при создании документов. Отпадает необходимость иметь и программные средства для просмотра графических объектов.

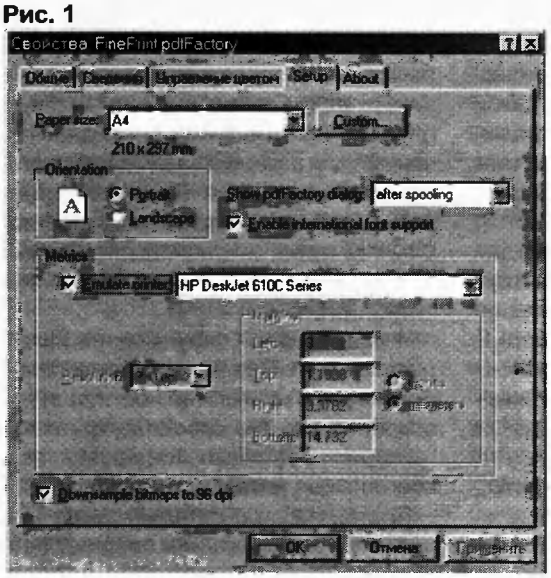
Основные достоинства PDF:

- компактность;
- программа для просмотра файлов распространяется бесплатно;
- в файлах могут быть представлены как растровые, так и векторные графические объекты;
- файлы могут содержать закладки (bookmarks) и гипертекстовые ссылки.

Для создания файлов формата PDF совсем не обязательно устанавливать программные комплексы Adobe Acrobat или Adobe PageMaker. Существует определенный класс программ-драйверов, с помощью которых исходные файлы можно преобразовывать в файлы формата PDF, например, pdfFactory. После установки этой программы, в папке "Принтеры" появляется новое устройство "FinePrint pdfFactory". С его помощью из любого Windows-приложения можно создать файл формата PDF. Для этого в меню "Файл" воспользуемся командой "Печать...", и в списке принтеров выбираем устройство "FinePrint pdfFactory". При просмотре в Acrobat Reader'е полученных файлов могут возникнуть проблемы с читаемостью русского текста. Для решения этой пробле-

мы нужно дополнительно установить в системе кириллические TrueType-шрифты фирмы Adobe и использовать их в своих исходных документах вместо "майкрософтовских". Кроме того, в настройках программы, на вкладке Setup необходимо включить опцию "Enable international font support". Также на этой вкладке можно задать нужный формат листа, книжную или альбомную ориентацию (**рис.1**).

Рис. 1



Включение опции "Emulate printer" позволит сохранить в структуре PDF-файла те же разрывы строк и разделители страниц, как если бы он был напечатан на выбранном принтере. При этом будут использоваться разрешение и поля выбранного принтера. Отмена этой опции позволит установить свои собственные поля и разрешение.

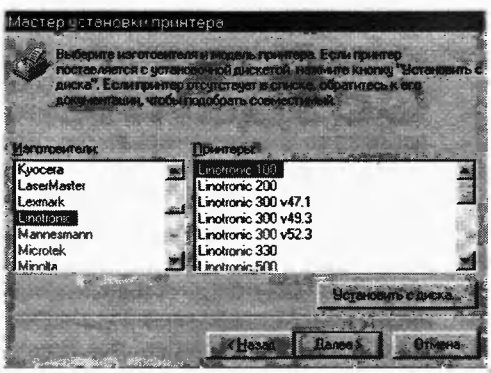
Определить, какая фирма "причастна" к созданию конкретного шрифта, не сложно — надо навести курсор на файл шрифта и нажать клавишу Enter или дважды кликнуть мышью по файлу. Загрузится программа просмотра шрифтов, и в ее окне можно прочитать информацию о фирме-создателе.

Единственный существенный недостаток, обнаруженный при работе с программой-драйвером pdfFactory — довольно "громоздкий" размер получаемых PDF-файлов. Увеличение размера файлов PDF происходит, в основном, за счет внедрения (embedding) кириллических шрифтов. Желающие использовать эту программу могут найти ее на компакт-диске, прилагающемся к журналу "Chip" №9/2001.

Второй, более сложный способ получения файлов формата PDF практически лишен каких-либо недостатков. В этом случае создание PDF-файлов происходит в два этапа. Сначала исходный документ следует преобразовать в формат PostScript. Затем с помощью программы Acrobat Distiller полученный файл формата PostScript нужно конвертировать в формат PDF. Для того чтобы появилась возможность конвертировать исходные документы в файлы формата PostScript, нужно установить драйвер устройства Linotronic. Для этого двойным щелчком мыши по значку "Установка принтера", который находится в папке "Принтеры", производим установку драйвера устройства Linotronic. Из списка принтеров можно установить любой. Они отличаются между собой только поддерживаемыми форматами листа (**рис.2**).

Linotronic — это фотонаборный аппарат фирмы Linotype. Областью применения этого оборудования является получение фотооригиналов и офсетных печатных фотоформ для высококачественной полиграфической печати. Драйверы устройства Linotronic есть в дистрибутиве Windows

Рис. 2

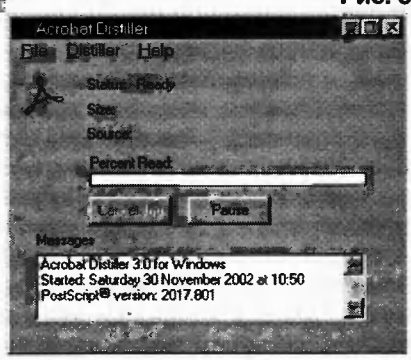


(спасибо Microsoft!), остается только вставить инсталляционный компакт-диск. Получить файл формата PostScript несложно. В свойствах Linotronic нужно обязательно включить опцию "Печатать в файл". Как и в случае с pdfFactory, в меню "Файл" используем команду "Печать...", но из списка принтеров выбираем уже "Linotronic". Полученный файл формата PostScript по умолчанию имеет расширение PRN (file_name.prn). Перед конвертированием в формат PDF расширение файла необходимо изменить на PS (file_name.ps).

Далее, из пакета Adobe PageMaker в "выборочном" режиме (Custom) нужно установить только одну программу — Acrobat Distiller. Эта программа оптимизирует и конвертирует код исходного файла формата PostScript в формат PDF. Acrobat Distiller, по сравнению с pdfFactory, обладает более широкими возможностями и большим количеством параметров. Причем получать файлы формата PDF в Acrobat Distiller можно как через запуск программы, далее в меню "Open", так и двойным щелчком мыши по файлу с расширением PS (file_name.ps). Файлы PDF после преобразования Acrobat Distiller'ом получаются очень компактными. Например, двухстраничный текст, содержащий 1400 слов (без графических объектов), умещается в файле размером 44 Кб. Проблем с чтением русского текста в файлах формата PDF, полученных Acrobat Distiller, замечено не было. Поэтому дополнительно подключать кириллические шрифты Adobe, как в случае с программой pdfFactory, нет необходимости. Из негативных моментов можно отметить лишь то, что программа при установке на жесткий диск создает в корневом каталоге свои рабочие папки "Acrobat3" и "AdobeApp", не спрашивая на это согласия пользователя. На **рис.3** показано рабочее окно программы Acrobat Distiller.

И последнее. В обеих программах есть опция "Downsampling bitmaps". Включение этой опции уменьшает размеры файлов PDF, содержащих графические объекты точечной графики. Это происходит за счет сжатия и уменьшения разрешения растровых графических объектов. Поэкспериментировав с параметрами обеих программ, можно подобрать для себя оптимальные значения размеров файлов PDF и качества изображений.

Рис. 3



В **таблице** указаны версии программ, использованных автором.

Программы	Версии
FinePrint pdfFactory	1.0
Adobe PageMaker	6.5
Acrobat Distiller	3.0

MATLAB. РАБОТА В КОМАНДНОМ РЕЖИМЕ

Б.КИСЕЛЕВ,
г.Минск, БГАТУ, каф.ВТ.

(Окончание. Начало в №6/2003)

Некоторые функции для работы с матрицами

- **eye** — создание единичной матрицы;
- **ones** — создание матрицы из единиц;
- **zeros** — создание матрицы из нулей;
- **linspace** — формирует вектор-строку равноотстоящих узлов;
- **cat** — конкатенация матриц;
- **diag** — создание матриц с заданной диагональю;
- **fliplr** — перестановка столбцов матрицы относительно вертикальной оси;
- **flipud** — перестановка строк матрицы относительно горизонтальной оси;
- **perms** — формирует матрицу из перестановок элементов вектора;
- **prod** — произведение элементов массива;
- **cumprod** — произведение элементов массива с накоплением;
- **sum** — сумма элементов массива;
- **cumsum** — сумма элементов массива с накоплением;
- **repmat** — копирование матрицы;
- **reshape** — выборки из матрицы;
- **rot90** — поворот матрицы на 90°;
- **tril** — выделяет нижнюю треугольную часть матрицы;
- **triu** — выделяет верхнюю треугольную часть матрицы;
- **length** — количество элементов в *одномерном* массиве A;
- **size** — размер массива (например, число строк и число столбцов);
- **ndims** — размерность массива, т.е. количество индексов.

Количество функций достаточно велико, при необходимости можно обратиться к системе Help.

Все перечисленные функции имеют несколько вариантов применения с разным числом аргументов. Далее мы будем описывать лишь требуемые для примеров варианты использования тех или иных функций.

Например, для определенных выше **v** и **ans**:

```
>> d=length(v)
d =
     9

>> s = size(ans)
s =
     3     3

>> n = ndims(ans)
n =
     2
```

Можно использовать вложение функций:

```
d=log((cumprod(1:6))')
d =
         0
    0.6931
    1.7918
    3.1781
    4.7875
    6.5793
```

Здесь **(1:6)** — формирование массива **[1 2 3 4 5 6]**; функция **cumprod** вычисляет факториалы элементов полученного массива — **[1 2 6 24 120 720]**; апостроф обозначает операцию транслонирования, и наконец, вычисляются логарифмы от факториалов, и результат присваивается переменной **d**.

Простейшие операции с векторами и матрицами

Уже из названия системы следует [3], что MATLAB специально предназначена для выполнения операций над матрицами. Даже обычную константу MATLAB рассматривает как матрицу размером (1,1). В этом легко убедиться:

```
>> size(5)
ans =
     1     1
```

В простых вычислениях пользователь не замечает такого представления данных, а в более сложных это необходимо учитывать. В дальнейшем нам такие случаи встретятся.

MATLAB представляет вектор-строку с N элементами как матрицу размером (1,N), а аналогичный вектор-столбец — как матрицу размером (N,1).

Правила действий над векторами и матрицами делятся на две группы:

- правила, которые предусмотрены *векторным исчислением* в математике;
- правила по *преобразованию элементов*, которые не предусмотрены математикой, но удобны для программирования.

Действия первой группы обозначаются так же как действия над обычными математическими объектами. Сюда добавляется знак транспонирования ' и функция обращения матрицы **inv(A)**. Например:

```
>> A = [1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
>> B = A'
B =
     1     4
     2     5
     3     6
>> C = [1 2
        3 4];
>> D = inv(C)
D =
   -2.0000    1.0000
    1.5000   -0.5000
>> G = inv(D)
G =
    1.0000    2.0000
    3.0000    4.0000
>> E=C*D
E =
    1.0000    0
    0.0000    1.0000
```

Здесь матрица D получена обращением матрицы C, а G — обращением D. Видно, что мы получили матрицу, равную исходной. Матрица E получена умножением матрицы C на обратную матрицу C^{-1} (т.е. D). Естественно, что E — единичная матрица.

Операцию обращения матрицы можно записывать в более естественном виде:

```
>> H = C^(-1)
H =
   -2.0000    1.0000
    1.5000   -0.5000
```

Видно, что матрица H совпала с матрицей D.

Для удобства работы в MATLAB добавлены две операции, которых в математике нет: деление матриц слева направо **/**, и деление справа налево ****.

Операция **B/A** эквивалентна действиям **B*inv(A)**, а операция **A\B** эквивалентна действиям **inv(A)*B**.

Эти операции удобны при решении системы линейных уравнений. Например, чтобы решить уравнение

$$A * X = B,$$

можно умножить слева это уравнение на обратную матрицу A^{-1}, и получить решение:

$$X = A^{-1}B.$$

Так, решение системы

$$x_1 + 2x_2 + 3x_3 = 14$$
$$2x_1 - x_2 - 5x_3 = -15$$
$$x_1 - x_2 - x_3 = -4$$

на MATLAB можно записать в следующем виде:

```
>> A=[1 2 3;2 -1 -5;1 -1 -1]
A =
     1     2     3
     2    -1    -5
     1    -1    -1
>> B=[14;-15;-4]
B =
    14
   -15
    -4
>> X=A\B
X =
    1.0000
    2.0000
    3.0000
```

Отметим, что при решении систем линейных уравнений лучше пользоваться записью **X=A\B**, а не **inv(A)*B**, т.к. первая использует алгоритм Гаусса, а вторая — обращение матрицы **A**.

С помощью операции деления можно обращение матрицы A записать как **E/A**, где E — единичная матрица.

Ко второй группе относятся операции поэлементного умножения, деления и возведения в степень. Эти действия обозначаются добавлением точки перед символом операции. Например:

```
>> A = [1 2;3 4]
A =
     1     2
     3     4
>> B = [2 1;3 1]
B =
     2     1
     3     1
>> C = A .* B
C =
     2     2
9     4
```

Сравним с обычным умножением матриц:

```
>> C1=A*B
C1 =
     8     3
    18     7
```

Если взять функцию от матрицы, то получим матрицу функций от ее элементов. Например:

```
>> D = [0    pi/6
       pi/3 pi/2]
D =
         0    0.5236
    1.0472    1.5708
>> F = sin(D)
F =
         0    0.5000
    .8660     1.0000
```

Сохранение и считывание данных сеанса работы (сессии)

Все значения переменных, вычисленные в течение текущего сеанса работы (его обычно называют *сессией*), хранятся в специально зарезервированной области памяти компьютера, называемой *рабочим пространством системы MATLAB* (**Workspace**). Посмотреть, что в нем находится, можно командой **who** или выбором подокна **Workspace** (если подокна нет, его можно открыть, используя **View — Workspace**). Для просмотра значения любой переменной из рабочего пространства, достаточно сделать двойной щелчок левой кнопкой мыши (ЛКМ) на имени переменной или набрать ее имя в командном окне и нажать **Enter.**

Рабочее пространство можно очистить командой **clear,** а для удаления конкретных переменных следует использовать **clear *имя1 имя2 ....***

После окончания работы и выхода из MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить рабочее пространство до следующего сеанса, необходимо: выбрать в меню **File — Save Workspace As...(Файл — Сохранить Рабочее Пространство как...)**. В открывшемся окне **Save to MAT-file (Сохранить в .mat-файл)** в строке **File name (Имя файла)** набрать ***имя файла***, при необходимости выбрать нужную папку и нажать кнопку **Save (Сохранить)**. Рабочее пространство будет сохранено в файле ***имя файла*** с расширением **.mat**.

По умолчанию рабочее пространство сохраняется в *текущей (активной) папке* в файле ***имя файла*** с расширением **.mat**. Обычно текущей папкой является папка **work** системы MATLAB. Чтобы сделать текущей другую папку, нужно воспользоваться раскрывающимся списком **Current Directory (Текущая Директория)** главного окна MATLAB и кнопкой справа от списка.

При поиске файлов пользователя система начинает просмотр с текущей папки, а затем просматривает все папки, указанные в списке путей, известных системе. Доступ к списку осуществляется через меню **File — SetPath...** .

Загрузить рабочее пространство перед продолжением работы можно через меню: **File — Open... (Файл — Открыть...)**.

Упражнения:

1. Ввести матрицу

$$A=\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ pi & 2i & 4j & 2\cdot pi \\ 1+2i & 0 & -3 & pi/2 \end{pmatrix}$$

различными способами: в естественном виде; в виде одной строки; в виде одного столбца; в виде 2×8 и 8×2.

2. Введенную в п.1 матрицу *A* преобразовать в вектор-столбец, затем — в вектор-строку.

3. Убедиться, что после выполнения действий

```
>> A = [1 2;3 4]; B = [4 3;2 1];
   C = [5 6;7 8]; D = [8 7;7 6];
>> F=[A B;C D];
>> F(:,1)=[]; F(:,3)=[]; F(1,:)=[];
F(3,:)=[]
```

получится следующий результат:

```
F =
     4     2
     6     8
```

4. С помощью оператора **":"** создать массив x = [0, 1, 2, 3, 4, 5], затем вычислить cos(x) и $\frac{sin(x)}{x}$. Если во второй функции получится одно число — подумать, в чем дело.

5. В полученном массиве x изменить порядок элементов на обратный и вычислить сумму элементов с накоплением, т.е. $x_k = \sum_{i=1}^{k} x_i$, $k = 1,2,...6$.

Воспользоваться функциями **fliplr** и **cumsum.**

6. Из массива x (п.5) постройте матрицу Вандермонда. Напомним общий вид такой матрицы:

$$W=\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & x_2^3 & \dots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & x_3^3 & \dots & x_3^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & x_n^3 & & x_n^{n-1} \end{pmatrix}.$$

Желательно все сделать одной командой, можно воспользоваться функциями **repmat, cumprod;**

Вопросы для самопроверки

1. Как осуществляется ввод-вывод информации в командное окно?
2. Что такое системные переменные?
3. Как используется оператор "двоеточие"?
4. Две группы операций над векторами и матрицами.
5. Как сохранить рабочее пространство?
6. Когда перед знаком операции ставится точка?

Литература

1 В.Дьяконов, И.Абраменкова. MATLAB 5. Система символьной математики. — М.: Нолидж, 1999, 640 с.

2 В.Потемкин. Система инженерных и научных расчетов MATLAB 5.x (в 2-х томах). — М.: Диалог-МИФИ, 1999.

3 Киселев Б.М. Компьютерная математическая система MATLAB. — Радиомир. Ваш компьютер, 2003, №5, С.2.

ЛИСТАЯ СТРАНИЦЫ

Невсмотря на существенное снижение цен на LCD-мониторы, дисплеи нв электронно-лучевых трубках все вще пользуются спросом, причем достойное место среди них занимают, казалось бы, совсем устаревшие 15-дюймовые модели. Это можно объяснить только очень низкими ценами на такие мониторы при сравнительно высоких качественных показателях. **Результатам тестирования пяти ЭЛТ-мониторов с диагональю 15 дюймов** посвящена статья *В.Дмитриева* "Маленький обзор малвньких экранов" **(Мир ПК, №1/2003, С.12...16).**

Еще несколько лет назад некоторые производители заявляли о грядущем снятии с производства 15-дюймовых ЭЛТ-мониторов. Одни сдержали обещание, другие не спешили с исполнением своих угроз, а место ушедших заняли новые игроки из Кореи и с Тайваня с неизвестными доселе отечественному потребителю именами. Пример тому — марка RoverScan, менее чем за два года с начала выпуска мониторов ставшая чрезвычайно популярной в России. Благодаря отличному соотношению цена/качество, успешно осваивают наш рынок мониторы Hansol и GreenWood.

Хотя доля 15-дюймовых мониторов в общем объеме продаж неуклонно снижается, но до их окончательного исчезновения еще далеко, считает аатор. Что же касается качества, то большинство моделей с новыми именами, как показало тестирование, ни в чем не уступает лучшим изделиям прежних грандов мониторостроения.

Монитор RoverScan 105SF оснащен трубкой Sony FD Trinitron с абсолютно плоским экраном, что обеспечивает достаточно высокое качество изображения. Дисплей имеет семый большой запас по яркости, выдает идеально чистый белый цвет, обладает яркой, насыщенной цветовой палитрой. Его можно назвать чемпионом контрастности. Присутствовавший изначально легкий муар без труда удалось убрать настройками, благо, аппарат оснащен удобным и наглядным меню, предоставляющим весь необходимый набор настроек. Небольшое несведение лучей отмечалось только по краям и в углах экрана. Фокусировка оказалась безупречной по всему полю экрана, за исключением левого нижнего угла, где наблюдалось легкое размывание мелких деталей. К недостаткам изделия можно отнести некоторые огрехи в геометрии экрана, выразившиеся в том, что при тестировании так и не удалось полностью устранить бочкообразные искажения, присутствовавшие при включении монитора.

RoverScan 115GS имеет некоторые особенности, не сразу бросающиеся в глаза. Во-первых, вго глубина на 2...2,5 см меньше, чем у остальных аппаратов обзора. Во-вторых, своеобразные линии передней панели удачно скрадывают выпуклость экрана, создавая в сравнении с другими мониторами с FST-трубкой впечатление более плоской поверхности. Очевидно, модель задумывалась как самый простой и дешевый вариант офисного аппарата. Об этом говорят не только средние технические характеристики, уменьшенная глубина корпуса, но и сокращенное меню настроек. В частности, нет регулировок цветовой гаммы и подавления муара, хотя обе они были бы явно не лишними: легкий муар у протестированного экземпляра все же присутствовал, а цветовая гамма, вполне приличная на одних цветовых шаблонах, на других слегка грешила избытком зеленого. Монитор продемонстрировал неплохой уровень сведения лучей и фокусировки, запас по яркости оказался средним, а огрехи геометрии легко выправлялись.

Монитор GreenWood FD570T оснащен кинескопом с плоским экраном Sony FD Trinitron. Аппарат имеет весьма привлекательный внешний вид, передней панелью напоминая изделия фирмы Sony. Кнопки управления оформлены в виде секторов круга, похожего на джойстик. Модель оснащена очень удобным экранным меню, изюминкой которого является функция i-Video, оптимизирующая работу с мультимедийными приложениями (теплая цветовая гамма, увеличение яркости, развертывание изображения на весь экран), что было особенно важно для предоставленного на тестирование экземпляра, поскольку в обычном режиме он не мог похвастать сколько-нибудь существенным запасом по яркости.

По несведению лучей и фокусировке изделие GreenWood оказалось на том же очень приличном уровне, что и RoverScan 105SF с аналогичной трубкой. Муар и геометрические искажения легко устранились настройками. При отображении большинства тестовых фотографий монитор обеспечивал цветопередачу, очень близкую к натуральной, хотя добиться чистого белого цвета удалось не сразу — при исходной заводской настройке чувствовалось заметное преобладание крвсного.

Дизайн мониторв Hansol 510P — совершенно стандартный, без каких-либо запоминающихся деталей. К сожалению, конструкторы не взяли на себя труд хоть как-то скрыть линиями передней панели выпуклость экрана. Однако, как выяснилось, монитор обладает отличными техническими характеристиками. Он стал лидером обзора по поддерживаемой частоте регенерации во всех режимах разрешения. Сведение лучей тоже оказалось на редкость точным по всей площади экрана, за исключением верхних углов, что, видимо, и вызвало некоторую расфокусировку — текст в этих местах выглядел слегка размытым. Запас по яркости — выше среднего. Имеющиеся в меню настройки для подавления как горизонтального, так и вертикального муара не понадобились, поскольку ни в одном режиме муара не наблюдалось. Не вызвала нареканий и геометрия — бочкообразные искажения легко убирались настройками. А вот что не понравилось, так это посредственное качество цветопередачи. Тестовые изображения выглядели несколько блеклыми, а белый цвет передавался как серо-желтый. Исправить этот недостаток имеющимися настройками, как показал опыт, задача непростая. Тем не менее, если выбирать офисный монитор по соотношению цена/качество, то лидером в данном обзоре, скорее всего, стала бы именно эта модель.

И, наконец, монитор Hyundai ImageQuest V570. Этот аппарат оказался единственным, не вызвавшим нареканий по поводу геометрии изображения. Кроме того, он имеет приличный запас по яркости, великолепное сведение лучей и полное отсутствие муара — тут монитор безоговорочно заслужил твердую оценку "очень хорошо". Близкой к идеалу у модели Hyundai была и точность передачи белого цвета; среди протестированных мониторов по этому показателю ее опередил лишь RoverScan 105SF. А вот яркость и насыщенность цветов оставляют желать лучшего — тестовые изображения на экране выглядели несколько блеклыми и пересаеченными. Увы, подкачала и фокусировка — мелкий шрифт смотрелся чуть более мутным, чем у остальных изделий.

Основные параметры мониторов приведены в твблице.

Модель	Тип ЭЛТ	Шаг пикселов, мм	Диагональ видимой области экрана, дюймов	Кинескоп	Диапазон строчной развертки, кГц	Диапазон кадровой развертки, Гц	Максимальное разрешение/частота кадровой развертки, Гц	Габариты, мм	Масса, кг	Потребляемая мощность, Вт	Ориентировочная цена, USD
GreenWood FD570T	Апертурная решетка	0,24	14	Sony FD Trinitron	30...72	50...130	1280x1024/67	360x372x389	-	90	170
Hansol 510P	Теневая маска	0,28	13,8	Samsung	30...72	50...160	1280x1024/70	371x398x375	12	75	125
Hyundai Image Quest V570	Теневая маска	0,28	13,8	-	30...70	50...150	1280x1024/65	360x377x392	11,7	75	145
RoverScan 105SF	Апертурная решетка	0,24	14	Sony FD Trinitron	30...72	50...130	1280x1024/67	370x334x395	13,6	85	165
RoverScan 115GS	Теневая маска	0,28	13,8	LG-Philips	30...70	50...120	1280x1024/67	382x380x367	12,5	85	125

Новому процессору для мобильных компьютеров посвящена статья *С.Пахомова* "Intel Centrino — новая эра мобильных компьютеров" **(КомпьютерПресс, №3/2003, С.136...141)**.

Современный ноутбук, по мнению авторе, — это прежде всего мобильный компьютер. Поэтому первостепенным фактором является его вес, который напрямую зависит от его размеров. В то же время, желательно, чтобы не слишком большой вес по возможности сочетался с приемлемым размером экрана. Вторым немаловажным фактором является продолжительность автономной работы ноутбука от аккумулятора,

что, по сути, и делает ноутбук мобильным компьютером. Кроме того, ноутбук, опять-таки по соображениям мобильности, должен по возможности обеспечивать соединение с локальной сетью и с Интернетом. И конечно, не последнее место занимает производительность ноутбука.

Объединить эти четыре фактора в одну платформу оказалось не так-то просто. Некоторые из перечисленных требований являются в определенной степени взаимоисключающими. К примеру, продолжительность автономной работы ноутбука от аккумуляторных батарей напрямую зависит от размера ЖК-матрицы и производительности процессора. Таким образом, чем более функциональным и производительным является ноутбук, тем меньше он сможет работать автономно.

Чтобы удовлетворить все возможные потребности пользователей, на протяжении многих лет выпускались ноутбуки нескольких разновидностей. К примеру, для тех, кто предпочитает использовать ноутбуки в основном в качестве стационарного компьютера, когда время автономной работы не является критичным, позиционировались полноразмерные ноутбуки. Они представляют собой полнофункциональные ПК с размером ЖК-матрицы 15 дюймов и более, и отличаются высокой производительностью, поскольку размеры корпуса позволяют создавать необходимую для охлаждения процессора систему теплоотвода. Но подобные ноутбуки трудно назвать мобильными, поскольку вес и длительность автономной работы от батареи не позволяют эффективно использовать их в командировках или вне стационарного рабочего места.

Для той категории пользователей, которые во главу угла ставят именно мобильность ноутбуков, предназначаются тонкие (Slim) и даже сверхтонкие (UltraSlim) ноутбуки. В дизайне таких ноутбуков особое значение имеют вес, размеры и продолжительность автономной работы. Кроме того, подобные ноутбуки, как правило, имеют средства доступа к локальной сети и в Интернет, что обусловлено требованием мобильности. К подобным средствам относятся интегрированные сетевые адаптеры, интегрированные модемы и, реже, беспроводные решения. К недостаткам указанных ноутбуков относятся их подчас не слишком высокая функциональность и слабая производительность. Для решения последней проблемы и с целью увеличения времени автономной работы от батареи компанией Intel были разработаны специальные версии мобильных процессоров, например, Intel Pentium IIIM и Intel Pentium 4M. Основное различие между мобильными процессорами и процессорами для стационарных ПК заключается в средствах управления энергопотреблением, что, в свою очередь, позволяет увеличить продолжительность автономной работы ноутбука на мобильном процессоре.

Использование мобильных версий процессоров отчасти позволяет решить проблему производительности с одновременным увеличением времени работы ноутбука от батареи. Однако комплексного решения проблемы, которое обеспечивало бы реализацию всех четырех векторов развития современных мобильных ПК, до нынешнего года не существовало. И вот, 12 марта 2003 года корпорация Intel представила новую технологию для мобильных ПК — Intel Centrino. Эта технология представляет собой сочетание трех составляющих: процессора Intel Pentium M (ранее известного под кодовым названием Banias), чипсетов Intel 855 (ранее — Odem) и 855GM, известного как Montara-GM, и интегрированного беспроводного решения Intel PRO/Wireless network connection. При этом в новой платформе впервые реализован комплексный подход, позволяющий объединить в одной платформе производительность, жизнеспособность батарей, формфактор и возможность установления связи.

В основе платформы лежит принципиально новый процессор Intel Pentium M (с новой процессорной микроархитектурой). Выполнен процессор по 0,13-микронному техпроцессу и имеет 77 млн. транзисторов. При этом существуют три различных варианта процессора: Intel Pentium M, Intel Pentium M Low Voltage (LW) и Intel Pentium M Ultra Low Voltage (ULV). Различаются эти процессоры напряжением питания и возможными тактовым частотами. Стандартная версия процессора имеет сегодня тактовые частоты 1,3; 1,4; 1,5 и 1,6 ГГц. Процессор Intel Pentium M LW выпускается с тактовой частотой 1,1 ГГц, а процессор Intel Pentium M ULV — с частотой 900 МГц.

Напряжение питания нового процессора составляет от 0,85 до 1,5 В в зависимости от модели, а средняя потребляемая мощность — менее 1 Вт. Отличительной особенностью новой микроархитектуры процессора Intel является сочетание высокой производительности и низкого энергопотребления и, соответственно, тепловыделения.

Кроме пониженного энергопотребления, в новом процессоре реализована поддержка улучшенной технологии Intel SpeedStep и проведена оптимизация мощности с интеллектуальным распределением энергии. Эта технология допускает использование нескольких возможных напряжений питания и частот (рабочих точек), что позаоляет достичь лучшего соотношения напряжение/частота и более эффективного режима функционирования, когда производительность согласуется с рабочей нагрузкой. Крайние рабочие точки процессора задаются аппаратно, а промежуточные — программно.

Управление переходами между различными рабочими точками выполняется только самим процессором и блоком регулятора напряжения (VRM). Для установки требуемого напряжения процессор посылвет служебные последовательности непосредственно в VRM-модуль. При осуществлении перехода между рабочими состояниями процессора никакие другие компоненты системы не используются.

Переход между различными рабочими точками процессора происходит так, чтобы обеспечить работоспособность процессора во время перехода. Для того чтобы произвести переход на более высокую тактовую частоту, сначала, в течение примерно 100 мкс, меняется напряжение процессора до требуемого уровня. При этом частота процессора не меняется. Когда же напряжение достигнет требуемого уровня, происходит скачкообразное увеличение частоты процессора, которое длится около 10 мкс. Если требуется осуществить переход к меньшей частоте, сначала очень быстро, в течение 10 мкс, происходит изменение частоты, а после этого постепенно уменьшается напряжение самого процессора при неизменной частоте.

Архитектура процессора Intel Pentium M объединяет несколько новейших технологий. Во-первых, применяется системная шина с тактовой частотой 400 МГц, использующая технологию оптимизации энергопотребления. Эта технология позволяет снизить энергопотребление процессора. Как правило, процессоры оставляют свою системную шину в рабочем состоянии даже тогда, когда она не используется. При этом значительную долю энергии при использовании шины процессора потребляют усилители считывания. Эти усилители используются на шине данных (64 вывода), стробах данных (8 выводов) и для сигналов инверсии данных (4 вывода). Для сокращения энергопотребления процессор включает усилители считывания только непосредственно при приеме данных, что приводит к существенной экономии энергопотребления.

Во-вторых, используется усовершенствованный метод прогнозирования команд, который включает технологии прогнозирования ветвлений и предвыборки данных. Первая технология обеспечивает увеличение производительности посредством анализа поведения программы в прошлом и прогнозирования будущих операций. В процессоре Intel Pentium M реализованы блоки прогнозирования ветвлений трех типов:

- бимодальный (прогнозирование условных и безусловных переходов);
- локальный (прогнозирование циклов, завершающихся при определенном значении счетчика);
- глобальный (прогнозирование ветвлений путем отслеживания хода выполнения программы).

Для сравнения автор отмечает, что процессоры с микроархитектурой P6 содержали только блок локального прогнозирования.

В результате использования всех блоков удается снизить ошибки прогнозирования более чем на 20%.

Технология предвыборки данных подразумевает анализ кэш-памяти L1 и поиск в потоке запросов тех данных, к которым будет происходить обращение. Процессор отслеживает два типа потоковых операций:

- "поток вверх", то есть переход от низших адресов к высшим;
- "поток вниз", то есть переход от высших адресов к низшим.

При этом удается отследить одновременно до восьми операций типа "поток вверх" и до четырех — "поток вниз".

Процессор Intel Pentium III M, например, отслеживает только операции типа "поток вверх".

В-третьих, используется набор потоковых команд SIMD-расширений второго поколения, оптимизированных для процессора Intel Pentium 4. Это способствует увеличению производительности одновременно с ростом эффективности энергопотребления.

В-четвертых, применяется технология выделенного диспетчера стеков, которая обеспечивается специализированными аппаратными средствами. Традиционно управление стеком осуществляется программно с помощью команд PUSH, POP, RET и CALL. Использование этих команд приводит к непроизводительным затратам. В процессоре Intel Pentium M реализована технология аппаратного управления стеком, в результате чего происходит уменьшение числа микроопераций более чем на 5%.

Весь перечисленный комплекс новых технологий обеспечивает существенный прирост производительности без снижения времени работы от батареи. Другим новшеством этого процессора является увеличенный размер кэшей L1 и L2. Размер кэша L1 инструкций и данных составляет 64 Кб, а размер кэша L2 увеличен до 1 Мб.

Для нового процессора разработан и новый чипсет — Intel 855, который существует в двух вариантах: Intel 855PM и Intel 855GM. Первый предусматривает внешний по отношению к чипсету графический контроллер, а второй имеет интегрированную графическую подсистему.

Контроллеры памяти чипсетов Intel 855PM и Intel 855GM поддерживают работу с памятью DDR266/200 с максимальным объемом до 2 Гб, а взаимодействие с процессором происходит по 400-мегагерцовой шине с пониженным напряжением питания. В качестве южного моста в чипсетах используется хорошо известный хаб ввода-вывода ICH4-M, обеспечивающий стандартные функции: шесть портов USB 2.0, AC'97 v. 2.3, два канала ATA66/100 IDE, а также PCI-шина с частотой 33 МГц.

В наборе микросхем Intel 855 используются средства оптимизации энергопотребления. К таким средствам относятся: сниженное до 1,2 В напряжение питания ядра процессора, управление питанием памяти, управление питанием блока ввода-вывода Intel 855 DDR I/O, управление блоком FSB I/O, поддержка технологии DDR Read Throttling с помощью внешнего датчика температуры, управление вводом-выводом DDR I/O, сокращение питания в C3. Управление питанием памяти DDR подразумевает отключение питания при любой возможности и использование оптимизированного метода управления страницами, при котором количество одновременно открытых страниц сводится к минимуму.

Управление питанием блока ввода-вывода Intel 855 DDR I/O заключается в использовании сигналов управления с тремя дискретными состояниями, неполном запуске сигналов управления (во время циклов ожидания), а также сокращении числа переключений линий управления. Сокращение питания в C3 подразумевает возможность отключения интерфейса концентратора и Host PLL.

Одним из основных компонентов платформы Intel Centrino является модуль беспроводной связи, который интегрируется на PCI-шине (вставляется в разъем Mini PCI) и позволяет работать по стандарту IEEE 802.11b на скорости 11 Мбит/с, или по стандарту IEEE 802.11a на скорости 54 Мбит/с.

С целью увеличения времени работы от батареи, модуль беспроводной связи поддерживает энергосберегающий протокол PSP. Это позволяет использовать пять различных параметров мощности и выбирать наиболее эффективный режим применения батареи. Другим нововведением, реализованным в беспроводном модуле, стали разнесенные антенны. Такой подход позволяет в реальном времени оптимизировать эффективность подключения к сети WLAN. Реализован также режим регулируемого сканирования. Беспроводной модуль позволяет управлять частотой сканирования точек доступа, что отражается на уменьшении потребления мощности и, следовательно, продлевает время автономной работы от батареи.

Если вам необходимо объединить в небольшую локальную сеть несколько компьютеров, рекомендуем ознакомиться со статьей *М.Зверева* "Провода против ..." **(CHIP, №3/2003, С.102...105).**

В настоящее время существует такое многообразие стандартов проводной и беспроводной связи, что трудно понять, какой из них лучше, отмечает автор. Поэтому их сравнение выполняется при условии, что проведение небольшой сети (в доме или офисе) является наиболее простым, удобным и дешевым. При этом возможны два пути: или ваши байты будут бежать по проводам, или передаваться по радиоканалу. В первом случае возникает еще и вопрос: готовы ли вы прокладывать новые провода, или будете использовать уже существующие? На рынке борьба за потребителя ведется в основном между тремя технологиями с той и другой стороны:

- проводные решения — Ethernet (IEEE 802.3), передача информации по телефонным линиям (HomePNA) и передача информации по линиям электропитания (HomePlug, LONWORKS);
- беспроводные технологии — HomeRF, Bluetooth и IEEE 802.11a (IEEE 802.11b).

Стандарт Ethernet был предложен в 1970 г., а IEEE 802.3 впервые был опубликован в 1985 г. Эти стандарты во многом схожи, небольшие различия состоят лишь в том, что Ethernet, в отличие от IEEE, описывает только один физический уровень.

Ethernet является в настоящее время самым распространенным стандартом при построении локальных сетей. По сравнению с другими технологиями, Ethernet-решения имеют ряд преимуществ. Во-первых, это скорость передачи данных. Наиболее распространенные в настоящий момент варианты Ethernet позволяют передавать данные со скоростью до 100 Мбит/с, однако и гигабитный Ethernet постоянно расширяет свое присутствие на рынке, причем не только для офисов, но и для домашних сетей. Во-вторых, дешевизна устройств. Соединение ведется по витой паре, сетевые карты Ethernet также не принесут большого убытка.

Однако Ethernet имеет и недостатки. Первый и самый значительный — необходимость прокладывать новые кабели для создания сети. Это не всегда приемлемо для небольших офисов или уже загруженных проводами домашних хозяйств. Кроме того, для подключения более чем двух устройств потребуется введение в сеть коммутаторов (хабов), что увеличивает стоимость создаваемого решения. Максимальное расстояние между двумя устройствами при создании сети на основе Ethernet не должно превышать 100 м.

При всех плюсах и минусах, по данным аналитической компании IDC, к 2004 г. около 15% рынка домашних сетей будут составлять именно Ethernet-решения.

Сети на основе телефонных линий (HomePNA) — это способ связать компьютеры, бытовую электронику и другую периферию, используя уже имеющиеся в доме кабели. Самый главный плюс данной технологии — отсутствие необходимости прокладывать новые кабели. Довольно часто в разных комнатах дома находится несколько телефонных розеток, соединенных параллельно, причем зачастую они располагаются рядом с компьютером. Это можно использовать. Соответствующий стандарт разрабатывает ассоциация Home Phoneline Networking Alliance (HomePNA). В настоящий момент более 200 компаний представляют продукты HomePNA, такие как USB-адаптеры, мосты Ethernet, роутеры, широкополосные модемы.

Последняя модификация данного стандарта HomePNA 2.0 позволяет передавать данные со скоростью до 10 Мбит/с по телефонным сетям при дальности связи до 350 м. Пробные тесты показали, что технология может быть внедрена в более чем 99% домов, где, естественно, есть телефон.

Главным преимуществом данной технологии является то, что для создания небольшой сети не требуется прокладывать новые кабели. Минусов тоже хватает: затухание сигнала и шум, сама топология телефонной сети, совместное (и не всегда корректное) сосуществование с другим телефонным оборудованием, невысокая производительность и необходимость наличия множества розеток.

Средой передачи информации по стандартам HomePlug и LONWORKS являются электрические линии. Согласно спецификации HomePlug 1.0, принятой в 2001 г., данные по электросетям передаются со скоростью 10 Мбит/с. Плюсы у данной технологии такие: электрические линии — наиболее простой способ для передачи информации, данное решение относительно дешево, его просто реализовать. Минусы также довольно значительны. Домашние линии электропередач изначально не создавались для передачи данных. Следовательно, невозможно передавать информацию по сетям без помех. Физическая топология домашних электросетей, физические возможности электрических ли-

ний, соединенные устройства, шум и помехи в электрических проводах — все вместе создает проблемы в использовании электрических линий в качестве сетевого решения.

Но перспективы данного направления все же не так плохи. Как явствует из исследования Cahners-InStat/MDR, рынок сетевых решений данного типа в 2002 г. достигнет 190 млн. USD, а по прогнозам экспертов, к концу 2006 г. будет достигнут уровень 706 млн. USD.

Технология HomeRF была создана с целью обеспечения дешевого решения для беспроводной передачи голосовых потоков данных и информации. Поддержка передачи голосовой информации обеспечивается стандартом DECT, использующим частоту 2,4 ГГц. Для передачи информации спецификация HomeRF использует упрощенную спецификацию поддержки TCP/IP. Техническая спецификация SWAP (Shared Wireless Access Protocol), описывающая технологию HomeRP, была разработана и оптимизирована для удовлетворения потребительских нужд рыночного сегмента SOHO (Small Office, Home Office) — небольших офисов и домашних хозяйств.

Этой технологией занимается рабочая группа HomeRF, которая была образована для разработки открытого стандарта беспроводной коммуникации между ПК и периферийными устройствами, независимо от того, внутри или снаружи дома они находятся. В результате работы группы, включающей в себя ведущие компании компьютерной индустрии, был разработан стандарт SWAP. Множество несовместимых сетевых стандартов ограничивают распространение сетевых технологий данного направления, поэтому рабочая группа HomeRF рассчитывает на то, что открытая SWAP-спецификация сломает эти барьеры, а также обеспечит гибкость и мобильность беспроводных решений. Насколько данная идея претворится в жизнь — покажет время. А теперь — ее плюсы и минусы. К плюсам относятся:

- изначальная ориентация на сегмент SOHO, так что при создании беспроводных решений учитываются четыре критерия — решения должны быть простыми, защищенными, надежными и финансово доступными;
- помимо довольно высокой скорости передачи данных (согласно стандарту HomeRF 2.0, анонсированному в 2000 г., она составляет 10 Мбит/с), осуществляется поддержка до восьми телефонов стандарта DECT, качественная передача потокового видео и музыки и стандартизированный роуминг;
- решение, объединяющее голосовые, потоковые и цифровые данные, прекрасно подходит для предоставления широкополосных услуг.

Минусов у данной технологии все же меньше, чем плюсов. До введения в жизнь стандарта HomeRF 2.0 его конкуренты основной упор делали на низкую скорость передачи данных (стандарт HomeRF 1.0 позволял передавать данные со скоростью лишь 1,6 Мбит/с). Теперь по скорости HomeRF вполне может бороться за потребителя с IEEE 802.11b, существенно обгоняя Bluetooth. Но хотя стандарт HomeRF 2.0 был принят на вооружение рабочей группой уже в 2000 г., продукты на его основе только начинают появляться, и до серьезного представительства на рынке им еще далеко.

Технология Bluetooth разрабатывается группой Bluetooth SIG, состоящей из лидеров компьютерной и сетевой индустрий. Она позволяет пользователям соединять широкий спектр устройств быстро и просто, без использования кабелей. Изначально Bluetooth создавалась как технология беспроводной связи на малые расстояния, для синхронизации данных между ПК, КПК и мобильными телефонами. Однако в настоящий момент этого оказывается недостаточно, и Bluetooth уже развивается как полноценный беспроводной стандарт связи.

Основные сведения о стандарте Bluetooth: рабочая частота — 2,4 ГГц, дальность приема — до 100 м, скорость передачи данных — до 720 Кбит/с в асимметричном режиме.

Плюсом данной технологии (как и двух ее беспроводных конкурентов) является отсутствие проводов. Главный минус (во всяком случае, на данный момент) — все еще очень малый радиус зоны уверенного приема. Сейчас, несмотря на заявления альянса, разрабатывающего данную технологию, расстояние, на котором вы можете быть уверенным, что ваши данные дойдут до адресата, ограничивается 10 м.

Стандарты IEEE 802.11b (IEEE 802.11a) разрабатывались одной и той же группой разработчиков и были приняты практически одновременно. Самым распространенным из них (поскольку был реализован первым), является старший — IEEE 802.11b. Его основные характеристики — передача данных со скоростью 11 Мбит/с, зона уверенного приема —до 100 м, рабочая частота — 2,4 ГГц. Данный стандарт уже получил свое признание в США и, по большему счету, является в настоящий момент главным и самым распространенным стандартом беспроводной связи. IEEE 802.11a имеет следующие показатели: скорость передачи данных — до 54 Мбит/с, уверенный прием — до 150 м, рабочая частота — 5 ГГц.

Плюсы (для IEEE 802.11b) — обеспечение хорошей (сравнимой с прочими беспроводными форматами) скорости передачи данных плюс относительно невысокая цена. В случае с IEEE 802.11a — более высокая стоимость устройств, зато лидерство по скорости среди беспроводных стандартов.

Подводя итоги, автор приводит несколько цифр, характеризующих перспективы беспроводных технологий. По данным исследований, рынок беспроводных устройств составил около 2 млрд. USD в 2000 г., а по оценкам Allied Business Intelligence, к 2005 г. достигнет 7,1 млрд. USD.

Все приведенное выше позволяет автору сделать вывод о том, что и у проводных технологий, и у связи по радиоканалу есть будущее. Технология, на основании которой вы собираетесь создавать свою локальную сеть, зависит от того, что конкретно вам требуется (высокая скорость, помехозащищенность, низкая стоимость организации сети или существующие коммуникации). К счастью, уже сейчас можно объединять решения в гибридные сети, при развертывании которых используются как проводные, так и беспроводные соединения. Такие сети позволяют гибко конфигурировать устройства и оперативно подключать новые.

Новым винчестерам компании Maxtor емкостью 200 и 250 Гб посвящена заметка "Жесткий диск толщиной 250 Гбайт" **(Мир ПК, №1/2003, С.37)**.

Емкость жестких дисков увеличивается очень быстро. В течение последних нескольких лет количество данных, вмещающихся на одну пластину жесткого диска (поверхностная плотность), удваивалось каждый год. Прошлым летом этот рост немного замедлился — "лишь" на 50% (с 40 до примерно 60 Гб). Со своими четырьмя пластинами и вместимостью 250 Гб жесткий диск компании Maxtor (скорость вращения — 5400 об/мин) являлся на декабрь 2002 г. самым большим по объему. Но Maxtor не остановилась на достигнутом и объявила о выпуске пластин емкостью 80 Гб.

В то время как жесткие диски разбухали в объеме, цены на них стремительно падали. Так, очень большой в свое время (октябрь 1999 г.) Maxtor DiamondMax 6800 емкостью 27,2 Гб стоил 399 USD. А сейчас модель той же фирмы объемом в 250 Гб продается уже за 299 USD, что означает почти вдесятеро большую емкость за сумму, на четверть меньшую.

Конечно, программное обеспечение занимает теперь больше места, чем раньше. Например, для Windows XP Professional требуется целых 1,5 Гб, тогда как Windows 98 SE, которая предустанавливалась на большинство систем три года назад, нуждалась не более чем в 255 Мб. Но даже и на том, по нынешним меркам жалком, винчестере на 27,2 Гб предостаточно места для размещения новейшей ОС компании Microsoft, полного набора приложении и приличного количества цифровой музыки и фотоснимков.

Таблица отражает динамику роста емкости винчестеров и падения цен за 1 Гб.

Месяц и год выпуска	Максимальная емкость, Гб	Средняя цена за 1 Гб, USD
Октябрь 1999	27,2	14,67
Март 2001	75,1	7,06
Март 2002	160	1,88
Ноябрь 2002	250	1,20

Олимпиадные задачи

Уважаемые читатели, приводим решения задач первого тура Витебской областной олимпиады по информатике 2002-2003 гг. среди школьников, условия которых были опубликованы в прошлом номере журнала. Решения задач предоставлены одним из участников олимпиады.

Задача 1. "Икосаэдр"

Решение задачи приведено в виде, наиболее понятном для понимания принципов ее решения, т.е. форматы ввода-вывода выбраны произвольно: входные данные — в виде констант (массив Sos), выходные — выводятся на экран монитора. Текст соответствующих процедур ввода-вывода не приводится (для экономии места), их написание не должно вызвать затруднений. Сделайте чертеж — вам сразу станет намного легче.

Текст решения:

```
Program Icosaedr;
Const Sos:Array[1..20,1..3] of integer=(...);
Var
   Svob      : array[1..kg] of boolean;
   Put       : array[1..kg] of integer;
   k,Sum,Min : integer;

procedure go (gl,n:integer);
Var Kon,nap,h:integer;
Begin
  For nap:=1 to 3 do
  begin
    Kon:=Sos[n,nap];
    if Svob[Kon] and (Sum<Min) then
    begin
      Put[gl+1]:=Kon;
      Svob[Kon]:=False;
      Sum:=Sum+1+(kg-gl)*Kon;
      if gl=kg-1 then
        begin
          for h:=1 to kg do write (Put[h]:3);
          Writeln (Sum:10);
          If Sum<Min then Min:=Sum
        end
      else go(gl+1,Kon);
      Sum:=Sum-1-(kg-gl)*Kon;
      Svob[Kon]:=true
    end
  end
end;

begin
  Min:=Maxint;
  for k:=1 to kg do Svob[k]:=True;
  for k:=1 to kg do
  begin
    Put[1]:=k;
    Svob[k]:=False;
    Sum:=kg*k;
    Go(1,k);
    Svob[k]:=true
  end;
  writeln('Минимальное время прохождения маршрута =',Min);
  readln
end.
```

Задача 2. "Арифметическая прогрессия"

Данная задача является самой простой из предлагавшихся. Думаю, что ее понимание ни у кого не вызовет трудностей. Привожу свой "полевой" вариант решения, поэтому не удивляйтесь, если найдете большие изъяны.

Текст решения:

```
program arifm_progr;
label no;
var
  n,m:word;
  d,count,pr,fir:longint;
  progr:array[1..10] of longint;

function prostoe(x:longint):boolean;
var
  i:longint;
begin
  prostoe:=true;
  if x=1 then begin prostoe:=false;exit;end;
  i:=2;
  while (i<trunc(sqrt(x))+1) and (x mod i<>0) do inc(i);
  if x mod i=0 then prostoe:=false
end;

begin
  readln(n);
  d:=1;
  count:=0;
  fir:=1;
  d:=1;
  while count<>n do
  begin
    d:=1;
    while (d<1000) and (count<>n) do
    begin
      pr:=fir;
      if prostoe(pr) then
                     begin
                       count:=1;
                       progr[1]:=pr
                     end
                     else count:=0;
      for m:=(count+1) to n do
      begin
        pr:=pr+d;
        if prostoe(pr) then
                       begin
                         inc(count);
                         progr[m]:=pr
                       end
                       else goto no;
      end;
no:   inc(d);
    end;
    inc(fir);
  end;
  for m:=1 to n do writeln(progr[m])
end.
```

Задача 3. "Мотоциклисты"

По-моему, эта задача является самой сложной из предлагавшихся и, в то же время, имеет некоторую неоднозначность решения. Организаторы олимпиады выдали задания, через некоторое время предложили участникам исправить значение в примере выходного файла, а в итоге вышло, что у них имеются только входные данные тестов. Но, если вы хорошенько поработаете над разборкой предложенного решения и вникнете в саму суть задачи, то поймете, что алгоритм решения —проще некуда (просто реализация получилась довольно объемной). Правда, на все это требуется много времени, а в условиях олимпиады...

Текст решения:

```
Program moto;
const InputFile='Moto.in';
      OutputFile='Moto.out';
      Eps=1e-5;
      MaxN=100;
var N:integer;
    Rs,WhRd:Array[0..MaxN+1] Of extended;
    Can: boolean;
    ResMax: extended;

function More(a, b:extended):boolean;
begin
  More:=(a-b)>Eps;
end;

function Eq(a, b:extended):boolean;
```

```
begin
  Eq:=Abs(a-b)<Eps;
end;

function IfPoss(Const k: Integer; Const Tm: extended): boolean;
var d:extended;
begin
  d:=(Tm/WhRd[k]);
  IfPoss:=Eq(d, Round(d)) And Eq(Frac(Int(d)/2), 0.5);
end;

function IfMay(Const R: extended): boolean;
var i: integer;
Begin
  IfMay:=False;
  For i:=1 To N Do If Not IfPoss(i, Rs[i]/R) Then Exit;
  IfMay:=True;
End;

Procedure Init;
Var i: Integer;
Begin
  ResMax:=-MaxLongInt;
  Assign(Input, InputFile);
  Reset(Input);
  Read(N);
  For i:=1 To N+1 Do
  Begin
   Read(Rs[i]);
   If i>1 Then Rs[i]:=Rs[i] + Rs[i-1];
  End;
  For i:=1 To N Do Read(WhRd[i]);
  Close(Input);
End;

procedure Done;
begin
  Assign(Output, OutputFile);
  Rewrite(Output);
  If Can Then WriteLn(ResMax:0:4)
  Else WriteLn("NO");
  Close(Output)
end;

procedure TryFind(k: Integer);
var i:integer;
    R:extended;
begin
  i:=1;
  R:=Rs[k]/(WhRd[k]*i);
  While (Not More(5, R)) And More(R, ResMax) do
  begin
    If IfMay(R) then
    begin
      ResMax:=R;
      Can:=True;
      Exit
    end;
    Inc(i,2);
    R:=Rs[k]/(WhRd[k]*i);
  end;
end;

procedure Solve;
var i: integer;
begin
  Can:=False;
  for i:=1 to N do TryFind(i);
end;

begin
  Init;
  Solve;
  Done
end.
```

От редакции: *для экономии места в журнале, тесты к задачам вынесены на веб-страницу журнала http://radiomir.com (файл test.zip).*

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

А.КОРБИТ, А.ЩЕРБАКОВ,
г.Минск, БГУИР.

Программирование в среде Visual C++ 6.0

Элементы управления ListBox

Цель этой статьи — научиться использовать элемент управления ListBox, основные методы класса CListBox, а также возможность контроля правильности ввода исходных данных.

1. Класс CListBox

Класс CListBox обеспечивает функционирование окна списка. В окне списка могут отображаться различные элементы, которые пользователь может просматривать и выделять. Окно списка имеет два режима выделения элементов:

- единичный выбор, т.е. может быть выделен только один элемент списка;
- множественный выбор, т.е. в списке одновременно могут быть выделены несколько элементов.

Окно списка может содержать как один столбец, так и несколько столбцов с элементами.

Рассмотрим основные методы данного класса.

Для добавления строки в список, в классе CListBox объявляется функция

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

Здесь параметр lpszItem — указатель на нуль-терминированную строку, добавляемую в список. Данная функция возвращает номер строки в списке, перед которой добавлялся текст, причем первая строка имеет нулевой индекс.

Для очистки всего списка объявляется функция

```
void ResetContent( );
```

Эта функция не получает и не возвращает параметры.

Кроме того, класс CListBox имеет следующие методы:

- `int GetCount() const;` — при успешном выполнении эта функция возвращает количество элементов в окне списка;
- `void SetColumn(int Wingth);` — устанавливает ширину всех колонок списка;
- `int GetCutSel();` — возвращает индекс выделенного элемента списка;
- `int DeleteString(UINT n);` — удаляет строку с индексом n;
- `int InsertString(int n, LPCTSTR lpszItem);` — вставляет строку в позицию списка, указанную индексом n. Если значение n равно -1, то строка будет добавлена в конец списка.

В случае, если в процессе работы функций возникла ошибка, то возвращается величина LB_ERR; если эта ошибка была связана с нехваткой памяти для хранения новой строки, то возвращается величина LB_ERRSPASE.

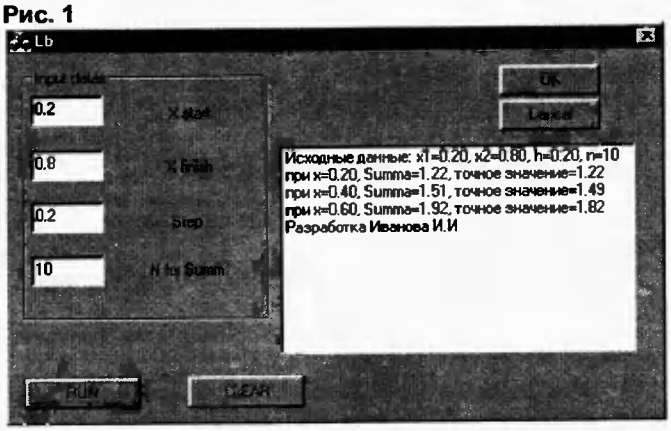
Пример написания программы

Необходимо написать и отладить программу, которая выводит таблицу функций

$$s(x) = \sum_{i=0}^{n} x^i / i! \text{ и } Y(x) = \exp(x)$$

для x, изменяющихся от x1 до x2 с шагом h. Программа должна выглядеть, как представлено на **рис.1**.

Рис. 1



Для решения задачи следует выполнить следующие действия.

1. Создать при помощи MFCAppWizard(exe) новый проект типа Dialog based (предварительно выбрав для него путь и имя — например, Lb).

2. Используя панель конструктора ресурсов "Controls", нанести на панель Dialog based по четыре элемента типа EditBox и StaticText (объединив их элементом GroupBox), два элемента Buttons и один элемент ListBox для вывода результатов. Установите нужные свойства и надписи.

3. При помощи ClassWizard через закладку Member Variables свяжите поле Edit1 с переменной m_x1. Для этого нужно выполнить команду ClassWizard/Member Variables..., далее сделать щелчок на IDC_EDIT1, затем щелчок на Add Variable... и в появившейся закладке в поле ...name: m_x1 установить тип double. Подтвердите изменения щелчком на OK и установите диапазон значений: Minimum Value — 0,1; Miaximum Value — 0.3.

По аналогии увяжите поле Edit2 с переменной m_x2 типа double, диапазон — 0,7...-0,9; поле Edit3 с переменной m_h типа double, диапазон — 0,1...-0,2; поле Edit4 с переменной m_n типа int, диапазон — 5...-20.

Элемент ListBox свяжите с переменной m_l. Для этого необходимо назначить Category: Control, установив тем самым для данного объекта тип **CListBox**. Теперь для данного объекта можно вызывать методы по всей цепи иерархии данного класса MFC. Так, для стирания окна вывода в обработчике щелчка на кнопке CLEAR можно вызвать метод **m_l.ResetContent()**.

В обработчик щелчка на кнопке RUN впишите текст программы решения поставленной задачи. Используйте циклические алгоритмы (внешний цикл — по x, а внутренний — по параметру i для вычисления суммы текущего алгоритма) для введенных в класс диалога переменных m_x1, m_x2, m_h и m_n.

Для вывода информации о разработчике и введенных исходных данных воспользуйтесь методом класса ListBox — AddString:

```
UpdateData(TRUE);
str.Format("Разработка Иванова И.И");
m_l.AddString(str);
UpdateData(FALSE);
str.Format("Исходные данные: x1=%.2lf, x2=%.2lf, h=%.2lf,
n=%2d", m_x1,m_x2,m_h,m_n);
m_l.AddString(str);
UpdateData(FALSE);
```

Участок вывода текущей строки таблицы в окно результатов может быть таким:

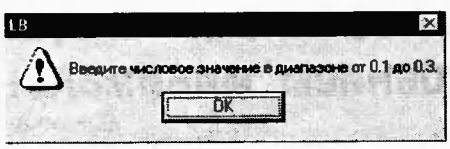
```
str.Format("при x=%.2lf, Summa=%.2lf, точное значение=%.2lf",
x,s,y);
m_l.AddString(str);
UpdateData(FALSE);
```

4. Находясь в режиме конструктора диалога, командой Layout/Tab Order установите очередность перехода между элементами управления:

1 — Edit1; 2 — Edit2; 3 — Edit3; 4 — Edit4; 5 — Button RUN; 6 — ListBox; 7 — Button CLEAR; 8 — Button OK; 9 — Button Cancel, а для кнопки RUN установите свойство Default.

5. Проверьте работу программы. Для этого запустите проект, и в появившемся диалоговом окне щелкните на OK. Появится MessageBox с подсказкой о диапазоне вводимого x1 (**рис.2**). Щелкните OK, введите начальное значение аргумента **x** (поле с подсветкой) и опять щелкните на OK. По аналогии, введите значения для x2, h, n. Для выполнения задания щелкните на RUN. Щелчок на кнопке CLEAR стирает значения в окне вывода.

Рис. 2



3. Индивидуальные задания

По аналогии с рассмотренным примером, необходимо написать и отладить программу, которая выводит таблицы значений функций в соответствии с предложенными вариантами.

1. $S(x)=\sum_{k=0}^{n}\frac{\ln^k 3}{k!}x^k,\ Y(x)=3^x-1.$

2. $S(x)=\sum_{k=1}^{n}\frac{\cos(kx)}{k},\ Y(x)=-\ln\left|2\sin\frac{x}{2}\right|.$

3. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k+1}}{(2k+1)!},\ Y(x)=\sin(x).$

4. $S(x)=\sum_{k=0}^{n}\frac{x^k}{k!},\ Y(x)=e^x.$

5. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{\sin(kx)}{k},\ Y(x)=\frac{x}{2}.$

6. $S(x)=\sum_{k=0}^{n}\frac{\cos\left(\frac{k\pi}{4}\right)}{k!}x^k,\ Y(x)=e^{x\cos\frac{\pi}{4}}\cos\left(x\sin\left(\frac{\pi}{4}\right)\right).$

7. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k}}{(2k)!},\ Y(x)=\cos(x).$

8. $S(x)=\sum_{k=1}^{n}x^k\sin^k\left(\frac{\pi}{4}\right),\ Y(x)=\frac{\left(x\sin\left(\frac{\pi}{4}\right)\right)}{1-2x\cos\left(\frac{\pi}{4}\right)+x^2}.$

9. $S(x)=\sum_{k=0}^{n}\frac{x^{4k+1}}{4k+1},\ Y(x)=\frac{1}{4}\ln\frac{1+x}{1-x}+\frac{1}{2}arctgx.$

10. $S(x)=\sum_{k=0}^{n}\frac{\cos(kx)}{k!}, Y(x)=e^{\cos x}\cos(\sin(x))$.

11. $S(x)=\sum_{k=0}^{n}\frac{2k+1}{k!}x^{2k}, Y(x)=\left(1+2x^2\right)e^{x^2}$.

12. $S(x)=\sum_{k=1}^{n}\frac{x^k\cos\frac{k\pi}{3}}{k}, Y(x)=-\frac{1}{2}\ln\left(1-2x\cos\frac{\pi}{3}+x^2\right)$.

13. $S(x)=\sum_{k=0}^{n}\frac{1}{2k+1}\left(\frac{X-1}{X+1}\right)^{2K+1}, Y(x)=\frac{1}{2}\ln(x)$.

14. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{\cos(kx)}{k^2}, Y(x)=\frac{1}{4}\left(x^2-\frac{\pi^2}{3}\right)$.

15. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k+1}}{4k^2-1}, Y(x)=\frac{1+x^2}{2}arctg(x)-\frac{x}{2}$.

Литература

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

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

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

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

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

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

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

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

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

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

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

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

Решение задач на графах построением минимального остовного дерева

М.ДОЛИНСКИЙ, г.Гомель.

(Окончание. Начало в №№5-6/2003)

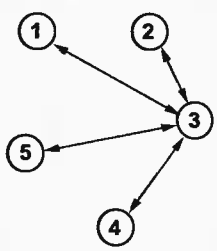
6. Задача "Метро"

(Белорусская республиканская олимпиада 2002 г.)

В некотором городе есть метро, состоящее из N ($1\le N\le 1000$) станций и M ($0\le M\le 500000$) линий, их соединяющих. Каждая линия обеспечивает проезд между какими-то двумя станциями в обе стороны. Между любой парой станций проведено не более одной линии. Сеть метро построена таким образом, чтобы с каждой станции можно было проехать на каждую (возможно, через промежуточные станции). Назовем это свойство связанностью метро.

В связи с изобретением принципиально нового вида транспорта, метро стало убыточным, и его работу решили прекратить. На заседании мэрии города постановили закрывать каждый год по одной станции, но так, чтобы связанность метро каждый раз сохранялась. При закрытии какой-либо станции, линии, ведущие от этой станции к другим, естественно, тоже перестают функционировать.

Задание



По введенной информации о сети метро разработать какой-либо порядок закрытия станций, при котором метро всегда будет оставаться связанным. Например, пусть метро выглядит так, как показано **на рисунке**. Тогда станции можно закрывать, например, в порядке 1, 2, 4, 3, 5. А порядок 3, 1, 2, 4, 5 не подходит, так как после закрытия 3-й станции, метро распадется на четыре не связанные между собой части.

Ввод

Первая строка входного файла будет содержать числа N и M. В следующих M строках находится информация о линиях. Каждая из этих строк содержит через пробел числа Ai и Bi — номера двух станций, которые соединяет i-я линия.

Вывод

Выходной файл должен состоять из N строк. Каждая строка должна содержать одно число — номер станции. Вывести станции нужно в порядке их закрытия.

Пример:

Input.txt	Output.txt
5 4	1
3 1	2
3 2	4
3 4	3
3 5	5

Естественно представить станции метро вершинами графа, а линии их соединяющие — ребрами графа. По условиям задачи длины линий не имеют значения, поэтому мы можем задать значения весов всех ребер равными 1. В целом, ввод исходных данных осуществляется следующим образом:

```
procedure InputData;
  begin
    assign (input,'input.txt'); reset(input);
    readln(N,P);
    for i:=1 to P do
      begin
        readln(B[i],E[i]);
        R[i]:=1;
      end;
    close(input);
  end;
```

Теперь нам достаточно найти остовное дерево (минимальность не требуется, и сортировку вызывать нет необходимости), а затем обойти исходный граф поиском в глубину, используя только ребра, включенные в остовное дерево.

Тело главной программы будет выглядеть следующим образом:

```
begin
  InputData;
  MinTree_Kruskal;
  assign (output,'output.txt'); rewrite(output);
```

```
  for i:=1 to N do Color[i]:=FREE;
  DFS(1);
  close(output);
end.
```

Здесь MinTree_Kruskal — процедура построения остовного дерева, сохраняющая найденный подграф с помощью двух массивов:

- kG[U] — массив количества ребер, связанных с вершиной U;
- G[u,j]=v — массив дуг (из вершины U дуга с номером j ведет в вершину V).

Текст процедуры:

```
procedure MinTree_Kruskal;
  begin
    for i:=1 to N do kG[i]:=0;
    for i:=1 to N do Makeset(i);
    for i:=1 to P do
      if (Findset(B[i])<>FindSet(E[i]))
        then
          begin
            inc(kG[B[i]]); G[B[i],kG[B[i]]]:=E[i];
            inc(kG[E[i]]); G[E[i],kG[E[i]]]:=B[i];
            Union(B[i],E[i]);
          end;
  end;
```

Массив Color[i] обеспечивает работу процедуры DFS — обход графа в глубину и вывод номеров вершин в порядке "от листьев к корням":

```
procedure DFS(U:longint);
  var
    j : longint;
  begin
    color[U]:=USED;
    for j:=1 to kG[U] do
      if Color[G[u,j]]=FREE then DFS(G[U,j]);
    writeln(U);
  end;
```

Неформально поиск в глубину с помощью массива Color можно пояснить следующим образом.

При заходе в очередную вершину мы выясняем, есть ли из нее дуга к неокрашенным (непосещенным) вершинам. Если есть — идем туда.

Если же у текущей вершины больше нет "потомков", значит, это "листовая" вершина, и ее можно выводить в результат.

Полный текст решения задачи "Метро" имеет вид:

```
program us02ope2;
Const
  MaxN =   1000   ;
  MaxP =  500000  ;
  MaxShortInt = 32767;
  Free =0;
  Used =1;
var
  N                          : 1..MaxN;
  P                          : 1..MaxP;
  i,j                        : integer;

  Pred,Rank,Color    : array [1..MaxN]   of integer;
  B, E               : array [1..MaxP]   of integer;

  R   : array[1..MaxP] of integer;

  kG  : array[1..MaxN] of integer;
  G   : array[1..MaxN,1..MaxN] of integer;

procedure InputData;
  begin
    assign (input,'input.txt'); reset(input);
    readln(N,P);
    for i:=1 to P do
      begin
        readln(B[i],E[i]);
        R[i]:=1;
      end;
    close(input);
  end;

  procedure MakeSet(x:longint);
    begin
      Pred[x]:=x;
      rank[x]:=0;
    end;

  function FindSet(x:longint):longint;
    begin
      if x<>pred[x]
        then Pred[x]:=FindSet(pred[x]);
      FindSet:=Pred[x];
    end;

  Procedure Link(x,y:longint);
    begin
      if rank[x]>rank[y]
        then pred[y]:=x
        else
          begin
            pred[x]:=y;
            if rank[x]=rank[y]
              then inc(rank[y]);
          end;
    end;

  Procedure Union(x,y:longint);
    begin
      Link (FindSet(x),FindSet(y));
    end;

procedure MinTree_Kruskal;
  begin
    for i:=1 to N do kG[i]:=0;
    for i:=1 to N do Makeset(i);
    for i:=1 to P do
      if (Findset(B[i])<>FindSet(E[i]))
        then
          begin
            inc(kG[B[i]]); G[B[i],kG[B[i]]]:=E[i];
            inc(kG[E[i]]); G[E[i],kG[E[i]]]:=B[i];
            Union(B[i],E[i]);
          end;
  end;

  Procedure DFS(U:longint);
    var
      j : longint;
    begin
      color[U]:=USED;
      for j:=1 to kG[U] do
        if Color[G[u,j]]=FREE then DFS(G[U,j]);
      writeln(U);
    end;

begin
  InputData;
  MinTree_Kruskal;
  assign (output,'output.txt'); rewrite(output);
  for i:=1 to N do Color[i]:=FREE;
  DFS(1);
  close(output);
end.
```

7. Задача "Network"

(ACM ICPC 2001. NorthEastern European Region, Northern Subregion)

По условиям соревнований студенческого ACM первенства мира по программированию, условия задач на всех стадиях соревнований (от четвертьфиналов до финалов) предъявляются участникам на английском языке. Ниже будет приведена математическая постановка задачи.

Andrew is working as system administrator and is planning to establish a new network in his company. There will be N

hubs in the company, they can be connected to each other using cables.

Since each worker of the company must have access to the whole network, each hub must be accessible by cables from any other hub (with possibly some intermediate hubs). Since cables of different types are available and shorter ones are cheaper, it is necessary to make such a plan of hub connection, that the maximum length of a single cable is minimal. There is another problem — not each hub can be connected to any other one because of compatibility problems and building geometry limitations. Of course, Andrew will provide you all necessary information about possible hub connections.

You are to help Andrew to find the way to connect hubs so that all above conditions are satisfied.

Input data:

The first line of the input file contains two integer numbers: N — the number of hubs in the network ($2 \le N \le 1000$) and M — the number of possible hub connections ($1 \le M \le 15000$). All hubs are numbered from 1 to N. The following M lines contain information about possible connections — the numbers of two hubs, which can be connected and the cable length required to connect them. Length is a positive integer number that does not exceed 10^6. There will be no more than one way to connect two hubs. A hub cannot be connected to itself. There will always be at least one way to connect all hubs.

Output data:

Output first the maximum length of a single cable in your hub connection plan (the value you should minimize). Then output your plan: first output P — the number of cables used, then output P pairs of integer numbers — numbers of hubs connected by the corresponding cable. Separate numbers by spaces and/or line breaks.

Example:

Input.txt	Output.txt
4 6	1
1 2 1	4
1 3 1	1 2
1 4 2	1 3
2 3 1	2 3
3 4 1	3 4
2 4 1	

Математическая постановка задачи сводится к следующему.

Для заданного взвешенного графа требуется найти такое остовное дерево, чтобы максимальный из весов ребер этого остовного дерева был минимальным, и вывести этот вес, количество ребер в остовном дереве и сами ребра (номера вершин, соединенных соответствующим ребром).

По построению минимального остовного дерева алгоритмом Крускала получается, что оно всегда является и решением поставленной задачи. Таким образом, решение задачи сводится к нахождению минимального остовного дерева алгоритмом Крускала, а затем к нахождению максимального по весу из ребер, включенных этим алгоритмом в минимальное остовное дерево.

Рассмотрим более подробно некоторые детали реализации.

Поскольку количество вершин (до 1000) и количество ребер (до 15000) настолько велики , что требуемые алгоритму Крускала структуры данных не поместятся в статической памяти Турбо-Паскаля, массивы начал, концов и весов ребер (B, E и R соответственно) размещаются в динамической памяти. Тело главной программы выглядит следующим образом:

```
begin
  InputData;        {Ввод исходных данных}
  QuickSortR(1,M); {Сортировка ребер по возрастанию весов}
  MinTree_Kruskal; {Построение минимального остовного дерева}
  Max:=R^[Key[1]]; {Нахождение максимального ребра в ост.дереве}
  for i:=1 to j do
    if R^[Key[i]]>Max then Max:=R^[Key[i]];
  OutputData;       {Вывод результата в требуемом формате }
end.
```

Процедуры QuickSortR, MinTree_Kruskal подвергнуты минимальным изменениям, вызванным переходом от использования статических массивов B, E, R к динамическим — B^, E^, R^.

Полный текст решения задачи:

```
program n401d1t3;
Const
  MaxN =   1000   ;
  MaxP =  15000   ;
Type
  TMas = array [1..MaxP]   of longint;
  PMas = ^TMas;
var
  N                          : 1..MaxN;
  M                          : 1..MaxP;
  i,j                        : integer;

  Key               : array [1..MaxN-1] of integer;
  Pred,Rank          : array [1..MaxN]   of integer;
  B, E, R            : PMas;

  Max                : longint;

procedure InputData;
  begin
    assign (input,'input.txt'); reset(input);
    readln(N,M);
    B:=New(PMas); E:=New(PMas); R:=New(PMas);
    for i:=1 to M do readln(B^[i],E^[i],R^[i]);
    close(input);
  end;

  function Partition(p,rr:longint):longint;
    var
      i,j,z : longint;
      x,y   : longint;
    begin
      x := R^[p];
      i:=p-1;
      j:=rr+1;
      while i<j do
        begin
          repeat j:=j-1 until R^[j]<=x;
          repeat i:=i+1 until R^[i]>=x;
          if i<j
            then
              begin
                y:=R^[i]; R^[i]:=R^[j]; R^[j]:=y;
                z:=B^[i]; B^[i]:=B^[j]; B^[j]:=z;
                z:=E^[i]; E^[i]:=E^[j]; E^[j]:=z;
              end;
        end;
      Partition:=j;
    end;

  procedure QuicksortR(p,rr:longint);
    var
      q : longint;
    begin
      if p<rr
        then
          begin
            q:= Partition(p,rr);
            QuicksortR(p,q);
            QuicksortR(q+1,rr);
          end;
    end;

  procedure MakeSet(x:longint);
```

```
  begin
    Pred[x]:=x;
    rank[x]:=0;
  end;

function FindSet(x:longint):longint;
  begin
    if x<>pred[x]
      then Pred[x]:=FindSet(pred[x]);
    FindSet:=Pred[x];
  end;

Procedure Link(x,y:longint);
  begin
    if rank[x]>rank[y]
      then pred[y]:=x
      else
        begin
          pred[x]:=y;
          if rank[x]=rank[y]
            then inc(rank[y]);
        end;
  end;

Procedure Union(x,y:longint);
  begin
    Link (FindSet(x),FindSet(y));
  end;

procedure MinTree_Kruskal;
  begin
    j:=0;
    for i:=1 to N do Makeset(i);
    i:=1;
    While J<>N-1 do
      begin
        if (Findset(B^[i])<>FindSet(E^[i]))
          then
            begin
              inc(j); Key[j]:=i;
              Union(B^[i],E^[i]);
            end;
        inc(i);
      end;
  end;

Procedure OutputData;
  begin
    assign (output,'output.txt'); rewrite(output);
    writeln(Max);
    writeln(J);
    for i:=1 to J do writeln (B^[Key[i]],' ',E^[Key[i]]);
    close(output);
  end;

begin
  InputData;
  QuickSortR(1,M);
  MinTree_Kruskal;
  Max:=R^[Key[1]];
  for i:=1 to j do
    if R^[Key[i]]>Max then Max:=R^[Key[i]];
  OutputData;
end.
```

8. Задача "Earthquake"

(USACO Open 2001, Green Division)

Землетрясение разрушило ферму фермера Джона, и он решил отстроить ее заново. Построив все N полей (1≤N≤400), он решил соединить их дорожками так, чтобы всегда имелся путь от любого поля к любому.

После изучения местности ФД пришел к выводу, что могут быть построены M (1≤M≤10000) двунаправленных дорожек. Он хотел бы завершить проект с минимальными финансовыми затратами.

Коровы создали инженерную консультативную фирму, которая специализируется на восстановлении дорог, разрушенных землетрясениями. Известно, что коровы умные, и берутся только за работу, которая принесет им прибыль. Они договариваются о цене F (1≤F≤2000000000), которую им заплатят за восстановление дорог. Им предоставляется таблица возможных дорог, время (в часах) на построение каждой дороги (1≤t≤2000000000) и стоимость построения каждой дороги (1≤c≤2000000000). Таблица может содержать более одной дороги, соединяющей одну и ту же пару полей. Всегда есть возможноть соединить все поля по заданным входным данным, хотя это может быть и не выгодно.

Определите значение максимальной прибыли, которую может принести восстановление дорог на ферме.

Формат ввода:

Строка1: три целых числа — N, M и F.

Строки 2...M+1: каждая строка содержит четыре целых числа, разделенных пробелом, (i j c t), соответственно обозначающих два поля, которые соединяет дорога; стоимость и время, требуемые для восстановления дороги.

Пример ввода:

```
5 5 100
1 2 20 5
1 3 20 5
1 4 20 5
1 5 20 5
2 3 23 1
```

Формат вывода:

Одна строка, содержащая одно число с четырьмя десятичными позициями после запятой — максимальная прибыль в час, которую коровы могут получить. Если прибыль не положительная, выводите 0,0000.

Пример вывода:

```
1.0625
```

Пояснения:

Коровы подрядились восстановить четыре последние дороги. Суммарные затраты на востановление — 83, и время на востановление — 16 часов. Оплата равна 100, поэтому коровы получат прибыль 100-83 за 16 часов, т.е. прибыль в час — 17/16=1,0625.

Поскольку в условии задачи сказано, что фермер хочет обеспечить связность всех полей, то очевидно, что придется строить остовное дерево. На первый взгляд кажется, что эта задача тоже на построение минимального остовного дерева. К сожалению, оказывается, что это не совсем так. Дело в том, что формула для расчета максимальной прибыли в час, которую могут получить коровы:

$$Max=(F-S)/T,$$

где:

F — фиксированная оплата фермера коровам;

S — затраты на восстановление всех дорог;

T — время, потраченное на восстановление всех дорог;

не позволяет просто вычислить фиксированный вес ребра!!!

Вес ребра (выгодность востановления конкретной дороги) зависит от того, какие ребра уже включены в остов. Более того, ценность ребер, которые добавлены в остов, **изменяется** после добавления в остов новых ребер.

Таким образом, непосредственное применение методов Прима или Крускала не дает оптимальных результатов. Как же решать эту задачу?

Оставляем это вопрос в качестве самостоятельной работы для вдумчивых читателей. Лучшие решения будут опубликованы!

Заключение

Данная статья продолжает серию публикаций [1...13], ориентированных на самообучение программированию и подготовку к областным и республиканским олимпиадам по информатике. Очевидно, что для закрепления данного материала необходимо проверить полученные знания на компьютере как самостоятельным решением поставленных в данном материале задач, так и применением полученных знаний к другим задачам. Удобно пользоваться сайтом http://dl.gsu.unibel.by для контроля правильности решения задач.

Полезно также ознакомиться с соответствующими материалами учебных пособий, например, [14-17].

Литература

1. М.Долинский. Памятка участнику олимпиады по информатике среди школьников. — Радиолюбитель. Ваш компьютер, 2000, №1, С.20.
2. М.Долинский. Начинаем программировать самостоятельно. — Радиолюбитель. Ваш компьютер, 2000, №№1...6.
3. М.Долинский. Решение задач на тему "Геометрия на плоскости" — Радиолюбитель. Ваш компьютер, 2000, №№8, 9.
4. М.Долинский. Разработка программ с использованием определяемых программистом типов данных. — Радиолюбитель. Ваш компьютер, 2001, №№1, 3.
5. М.Долинский. Решение задач с помощью очереди и стека. — Радиолюбитель. Ваш компьютер, 2001, №№4...6.
6. М.Долинский. Обзор возможностей языка программирования Паскаль. — Радиомир. Ваш компьютер", 2001, №7, С.23.
7. М.Долинский. Алгоритмы генерации комбинаторных объектов. — Радиомир. Ваш компьютер, 2001, №№10, 11.
8. М.Долинский. Некоторые факты из теории чисел. — Радиомир. Ваш компьютер, 2001, №9, С.20.
9. М.Долинский. Использование рекурсивных функций и процедур. — Радиомир. Ваш компьютер, 2002, №№1, 2.
10. М.Долинский. Решение задач с помощью рекуррентных соотношений. — Радиомир. Ваш компьютер, 2002, №№3...6.
11. М.Долинский. Решение задач на графах. — Радиомир. Ваш компьютер, 2002, №№7-11.
12. М.Долинский. Использование динамической памяти в прикладных программах. — Радиомир. Ваш компьютер, 2002, №12; 2003, №1.
13. М. Долинский. Решение задач методом дихотомии. — Радиомир. Ваш компьютер, 2003, №№1-3.
14. Котов В.М., Волков И.А., Харитонович А.И. Методы алгоритмизации. Учебное пособие для 8-го класса общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 1996.
15. Котов В.М., Волков И.А., Лапо А.И. Методы алгоритмизации. Учебное пособие для 9-го класса общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 1997.
16. Котов В.М., Мельников О.И. Методы алгоритмизации. Учебное пособие для 10-11-го классов общеобразовательной школы с углубленным изучением информатики. — Мн.: Народная асвета, 2000.
17. Кормен Т., Лейзерсон Т., Ривест Р. Алгоритмы: построение и анализ. — МЦНМО, Москва, 1999, 960 с.

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

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

(Продолжение. Начало в №№9-12/2001, №№1-10, 12/2002, №№1-3,5-6/2003)

WindowsGDI: более подробно о линиях

#1. Основой любой машинной графики является точка. Нарисовать ее можно при помощи функции *SetPixel*:

```
COLORREF SetPixel(
  HDC hdc,              // хэндл DC
  int X,                // x-координата пиксела
  int Y,                // y-координата пиксела
  COLORREF crColor      // цвет пиксела
);
```

Переменная типа *COLORREF* — это двойное слово, младшие три байта которого задают *красную*, *зеленую* и *голубую* составляющие цвета точки. Такая система кодирования цвета называется *RGB* (***Red, Green, Blue***). Учитывая, что каждый байт может принимать значения от 0 до 255, нетрудно подсчитать, что таким образом мы можем закодировать 2^{24}, то есть 16777216 цветов (**рис.1**).

Рис. 1

Практически в каждом языке программирования существует макрос, выполняющий "смешивание цветов". В *MASM32* он выглядит следующим образом:

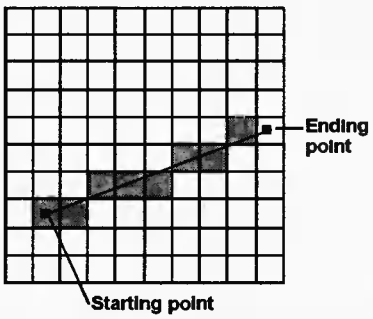
```
;; -------------------------------------
;; INPUT red, green & blue BYTE values
;; OUTPUT DWORD COLORREF value in eax
;; -------------------------------------
  RGB MACRO red, green, blue
    xor eax, eax
    mov al, blue    ; blue
    rol eax, 8
    mov al, green   ; green
    rol eax, 8
    mov al, red     ; red
  ENDM
```

В случае успешного выполнения функция возвращает RGB-значение цвета, которым был отрисован пиксел, в противном случае — *1*. Однако имейте в виду, что отнюдь не все графические устройства могут рисовать точку.

#2. Линия — это набор подсвеченных пикселов (или точек на напечатанной странице), описанный координатами двух точек — начальной и конечной. Причем начальная точка принадлежат линии (т.е. в этой координате прорисовывается пиксел), а вот конечная — не принадлежит (пиксел в конечной координате не прорисовывается).

Правильный способ рисования линии был рассмотрен в предыдущей главе — это парочка функций *MoveTo* (установка позиции пера) и *LineTo* (рисование линии от текущей позиции пера до заданной точки). Сейчас же рассмотрим неправильный способ, но зато более интересный ;).

Рис. 2



Для просчитывания координат всех точек, принадлежащих линии, подсистема *GDI* использует *цифровой дифференциальный анализатор* (*digital differential analyzer, DDA*). На **рис.2**, выдранном из *MSDN*'а, показаны начальная и конечная точки, а также все точки, которые, по мнению *DDA*, принадлежат этой прямой.

Наиболее известный и часто используемый *DDA* — это инкрементальный цифровой анализатор *Bresenham*'а. Именно его продвинутая версия рисует линии в Windows. Это очень простой алгоритм, однако линии, которые рисуются с его использованием, не всегда (по утверждению *MSDN*) соответствуют именно тем линиям, которые необходимы приложению, так как координаты пикселов округляются до ближайшего целочисленного значения. Также *Microsoft* утверждает (в своем репертуаре), что тот *DDA*, который используется *GDI*, подобного недостатка лишен, в результате чего их линии более красивы. И, наверное, осознавая всю неопределенность такого понятия как красота, заботливо оставляет нам возможность рисовать линии с использованием "старого" алгоритма — при помощи функции *LineDDA* и callback-функции *LineDDAProc*.

```
BOOL LineDDA(
  int nXStart,              // x-координата начальной точки
  int nYStart,              // y-координата начальной точки
  int nXEnd,                // x-координата конечной точки
  int nYEnd,                // y-координата конечной точки
  LINEDDAPROC lpLineFunc,   // callback-функция
  LPARAM lpData             // данные, определенные приложением
);

VOID CALLBACK LineDDAProc(
  int X,                    // x-координата точки
  int Y,                    // y-координата точки
  LPARAM lpData             // данные, определенные приложением
);
```

Тип функции *CALLBACK* означает, что ее код будет располагаться в нашей программе, а вот вызываться она будет операционной системой (точно таким же образом, как и хорошо нам известная процедура окна). Вот ключевая часть исходника, демонстрирующая этот "неправильный" способ рисования линии (хотя... если речь идет не о рисовании, а просто нужна простыня координат, то почему бы и не использовать математический алгоритм, встроенный в операционную систему, вместо того чтобы включать код соответствующей функции в собственную программу?):

```
...

.data?

  cxClient    dd ?
  cyClient    dd ?
  hDC         dd ?

.code

...

;-------------------------------------------------------------
; процедура окна
;-------------------------------------------------------------
WndProc proc hWnd :HWND, uMsg :UINT, wParam :WPARAM, lParam :LPARAM

  local ps  :PAINTSTRUCT
  local x1 :dword, y1 :dword, x2 :dword, y2 :dword

  mov eax,[uMsg]

;----- рисуем

  .if eax==WM_PAINT

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

    invoke LineDDA, 0, 0, [cxClient], [cyClient], addr
LineDDAProc, 0
    invoke LineDDA, 0, [cyClient], [cxClient], 0, addr
LineDDAProc, 0

    invoke EndPaint,hWnd,addr ps

;----- узнаем координаты клиентской области

  .elseif eax==WM_SIZE

    movzx  eax, word ptr lParam[0]   ; low word
    mov    [cxClient], eax
    movzx  eax, word ptr lParam[2]   ; hi word
    mov    [cyClient], eax

;----- выходим

  .elseif eax==WM_DESTROY

    invoke PostQuitMessage, 0

  .else

    invoke DefWindowProc,hWnd,eax,wParam,lParam
    ret

  .endif

    xor    eax,eax
    ret

WndProc endp

;-------------------------------------------------------------
; Еще одна CALLBACK процедура
;-------------------------------------------------------------

LineDDAProc proc X :dword, Y :dword, Color :LPARAM
  invoke SetPixel, hDC, X, Y, Color
  ret
LineDDAProc endp

...
```

Функция *LineDDA*, которой передаются координаты начальной и конечной точек, осуществляет вычисление координат всех пикселов, которые должны быть подсвечены. При вычислении очередной координаты функция *LineDDA* вызывает callback-функцию *LineDDAProc*, которой и передает вычисленные координаты. В нашем случае эти координаты используются для вывода точки при помощи функции *SetPixel*.

В случае успешного выполнения функция *LineDDA* возвращает *не-0*, в противном случае — *0*.

#3. В том случае, если необходимо отрисовать несколько линий, нужно использовать функцию *PolyLine*, которая последовательно соединяет линиями энное количество точек, образуя при этом то, что в школьных учебниках по геометрии называется "ломаная линия":

```
BOOL Polyline(
  HDC hdc,              // хэндл DC
  CONST POINT *lppt,    // массив конечных точек
  int cPoints           // число точек в массиве
);
```

где **POINT** — это структура, определенная следующим образом:

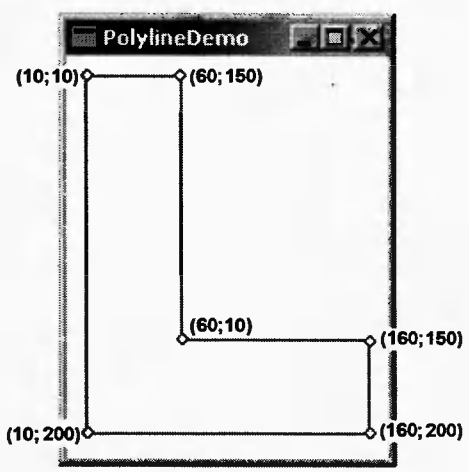
```
typedef struct tagPOINT {
  LONG x;
  LONG y;
} POINT, *PPOINT;
```

Вот исходник, демонстрирующий использование этой функции:

```
...

  PolyData    dd 10,  10
              dd 10,  200
              dd 160, 200
              dd 160, 150
              dd 60,  150
              dd 60,  10
              dd 10,  10

  nPolyPoints = ($ - PolyData) / 8

...

  invoke Polyline, hDC, addr PolyData, nPolyPoints

...
```

Окно этой нехитрой программы выглядит, как показано на **рис.3** (координаты — это авторская отсебятина, программа их не рисует).

Рис. 3



Есть еще одна похожая функция для рисования ломаных линий — *PolylineTo*. Ее отличие только в том, что линия начинает отрисовываться с текущей координаты пера, и координаты пера при этом изменяются от точки к точке.

В случае успешного выполнения обе эти функции возвращают *не-0*, в противном случае — *0*.

#4. Для рисования при помощи линий более сложных фигур используется функция *PolyPolyline*:

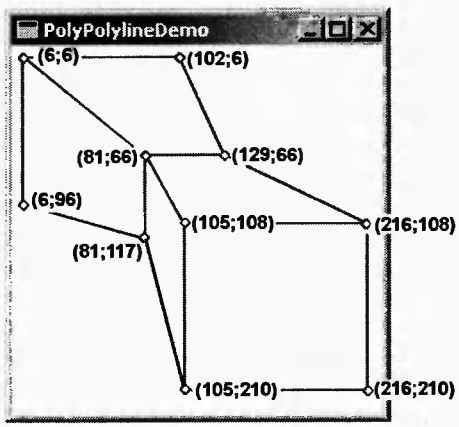
```
BOOL PolyPolyline(
  HDC hdc,                      // хэндл DC
  CONST POINT *lppt,            // массив точек
  CONST DWORD *lpdwPolyPoints,  // массив значений
  DWORD cCount                  // число значений
);
```

Вот небольшой пример ее использования:

```
  ...

  PolyData  dd 6,96,    6,6,     102,6
            dd 81,117,  81,66,   129,66

            dd 105,210, 105,108, 216,108, 216,210, 105,210
            dd 6,96,    81,117,  105,210
            dd 6,6,     81,66,   105,108
            dd 102,6,   129,66,  216,108

  xm        dd 3, 3, 5, 3, 3, 3

  nPolyPoints = ($ - xm) / 4

  ...

  invoke PolyPolyline, hDC, addr PolyData, addr xm,
nPolyPoints

  ...
```

Рисунок, который рисует эта программа, выглядит как показано на **рис.4** (координаты точек — это снова авторская отсебятина).

Рис. 4



Как следует из названия, функция *PolyPolyline* отрисовывает энное количество ломаных линий. Причем для каждой из них число конечных точек может быть разным, поэтому их количество необходимо задавать в отдельном массиве. Так, в нашем примере мы сначала подготавливаем массив точек *PolyData*, а в массиве *xm* указываем, что для отрисовки первой ломаной линии необходимо использовать из *PolyData* последовательно: первые три координаты (первая ломаная линия), вторые три координаты (вторая), следующие пять координат (третья) и т.д. И само собой, мы должны указывать общее число ломаных линий.

В случае успешного выполнения функция возвращает *не-0*, в противном случае — *0*.

На web-странице журнала (http://radio-mir.com) выложены полные версии приведенных в данном тексте исходных кодов.

(Продолжение следует)

Рисунок О.Попова

Самый короткий путь к IPC

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

Рано или поздно у любого человека, написавшего больше одной программы, появляется желание заставить эти программы как-то общаться между собой. Это желание со временем растет и крепнет, и в конце концов достигает таких огромных размеров, что его просто необходимо удовлетворить. Обычно такой человек в поисках удовлетворения хватается за умные книжки (кстати, по-научному обмен данными между приложениями называется interprocess communication, сокращенно — IPC), зарывается в MSDN или ищет готовые примеры в Интернете. Но что в итоге находит этот человек? Ответ на вопрос "как бы мне переслать пару байтов из одной программы в другую"? Как бы не так! Он находит в книжках множество умных фраз о мютексах, семафорах, memory-mapped-файлах и прочая, прочая, прочая. И все это нередко сопровождается многокилобайтными листингами на языке Це. Для профи, возможно, эти листинги выглядят чем-то вроде детских книжек с картинками, но человеку, впервые столкнувшемуся с проблемой коммуникации между процессами, эти коды сильно напоминают китайскую грамоту. Хотя, казалось бы, простая задача должна иметь столь же простое решение. Вот этим мы и займемся — попытаемся установить общение между программами, уложившись при этом в два десятка строчек кода.

Одним из способов обмена данными между приложениями являются сообщения (messages). В ОС Windows вместе с сообщением можно передать два четырехбайтных значения, которые называются wParam и lParam соответственно. Отправка сообщений не составляет никакой особой сложности и реализуется при помощи функций WinAPI SendMessage, SendMessageEx, PostMessage или SendNotifyMessage. Я предпочитаю использовать SendMessageEx, но в приведенном ниже примере будет использоваться исключительно SendMessage (единственно по той причине, что вызов этой функции выглядит несколько проще). Большинство сообщений Windows предназначено для отправки конкретным окнам, хотя возможна и "широковещательная" отправка сообщения всем окнам верхнего уровня, независимо от таких их атрибутов как видимость, активность и т.п. Традиционно обработка сообщений производится внутри оконной процедуры. В приложениях, не имеющих ни одного окна, для получения сообщений приходится прибегать к более сложным приемам, например, к использованию хуков. Стало быть, если мы хотим иметь возможность получать сообщения, желательно, чтобы наша программа имела хотя бы одно окно. Теперь нам осталось только определиться с "частотой волны", на которой мы будем передавать наши сообщения.

В принципе, вы можете передавать информацию при помощи сообщения WM_COPYDATA. Достоинство такого подхода в том, что это сообщение позволяет передавать объемы информации, значительно превышающие восемь байтов. Однако у этого достоинства есть и довольно неприятная обратная сторона — наверняка ваша программа далеко не единственная в системе, и кто-нибудь другой тоже мог использовать этот прием в своем софте. И если вы случайно (или умышленно) отправите некий блок данных чужой программе, невозможно предсказать, как эта программа прореагирует на такое вторжение. Если несчастная программа не рухнет сразу же с GPF, она вполне может попытаться отплатить вам аналогичным сообщением, но со своими данными — и тогда уже вашей программе придется интерпретировать информацию, которая для нее вообще-то не предназначена. В общем, безнаказанно отправить такое сообщение в "широковещательном" режиме (то есть указав в качестве хэндла окна-получателя HWND_BROADCAST) вряд ли удастся. Чтобы избежать подобных неприятностей, обычно используют уникальные идентификаторы, которые добавляют в начало или конец каждого пересылаемого блока данных — это позволяет однозначно идентифицировать приложение-отправителя сообщения и принять решение о методе обработки данных. Кроме того, перед отправкой таких сообщений, опять же, во избежание конфликтов с другими программами, приходится "вручную" производить поиск всех окон, которым будет отправлено сообщение, при помощи функции EnumWindows, что не лучшим образом сказывается на быстродействии программы. Так что если ваша задача заключается в том, чтобы просто "просигналить" программе о том, что пришла пора выполнить те или иные действия, использование столь сложного механизма вряд ли уместно.

В некоторых программах для обмена информацией используются сообщения WM_USER+N, в частности, именно так реализован механизм IPC в WinAmp. Однако Microsoft имеет по этому поводу свое особое мнение — согласно MSDN, сообщения от WM_USER+0 до WM_USER+3FFFh включительно используются только для передачи данных внутри или между окнами одного класса. А чтобы вы почем зря не нарушали данные свыше инструкции, я расскажу вам, чем может кончиться несоблюдение рекомендаций MSDN. Несколько лет назад мне встретился пакет программ, в котором для обмена информацией между отдельными приложениями использовались сообщения WM_USER+... . Сами программы работали как надо, но вот WinAmp во время их работы был совершенно непригоден к употреблению. Как только компоненты пакета начинали обмениваться информацией, сей странный mp3-проигрыватель просто сходил с ума. Причина была простая — все та же "широковещательная" рассылка сообщений. Так что если вы не жаждете стать невинной жертвой, в клочья разорванной ордами ВинАмпопоклонников — потрудитесь соблюдать установленный порядок.

Ну вот, теперь мы узнали самое главное — как не надо организовывать обмен данными между программами, и пусть это знание ведет нас через дебри багов и глюков к светлому будущему. Остался сущий пустяк — выяснить, как все-таки можно переслать другой программе пару байтов, никому при этом не помешав и ничего не испортив. В поисках ответа на сей глубоко философский вопрос обратите свой взор на функцию

RegisterWindowMessage, которая принимает одинединственный параметр — указатель на ASCIIZ-строку, однозначно идентифицирующую сообщение. Возвращает эта функция номер сообщения, которое отныне и до окончания текущей сессии будет связано с этой строкой. При первом вызове этой функции Windows резервирует и отдает в наше полное распоряжение первое попавшееся ей на глаза свободное сообщение. Если текстовая строка уже была поставлена в соответствие какому-либо сообщению, при повторном вызове функции с тем же параметром в качестве результата мы получим именно номер этого сообщения. Образно говоря, первый вызов этой функции "открывает канал" для приема-передачи данных, *а* все последующие позволяют настроиться на ранее зарегистрированную "волну". Всего Windows позволяет зарегистрировать 4000h различных сообщений с номерами в интервале от 0C000h до 0FFFFh (как видите, достигнуть этого предела не так-то просто — если, конечно, специально к этому не стремиться). Но если, ценой тяжких трудов и ужасных лишений, вам все-таки удастся исчерпать этот лимит, и вы попытаетесь зарегистрировать 4001h-е сообщение, Windows вам откажет и в качестве результата функции вернет 0. К сожалению, Microsoft не предусмотрела каких-либо функций для "разрегистрации" ненужных сообщений, а потому однажды зарегистрированное сообщение будет занято в течение всей сессии.

Итак, позвольте на этом закончить теоретическую часть и перейти к самому интересному — изучению вполне работоспособного примера программы (а если быть до конца точным, то двух программ: клиента, отправляющего сообщения, и сервера, эти сообщения принимающего), демонстрирующей предлагаемую технику. Полные тексты примеров вместе с файлом проекта RadASM (для упрощения самостоятельных экспериментов) вы можете найти на сайте журнала (http://www.radiomir.com). Автор же, в приливе нечеловеческого альтруизма и понимания, сколь трудно подчас бывает написать десяток строчек работающего кода, предоставляет всем читателям сего манускрипта право всячески цитировать код примеров и безнаказанно использовать его в своих проектах.

Оба примера в начале содержат практически идентичные участки кода, отвечающие за регистрацию сообщения, при помощи которого мы будем осуществлять коммуникацию между процессами:

```
invoke RegisterWindowMessage,ADDR MsgString
mov Message_for_IPC,eax

.if !eax
 invoke MessageBox,0,ADDR msgMNR, ADDR msgError,MB_APPLMODAL
        or MB_ICONERROR
.else
 invoke DialogBoxParam,hInstance,IDD_DIALOG1,NULL,addr
        DlgProc,NULL
.endif
```

Нетрудно догадаться, что в этих наполненных глубинным смыслом строках мы пытаемся зарегистрировать сообщение, сохраняем полученный код для дальнейшего использования в программе и затем, в зависимости от того, удалось нам зарегистрировать сообщение или нет, вызываем диалог либо отображаем сообщение об ошибке. Теперь пришло время обратить взоры к строке MsgString, которая играет роль идентификатора сообщения. Правильный выбор строки-идентификатора имеет огромное, если не сказать судьбоносное, значение для нормальной работы программы. Вы ведь не хотите, чтобы ваша программа реагировала на каждый "чих" какой-нибудь не вашей программы, которая использует тот же идентификатор?

Конечно, вы можете взять в качестве идентификатора первую попавшуюся на глаза текстовую строку, например, название своей программы. Но практика показала, что в деле называния своих творений полет фантазии у программистов практически нулевой, повсеместно расплодившиеся PowerPad'ы и CoolClock'и — живой тому пример. Поэтому дружно проигнорируйте поданный мною дурной пример (идентификатор "Message_for_IPC" на удивление неординарен, не так ли?). Будьте оригинальнее! Например, переведите название своей программы на валлийский язык и возьмите его в качестве идентификатора. Поскольку валлийским языком во всем мире владеет всего несколько сот человек, и программистов среди них не так уж много, несчастливые совпадения названий программ весьма маловероятны. Если же вы среди всех направлений искусства программирования предпочитаете примитивизм, просто возьмите микрософтовский генератор уникальных идентификаторов, запустите его и воспользуйтесь результатом.

После того как мы узнали номер сообщения, при помощи которого наши программы будут обмениваться байтиками, мы можем смело начинать эти байтики пересылать. Более того, мы можем пересылать даже сдвоенные и счетверенные байтики, а если как следует постараться — то и все восемь байтов! Дурное дело, как известно, нехитрое:

```
invoke SendMessage,HWND_BROADCAST,Message_for_IPC,wParam ,lParam
```

Байтики, которые мы хотим переслать, нужно всего лишь "зарядить" в качестве параметров wParam и lParam, каждый из которых имеет тип DWORD. В моем примере принципиальной необходимости во втором параметре не было, поэтому он равен нулю. В результате все окна верхнего уровня получат "подарочек" в виде нашего сообщения и двух его параметров. В нашем примере мы отправляем в качестве параметра номер нажатой кнопки, который вычисляем как ID кнопки минус 1000:

```
.elseif (eax==IDC_BTN1) || (eax==IDC_BTN2) || (eax==IDC_BTN3)
      sub eax,1000
      invoke SendMessage,HWND_BROADCAST,Message_for_IPC,eax,0
     .endif
```

Теперь посмотрим, как получить отправленные таким образом данные в своей программе. В принципе, тут тоже нет ничего сложного — обычная обработка сообщения в оконной процедуре:

```
mov eax,uMsg
.if eax== Message_for_IPC
; Если получено зарегистрированное нами сообщение
invoke wsprintf,ADDR Buffer, ADDR FmtStr,wParam
invoke SetDlgItemText,hWin,IDC_STC1,ADDR Buffer
```

Как видно из текста, здесь мы всего лишь переводим номер нажатой кнопки в текстовое представление и выводим полученный текст в соответствующее окно.

Вот и все! Впишите приведенные куски кода в свои программы, и проблемы с обменом байтиками между процессами у вас больше не возникнет. Желающие могут даже посчитать, сколько строчек текста нам на это потребовалось, и в какое количество байтов эти строчки откомпилировались. Ну и, конечно, в очередной раз восхититься языком, который предоставляет возможность писать самые короткие программы.

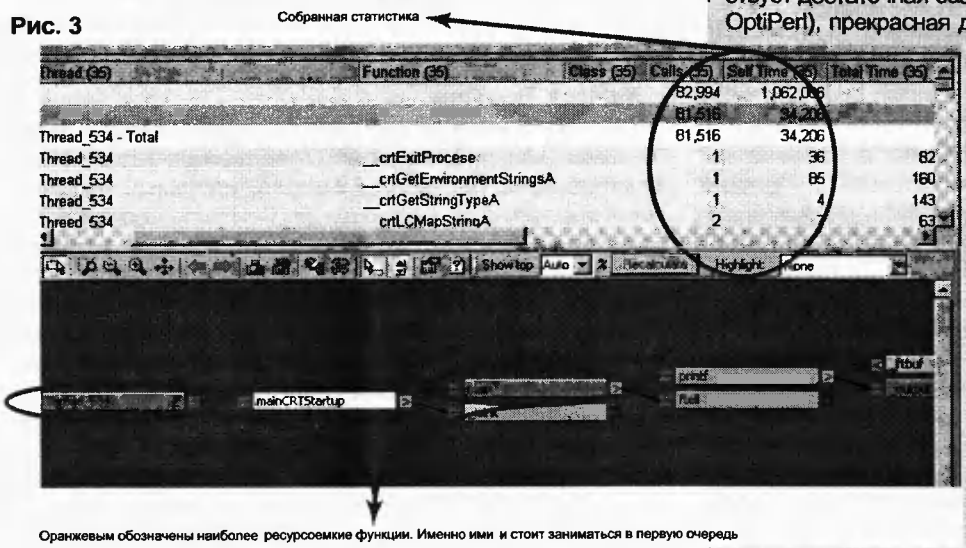
В.ГРИЦАН /HI-TECH,
г.Торонто, Канада,
WWW: http://wasm.ru/

Современные профилировщики — что выбрать?

(Окончание. Начало в №6/2003)

Теперь посмотрим на метод Call graph. Еще раз напомню, Sampling собирает данные по всем активным процессам, Call graph — только по конкретному подопытному приложению. Здесь большое значение играет цвет функции **(рис.3)**.

Рис. 3



Итак, функции представлены в виде дерева. Оранжевый цвет функции говорит о том, что именно в ней проводится большая часть времени. Толщина стрелок, соединяющих функции и их цвет, тоже играют большую роль. О назначении цвета, думаю, догадаетесь сами, толщина стрелки означает время, проведенное в вызывающей функции. Также на графике показывается, какая функция "позвала" остальные; время, затраченное на выполнение конкретной функции и т.п. Здесь тоже много чего можно узнать. Мне представляются важными показатели, приведенные в **табл.2**.

Табл. 2

calls	Число вызовов дочерней функции родительской функцией
self time	Время (microseconds), проведенное внутри функции. Не включает в себя время, затраченное на вызовы других функций
total time	Время (microseconds), проведенное внутри функции. Включает в себя и время, проведенное в дочерних функциях, т.е. ПОЛНОЕ время всей функции и всех порожденных ею функций
callers	Число "родителей", вызвавших данную функцию
callees	Число дочерних функций, которые были вызваны данной функцией.

У VTune есть еще одна очень приятная возможность, начисто отсутствующая у конкурентов — статический анализ ассемблерного кода (File/Open Static Module Viewer). Здесь возможности вообще изумительные — можно посмотреть время выполнения конкретных инструкций на конкретном процессоре (разумеется, Intel-линейки); сразу получить совет, как можно устранить ту или иную задержку; узнать, что означает тот или иной термин. Словом — прелесть, ассемблерщики оценят! Справедливости ради замечу, что у AMD тоже есть нечто подобное — CodeAnalyst, на момент написания статьи — версии 1.2. С ним я не "игрался", т.к. у меня Intel'овский процессор. Однако профилировщик свободно доступен для закачки с сайта AMD, так что no problem!

Однако обзор VTune был бы не полным, если бы мы не затронули тему профилировки интернет-приложений. К таковым здесь будем относить ASP-/Java-страницы, ISAPI-dll, а также CGI-программы, написанные на компилируемых (а не интерпретируемых, как Perl — хотя, как известно, сейчас уже можно получать Perl-exe-файлы) языках.

CGI-приложения на компилируемых языках

Сейчас стало модным писать CG-приложения на Perl. Спору нет, язык действительно очень приятный. Уже существует достаточная база утилит (Visual Perl, например, или OptiPerl), прекрасная документация, множество примеров и пояснений. Однако что делать тем, кто в силу самых различных причин вынужден писать CGI-программы с использованием C/C++? Отладка и профилировка таких программ — дело ох какое сложное. Здесь существует несколько трюков, которые позволяют облегчить жизнь разработчику. Лично я не люблю вставку printf в код в отладочных целях — не та степень интерактивности, не та степень контроля над программой, поэтому:

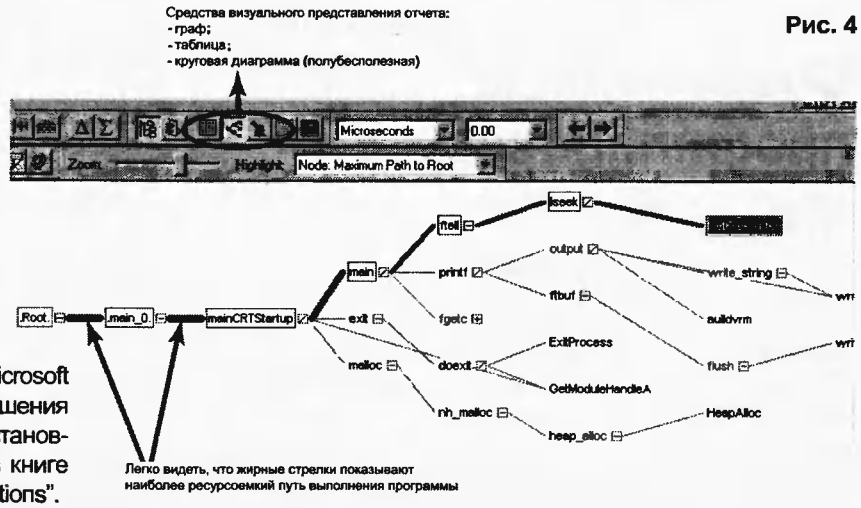
- компилирование приложения со вставкой __asm int 3; ***после*** получения программой CGI QUERY_STRING. Как известно, int 3 вызывает всплытие отладчика — таким образом, мы можем продолжать отладку как ни в чем не бывало;
- вставка в код sleep-функций (например, функции WinAPI Sleep(DWORD)). Поток на какое-то время "засыпает", и за это время нам надо сделать "Attach to process" в отладчике. Потом — по накатанной дорожке;
- нормально спроектированные отладчики обязаны допускать, чтобы программа имела возможность получать на вход аргументы командной строки. Поэтому просто вставьте QUERY_STRING ***без*** знака "?" в command-line arguments-опцию VTune или Visual Studio Debugger и наслаждайтесь. При необходимости можно добавить переменные окружения CONTENT_TYPE, CONTENT_LENGHT и т.п (Win + break / Advanced / Environment Variables).

Работоспособность первого метода под IIS у меня вызывает сильные сомнения. Да, действительно, исключительная ситуация возникает, однако отладчик Visual Studio (под Soft-Ice я не пробовал) не активируется. Почти наверняка, это происходит из-за модели работы IIS (см. статью Skonnard "Understanding the IIS Architecture" на MSDN).

Да, и еще одна вещь, омрачающая жизнь разработчиков CGI-приложений под Win32. **Для каждого http-запроса Windows создает новый процесс.** А это есть очень нехорошо, т.к. создание процесса под Windows — вещь сложная, долгая и ресурсоемкая. Здесь отправляю к статье Osterlund "What Goes On Inside Windows 2000: Solving the Mysteries of the Loader". Можно, конечно, слегка снизить время загрузки, использовав специальные утилиты Мэтта Питрека, или, скажем, "привязать" exe-файл к конкретной ОС утилитой bind, входящей в состав Visual Studio (bind -u filename), но это не изменит принцип. Остается только одна альтернатива, если действительно важна производительность — использовать Fast-CGI (www.fastcgi.com).

Для этого Intel написала еще одно приложение — Intel Enterprise Analyzer for Web Applications. Это "нашлепка" на Perfomance Analyzer, превращающая VTune в одну здоровенную среду профилировки. Enterprise Analyzer ***очень*** ресурсоемок, требует установки JDK, запускает MySQL и даже Tomcat. Все это дело тормозит просто жутко, и весьма сложно в настройке. Поэтому для профилировки таких решений (managed code) рекомендуется использовать Java-решения от NuMega или продукты от Rational с активированной опцией "managed code". Также фирма Microsoft (как всегда :-), молодцы) предлагает свои решения для профилировки ASP-программ. Процесс установки и отладки достаточно детально описан в книге "Perfomance testing Microsoft .NET Web applications".

Рис. 4



Пару слов о недостатках. VTune — продукт глючный, часто бывают заскоки. Иногда встречаются непонятные проколы, вроде требования предъявить .dsp-файл, в то время как VS.NET (а у VTune заявлена совместимость с VS.NET) уже ***не*** использует .dsp/.dsw-файлы, заменяя их своими экзотическими .sln/.suo/.ncb/.vcproj-файлами. Более подробно об этом см. "Использование VC.NET" Кейта Грегори. Поэтому имеет смысл часто проверять ftp-сайт Intel на новые версии программы.

В завершение обзора этого профилировщика хочется сказать, что фирма Intel создала классный интерактивный курс по Vtune. Он доступен на Web-сайте Intel, а также поставляется вместе с VTune. После того как прочтете, "поиграйтесь" с этим курсом, и можете смело считать себя специалистом по профилировщикам.

Ну вот, вроде, все. Теперь со спокойной совестью можно переходить к PurifyPlus.

Rational Quantify

Что ж, продукт быстрый, достаточно легкий в использовании. Тесно интегрирован с VS.NET. По возможностям явно проигрывает VTune, но меня лично приводит в ужас мысль о профилировании действительно ресурсоемкой программы с помощью VTune — это будет слишком медленно и глючно. За время использования, VTune у меня глючил очень много раз, Quantify — ни одного. Стабильная и быстрая утилита.

Итак, Quantify предоставляет информацию только в виде Call graph. Иными словами, мы можем видеть то же дерево выполнения программы с наиболее ресурсоемкими функциями (**рис.4**). Никакой предварительной подготовки программы не требуется. Правда, и к исходному коду доступа нет, не говоря уже о встроенном дизассемблере и т.п. Нет возможности получить советы по оптимизации, как в VTune, нет встроенной помощи. Словом, отсутствуют все те вещи, которые мы так ценим в справочной системе. Однако фирма Rational создала пару достаточно неплохих pdf-файлов с описанием пакетов. Требуется регистрация, однако сколько стоит зарегистрироваться?

Принцип работы — тот же, что и у VTune. Ничего сложного. Единственное, что может немного смутить, это графа F+D time в таблице. F+D означает function+descendents, т.е. родительская функция плюс дочерние. Иными словами, это эквивалент total time в VTune. Во всем остальном используется схожая терминология.

Quantify способен производить профилировку Java/VB/C/C++/C#/J#. По-моему, список весьма неплох. Работать можно.

На этом мы распрощаемся с Quantify.

NuMega TrueTime

Забавно отметить, что сама Microsoft в своей книге "Perfomance testing Microsoft .NET Web applications" рекомендует использовать DevPartner 7.0 для тестирования и профилировки managed-code. Честно говоря, DevPartner производит не самое лучшее впечатление. Те части DevPartner, над которыми работал Мэтт Питрек — BoundsChecker — производят очень приятное впечатление. Все работает быстро, красиво, четко. Хотя отдельные глюки все же присутствуют. Например, несмотря на великолепную интеграцию DevPartner с Visual Studio, данные об исходном файле все равно не появляются. И это происходит даже тогда, когда BoundsChecker (ныне Error Detection) вызывается прямо из активной в данный момент solution-сессии. О TrueTime и TrueCoverage я уж из вежливости умолчу. Если в случае Rational, для того чтобы разобраться с продуктом и получить осмысленные результаты, у меня ушло несколько минут, то, промучавшись пару часов над TrueTime и не получив ничего, кроме времени выполнения Win-API-функций (интересно, а зачем оно мне?), я сдался. Видно, что создатели Purify очень неплохо продумали свои программы, в то время как NuMega создала монстра, которому очень трудно разъяснить, что ты от него, собственно, хочешь.

Итак, если есть желание по-прежнему использовать продукты NuMega, то хорошей идеей будет использование BoundsChecker. Продукт на высоте. Я не проверял его работоспособность с C#/VB, однако, думаю, и здесь он не подведет. TrueTime/TrueCoverage лучше не пользуйтесь — берите продукты от Rational.

Подведение итогов

Автор выражает свою искреннюю признательность vkim за ценные технические советы и Андрею Бабию — за разбор программных алгоритмов. Также моя признательность Alexela и CyberManiac, взявшим на себя труд эту статью вычитать, а также не полениться попинать автора и мужественно выдержать его пинки в ответ. Спасибо, ребята!

Увы, в одной короткой статье нельзя осветить все. К сожалению, лишь очень бегло освещены профилировщики managed-code, совершенно не уделено внимание Java-байткодам, а также решениям NuMega в этой области (JCheck и т.п.). Однако надеюсь, путеводная нить у вас появилась.

Да, чуть не забыл. Если возникают какие-то проблемы при инсталляции программ — возможно, вы найдете ответ на wasm.ru в статье "Обзор некоторых защит на основе FlexLm SDK".

Успехов в работе.

Преобразование псевдографических таблиц в формат HTML

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

(Окончание. Начало в №№5-6/2003)

Работа с программой.
Входные и выходные параметры

Входные параметры передаются программе через командную строку следующим образом:

```
tab2html.exe (ini_filename) <in_filename> <out_filename>.
```

Имя файла может содержать полный путь к файлу. При указании полного пути символ "\" после имени диска и двоеточия обязательно должен присутствовать. Если путь к файлу не указан, файл считается находящимся в текущем каталоге.

Имя ini-файла — необязательный параметр. Если он не указан, используется файл с именем default.ini, находящийся в том же каталоге, где расположен выполняемый файл программы.

Если указать имя ini-файла в квадратных скобках, это будет означать, что файл находится в том же каталоге, где расположен выполняемый файл программы. Таким образом, вы можете создать в этом каталоге несколько ini-файлов с различными значениями параметров (на разные случаи), а потом описанным выше образом указывать имя нужного ini-файла.

Как можно заметить, ситуация, когда при вызове программы имя ini-файла не указано, полностью аналогична той, как если бы мы указали это имя в виде [default.ini].

Пример вызова программы:

```
tab2html.exe n.ini e:\text1.txt c:\work\text1.htm
```

Для этого примера ini-файл с именем n.ini будет взят из текущего каталога, исходный файл с именем text1.txt будет взят из каталога e:\, получаемый файл будет записан с именем text1.htm в каталог c:\work.

Исходный текстовый файл должен быть в альтернативной кодировке, с кодами конца строк (13,10), за исключением, может быть, последней строки текста, у которой код конца строки может отсутствовать. Исходный файл может содержать символы табуляции, которые будут правильно обработаны.

Выходной файл формируется также с кодами конца строк (13,10). Кодировка выходного файла может быть ALT, WIN или KOI — это зависит от значения параметра Recode, указываемого в ini-файле. Выходной файл не содержит символов табуляции, даже если они были в исходном файле.

Ini-файл, как и исходный файл, должен быть в альтернативной кодировке и с кодами конца строк (13,10) (кроме, может быть, последней строки), но табуляций в нем не должно быть. Подробную информацию об указываемых в этом файле параметрах вы найдете в следующем разделе статьи.

Рис. 11

```
Pseudographic table -> HTML table convertor
                version 1.0
        (c) Ivan Roshin, Moscow, 2003
          E-mail: bestview@mtu-net.ru
           WWW: http://www.ivr.da.ru
```

После запуска на экран выводятся сведения о программе (**рис.11**).

Затем, если все нормально, вы увидите примерно следующее (каждая строка соответствует своему этапу выполнения программы; во второй строке указывается количество найденных ячеек, а в третьей — количество таблиц. У вас, конечно, эти значения будут другими):

```
Pass 1... OK.
Pass 2... OK, found 39 cells.
Pass 3... OK, found 2 tables.
Pass 4... OK.
```

После этого программа завершит свое выполнение.

Программа возвращает операционной системе код завершения. 0 — признак успешного завершения, ненулевое значение свидетельствует об ошибке (**табл.2**). Сообщение об ошибке также выводится на экран.

Табл. 2

Возвращаемое значение	Комментарий
1	Неправильно указаны параметры при вызове программы
2	Ошибка при открытии входного файла
3	Ошибка при открытии выходного файла
4	Ошибка при открытии ini-файла
5	Ошибка при чтении параметров из ini-файла (более подробное сообщение о характере ошибки выводится нв экран)
6	Во входном файле встретилась слишком длинная строка
7	Во входном файле слишком много строк, предположительно относящихся к таблицам
8	Не найдено ни одной таблицы
9	Количество найденных ячеек таблиц превышает максимально допустимое
10	Количество найденных таблиц превышает максимально допустимое
11	Количество пустых ячеек, добавляемых к одной из таблиц непрямоугольной формы, оказалось больше допустимого
12	Ошибка при записи в выходной файл
13	Ошибка при закрытии входного файла
14	Ошибка при закрытии выходного файла

Если программа была вызвана из bat-файла или из другой программы, код завершения может быть в них проанализирован, и, в зависимости от его значения, могут быть предприняты какие-либо действия.

Содержимое Ini-файла

В этом файле задаются параметры, определяющие внешний вид получаемых таблиц и кодировку выходного файла.

Символ ";" в начале строки обозначает, что эта строка является комментарием. Такие строки, а также пустые строки игнорируются.

Общее количество строк и максимальная длина строк в ini-файле явно не ограничены, однако максимальная суммарная длина строковых переменных не может превосходить 5000 байтов (эта величина определяется значением константы MAX_STRING_VARS в тексте программы).

Ниже приведены описания всех параметров ini-файла. Значение каждого параметра должно быть определено, т.е. никаких "значений по умолчанию" не существует.

Для понимания смысла многих параметров необходимо знание языка HTML.

Параметр *AttributesTable* — атрибуты тега <table>.

Пример:

AttributesTable={align=center border=1}.

Параметр *UseCaption* определяет, будет ли у формируемых таблиц заголовок, содержащий номер таблицы (Y — да, N — нет).

Пример:

UseCaption=Y.

Параметр *AttributesCaption* — атрибуты тега <caption>.

Пример:

AttributesCaption={align=bottom}.

Параметры *FirstTagsInCaption* и *LastTagsInCaption* определяют теги, которые будут добавлены соответственно после тега <caption> (т.е. перед текстом заголовка таблицы) и перед тегом </caption> (т.е. после текста заголовка). Может быть указано несколько тегов или вообще ничего не указано.

Пример:

Пусть заголовок должен выводиться наклонным шрифтом. В этом случае значения параметров должны быть следующими:

FirstTagsInCaption={}

LastTagsInCaption={}.

Параметр *TextCaption* — содержимое заголовка таблицы.

Пример:

TextCaption={Табл. }.

Параметр *FirstCaptionNum* — номер, с которого начнется нумерация таблиц. Номер таблицы включается в заголовок сразу после текста, заданного параметром TextCaption.

Пример:

FirstCaptionNum=1.

Параметр *AttributesTR* — атрибуты тега <tr>. Значением этого параметра может быть либо одна строка, либо число строк (1...255) и далее эти строки. В последнем случае для первого тега <tr> в каждой таблице будут использованы атрибуты, взятые из первой строки параметра, для второго тега <tr> — из второй строки, и так далее (после того как атрибуты будут взяты из последней строки параметра, произойдет переход к первой). Это удобно использовать, например, чтобы делать таблицы "полосатыми".

Пример:

Пусть надо, чтобы цвет фона в строках таблицы чередовался: в верхней строке — #ccffff, в следующей строке — #ccffcc, потом опять #ccffff, и так далее. Тогда значение параметра будет таким:

AttributesTR=2 {bgcolor=#ccffff} {bgcolor=#ccffcc}.

Параметры *UpTH* и *LeftTH* показывают, сколько первых строк сверху и первых столбцов слева в каждой таблице будут содержать тег <th> вместо <td>.

Пример:

UpTH=1

LeftTH=0.

Параметр *AttributesTH* — атрибуты тега <th>. Значением этого параметра может быть либо одна строка, либо число строк (1...255) и далее эти строки. В последнем случае в каждой строке таблицы для первого тега <th> будут использованы атрибуты, взятые из первой строки параметра, для второго тега <th> — из второй строки параметра, и так далее (после того как атрибуты будут взяты из последней строки параметра, произойдет переход к первой).

Пример:

Пусть никаких атрибутов задавать не надо. Тогда значение параметра будет таким:

AttributesTH={}.

Параметры *FirstTagsInTH* и *LastTagsInTH* определяют теги, которые будут добавлены соответственно после тега <th> (то есть перед содержимым ячейки) и перед тегом </th> (т.е. после содержимого ячейки). Значением каждого из этих параметров может быть либо одна строка, либо число строк (1...255) и далее эти строки. В последнем случае (на примере параметра FirstTagsInTH, для параметра LastTagsInTH — аналогично) в каждой строке таблицы после первого тега <th> будут добавлены теги, взятые из первой строки параметра FirstTagsInTH, после второго тега <th> — из второй строки, и так далее (после того как атрибуты будут взяты из последней строки параметра, произойдет переход к первой).

Пример:

Пусть в таблице три столбца, и надо, чтобы в <th>-ячейках, находящихся в первом столбце, текст был наклонным, а во втором и третьем столбцах — нет. Тогда указываем:

FirstTagsInTH=3 {} {} {}

LastTagsInTH=3 {} {} {}.

Параметр *AttributesTD* — атрибуты тега <td>. Значением этого параметра может быть либо одна строка, либо число строк (1...255) и далее эти строки. В последнем случае в каждой строке таблицы для первого тега <td> будут использованы атрибуты, взятые из первой строки параметра, для второго тега <td> — из второй строки, и так далее (после того как атрибуты будут взяты из последней строки, произойдет переход к первой).

Пример:

Пусть в таблице три столбца, и надо, чтобы в <td>-ячейках, находящихся в первом столбце, текст был выровнен по правому краю, а во втором и третьем столбцах — по центру. Тогда указываем:

AttributesTD=3 {align=right} {align=center} {align=center}.

Параметры *FirstTagsInTD* и *LastTagsInTD* — то же самое, что и описанные выше FirstTagsInTH и LastTagsInTH, но относятся не к <th>, а к <td>.

Пример:

Пусть в таблице три столбца, и надо, чтобы текст в <td>-ячейках, находящихся во втором столбце, был выделен жирным шрифтом. Тогда значения параметров будут следующими:

FirstTagsInTD=3 {} {} {}

LastTagsInTD=3 {} {} {}.

Параметр *DelSpaceInCell* определяет, надо ли удалять пробелы сверху и снизу, слева и справа от содержимого ячеек (Y — да, N — нет).

Пример:

DelSpaceInCells=Y.

Параметр *UseNbspInSpaceCells* определяет, нужно ли добавлять символ   в пустые ячейки таблицы, чтобы вокруг этих ячеек рисовалась рамка (Y — да, N — нет).

Примечание: парамер не оказывает влияния на пустые ячейки, которые добавляются к таблицам непрямоугольной формы.

Пример:

UseNbspInSpaceCells=Y.

Параметр *Recode* — кодировка выходного файла: 0 — ALT, 1 — WIN, 2 — KOI.

Пример:

Recode=1.

Параметр *UseLRaquo* определяет, на что будет заменен символ кавычки в содержимом ячеек: если N — всегда на ", если Y — первый символ в ячейке будет заменен на «, второй — на », следующий — опять на « и так далее.

Пример:

UseLRaquo=Y.

Remote Controller for Winamp

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

Уверен, у многих из вас бывает, что после того как прочитаешь какую-нибудь статью или увидишь программу с интересной концепцией, начинают возникать многочисленные идеи по реализации и усовершенствованию увиденного. Зачастую, при отсутствии лени и наличии свободного времени, эти идеи развиваются с астрономической скоростью, и в итоге получается продукт, по многим параметрам превосходящий прототип. Так произошло и со мной после прочтения статьи [1].

У меня возникло множество интересных идей по реализации "пульта дистанционного управления" (ПДУ) для Winamp'а. Среди них я выделил три наиболее удобные и практически осуществимые. О них я и собираюсь рассказать.

Способ первый — System Tray

Этот способ является наиболее трудным в реализации и максимально приближенным к программистскому (т.е. наиболее автоматизированному). Суть его заключается в помещении в System Tray (область Taskbar, располагающуюся рядом с часами) иконок, которые и будут служить управляющими клавишами. Я написал программу, основанную на данном способе. Для ее создания я выбрал среду Delphi, так как язык Object Pascal больше чем другие подходит для быстрого и комфортного создания приложений среднего уровня сложности. Так, если у меня на создание этой программы с использованием Delphi ушел один день, то при использовании других языков и сред интегрированной разработки на ее создание ушло бы уже несколько дней. Рабочее окно программы показано на **рис.1**.

Рис. 1

Функциональные кнопки разделены на две группы — кнопки управления Winamp'ом и кнопки управления самим ПДУ. Для управления проигрывателем я реализовал только самые необходимые функции: включение/выключение, воспроизведение, пауза, стоп, переход к следующей/предыдущей композиции, шаг на 5 с вперед/назад. Кнопки воспроизведения и включения выполняют также функции паузы и выключения соответственно. Я считаю, что добавлять ПДУ еще какие-либо функции управления не стоит — тогда уже лучше будет самим Winamp'ом воспользоваться. Самая же главная функция программы раскрывается при ее минимизации, когда она вместо сворачивания в Taskbar помещается в System Tray. В режиме паузы и остановки свернутая программа выглядит как показано на **рис.2**, в режиме воспроизведения — как на **рис.3**.

Рис. 2

Рис. 3

Рис. 4

Если вы расширите Taskbar до двух строк, предоставив таким образом больше места не только для System Tray, но и для Quick Launch, вид свернутой программы будет соответствовать **рис.4**.

Как вы видите, программа добавляет в System Tray пять иконок, четыре из которых используются для управления Winamp'ом, а пятая — для возвращения программы в развернутый режим работы. Кнопка воспроизведения, как и в развернутом режиме, выполняет также и функцию кнопки паузы. Сделано это для экономии дефицитного места в области состояния. Ведь System Tray — не "резиновая", а каждая современная программа так и норовит всунуть туда свой значок, которым пользователь воспользуется от силы раз десять. После рассмотрения основных принципов построения программы, перейдем к аспектам ее реализации.

Главной трудностью для начинающих при написании программ подобного рода, безусловно, является работа с иконками в System Tray (или в области состояния — taskbar status area). Поэтому на этом вопросе я остановлюсь более подробно. В WinAPI имеется всего одна функция, предназначенная для этих целей — Shell_NotifyIcon (находится в библиотеке ShellApi). Вот ее подробный формат:

```
Shell_NotifyIcon(DWORD dwMessage, PNOTIFYICONDATA pnid);
```

Параметр *dwMessage* определяет действия, которые следует произвести со значком в System Tray. Он может принимать одно из следующих значений:

- NIM_ADD — добавить иконку в область состояния;
- NIM_DELETE — удалить иконку из области состояния;
- NIM_MODIFY — изменить иконку в области состояния.

Параметр *pnid* — это указатель на структуру типа NOTIFYICONDATA. Содержание структуры зависит от значения параметра *dwMessage*.

Если операция проходит успешно, то функция возвращает ненулевое значение. В противном случае — ноль.

Теперь более подробно рассмотрим тип NOTIFYICONDATA:

```
NOTIFYICONDATA {
    DWORD cbSize;
    HWND hWnd;
    UINT uID;
    UINT uFlags;
    UINT uCallbackMessage;
    HICON hIcon;
    char szTip[64];
};
```

Поля структуры:

- *cbSize* — размер структуры NOTIFYICONDATA;
- *hWnd* — дескриптор окна, которое будет получать сообщения от значка;
- *uID* — идентификатор значка;
- *uFlags* — набор флагов, который определяет, какие поля записи содержат корректные данные. Это поле может содержать комбинацию следующих значений:
- NIF_ICON — поле *hIcon* заполнено корректно;
- NIF_MESSAGE — поле *uCallbackMessage* заполнено корректно;
- NIF_TIP — поле *szTip* заполнено корректно;
- *uCallbackMessage* — идентификатор сообщения. Система использует этот идентификатор для посылки окну, определенному дескриптором *hWnd*, сообщений при возникновении событий мыши над иконкой;
- *hIcon* — дескриптор иконки, которую необходимо добавить, изменить или удалить;
- *szTip* — текст для всплывающего сообщения, которое появляется при остановке курсора над иконкой.

Итак, для запуска **Shell_NotifyIcon** предварительно нужно создать структуру типа NOTIFYICONDATA. Обязательными для заполнения являются только поля cbSize, hWnd и uFlags.

Остальные заполняются в зависимости от выполняемых действий и потребностей программиста. Поле uFlags заполняется в соответствии с содержимым вышеназванных полей. Например, если мы заполним поля uCallbackMessage и hIcon, то содержание uFlags следует сделать следующим:

```
uFlags:=NIF_ICON or NIF_MESSAGE;
```

Если мы хотим добавить иконку в System Tray, то необходимо вызвать функцию Shell_NotifyIcon с параметром dwMessage равным NIM_ADD. При правильном заполнении полей структуры pnid, в System Tray должна появится новая иконка.

Для изменения значка или всплывающей подсказки уже созданной иконки необходимо вызвать функцию Shell_NotifyIcon с параметром dwMessage, равным NIM_MODIFY. Теперь для заполнения обязательными являются поля cbSize, hWnd, uFlags и те поля, значения которых вы планируете изменить.

Чтобы удалить иконку из System Tray, необходимо заполнить асе те же три поля и вызвать функцию с параметром dwMessage, равным NIM_DELETE.

Особо следует упомянуть об обработке событий, происходящих в области иконки. В качестве идентификатора сообщения uCallbackMessage можно использовать WM_USER + n, где n — целое число из [1; 7FFF]. При этом первый параметр получаемого сообщения (WParam) будет содержать идентификатор иконки (uID), а второй (LParam) — тип сообщения. Это позволяет сделать довольно элегантный обработчик событий от нескольких иконок. При этом каждой иконке в качестве идентификатора сообщения присваивают одно и то же сообщение. Вместо того чтобы писать для каждой иконки свою процедуру обработки событий, мы напишем для всех иконок одну общую процедуру. Так как все иконки будут посылать сообщения по одному адресу, то нашей процедуре надо запускаться при получении события по этому адресу. Вначале надо проверить, тот ли тип события, который нам необходим (в нашем случае — одинарное нажатие левой клавиши мыши), произошел в System Tray. Для этого просматриваем LParam полученного сообщения. Если он удовлетворяет нашим требованиям, то, просматривая WParam, определяем иконку, над которой произошло событие, и аыполняем соответствующий этой иконке участок кода. При количестве иконок больше двух для этих целей удобно использовать оператор выбора *case*. Думаю, что вы знакомы с основными событиями, которые могут произойти в System Tray, а если нет — почитайте об этом в [2].

Это, по-моему, необходимый минимум знаний для успешной работы с System Tray, а теперь возвратимся к программе.

О внешнем виде программы. Я применил только базовые элементы "украшения" (регион, кнопки объединены а две группы с помощью Shape). Конечно, можно сделать и намного красивее, но ведь эта программа будет постоянно "висеть" в области состояния и отнимать у системы ресурсы оперативной памяти и процессорное время. Как я уже сказал, в программе применена непрямоугольная форма, полученная из одного региона. Более подробно об этом можно узнать в [3]. Кстати, в этой программе я применил другую реализацию процедуры "таскания" формы мышкой:

```
procedure TForm1.WMNCHITTEST(var Msg: TMessage);
begin
  inherited;
  Msg.Result := HTCAPTION
end;
```

Недостатком данной процедуры является невозможность обработки событий от правой кнопки мыши (например, при создании всплывающих меню). Зато этот способ действует, когда пользователь "хватается" не только за свободное место формы, но и за некоторые компоненты (например, Label), помещенные на форму.

В комплекте с программой идут 5 файлов иконок, которые должны быть помещены в ту же директорию, где будет размещен exe-файл программы. Я специально не помещал их в ресурс, чтобы пользователь мог легко их поменять, не прибегая к помощи программ такого рода как Resource Hacker.

При своих богатых возможностях программа получилась не очень большой по объему. Вот ее полный листинг:

```
unit RemoteController_;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls, Registry, ShellApi;
const
  WM_TRAYICON=WM_USER+1;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Shape1: TShape;
    Shape2: TShape;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    Button9: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    { Private declarations }
    procedure WMNCHITTEST(var Msg:TMessage); message
WM_NCHITTEST;
    procedure WMTRAYICONNOTIFY(var Msg:TMessage); message
WM_TRAYICON;
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  hrgn1:HRGN;
  WinampHWND:HWND;
  reg:TRegistry;
  pos,pos2:byte;
implementation
{$R *.DFM}
procedure CreateSystrayIcon(IconName:string;Id:integer);
var
  trayicon: TNotifyIconData;
  Ic: TIcon;
begin
  Ic := TIcon.Create;
  Ic.LoadFromFile(IconName);
  with trayicon do begin
    cbSize := SizeOf(TNotifyIconData);
    Wnd:=Form1.Handle;
    uID:=Id;
    uFlags:=NIF_ICON or NIF_MESSAGE;
    uCallBackMessage:=WM_TRAYICON;
    hIcon:=Ic.Handle
  end;
  Shell_NotifyIcon(NIM_ADD,@trayicon);
  Ic.Destroy
end;
procedure DestroySystrayIcon(id:integer);
var
  trayicon:TNotifyIconData;
```

```
begin
  with trayicon do
  begin
    cbSize:=SizeOf(TNotifyIconData);
    Wnd:=Form1.Handle;
    uID:=id
  end;
  Shell_NotifyIcon(NIM_DELETE,@trayicon)
end;
procedure ModifySystrayIcon(IconName:string;id:integer);
var
  trayicon:TNotifyIconData;
  Ic: TIcon;
begin
  Ic := TIcon.Create;
  Ic.LoadFromFile(IconName);
  with trayicon do
  begin
    cbSize:=SizeOf(TNotifyIconData);
    Wnd:=Form1.Handle;
    uID:=id;
    hIcon:=Ic.Handle
  end;
  Shell_NotifyIcon(NIM_MODIFY,@trayicon)
end;
function WinampUser(data:integer;id:integer):integer;
begin
  WinampHWND:=findwindow('Winamp v1.x',nil);
  if WinampHWND<>0 then
    result:=SendMessage(WinampHWND,WM_USER,data,id)
  else result:=-1
end;
procedure WinampCommand(Command:integer);
begin
  WinampHWND:=findwindow('Winamp v1.x',nil);
  if WinampHWND<>0 then
    SendMessage(WinampHWND,messages.WM_COMMAND,command,0)
end;
procedure StartWinamp;
var
  WinampExe:string;
begin
  reg:=TRegistry.Create;
  reg.RootKey:=Windows.HKEY_LOCAL_MACHINE;
  reg.OpenKey('Software\CLASSES\Winamp.File\shell\open\command',False);
  WinampExe:=reg.ReadString('');
  reg.Free;
  if WinampExe<>'' then
  begin
    WinampExe:=copy(WinampExe,2,Length(WinampExe)-7);
    WinExec(pchar(WinampExe),SW_SHOWNORMAL)
  end
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  hrgn1:=CreateRoundRectRgn(0,0,210,255,20,20);
  SetWindowRgn(Form1.Handle,hrgn1,False);
  pos:=1;
  pos2:=1
end;
procedure TForm1.WMNCHITTEST(var Msg: TMessage);
begin
  inherited;
  Msg.Result := HTCAPTION
end;
procedure TForm1.WMTRAYICONNOTIFY(var Msg:TMessage);
var
  i:byte;
begin
  if Msg.LParam=WM_LBUTTONDOWN then
  case Msg.WParam of
  1: WinampCommand(40044);
  2: if WinampUser(0,104)=1 then WinampCommand(40046) else
WinampCommand(40045);
  3: WinampCommand(40047);
  4: WinampCommand(40048);
  5: begin
       for i:=1 to 6 do DestroySystrayicon(i);
       Form1.Show;
       pos:=1;
     end;
  end;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if pos=1 then
  begin
  if WinampUser(0,104)=1 then Button1.Caption:='Pause'
    else Button1.Caption:='Play';
  WinampHWND:=findwindow('Winamp v1.x',nil);
  if WinampHWND<>0 then Button4.Caption:='Switch Off Winamp'
    else Button4.Caption:='Switch On Winamp'
  end
  else
  if (WinampUser(0,104)=1)then
    if pos2=1 then
      begin
        ModifySystrayicon('Icon2a.ico',2);
        pos2:=0
      end
    else
  else if pos2=0 then
      begin
        ModifySystrayicon('Icon2.ico',2);
        pos2:=1
      end
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  If WinampUser(0,104)=1 then WinampCommand(40046)
                         else WinampCommand(40045)
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  WinampCommand(40044)
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
  WinampCommand(40048)
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
  WinampHWND:=findwindow('Winamp v1.x',nil);
  if WinampHWND<>0 then WinampCommand(40001) else StartWinamp
end;
procedure TForm1.Button5Click(Sender: TObject);
begin
  CreateSystrayIcon('Icon1.ico',1);
  CreateSystrayIcon('Icon2.ico',2);
  CreateSystrayIcon('Icon3.ico',3);
  CreateSystrayIcon('Icon4.ico',4);
  CreateSystrayIcon('Icon5.ico',5);
  pos:=0;
  Form1.Hide
end;
procedure TForm1.Button6Click(Sender: TObject);
begin
  WinampCommand(40047)
end;
procedure TForm1.Button7Click(Sender: TObject);
begin
  WinampCommand(40144)
end;
procedure TForm1.Button8Click(Sender: TObject);
begin
  WinampCommand(40148)
end;
procedure TForm1.Button9Click(Sender: TObject);
begin
  Close
end;
procedure TForm1.FormDestroy(Sender: TObject);
var
  i:byte;
begin
  if pos=0 then for i:=1 to 5 do DestroySystrayicon(i)
end;
end.
```

Скомпилированный файл программы вы можете взять на моей страничке или на сайте редакции.

Ну вот практически и все, что я хотел рассказать о первом способе реализации ПДУ для Winamp'а. Кроме описания самого ПДУ, я подробно рассказал о работе с иконками в System Tray, что будет полезно для начинающих программистов.

(Окончание следует)

А.ГЕРАЩЕНКО,
г.Лутугино, Луганской обл.

Укрощение shareware-игр

Как часто вам приходилось удалять понравившуюся игрушку только из-за того, что закончилось время ее использования, или когда, поработав пару минут, программа выдает сообщение, что для продолжения игры нужно платить "зеленые денежки"? Произвести оплату в наших условиях довольно проблематично. Попробуем преодолеть возникшую "преграду". Начнем с подбора инструментов.

Во-первых, нам потребуется отладчик SoftIce 3.24 или более поздней версии. Использование более ранних версий не проверялось.

Во-вторых, будет нужен дизассемблер для дизассемблирования исследуемой программы. Для этой цели подойдет Wdasm 8.9.

Также понадобится какой-нибудь hex-редактор. Можно взять Hiew, Qview или любой другой. Этого, пожалуй, хватит на первое время.

Ну-ссс... Приступим. Для примера возьмем игру Ricochet (v1.01 Build 51 by Reflexive Entertainment, Inc). В ней постоянно ненавязчиво предлагают зарегистрироваться, рассказывают о преимуществах зарегистрированной версии и дают поиграть всего два часа, а затем — платите денежки (**рис.1**).

Рис. 1

Я не случайно для примера взял именно эту игру, а не какую-нибудь другую шароварную программку. Обычно в программах весь ввод-вывод основан на функциях API Windows, и там невозможно что-либо скрыть. А Ricochet написан под DirectX, и многие API-функции заменены на аналогичные, но написанные для DirectX. Поэтому их труднее найти и использовать.

Способ, которым мы будем ломать — не из легких, и не является универсальным, т.е. для другого вида защит нужно будет искать другой подход. Но предлагаемый метод дает 100%-ный результат для данного типа защиты и хорошо подходит для ознакомления с приемами работы с SoftIce.

Рис. 2

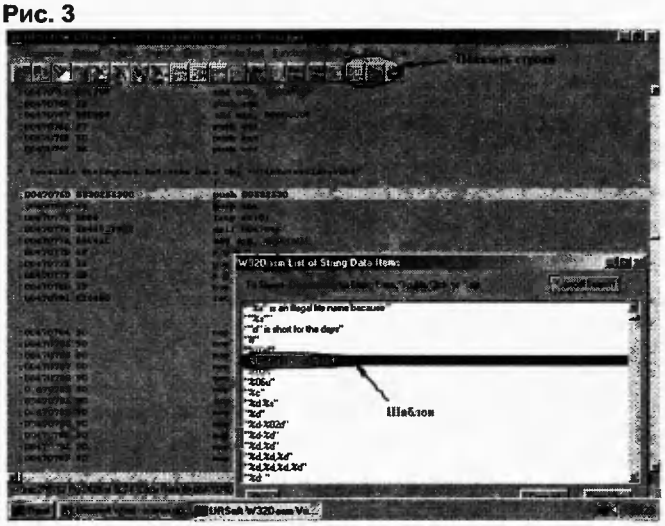
Для начала внимательно рассмотрим рабочее окно исследуемой игры. Мы видим справа таймер, который отсчитывает время до момента X (**рис.2**). Цифры выведены таким образом:

XX:XX:XX

Запомним это на будущее. Теперь нужно найти в программе место, где цифры выводятся на экран именно в таком формате. И вполне вероятно, что где-то в том же районе также будет выполняться проверка, зарегистрирована программа или нет.

Запустим Wdasm и дизассемблируем исполняемый модуль игры с именем Ricochet.exe. Теперь найдем строку в программе, где используется записанный ранее шаблон (**рис.3**). В Windows есть функция wsprintf, которая создает форматированную строку из заданных частей. Одним из параметров этой функции является шаблон, по которому создается строка. И, как вы уже, возможно, догадались, мы будем искать записанную ранее строку.

Рис. 3



Откроем окно со всеми найденными строками и дважды кликнем на строке, имеющей вид:

%02d%s%02d%s%2d

где:

- %02d — две цифры десятичного числа;
- %s — текстовая строка (в нашем случае двоеточие).

Как видите, нужная нам строка склеивается из двух цифр (значение часов), двоеточия, двух цифр (значение минут), двоеточия и двух цифр (значение секунд). На экране появится:

```
* Referenced by a CALL at Addresses:
|:004A5E52   , :004C1C09   , :004C24B2
|
...                                .
:0047076A 57          push edi
:0047076B 50          push eax
:0047076C 56          push esi

* Possible StringData Ref from Data Obj ->"%02d%s%02d%s%02d"
                                        |
:0047076D 6830255300  push 00532530
:00470772 53          push ebx
:00470773 DDD8        ftp st(0)
:00470775 E846FEFFFF  call 004705C0
:0047077A 83C41C      add esp, 0000001C
:0047077D 5F          pop edi
:0047077E 5E          pop esi
:0047077F 5B          pop ebx
:00470780 59          pop ecx
:00470781 C20400      ret 0004
```

Функция, в которой используется заданный шаблон, в программе вызывается из трех точек. Чтобы определить, какая из них нам нужна, запустим Ricochet, и во время игры (когда виден отсчет таймера) войдем в SoftIce, нажав комбинацию клавиш Ctrl-D.

Управление в отладчике производится из командной строки. Вот некоторые команды и управляющие комбинации SoftIce:

- A — вставка кода;
- D — просмотр памяти;
- E — правка памяти;
- R — изменение регистров;
- U — просмотр кода;
- F5 — запуск программы;
- F8 — пошаговое выполнение с заходом в тело функции;
- F10 — пошаговое выполнение без захода в тело функции;
- F11 — выход из функции.

Для получения более подробной информации о командах отладчика используйте команду H.

Установим точку останова по адресу 0047076D:

```
bpx 0047076D
```

Продолжим выполнение программы, нажав клавишу F5. Программа снова выйдет в Softice. Теперь трассируем программу до выхода из функции, и видим приблизительно такой код:

```
:004A5E49 8B4C244C          mov ecx, dword ptr [esp+4C]
:004A5E4D 51                push ecx
:004A5E4E 8D4C2410          lea ecx, dword ptr [esp+10]
:004A5E52 E8A9A8FCFF        call 00470700; Отсюда вызывалась
:004A5E57 6A00              push 00000000; функция с шаблоном
:004A5E59 6A02              push 00000002
:004A5E5B 8D4C2414          lea ecx, dword ptr [esp+14]
:004A5E5F E82CADFCFF        call 00470B90
```

Отключим точку останова:

```
bd 0
```

и запустим программу. Теперь мы знаем, что продолжать исследование программы нужно с адреса 004A5E52. Переключимся на Wdasm. Двигаемся вверх и обнаруживаем такой код:

```
1):004A5DEB E8309FFFFF      call 0049FD20; Вызов интересной функции
2):004A5DF0 84C0            test al, al; Если возвращается 0,
3):004A5DF2 7420            je 004A5E14 ; то переход на вывод времени,
4):004A5DF4 8B8E98020000    mov ecx, dword ptr [esi+00000298]; иначе -
5):004A5DFA 8B5154          mov edx, dword ptr [ecx+54]; то, что нас
6):004A5DFD 8D4C240C        lea ecx, dword ptr [esp+0C];
интересует
7):004A5E01 8BC2            mov eax, edx
8):004A5E03 8954244C        mov dword ptr [esp+4C], edx
9):004A5E07 50              push eax

* Possible StringData Ref from Data Obj ->"%d:%02d:%02d"
                                        |
10):004A5E08 68844E5300     push 00534E84
11):004A5E0D E87EA8FCFF     call 00470690
12):004A5E12 EB43           jmp 004A5E57; Продолжить выполнение.
13):004A5E14 ...
```

Чтобы программа работала как зарегистрированная, нужно, чтобы в любом случае выполнялся код по адресу 004A5DF4. Можно заменить строки 1, 2, 3 на MOV AL,1 и цепочку NOP'ов, но это может не сработать — в игре может быть несколько таких проверок. Поэтому продолжим.

Нам показалась интересной функция по адресу 0049FD20. Перейдем туда:

```
* Referenced by a CALL at Addresses:
|:00406217   , :004A187A   , :004A5DEB   , :004A5EF3   , :004A61FF
|
:0049FD20 A09D125E00        mov al, byte ptr [005E129D];Если этот байт
:0049FD25 84C0              test al, al  ; не равен 0,
:0049FD27 7512              jne 0049FD3B ; то выйти из функции, иначе
:0049FD29 C6059D125E0001    mov byte ptr [005E129D], 01; сделать его 1.
:0049FD30 E81B000000        call 0049FD50
:0049FD35 A29C125E00        mov byte ptr [005E129C], al
:0049FD3A C3                ret

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0049FD27(C)
|
:0049FD3B A09C125E00        mov al, byte ptr [005E129C]; Здесь всегда
:0049FD40 C3                ret; должна быть 1
```

Из этого фрагмента кода видно, что по адресу 005E129D всегда будет 1, а возвращаемое значение зависит от байта 005E129C. Переключимся на Ricochet и войдем в SoftIce. Посмотрим содержимое памяти по адресу 005E129C:

```
E 005E129C.
```

Ага! Ноль, а нам нужна единица. Нажмем F5 — YES!!! Таймер считает вперед! Игра зарегистрирована. Но не спешите радоваться — при следующем перезапуске она будет снова выдавать предложение зарегистрироваться.

Такой принцип защиты, при котором в памяти находится флажок, отвечающий за регистрацию, характерен для большинства шароварных программ. Чтобы полностью сломать защиту, нужно при каждом запуске программы устанавливать флажок в единицу. Сделать это можно, поменяв в программе все команды установки флажка в единицу.

Переключимся на Wdasm и зададим для поиска строку

```
005E129C
```

Из всех команд, обращающихся к флажку, нам нужны только те, которые его устанавливают:

```
MOV byte ptr[005E129C],xxx
```

Здесь xxx может быть регистром или константой. Использование других команд маловероятно, потому что все программы пишутся на языках высокого уровня, и компиляторы используют ограниченное число команд для компиляции.

В программе обнаружились три такие команды, при этом управление по адресу 49FD3A практически никогда не передается (если не верите, поставьте точку останова и убедитесь), поэтому мы исключим и эту команду. Теперь остались две команды установки флажка. Сделаем два изменения в программе.

Первое изменение:

```
:0049FC82 E889030000        call 004A0010
:0049FC87 8B0DD4135D00      mov ecx, dword ptr [005D13D4]
1):0049FC8D 881D9C125E00    mov byte ptr [005E129C], bl
2):0049FC93 881D9D125E00    mov byte ptr [005E129D], bl
:0049FC99 53                push ebx
:0049FC9A 8B11              mov edx, dword ptr [ecx]
```

В этом фрагменте нужно заменить строки 1 и 2 на

```
:0049FC8D 66C704259C125E000101   mov word ptr [005E129C], 0101
:0049FC97 B301mov bl, 01
```

Или в исполняемом модуле:

```
9FC8D: 88 1D 9C 12 5E 00 88 1D 9D 12 5E 00
```

на

```
9FC8D: 66 C7 04 25 9C 12 5E 00 01 01 B3 01
```

Обратите внимание, что в дизассемблированном тексте программы команды находятся по адресам в памяти, а в EXE-шнике — относительно начала файла (по смещению). Смещение можно найти, например, при помощи команды поиска Qview, задав для поиска образец из дизассемблированного текста.

И второе изменение:

```
* Referenced by a CALL at Addresses:
|:00493382   , :004A0C83   , :004A455B   , :004A6BAF   , :0Q4BE7D8
|:004C5CB6
|
:0049FE40 A09D125E00        mov al, byte ptr [005E129D]
:0049FE45 84C0              test al, al
:0049FE47 7511              jne 0049FE5A
1):0049FE49 C6059D125E0001  mov byte ptr [005E129D], 01
2):0049FE50 E8FBFEFFFF      call 0049FD50
```


```
3):0049FE55 A29C125E00       mov byte ptr [005E129C], al

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0049FE47(C)
|
:0049FE5A 8A0D9C125E00       mov cl, byte ptr [005E129C]
:0049FE60 33C0               xor eax, eax
:0049FE62 84C9               test cl, cl
:0049FE64 0F94C0             sete al
:0049FE67 C3                 ret
```

В этом фрагменте нужно заменить строки 1, 2, 3 на

```
:0049FE49 E802FFFFFF                call 0049FD50
:0049FE4E 66C704259C125E000101      mov word ptr [005E129C],
0101
:0049FE58 90                        nop
:0049FE59 90                        nop
```

Или в исполняемом модуле:

```
9FE49: C6 05 9D 12 5E 00 01 E8 FB FE FF FF A2 9C 12 5E 00
```

на

```
9FE49: E8 02 FF FF FF 66 C7 04 25 9C 12 5E 00 01 01 90 90
```

ВНИМАНИЕ!!! Перед изменением файла Ricochet.exe сделайте его резервную копию, например, скопировав в другой каталог.

Теперь запустим hex-редактор и исправим в файле Ricochet.exe байты:

```
9FC8D: 88 1D 9C 12 5E 00 88 1D 9D 12 5E 00
```

на

```
9FC8D: 66 C7 04 25 9C 12 5E 00 01 01 B3 01
```

и

```
9FE49: C6 05 9D 12 5E 00 01 E8 FB FE FF FF A2 9C 12 5E 00
```

на

```
9FE49: E8 02 FF FF FF 66 C7 04 25 9C 12 5E 00 01 01 90 90
```

Запускаем Ricochet и видим, что игра зарегистрирована. На сэкономленные 10 USD можете купить пива.

Основная работа проделана, но если вы хотите поделиться своими успехами с друзьями, то нужно будет сделать крек.

Его мы напишем на языке C. Мне нравится компилятор Borland C++ 5.02, но можно использовать и другой.

Выполним подготовительный этап к написанию крека. Для начала нужно сравнить резервную копию и модифицированный EXE-файл, и различия занести в промежуточный файл. Это можно сделать и вручную, но следующая программка справится с этим быстрее:

```
#include <windows.h>
#pragma argsused
int APIENTRY WinMain(
HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine,
int nCmdShow)
{
HANDLE file1,file2,res;
BYTE bt1,bt2;
DWORD readed;
int i,total1,total2;
file1=CreateFile("c:\\temp\\ricochet.exe",
GENERIC_READ, //Нахождение резервной копии
FILE_SHARE_READ,
NULL,
OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL,
NULL);
file2=CreateFile(
"c:\\program files\\ricochet\\ricochet.exe",
GENERIC_READ, //Нахождение EXE-шника игры
FILE_SHARE_READ,
NULL,
OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL,
NULL);
res=CreateFile("rico.dat",//Выходной файл данных
GENERIC_WRITE,
FILE_SHARE_WRITE,
NULL,
CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL,
NULL);
total1=GetFileSize(file1,NULL);
total2=GetFileSize(file2,NULL);
if(total1==total2)
       {
   for(i=0;i<total1;i++)
       {
              ReadFile(file1,&bt1,1,&readed,NULL);
      ReadFile(file2,&bt2,1,&readed,NULL);
      if(bt1!=bt2)
       {
         WriteFile(res,&i,4,&readed,NULL);
         WriteFile(res,&bt1,1,&readed,NULL);
         WriteFile(res,&bt2,1,&readed,NULL);
         }
       }
   }
CloseHandle(file1);
CloseHandle(file2);
CloseHandle(res);
return 0;
}
```

Эта программка побайтно сравнивает наш исполняемый модуль игры и его резервную копию, и найденные различия записывает в файл Rico.dat.

Теперь собственно крек:

```
#include <windows.h>
#define REPLACE 1
typedef struct _Replace
{
DWORD Offset;
BYTE Src;
BYTE Dst;
}Replace;
#pragma argsused
int APIENTRY WinMain(
HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine,
int nCmdShow)
{
Replace *RepData;
HANDLE file;
HRSRC RepRes;
DWORD offset,cb;
BYTE src;
BOOL flag=FALSE;
int i;
file=CreateFile("ricochet.exe",//изменяемый файл
GENERIC_WRITE|GENERIC_READ,
FILE_SHARE_WRITE|FILE_SHARE_READ,
NULL,
OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL,
NULL);
if(file!=INVALID_HANDLE_VALUE)
       {
       RepRes=FindResource(hInstance,
MAKEINTRESOURCE(REPLACE),
RT_RCDATA);
       RepData=LockResource(LoadResource(
hInstance,RepRes));
       for(i=0;i<SizeofResource(hInstance,RepRes)/
sizeof(Replace);i++)
       {
      SetFilePointer(file,RepData[i].Offset,NULL,FILE_BEGIN);
      ReadFile(file,&src,1,&cb,NULL);
      if(!flag&&src!=RepData[i].Src)
       {
         if(MessageBox(NULL,"Неправильные данные.\nПродолжить?",
```

```
"Внимание!",MB_YESNO|MB_ICONQUESTION)==IDNO)
       break;
         flag=TRUE;
         }
      SetFilePointer(file,RepData[i].Offset,NULL,FILE_BEGIN);
      WriteFile(file,&RepData[i].Dst,1,&cb,NULL);
      }
       CloseHandle(file);
   MessageBox(NULL,
"Файл исправлен",
"Сообщение",
MB_OK|MB_ICONINFORMATION);
   }
else
       {
   MessageBox(NULL,
"Не могу открыть файл",
"Ошибка!",
MB_OK|MB_ICONWARNING);
   }
```

```
return 0;
}
```

В проект также включен файл Rico.rc:

```
/*************************************************************

rico.rc

produced by Borland Resource Workshop

*************************************************************/

#define REPLACE 1

REPLACE RCDATA "Rico.dat"
```

Эти программы можно использовать для создания крекков других программ. Нужно только изменить имена взламываемого файла, резервной копии и выходного файла.

А.ТУГБАЕВ,
Удмуртская р-ка, п.Игра.

Welcome to Life 2

В [1] была опубликована игровая программа М.Фахриева "Welcome to Life" на Basic для "ZX-Spectrum". Но данная версия программы не подойдет пользователям PC, на которых установлен стандартный Basic (по крайней мере, интерпретатор QBasic сообщает об ошибке в строке 30). Дело в том, что QBasic не знает такую последовательность команд — AT PAUSE, NEW. Заменить их можно на последовательность LOCATE, SLEEP, RUN 10. Причем параметр оператора SLEEP указывает на приостановление программы на заданное количество секунд. Например, команда SLEEP 10 задает остановку на 10 секунд. Кстати, удобнее использовать вместо оператора INPUT цикл DO...LOOP, например, DO: a $ = INKEY: LOOP WHILE a $ = "". Если при использовании оператора INPUT для сбрасывания кубиков нужно нажать клавишу "R", а затем ENTER, то при использовании цикла достаточно лишь одного нажатия "R". Ниже приведен листинг программы "Welcome to Life", опитимизированной под Qbasic.

```
Begin:  ' Welcome To Life. New version (' - аналог REM)
        CLS
        LOCATE 1,1 : PRINT "Welcome To Life"
        SLEEP 5          ' Пауза 5 сек.
        LOCATE 2,1: PRINT "You wanna play withs syndicate,"
        LOCATE 3,3: PRINT "or you playing on the street (1 or 2)?"
Play:DO: a$=INKEY$: LOOP WHILE a$="" ' Пока клавиша не нажата,
                                     ' опрашивать клавиатуру.
        IF a$="1" THEN GOTO Syndicate
        IF a$="2" NHEN GOTO Street
        GOTO Play
Syndicate:CLS
        LOCATE 3,1: PRINT "Let's get death..."
In1:    DO: a$=INKEY$: LOOP WHILE a$=""
        IF a$="R" OR a$="r" THEN GOTO Game1
        GOTO In1
Game1:y=INT(12*RND)            ' Добавим переменные y и m
        m=INT(12*RND)          ' RANOOMIZE TIMER
        IF y<m THEN GOTO Yourself
        LOCATE 6,2: PRINT "Sometimes I loose..."
        LOCATE 7,2: PRINT m
Write:PRINT "Continue? (Y/N)"
In2:    DO: a$=INKEY$: LOOP WHILE a$=""
        IF a$="Y" OR a$="y" THEN GOTO Syndicate
        IF a$="N" OR a$="n" THEN GOTO Begin
        IF a$=CHR$(27) THEN GOSUB Exit ' Выход при нажатии Esc
        GOTO In2
Street: CLS
        LOCATE 11,10: PRINT "Is on"
        LOCATE 5,1: PRINT "Player 1"
In3:    DO: a$=INKEY$: LOOP WHILE a$=""
        IF a$="R" OR a$="r" THEN GOTO Game2
        GOTO In3
Game2:  g=INT(12*RND)
        LOCATE 6,1: PRINT G
        LOCATE 6,10: PRINT "Player 2"
In4:    DO: a$=INKEY$: LOOP WHILE a$=""
        IF a$="R" OR a$="r" THEN GOTO Game3
        GOTO In4
Game3:  j=INT(12*RND)' Также можно добавить RANDOMIZE TIMER
        LOCATE 7,10: PRINT j
PRINT "Continue? (Y/N)"
In5     DO: a$=INKEY$: LOOP WHILE a$=""
        IF a$="Y" OR a$="y" THEN GOTO Street
        IF a$="N" OR a$="n" THEN GOTO Begin
        IF a$=CHR$(27) THEN GOSUB Exit
        GOTO In5
Yurselt:        LOCATE 6,2: PRINT "You player yourselt"
        LOCATE 7,2: PRINT Y
        GOTO Write
Exit: BEEP ' В отличие от SPECTRUM, BEEP используется без
           ' параметров, для написания мелодии - SOUND x%,y%
        LOCATE 5,20: PRINT "Exit. You are sure? (Y/N)"
In6:    DO: a$=INKEY$: LOOP WHILE a$=""
        IF a$="Y" OR a$="y" THEN END ' END - аналог STOP
        IF a$="N" OR a$="n" THEN LOCATE 5,20: PRINT"  ": RETURN
        GOTO In6
```

Для некоторого упрощения программы, вместо фрагмента кода

```
InX:    DO: a$=INKEY$: LOOP WHILE a$=""
        IF a$="R" OR a$="r" THEN GOTO GameY
        GOTO InX
GameY:
```

следует использовать

```
InX: DO: a$=INKEY$: LOOP WHILE a$<>"R" OR a$< >"r"
GameY:
```

Кстати, в первоначальном варианте программы обнаружились две ошибки: одна синтаксическая — в строке 230 вместо разделителя операторов ";" написано";)" и вторая — пропущено слово THEN в строке 210.

Литература

1. М.Фахриев. Welcome to Life. — Радиомир. Ваш компьютер, 2002, №11, С.44.

С.РЮМИК,
г.Чернигов.

Чип-"невидимка" в "PlayStation"

(Окончание. Начало в №6/2003)

Варианты подключения МЧ

За восемь лет производства PSX их было изготовлено более 90 млн. (!) экземпляров. Если представить, сколько из них подверглось "чиповке", то станет ясно, почему насчитывается несколько десятков вариантов подключения МЧ и еще столько же вариантов прошивок контроллеров.

Единого центра, устанавливающего стандарты на МЧ, не существует. Зато не перечесть фирм и фирмочек, инженеров-профессионалов и любителей, которые плодотворно освоили это ремесло.

Большим подспорьем для многих стал Интернет, где размещены сотни сайтов, посвященных теме установки, приобретения и самостоятельного изготовления МЧ. Если проанализировать размещенные в Интернете сведения, то можно сделать интересный вывод — рассказы о "самой-самой лучшей" прошивке МЧ, как бы ни хвалили себя разработчики, являются мифом.

Преодолеть региональную защиту можно разными способами, но с одинаковым результатом, используя различное количество проводов соединения и разные типы микросхем контроллеров. Нюансы сводятся разве что к скорости обхода защиты, но разница в пару секунд практически незаметна и несущественна.

Итак, рассмотрим схемы подключения МЧ в разных моделях PSX, не ставя перед собой сверхзадачу охватить абсолютно все вариации технических решений. Условно все МЧ будут разделены на 3 поколения.

Первое поколение МЧ. На **рис.3** приведена схема взаимосвязей стандартного чипа Райдера в моделях SCPH-100x...SCPH-700x. Из внутренних узлов PSX изображены только самые необходимые для понимания процессов. Назначение сигналов перечислено в **табл.2**. Там же даны варианты соединения выводов МЧ по 10-, 6-, 5- и 4-проводным схемам.

Табл. 2

Название сигнала	Назначение	Функция	Число проводов			
			10	6	5	4
VCC	Питание	Питание +5 В или +3,5 В	+	+	+	+
GND	Общий	Общий	+	+	+	+
DATA	Выход	Информационный сигнал, содержащий код обхода защиты	+	+	+	+
GATE	Выход	Сигнал блокировки информации с диска	+	+	+	+
PCLK	Вход	Тактовая частота 4...4,5 МГц	+	+	+	-
RES	Вход	Сигнал сброса от кнопки "RESET"	+	+	-	-
SYN1... SYN4	Вход	Четыре синхросигнала, по которым определяют момент выдачи сигнала DATA.	+	-	-	-

Исторически первым появился 10-проводный вариант. Он был применен в знаменитом чипе "Hong Kong", который содержал однократно программируемый контроллер PIC16C54 ("Microchip Technology").

Сигналы VCC, GND, PCLK, RES необходимы для нормального запуска и функционирования контроллера. Сигналы SYN1...SYN4 служат датчиками, по которым контроллер вычисляет момент выдачи информации по линии DATA. Сигнал GATE блокирует данные регионального кода, которые поступают с диска при инициализации системы.

Эксперименты, проведенные с чипом "Hong Kong", показали, что точная привязка сигнала DATA к сигналам SYN1...SYN4 не требуется. Достаточно ввести определенную задержку после включения питания и далее генерировать сигнал DATA в бесконечном цикле, пока ЦП не прочитает его код. Так появился 6-проводный вариант включения, использующий контроллеры PIC16C54 и PIC16C84 ("Microchip Technology").

Сократить еще одну линию связи позволил 5-проводный вариант. В нем отсутствует входной сигнал RES, поскольку применяются микросхемы Z86E0208PSC ("Zilog"), PIC12C508, PIC12C509 ("Microchip Technology"), AT90S2313 ("Atmel"), имеющие внутреннюю цепь начального сброса.

Исследование диапазона, в котором может изменяться сигнал DATA, показало, что кварцевая стабилизация формирования его длительности не обязательна. Достаточно применить внутренний RC-генератор и отказаться от внешнего тактового сигнала PCLK. Такая возможность имеется в контроллерах PIC12C508 (509). Получившийся 4-проводный вариант работает полностью автономно. На входы контроллера не поступает ни одного входного сигнала, зато имеются два выходных и два провода питания.

Можно ли еще больше сократить количество соединительных проводов? Скотт Райдер доказал, что можно, построив экспериментальный 3-проводный МЧ с сигналами VCC, GND, DATA для "американской" модели PSX. Однако этот эксперимент доказал лишь работоспособность концепции. На практике 3-проводный МЧ работает неустойчиво из-за сложности синхронизации цифрового потока.

На **рис.4** приведена структурная схема алгоритма работы стандартного чипа Райдера, которая соответствует 6-, 5- и 4-проводным вариан-

"PlayStation" SCPH-100x, SCPH-550x, SCPH-700x

Рис. 3

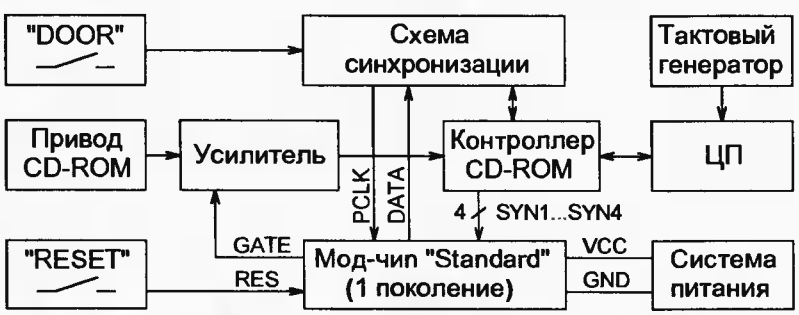
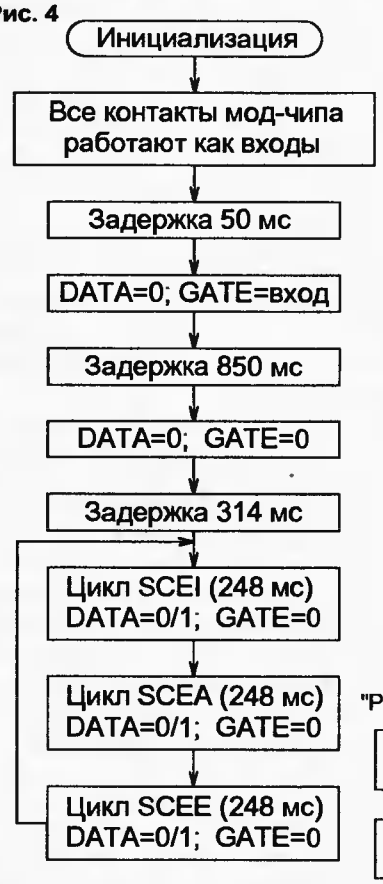


Рис. 4



там соединения. Временные диаграммы выходных сигналов DATA и GATE изображены на

Рис. 5

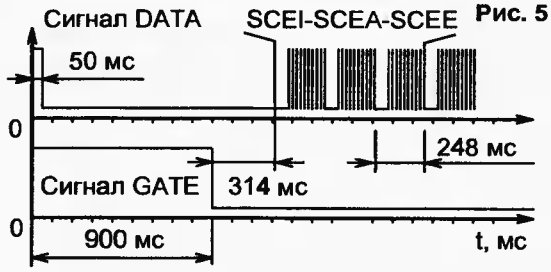


Рис. 6

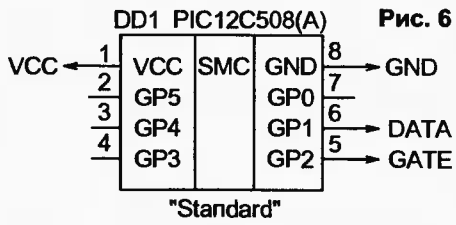


рис.5. Коды прошивки контроллера PIC12C508(A) приведены в **листинге 1**, схема его включения — на **рис.6**, точки подключения проводов к разным моделям PSX — в **табл.3**.

"PlayStation" SCPH-750x, SCPH-900x

Рис. 7

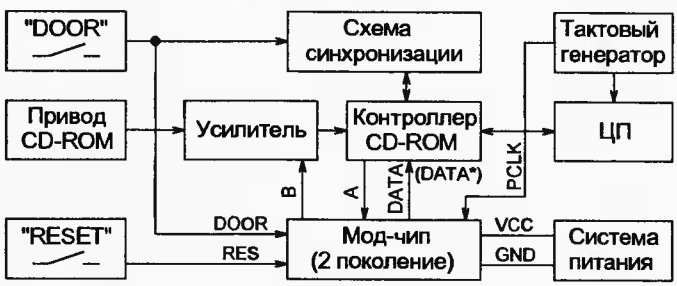


Табл. 3

Название сигнала	SCPH-100x	SCPH-550x	SCPH-700x	SCPH-750x	SCPH-900x	SCPH-10x "PSone" PM-41	PM-41(2)
VCC	78M05F:3 (+5 В), CN602:3 (+3,5 В)						
GND	78M05F:2						
DATA	SC4309xxPB:17			CXD2938Q:42		-	-
GATE	"O82":2	"118":7	"A3821":7	-	-	-	-
PCLK	SC4309xxPB:15		-	CY2081SL:6		-	-
RES	CN602:7	CN602:5				-	-
DATA*	-	-	-	CXD2938Q:42			CXD2941:35
A	-	-	-	CXD2938Q:5			CXD2941:76
B	-	-	-	"A2575N":17		-	-
DOOR	-	-	-	SC4309xxPB:19			
A18	-	-	-	-	-	M53403IE:31	
D2	-	-	-	-	-	M53403IE:31	
STL	-	-	-	-	-	SC4309xxPB:44	

Листинг 1. Коды прошивки PIC12C508 (1 поколение)

```
:100000002500000B320C2A00F90C2B000000EB023B
:10001000060AEA02040A03082700480C0309040C34
:10002000280000D0231092C006C02080C2900FB0C81
:10003000600040C03092C03F90C26000306FB0C34
:100040000307F90C0600040C0309E9021B0AF90C6A
:10005000600080C0309AD02E802110AE7020D0AC6
:100060000308E20153084308450849085308430 8B8
:0C00700045084108530843084508450 8AE
:10020000C20C0200FF0C060002092604FD0C0600C9
:10021000110C27000209E7020A0B4604F90C06003C
:10022000060C27000209E702120B0E0C03096D00F1
:040230000C09170B93
:021FFE00EA0FE8
:00000001FF
```

Второе поколение МЧ. На **рис.7** показана схема подключения МЧ в приставках SCPH-750x, SCPH-900x. Назначение сигналов перечислено в **табл.4.** Классификация вариантов, как и прежде, дана через количество соединительных проводов. Следует только помнить, что МЧ с одинаковым числом проводов в SCPH-100x и SCPH-900x функционально не взаимозаменяемы.

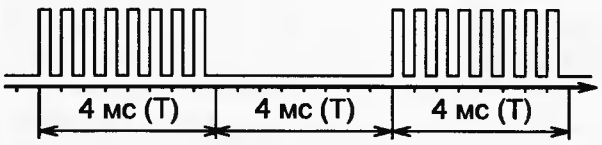
Сигналы RES и DOOR имеют разную полярность, в частности, активный уровень для RES — нулевой, для DOOR — единичный. Оба сигнала выполняют одну и ту же функцию, но используются раздельно — в зависимости от типа контроллера. Сигнал PCLK необходим, если контроллер не имеет встроенного задающего RC-генератора. Сигналы PCLK, RES, DOOR являются необязательными и могут отсутствовать.

Главное отличие МЧ второго поколения (по сравнению с первым) состоит в более сложной структуре сигнала DATA и в отсутствии сигнала GATE. Это связано с введением в схему "PlayStation" СБИС CXD2938Q (позиционное обозначение — IC732). Сигнал DATA, как и прежде, содержит огибающую с информационным кодом циклов SCEI, SCEA, SCEE, но теперь в каждом "единичном"такте имеются синхронизирующие "врезки" (**рис.8**).

Табл. 4

Название сигнала	Назначение	Функция	Число проводов 7	6	5	4
VCC	Питание	Питание +5 В или +3,5 В	+	+	+	+	+
GND	Общий	Общий	+	+	+	+	+
DATA	Выход	Информационный сигнал, содержащий код обхода защиты	+	+	+	+	-
DATA*	Выход	Информационный сигнал с "врезками" и кодом обхода защиты	-	-	-	-	+
A	Вход	Синхросигнал "врезок" 8 кГц	+	+	+	+	+
B	Выход	Повторение "врезок" 8 кГц с "открытым коллектором"	+	+	+	+	-
PCLK	Вход	Тактовая частота 4...4,5 МГц	+	+	+	-	-
RES	Вход	Сигнал сброса от кнопки "RESET"	+	-	-	-	-
DOOR	Вход	Сигнал поднятия крышки "DOOR"	-	+	-	-	-

Рис. 8 Сигнал DATA* с "врезками" 8 кГц

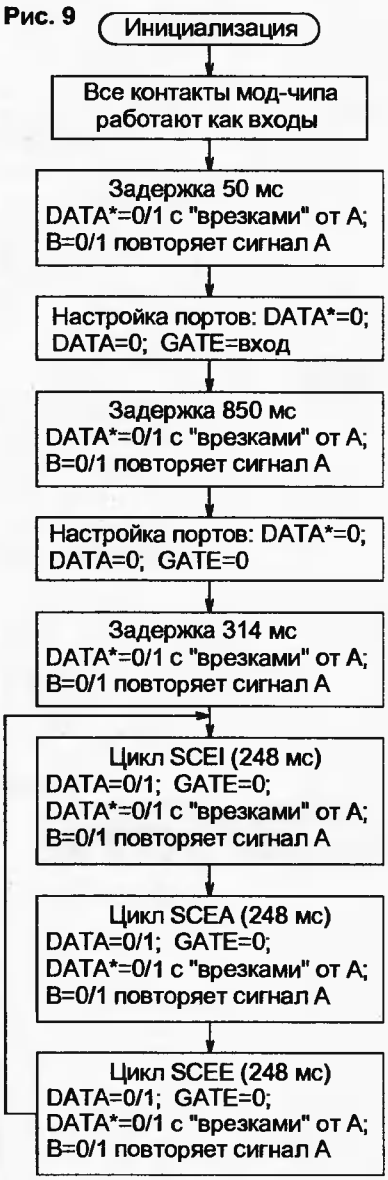


"Врезки" можно сформировать двумя путями. Хронологически первым стали применять способ внешнего добавления синхроимпульсов. Для этого сигнал A частотой 8 кГц, поступающий от контроллера CD-ROM, транслируется без изменений на выход B через схему "с открытым коллектором". Одновременно контроллер МЧ формирует обычный сигнал DATA (без "врезок") и подает его в PSX. Смешивание сигналов DATA и B происходит внутри СБИС CXD2938Q.

Второй способ заключается во внутреннем добавлении синхроимпульсов. Для этого контроллер МЧ смешивает при помощи операции "логическое И" входной сигнал A частотой 8 кГц с обычным сигналом DATA. Полученный выходной сигнал DATA* содержит "врезки" и подается в контроллер CD-ROM приставки. Таким образом, можно сократить число соединительных проводов по сравнению с первым вариантом, так как не требуется сигнал B. Единственный "недостаток" этого метода — в том, что он был разработан позже по времени и не получил достаточного распространения.

На **рис.9** приведена структурная схема алгоритма работы 4-проводного МЧ с внутренними "врезками". Заметим, что частота "врезок" — величина не постоянная, и может меняться в процессе загрузки игры. Более того, встречаются МЧ с начальной частотой синхроимпульсов не 8 кГц, а 44 кГц.

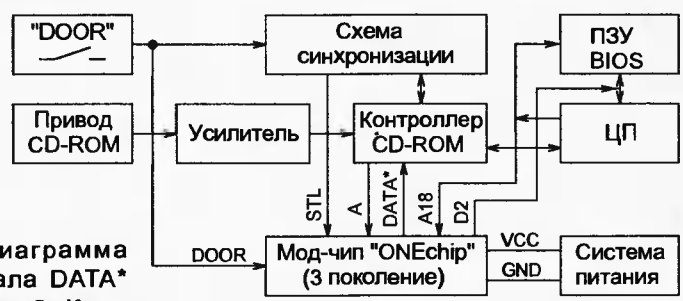
Рис. 9



Листинг 2. Коды прошивки PIC12C508 (2 поколение)

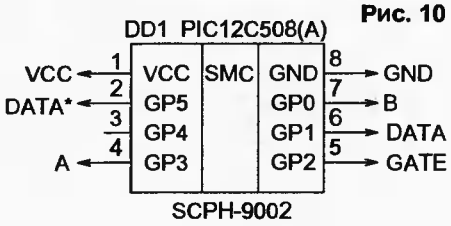
```
:100000002500000B320C2A00410C2B0066060E0561
:1000100066070E04F80C0E07D80C2E07D80C260025
:100020000E020600EB02060A0000000000000000BD
:1000300000000000EA02040A03082700480C030934
:1000400040C28000D0243092C006C02080C290046
:100050002E050E020600040C03092C0303062E05D0
:1000600003072E040E020600040C0309E9022D0A00
:100070002E040E020600080C0309AD02E802220A53
:10008000E7021E0A0308E201530843084508490820
:1000900053084308450841085308430845084508E4
:10020000C20C0200DF0C2E000600F80C26000209CA
:1002100060426042E040E020600110C2700020913
:10022000E7020F0B46044E040E020600060C2700E0
:100230000209E702180B0E0C03096D001D091D0BC6
:021FFE00EA0FE8
:00000001FF
```

"PSone" SCPH-10x



Временная диаграмма выходного сигнала DATA* соответствует рис.8. Коды

Рис. 10



прошивки контроллера PIC12C508(A) приведены в **листинге 2**, схема его включения — на **рис.10**, точки подключения проводов к разным моделям PSX — в табл.3.

Третье поколение МЧ. На **рис.11** показана схема подключения МЧ "ONEchip" к "европейской" модели SCPH-102 "PSone". Назначение сигналов перечислено в **табл.5**.

Полный 8-проводный вариант позволяет обходить как двойную защиту региональной принадлежности, так и запускать любые "антимод"-игры. Выходные сигналы DATA* и A имеют структуру, аналогичную МЧ второго поколения (с "врезками").

Обход новой региональной защиты выполняют сигналы A18 и D2. Контроллер МЧ анализирует состояние адресного входа A18 ПЗУ и, при определенном перепаде уровней на нем, через строго отмеренный промежуток времени формирует короткий "логический ноль" на линии D2 шины данных. В это время ЦП "PSone", считывая информацию с шины данных ПЗУ, получает ее в искаженном виде. Этого иска-

Рис. 11

Табл. 5

Название сигнала	Назначение	Функция	Число проводов 8	6	4
VCC	Питание	Питание +5 В или +3,5 В	+	+	+
GND	Общий	Общий	+	+	+
DATA*	Выход	Информационный сигнал с "врезками" и кодом обхода защиты	+	+	+
A	Вход	Синхросигнал "врезок" 8 кГц	+	+	+
A18	Вход	Разряд A18 шины адреса ПЗУ	+	+	-
D2	Выход	Разряд D2 шины данных ПЗУ	+	+	-
STL	Вход	Сигнал наличия "антимод"-игры	+	-	-
DOOR	Вход	Сигнал поднятия крышки "DOOR"	+	-	-

жения достаточно, чтобы пройти второй этап региональной защиты. Поскольку такая проверка происходит однократно, то в дальнейшем цепь D2 отключается от управления МЧ.

Сигнал STL служит стелс-индикатором начала проверок в "антимод"-играх. Логика разработчиков игровых программ следующая. Они знают, что в МЧ первых выпусков сигнал DATA генерируется постоянно, в бесконечном цикле. Проверка регионального кода, изначально заложенная в ПЗУ PSX, осуществляется только при загрузке диска, и в дальнейшем не проводится.

"Антимод"-игры имеют в своем составе специальную подпрограмму, которая периодически запускается в работу и выполняет аналогичные функции по проверке кода. Однако эта подпрограмма отслеживает не структуру импульсов кода, а просто их наличие. Если импульсы имеются, значит, МЧ установлен в приставке, если отсутствуют — значит, приставка фирменная.

"ONEchip", так же как и его предшественники, генерирует сигнал DATA* в бесконечном цикле, но через каждые 8 мкс проверяет логический уровень сигнала STL. Если он переходит из "1" в "0", то выдача сигнала DATA прекращается, и наоборот, если из "0" в "1", то возобновляется. Таким образом контроллер обходит "антимод"-защиту.

Сигнал DOOR, в отличие от МЧ второго поколения, не предназначен для начального сброса контроллера. Он выполняет более интеллектуальную функцию, превращая "ONEchip" в многорежимный ("multi-mode") МЧ. Сигнал DOOR меняет свое состояние каждый раз при поднятии или закрытии крышки доступа к диску: DOOR="1" — крышка открыта, DOOR="0" — крышка закрыта. Варьируя время нахождения крышки в открытом состоянии, можно установить один из трех режимов работы:

- "Stealth-mode" — основной режим, при котором обходятся все защиты. Крышка должна быть закрыта не позже чем через 2,5 с после включения питания;
- "Dino-mode" — запасной режим. Крышка должна быть закрыта (открыта) при включении питания, затем через 2,5 с открыта на время от 0 до 2,5 с (подбирается пользователем экспериментально по устойчивости ввода игры). При этом сигнал DATA меняет свое логическое состояние каждые 100 мс;
- "No ModChip" — режим отключения МЧ. Крышка должна быть открыта в течение первых 5 или более секунд после подачи питания. Этот режим полезен при диагностике неисправностей, например, если существует подозрение, что МЧ отрицательно влияет на работоспособность PSX. Более того, отключение МЧ бывает полезным, если используются внешние устройства типа "Action Replay/PS Hacker".

Кроме 8-проводного режима включения "ONEchip", в Интернете встречаются упоминания о 6-проводном режиме, когда сигналы DOOR и STL отключаются от контроллера, и вместо них постоянно подаются "логические нули". При этом будут вводиться все игры, за исключением "антимод". Практической пользы от такого варианта включения не видно, равно как и от 4-проводного, при котором отсутствуют сигналы DOOR, STL, A18, D2. В этом случае будут вводиться все фирменные и нефирменные диски, но только лишь стандарта PAL, и не имеющие "антимод"-защиты.

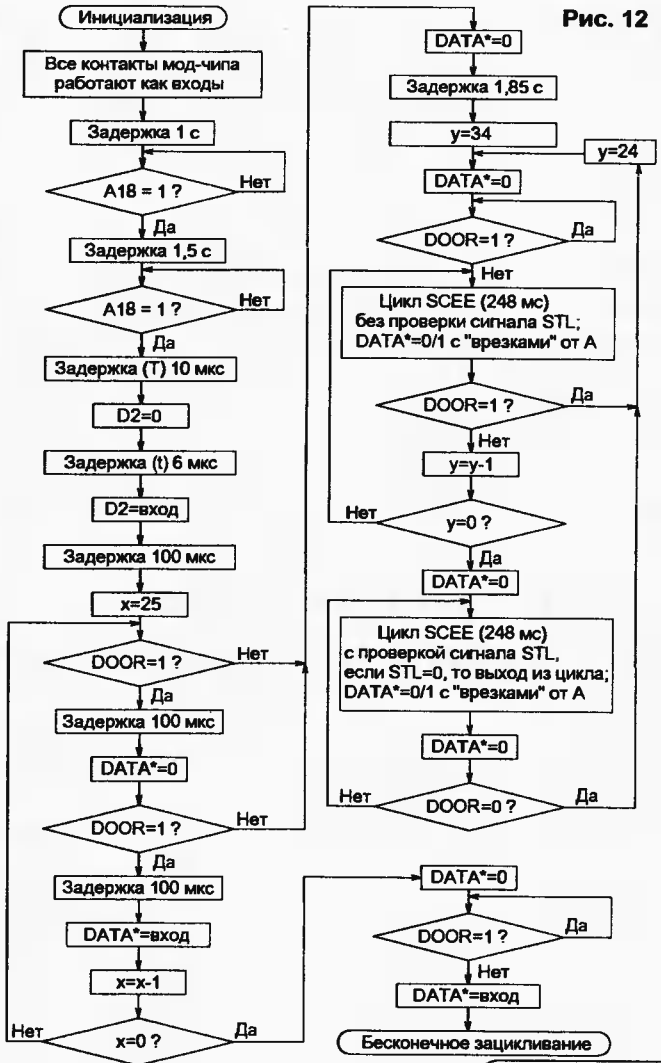


Рис. 12

На **рис.12** приведена структурная схема алгоритма работы 8-проводного "ONEchip". Схема будет полезна при попытке повторения его на другом типе контроллера. Временные диаграммы сигналов A18 и D2 изображены на **рис.13**, а сигнала DATA* — на рис.8. Коды прошивки контроллера PIC12C508(A) при-

Листинг 3. Коды прошивки PIC12C508 (3 поколение, "ONEchip")

```
:0C0000002500FF0C0600820C0200F20A32
:100010004F084E0845086308680869087008200 8FA
:0C0020005608310 82E0830083008200 86F
:10008000030C2F00FF0C2E000008320C2A00C80CB5
:100090002B00A6076E00EB02490AEA02470A000895
:1000A000FD0C0600F90C26000008FF0C0600FF0CF2
:1000B000260000 08FD0C0600F90C2600040C460A78
:1000C000FD0C0600040C2A00690C2B00FB0C4607F3
:1000D000F90C2600A6076E00EB02660AFB0C460729
:1000E000F90C26000000EA02640A00086D00040C06
:1000F000280011068 80A4009120C3200040C460937
:10010000EF02850A0E06B00A4009F2027E0A8C0A46
:10011000480C4609FF0C2F000D02B1092C006C029F
:10012000090C2900960A2C030307980A60099A0A09
:100130005A090000EF029F0A0E06B00A4009E902C0
:10014000930A5A09EF02A70A0E06B00A40095A0993
:10015000EF02AD0A0E06B00A4009AD02E8028C0AB1
:1001600000 08E2015308430845085086 40C4609A5
:10017000190C32000607CD0A640C4609500906071F
:10018000CD0A640C46095509F202BA0A500906065E
:10019000C70AFF0C06000000CB0A50091E0C2700FE
:1001A0004509E702D00AF90C2600FD0C0600070CF1
:1001B000270045 09E702D90A220C3000E10A180C91
:1001C000300050090606E20AFF0C310076090606E7
:1001D000DF0AF002E60A5009710076095009 606A6
:1001E000DF0AED0A140C33004509F302F40A66072E
:1001F000F70A86041D0C34004509F402FC0A660760
:10020000FF0A0000000000000000000000000000E5
:10021000EF0C0600000000000000000FF0C0600CC
:02022000B60A1C
:021FFE00EA0FE8
:00000001FF
```

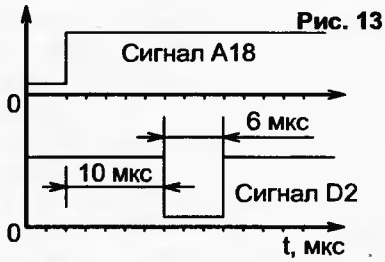


Рис. 13

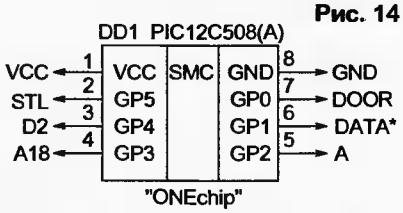


Рис. 14

ведены в **листинге 3**, схема его включения — на **рис.14**, точки подключения проводов к моделям "PSone" с процессорными платами PM-41 (выпуска 2000, 2001 годов) и PM-41(2) (выпуска 2002 года) — в табл.3.

В приставки прежних моделей (до "PSone") также можно ввести стелс-режим. Для этого необходим МЧ на контроллере PIC12C508(A) с прошивкой "Stealth 2.8a". На странице http://www.angelfire.com/tn/psxchips/installation.html имеются фотографии установки этого чипа в моделях SCPH-100x ("Old"), SCPH-100x ("New"), SCPH-550x, SCPH-750x, SCPH-900x. По адресу http://www.chipstation.it/supporto/download/st2p8a.zip можно бесплатно скачать HEX-коды его прошивки для японской, американской и европейской моделей PSX.

Литература

1. Рюмик С. "PlayStation" — история развития. — Радио, 2002, №11, С.21-23.

2. Рюмик С. "PlayStation" — ремонт блока адаптации. — Радио, 2000, №4, С.26-28, №5, С.31-33.

Доработка динамиков компьютерных колонок

И.ДЗЮБА,
г.Великий Новгород,
E-mail: il@mail.ru

Хочу продолжить тему создания компьютерных колонок из картонных коробок, которые можно собрать за 2 часа... [1]. В этот раз речь пойдет о доработке динамиков.

В своих АС я использовал динамики 5ГДШ-4 (5ГДШ-14), имеющие по паспорту сопротивление 4 Ом. Диаметр диффузора у них, если считать и гофр, составляет 14 см, что, в принципе, вполне достаточно для относительно неплохого воспроизведения низкочастотной части диапазона. Я имею в виду только площадь диффузора. Но на НЧ эти динамики звучат "не очень". При детальном рассмотрении одного из них было обнаружено, что ход диффузора очень маленький, хотя катушка может двигаться в достаточно больших пределах. Причина оказалась в слишком жестком подвесе самого диффузора. Да, именно гофр был слишком жестким, и не давал двигаться диффузору динамика в достаточных пределах. Центрирующая шайба, конечно, тоже имела определенную жесткость, но значительно меньшую, чем гофр. Было принято решение сделать подвес более мягким и гибким. Можно, конечно, поменять весь гофр на какой-нибудь резиновый, но это сложно, да и затратно как по времени, так и по материалам. Согласитесь, динамики того не стоят (напомню, что обошлись они мне по 15 рублей за штуку). Поэтому было решено пропитать гофр каким-нибудь составом, чтобы увеличить мягкость подвеса.

В ходе экспериментов над одним из динамиков, его гофр был условно разделен на 3 части, и каждая часть пропитана тем, что было подходящего под рукой — машинным маслом, литолом и маликотовой смазкой. Результаты были следующие.

Машинное масло не дало никакого результата, так как мягкость подвеса осталась практически той же. К тому же, так как бумага хорошо впитывала масло, оно быстро растекалось по поверхности диффузора, а не только гофра. Что, разумеется, нехорошо.

Литол показал себя лучше. Мягкость подвеса чуть-чуть повысилась, растеканий было меньше, но они все-таки появлялись, так как в литоле есть какое-то количество маслопродуктов.

Лучшей пропиткой оказалась маликотовая смазка — она не давала растеканий, не высыхала, не затвердевала и очень здорово смягчала гофр. На ней я остановился, и ею был обработан другой динамик.

Эксперимент дал очень неплохие результаты. При подключении доработанного и недоработанного динамиков к компьютеру и при использовании программы "Audio Test" в качестве генератора, были сделаны следующие выводы.

На "бывшей" нижней частоте, т.е. где-то в районе 72 Гц разница в ходе диффузоров динамиков была не очень заметной, но при уменьшении частоты она плавно увеличивалась в полтора раза. При подходе к частоте 25...30 Гц разница в колебаниях диффузора составляла 2 раза! Что примерно эквивалентно увеличению мощности на этих частотах в 4 раза!

На слух разница была очень даже заметной. Например, на недоработанном динамике частоту 27...30 Гц было просто не слышно, даже при акустическом оформлении. Жесткость подвеса была слишком большой. А вот на доработанном ее было слышно настолько хорошо, что в 2 часа ночи (когда я проводил тестирование) мне пришлось уменьшить громкость, чтобы не мешать соседям. Результат доработки определенно меня порадовал! АЧХ колонок стала еще более широкой в НЧ-диапазоне. На средних и высоких частотах отличий в звуке замечено не было, что очевидно, так как сами диффузоры динамиков изменениям не подвергались.

Следующий тест был проведен на реальных музыкальных произведениях. Результат был крайне интересен. Звук "исправился". Т.е. он стал, так сказать, более "правильным", более реальным, таким, каким он и должен бы быть. Оказа-

лось, что из-за жесткого подвеса динамика у него есть резонансная частота где-то в районе 72...90 Гц, и все частоты, попадающие в этот диапазон, усиливаются больше, чем остальные, т.е. динамик "окрашивает" звук в этом диапазоне, как бы усиливая и выделяя. На частотах ниже этих происходил достаточно быстрый спад АЧХ, и, как следствие, их почти не было слышно. Поэтому и появлялся призвук "китайских" магнитофончиков, бубнящий звук "консервной банки". После доработки этот резонанс пропал. Динамик стал воспроизводить больше низких частот и без резонансного подъема, звук стал более достоверным, "правильным". Впоследствии мною были доработаны и размещены в колонках еще шесть таких динамиков.

При прослушивании АС сначала было психологически трудно перейти от качества звучания "До", к качеству звучания "После". Дело в том, что звук стал казаться тише, так как раньше он "подусиливался" самим динамиком в области резонансной частоты и звучал на этих частотах более громко. Теперь резонанс и бубнящий звук пропали, расширилась НЧ-полоса, и на слух звучание кажется более тихим, но достоверность звучания объективно выше. К этому надо привыкнуть.

Еще несколько замечаний и советов.

Я наносил смазку на гофр просто пальцем — как показала практика, это вполне безопасно. После пропитки, особенно маликотом, гофр не сразу размягчится, а примерно через 45...60 мин. Для ускорения этого процесса и улучшения равномерности растяжения гофра я его... тренировал!

Для этого он подключался к выходу звуковой карты компьютера (Speaker), и с помощью вышеупомянутой программы "Audio Test" на выход подавался синусоидальный сигнал частотой 39 Гц максимальной амплитуды. Колебания диффузора динамика при этом получались достаточно большой амплитуды и частоты. Для динамика такой режим вполне безопасен, так как он рассчитан на мощность 5 Вт, а выход звуковой карты отдает только 3 Вт. При этом динамик и без ваших усилий достаточно быстро и равномерно размягчает свой гофр.

Также не стоит подавать большую выходную мощность на динамик с помощью ручки "Громкость", чтобы компенсировать разницу в громкости звука. Помните, что амплитуда движений диафрагмы стала в 2 раза большей. А значит, стало легче повредить катушку!

А вот центрирующую шайбу "размягчить" мне ничем не удалось. А жаль. Тоже хотелось...

Просто она ничем не пропитывается, а местные "подрезания" не дают желаемого результата. Если у кого-то получится — поделитесь технологией.

Кроме того, доработанные динамики лучше размещать уже в более хороших корпусах — например, деревянных или фанерных. Картон уже просто не может хорошо разделить переднюю и заднюю полуволны, излучаемые динамиком, отдача падает, соответственно, изменения. не так заметны.

Еще я сделал АС, в которую было встроено два динамика, благо, коробка позволяла. Динамики соединил последовательно, общее сопротивление стало 8 Ом. Думал, упадет громкость, так как на 8 Ом отдаваемая мощность меньше. Оказалось, что нет — громкость осталась той же, даже немного возросла. Увеличение общей площади диффузора сказывается в большей степени, чем увеличение сопротивления. Правда, увеличились и направленные свойства АС.

Итог — простыми средствами получаем, чуть ли не НЧ-динамик из широкополосника стоимостью всего 15 рублей. Так же можно доработать и другие отечественные и импортные широкополосные динамики.

Литература

1. И.Дзюба. Компьютерные колонки за 2 часа. — Радиомир. Ваш компьютер, 2003, №4, С.40.

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

Рациональное распределение ресурсов

Театр начинается с вешалки, а компьютер — с BIOS'а. Компьютерная поговорка

Очень часто производители материнских плат встраивают в свои изделия различные устройства — сетевые карты, модемы, звуковые карты, видеокарты и т.д. Многие из этих интегрированных устройств могут быть не задействованы пользователем, но, вне зависимости от того, используются они или нет, эти устройства все равно потребляют системные ресурсы. Если их отключить, можно высвободить в системе немало ресурсов (прерываний, адресов ввода/вывода, каналов DMA). Кроме того, отключение незадействованных устройств ускорит процесс загрузки операционной системы.

В этой статье я хочу поделиться опытом определения и устранения неисправностей, связанных с понижением производительности видеосистемы. Приведу один наглядный пример того, как из-за неправильного распределения ресурсов может существенно снизиться производительность видеосистемы. Возможно, кто-то из пользователей уже столкнулся или еще столкнется с подобной проблемой. Но для начала расскажу все по порядку.

Дело в том, что я периодически тестирую производительность своего компьютера. В частности, для тестирования видеосистемы я применяю программу 3DMark2000. Кроме того, я — любитель выжать из своего "железа" все возможное, т.е. занимаюсь разгоном, или overclocking'ом, как вам будет угодно. И вот как-то, в один из обычных дней, при тестировании своей разогнанной видеокарты я обнаружил резкое

падение производительности. Вместо прежних 74,9 кадров в секунду показания программы 3DMark2000 "застряли" на 19 FPS как вкопанные. Диагностика с помощью DirectX v8.1 тоже положительного эффекта не дала — трехмерный кубик вращался вяло и медленно. При просмотре видеофильмов изображение периодически подергивалось, и поэтому пришлось в настройках декодера снизить качество воспроизведения. Казалось, что ускорители 2D- и 3D-графики вышли из строя. Закралась мысль о порче видеокарты из-за разгона. "Доразгонялся" — подумал я про себя. Что я только ни делал, для того чтобы видеокарта снова заработала как надо. Переустанавливал драйвер видеокарты, DirectX v8.1, 3DMark2000 и даже полностью "сносил" операционную систему. Снимал видеокарту, чистил ее контакты, "вылизывал" лопасти вентилятора видеокарты. Все было напрасно, ничто не помогало. Мне ничего не оставалось, как только смириться с подобным положением дел, набраться терпения и копить денежки для новой видеокарты.

Но вскоре я одолжил на время другую, точно такую же видеокарту и протестировал ее работоспособность на своем компьютере. И что вы думаете? Вторая карта показала точно такую же заниженную производительность, как и моя. Стало ясно, что дело совсем не в порче карты — не могли же обе карты иметь один и тот же дефект. Я стал анализировать, в чем же заключается причина низкой производительности видеокарты. После того как стало ясно, что дело не в карте, а чем-то другом, решение проблемы пришло довольно быстро. Направил меня на верное решение не кто иной как компания Microsoft. Я вспомнил, что в папке Windows есть файл **display.txt**, в котором имеется раздел "Конфликты прерываний при использовании видеоадаптеров PCI". В нем говорится о конфликтах видеокарт с другими устройствами из-за прерываний. Причем все сказанное в этом файле относится не только к видеокартам, установленным на шине PCI, но и к наиболее распространенным на сегодняшний день картам, подключаемым к шине AGP.

Прочитав информацию из этого файла, я понял, что для нормальной работы видеокарты необходимо, чтобы было выделено прерывание, которое не "делилось" бы с другими устройствами. Просмотрев таблицу прерываний, я увидел, что видеокарте назначено прерывание IRQ11, и это же прерывание используется USB-контроллером. Сначала я попытался назначить другое прерывание для видеокарты, но поскольку в системе свободных неиспользуемых прерываний не оказалось, задуманное осуществить не удалось. Тогда я решил в Setup BIOS на время полностью отключить USB-контроллер. После загрузки операционной системы, сразу начал тестировать производительность видеокарты. При тестировании я испытал взрыв положительных эмоций, поскольку теперь показания в программе 3DMark2000 меня полностью устраивали.

Не устраивало меня лишь то, что нельзя было использовать USB-устройства. Стал я думать, как решить и эту проблему, потому что теперь невозможно было, например, пользоваться USB-сканером. Поскольку в моей материнской плате интегрировано два USB-контроллера, и каждый из них использует прерывание, а это означает, что можно использовать до четырех USB-устройств, то один неиспользуемый USB-контроллер я отключил в Setup BIOS. В итоге конфликт ресурсов у меня благополучно разрешился, видеокарта и USB-сканер стали использовать не одно общее, а каждый свое отдельное прерывание. Производительность видеокарты нормализовалась.

Дополнительно к вышесказанному, рекомендуется для решения проблем, вызванных конфликтами прерываний, устанавливать операционные системы Windows 2000/XP, при условии что в BIOS имеется и включена поддержка APIC (см. **рисунок**). Аббревиатура APIC расшифровывается как Advanced Programmable Interrupt Controller — улучшенный программируемый контроллер прерываний. APIC, в отличие от своего предшественника PIC, дает возможность назначать устройствам не 16 прерываний, а 24.

Заключение

При возникновении описанных в статье проблем поиск решения надо начинать именно с контроля над распределением прерываний, а также отключить в Setup BIOS или на вкладке "Устройства" все неиспользуемые контроллеры, порты и другие устройства с целью освободить для нужд системы как можно больше ресурсов. Если в "материнке" интегрирован звуковой контролер AC'97, то в случае, если джойстик и MIDI-порт не используются, имеет смысл их отключить в Setup BIOS, в разделе Integrated Peripherals. Правильное и рациональное распределение ресурсов поможет решить проблемы с прерываниями, подобные описанной, и увеличить производительность "заторможенных" узлов и компонентов. А значит, в результате возрастет быстродействие всей системы в целом.

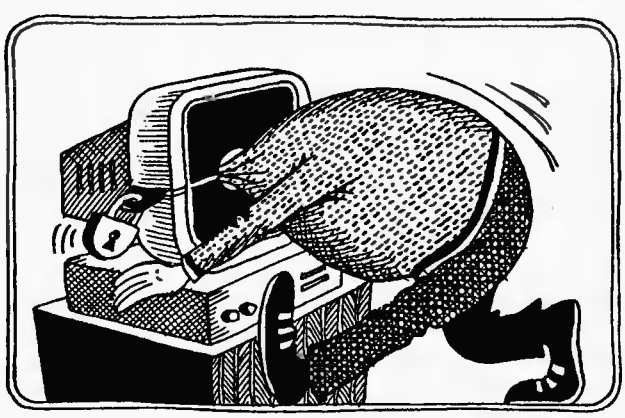
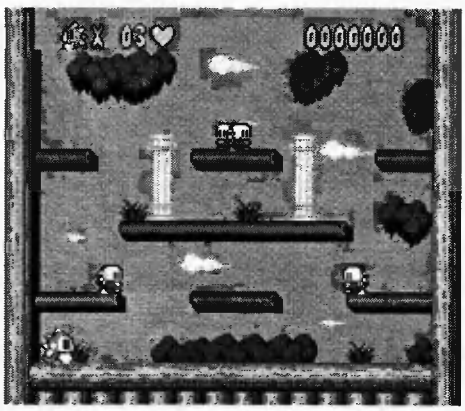


Рисунок О.Попова

Восьмибитное чудо

Р.КАРПАЧ,
E-mail: RomanKarpach@yandex.ru
www.chizhovka.narod.ru

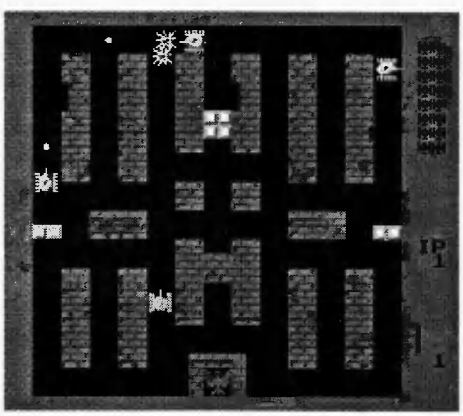
Было время, когда все компьютерные игры были добрыми и веселыми. В них не было того насилия, которое присуще их современным братьям. Однако это не мешало нам допоздна засиживаться перед телевизором с любимой приставкой, проходя бесконечные лабиринты "Принца Персии", спасая мир от злобных мутантов в "Контре", да и просто весело проводить свой досуг. Теперь никто из нас не будет покупать приставку, чтобы поиграть в "Марио". Да и зачем? Оказывается, любимые приставочные игры можно запускать и на вашем PC! В этом нам, как всегда, поможет Интернет.



История "Nintendo"

"Nindendo", "Lifa", "Dendy", "Ufa", "Сюбор" — у нее было множество названий. Приставка, которая покорила весь мир. Игровая платформа, ставшая самой популярной за всю историю развития развлекательной компьютерной индустрии. В разных странах ей давали разные имена: **"Nintendo"** — так ее звали в США; **"Famicom"** — это имя ей присвоили японцы; **"Dendy"** (слоненок) — название и фирменный логотип придумала фирма, распространявшая приставку на территории СНГ; в конце концов, просто **NES**. "Восьмибитка", как ласково в народе называли приставку, являлась самой популярной консолью, хотя на рынке уже в начале 90-х появились шестнадцатибитные "Sega". Ведь, по большому счету, именно с этой платформы началась история видеоигр. Игровая видеоприставка появилась в Японии в 1982 г., после чего была представлена всему миру в 1984 г. Но для начала следует обрисовать картину рынка видеоигр тех времен. С момента появления первого аркадного автомата прошло уже практически пять лет. К тому времени игровые залы стали обыденностью, а автоматы с играми "Defender" и "Galaxian", "Guruin" и "Bubble" можно было найти в любой забегаловке, в любом кинотеатре.

Такого понятия как игровая приставка тогда еще не существовало, и, пожалуй, никто даже представить себе не мог, что когда-то появятся домашние игровые машины. Фирма **Nintendo of Japan** к тому времени уже имела некоторый опыт производства видеоигр. На разработку приставки у нее ушло около года. И вот **"Pasofami"** (японское название "NES") явилась на суд игровой общественности. Покупатели, мягко говоря, несколько опешили. А потом начался настоящий бум на продукцию Nintendo. Приставки расходились по всему миру гигантскими партиями. Разработчики со всего света подписывали контракты на производство игр под эту приставку. А пользователи были на седьмом небе от счастья, сидели у телевизоров как приклеенные, играя во все подряд, боготворя героя **"Марио"** и ведя бои с ордами вражеских танков в **"Battle City"**.



И ведь было за что хвалить ф.Nintendo — такого успеха на видеорынке еще никто не добивался. Грамотная рекламная политика, забота о пользователе и большой объем выпущенных игр сделали свое дело — "NES" побила все рекорды по продажам, и на данный момент, даже на фоне новых 64-битных платформ, все еще является рекордсменом по количеству проданных экземпляров.

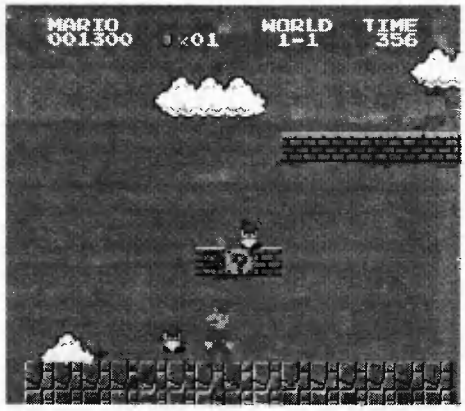
Вы, наверное, и не знаете, что представляет собой изнутри "Nintendo". Приставка снабжена процессором с тактовой частотой около 1,8 МГц. Ее выход пришелся на время резкого скачка популярности персональных компьютеров, поэтому приставки переживали серьезный кризис. Тем не менее, простота в обращении и новый дизайн сделали свое дело — эта платформа буквально в течение года стала самой популярной в мире. Этому, в частности, способствовала игровая поддержка, большой ассортимент игр и регулярно выходящие новинки. Приставка поддерживала разрешение 256x224, что было эталоном для 8-битных приставок, и палитру из 52 цветов, из которых 26 одновременно выводились на экран. Все это поддерживалось 5-канальным звуком. Официально для платформы вышло более 3000 игр. Хотя на Западе приставка не продается уже с начала 90-х, наши братья из дружественного Китая продолжают продавать их у нас в стране.

Компьютер и "Nintendo"

Время пролетело незаметно. И в один прекрасный день, увидев свою любимую приставку на рынке, я подумал, ведь есть же эмуляторы старых компьютеров, таких как Спектрум, Атари, Корвет?! Почему же их не может быть и для нашей любимой приставки? Придя домой, гложимый воспоминаниями, я зашел в Интернет. После продолжительных поисков мне открылись несколько сайтов на которых находились огромные коллекции некогда всеми любимых игр. По адресу http://emulators.wallst.ru/nintendomul.htm имеется внушительная коллекция эмуляторов "DENDY". Программы сами по себе небольшие. Для вас я бы порекомендовал эмулятор под названием **NESTICLE**. Хотя он и работает в среде DOS, но его функциональность и возможности могут только радовать. С помощью этой программы мы можем записывать музыку из любимых игр, менять раз-

решение экрана, делать скриншоты высокого качества. Самым же главным достоинством NESTICLE является то, что он не искажает звук. Хорошо, но ведь эмулятор — это только полдела. Следует также скачать восьмибитные игры. Все они хранятся в файлах с расширением *.NES. На сайте www.apm.boom.ru вы найдете все, во что вы когда-то играли. Например, любимого "МАРИО".

Единственное что может вызвать головную боль, так это нежелание некоторых игр запускаться. Поэтому



скачайте несколько разных эмуляторов. Ведь бывает и так, что игра не работает с NESTICLE, но работает с JNES. ftp.cs.albany.edu/pub/kesecki/nestcle — этот ftp-архив тоже должен порадовать вас своей большой коллекцией. В общей сложности мне удалось найти 94 игры.

Теперь, с чувством глубокого душевного удовлетворения, я могу сказать, что любимая "восьмибитка" живет не только в наших сердцах, но и на жестких дисках наших компьютеров!

КОМПЬЮТЕРНЫЕ ХРОНИКИ

Конкурс "Твоя Игра"

В декабре 2002 г. редакционной коллегией синклеровской газеты "Абз@ц" были подведены итоги конкурса "Твоя Игра". Конкурс был организован и проведен по инициативе редакции газеты и при активной спонсорской поддержке многих синклеристов, неравнодушных к состоянию рынка программного обеспечения для Sinclair-совместимых компьютеров и заинтересованных в существовании этого рынка в пределах правового поля — при соблюдении авторского права не на словах, а на деле.

Опровергая известную поговорку, этот "блин" не вышел "комом". География участников конкурса оказалась чрезвычайно широкой — от Бреста, что находится в Беларуси, до приморского Владивостока. Игровые программы, принявшие участие в конкурсе, принадлежали к разным жанрам. В основном были представлены аркадные игры (как в "чистом" виде, так и в смеси с другими жанрами), но присутствовала также "тактика" и "экономика".

Победителей конкурса определили сами читатели газеты "Абз@ц". Точнее, те из них, кто приобрел в редакции газеты легальный экземпляр дискеты с играми конкурса. С подробной статистикой подведенных итогов можно познакомиться в №14 газеты "Абз@ц". Приведем краткие сведения об играх, занявших призовые места.

1 место — игра "Abe's Mission (escape) part 1", созданная командой (c)Brothers. Жанр — аркадно-адвентюрный. Сюжет — разнообразно управляя достаточно нетрадиционным главным персонажем, нужно выручить из большой беды всех его соплеменников.

2 место (с минимальным отрывом от игры-победительницы) — "Курьер 2. Потерянный мир". Авторы — (c)Perspective group. Жанр — аркадно-логический. Сюжет — очередное спасение мира с попутным решением очень интересных и хитроумных головоломок.

3 место — "Super Mario Bros. Playable demo". Авторы — (c) Gogin production. Жанр — "чистокровный" аркадный. Сюжет — до боли знакомый очень многим геймерам по игровым приставкам. "Супер Марио" — одна из самых популярных компьютерных игр всех времен и народов.

По итогам конкурса победители были награждены дипломами и солидными денежными призами. Но самый главный итог — решение о проведении конкурса "Твоя Игра-2003". Иными словами, конкурс на лучшую компьютерную игру для "Sinclair ZX-Spectrum" продолжается. Организаторы приглашают к контактам спонсоров и авторов — как творческие коллективы, так и отдельных разработчиков, независимо от гражданства, пола и возраста, которые могут представить на конкурс до 1 ноября 2003 г. никогда ранее не публиковавшиеся игры любого жанра. Почтовый адрес для контактов: Россия, 160035, г.Вологда, а/я 136, Шушков Александр Дмитриевич.

Е.ИЛЯСОВ,
г.Балашов, Саратовской обл.

BestView 2.15

В апреле этого года вышла очередная версия BestView — универсальной программы для просмотра и запуска файлов на "ZX-Spectrum". Что же появилось нового?

Главное нововведение — появилась возможность просмотра HTML-файлов. Перед просмотром происходит их преобразование в текст. При этом удаляются теги, комментарии, стили, скрипты, обрабатываются наиболее распространенные &-последовательности.

Если полученный текст содержит длинные, не умещающиеся на экране строки, то для более удобного просмотра можно воспользоваться разбиением длинных строк (клавиша "W") или форматированием текста (клавиша "R").

Так как BestView может работать с кодировками ALT, WIN, KOI, то и просматриваемые HTML-файлы могут быть в любой из этих кодировок.

Еще одно улучшение — немного более удобным сделан выбор файла в истории просмотренных файлов. Теперь можно одним нажатием клавиши переходить к первому или последнему элементу истории.

Кроме того, как обычно, исправлены несколько замеченных ошибок.

При всем этом размер программы не только не возрос, но, напротив, уменьшился на два сектора (тут сыграло свою роль и предпринятое "облегчение" заставки программы).

Остается только добавить, что BestView 2.15, как и другие мои программы, можно скачать с моей web-страницы: http://www.ivr.da.ru. Если вы по какой-либо причине не можете это сделать, напишите мне, и я вышлю BestView по e-mail. Также при желании вы можете получать по e-mail сообщения о выходе новых версий BestView.

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

Blade of darknees

М.ВАЛЬПА,
E-mail: valpa@vpost.ru

Очередная игра-хит прибыла к нам из Испании. Мифический экшн, повествующий о борьбе четырех воинов Света со вселенским Злом. Выбрав для игры классическую тему и не создавая ничего оригинального, мастера-игроделы "выдали на гора" грандиозную повесть извечной борьбы Тьмы и Света, настолько углубившись в это противостояние, что даже игру в каком-то неведомом порыве окрестили "Лезвием Тьмы". Да, символика еще та, но, усмирив чувства, будем рассказывать обо всем плавно и как можно более подробно.

Что вообще представляет собой эта игрушка? Это смесь экшена и RPG. В принципе, в игре несомненно имеется огромное количество битв с разными тварями и так называемыми "боссами", но они не лезут на тебя толпами изо всех щелей, как, к примеру, в "Diablo", а нападают небольшими группами, встречающимися у вас на пути. И система боя не состоит из двух команд — бей и отбивай, а представляет собой сложную, разветвленную систему, состоящую из набора различных комбинаций и супер-ударов. Но хоть крови и много, думать тоже приходится, правда, чаще всего вся дума сводится к вопросам типа "Как открыть эту дверь?" или "Куда идти дальше?", иногда возникает вопрос: "Где эта чертова руна?". "Что еще за руна?" — наверно, спросите вы. Сейчас объясню. Прохождение игры основывается на одном принципе — вы должны дойти до самого главного "босса" и убить его. Для этого вам нужно пройти ряд миссий и "покрошить" кучу тварей и мелких "боссов". Вроде бы, все легко и просто. Однако не тут-то было, главного "босса"можно убить только специальным оружием (мечом Иоанны), а оно, в свою очередь, дается за шесть рун и четыре камня, собранных по ходу игры. Эти руны не просто так валяются на уровнях — их нужно искать в самых секретных местах и потаенных углах. Правда, если по ходу игры вы собрали не все руны, то вас возвращают на те уровни, где вы оные руны пропустили.

Камни же находятся гораздо легче. Дело в том, что разработчики подложили игрокам маленькую "свинью". В самом начале игры они мельком упомянули о рунах и о том, что те где-то валяются, и только в самом конце вам говорится, что руны обязательно нужно собрать. Про камни же, наоборот, все разъясняется заранее, и в основе половины миссий лежит поиск камней.

На кого же возлагается такая ответственная и отнюдь не легкая задача? Для этого вам предлагается четыре разных героя с индивидуальными характеристиками — рыцарь, карлик, амазонка и варвар. Для каждого из них существуют свои виды вооружения. Карлик орудует одноручными молотами и топорами; рыцарь специализируется на одноручных мечах; варвару подходит любое двуручное оружие; а амазонка владеет пиками и алебардами. Ну и конечно же, карлик машет двуручным оружием так, будто оно весит больше всей планеты, а из варвара выходит специалист по пикам и алебардам, как из слона балерина. Также у каждого героя для его оружия есть набор супер-ударов или финтов. Кроме обычного оружия, в игре существуют артефакты. Конечно же, самый главный артефакт — это меч Иоанны, но также существует много средних артефактов, таких как огненный меч, ледяной молот, меч королевы и другие. Для каждого из героев существует только одна (первая) отличающаяся миссия, все остальные (начиная со второй и дальше) — общие для всех.

Графика игры просто потрясает. Во-первых, текстуры очень богаты и красочны. Во-вторых, тени и освещение настолько приближены к реальности, что иногда теряешь границу между реальностью и игрой. В-третьих, движок игры не только привлекателен, но еще и позволяет вам в буквальном смысле расчленять ваших врагов.

В общем, игра удалась на славу — красивая графика, навевающая страх музыка и масса насыщенного действия не дадут вам спать спокойно.

C&C Generals

А.ВЕНДИЛОВСКИЙ,
г.Минск.

... В бездонной глубине космоса, в вечной темноте, обратив свои "стеклянные глаза" на Землю, безучастно висел спутник, улавливая малейшие изменения на поверхности планеты. С такой высоты все казалось мелким и не важным, даже если это несло смерть!

...Мы не можем позволить какой-то группе террористов диктовать нам условия! Мы — великая нация, и наше право и обязанность — нести демократию всему миру...

... Посмотрите на них, этих американских снобов, зажравшихся и зарвавшихся, возомнивших себя единственными обладателями истины, готовых смести любого, кто посмеет встать у них на пути, на их пути к мировому господству. Наши воины готовы умереть за свободу! Мы будем сражаться до последнего бойца! И когда их дома будут гореть, пусть знают, что кровь их жителей на их руках!...

... Наша страна и наше правительство единодушно осуждают военные действия и любые попытки разрешить конфликт военными методами, но предупреждает, что в случае угрозы для нашего народа готовы применить против любого агрессора весь арсенал имеющихся средств, вплоть до ядерного оружия!...

***

Новый мир, старая вражда, пока не останется только один...

Передовой отряд танковой бригады ворвался на улицы города, за ним — несколько бронемашин, поливающих свинцовым дождем разбегающихся в панике жителей. От разрушенных нефтяных вышек шел черный смрадный дым, который застилал глаза, и от которого першило в горле. Командир батальона GLA не верил в легкую победу — бойцы Поднебесной империи умели сражаться до последнего патрона. Дома превращались в маленькие крепости, а неповоротливые танки, зажатые в узких городских проходах, становились легкой добычей ракетчиков. Легкие треки были практически бессильны перед прицельным огнем из окон. В тылу уже были развернуты подземные переходы, через которые с основной базы уже подтягивались войска и первые инженеры, готовые приступить к постройке базы. Командир нервно поглядывал в небо — в любой момент там могли появиться "МиГи" или тяжелая авиация американцев. Схватка только начиналась, скоро, очень скоро тут будут тяжелые танки китайцев, перед толщиной брони и мощностью пушек которых пасовал любой танк их армии. А позади них будет шествовать тяжелая дальнобойная артиллерия, которая уничтожит любую цель на большом расстоянии, превратив сотни квадратных метров земли в один пылающий ад... И, наверно, они применят тактические ядерные снаряды малой мощности...

Тяжелые взрывы и ядовитые радиоактивные облака просто прокладывали "просеку" в городской "чаще", на корню подавляя сопротивление обороняющихся. Да и забывать о сверхдальней артиллерии тоже нельзя — кто забывал о таких вещах, недолго прохлаждался на этом свете. Но у него было что противопоставить им — мобильность и скорость его армии, яростная атака ракетных треков и химические снаряды, наводившие ужас на врагов. Фейдакины-смертники, бросающиеся на врага с взрывчаткой на поясе, умирая, забирали их с собой. Как можно сражаться с теми, кому безразлична собственная жизнь, и кого интересует только смерть противника?

... Тяжелый бомбардировщик вынырнул из облаков практически над базой, сбросив несколько кассетных бомб по заводам и казармам, превращая аккуратные домики в груду щебня. Но второй заход на цель сделать не удалось. Зенитная ракета ударила ему в крыло, разрывая корпус самолета словно лист картона. Через несколько мгновений град обломков обрушился на землю, недалеко от первого ряда оборонительной системы.

... Деревья ломались как спички, когда многотонные ракетные тягачи прокладывали себе путь через лес. Колонна сопровождения застыла в немом ожидании, когда стальные наконечники СКАДов уставились в небо. Аккуратно проведенная вылазка, уничтоженный охранный патруль... Замаскировавшаяся в дремучем лесу колонна готовилась нанести решительный удар по всем ключевым объектам китайской армии в этом регионе. Это станет началом широкой наступательной акции, которая позволит укрепиться на этой территории и взять инициативу в этой войне в свои руки.

Конечно, китайские радары тут же запеленговали старт ракет, но слишком мало времени было им дано, да и сбить летящую ракету не так-то и легко. И еще через мгновение военная база погрузилась в море зеленовато-желтого пламени, и то, что не смогла сделать ударная волна, уничтожали особые химические реагенты, гордость GLA. Но внезапно прямо из сердца этого ада вырвался яркий сноп света, и сквозь дым, словно нехотя, танцуя на столбе пламени, показался черный корпус ракеты. Русские умеют делать хорошее оружие, и ракета класса "Сатана" была их шедевром. Но никто не знал, что они продали эти технологии китайцам! Она могла стартовать даже из эпицентра ядерного взрыва воплощением сурового и необратимого возмездия.

...Словно тысячи солнц разом вспыхнули на горизонте, на доли мгновения контурные тени людей и построек замерли, чтобы навсегда остаться только тенями. Вырванные с корнем деревья, падающие башни, разваливающиеся здания внутри клокочущего шара огня, который гигантским уродли-

вым грибом стал подниматься к небесам. Там, где раньше находилась база GLA, теперь была радиоактивная пустыня. Война продолжалась...

***

Итак, мы наконец-то дождались появления на прилавках новой стратегии от "Westwood". Нас так долго кормили обещаниями и недомолвками, что казалось, будто с появлением этой игры для игроков начнется новая эра. Но вот восторженные голоса начинают сменяться несколько озадаченными возгласами. Любители ЦыЦ-серии уже разбились на два непримиримых лагеря и готовы сцепиться насмерть, отстаивая собственное мнение. Мы не будем поддаваться эмоциям, а попробуем детально рассмотреть саму игру с ее "плюсами" и "минусами".

Часть первая. Ода движку

Видимо, вагоны тухлых яиц и гнилых помидоров, которые были выпущены в отцов-основателей серии за ужасную, бездарную и устаревшую графику сделали свое дело. "Ах так!", — подумали они и бросили все силы на красочность и качество картинки. И, смею вас уверить, им это удалось. Посмотрите на скрины, на дальних планах картинка порадует гаммой цветов и количеством объектов, а если приблизить камеру, то становятся заметны такие мельчайшие детали как вращающиеся вентиляторы на заводе или пушки гатлинг-танка. На броне видна эмблема воюющей стороны. Люди теперь — не просто несколько ходячих пикселов, а четко рассматриваемая фигура с достаточно "человеческой" пластикой движений, и ошибиться, кто автоматчик, а кто гранатометчик, просто невозможно. Вообще, надо заметить, что над детализацией ВСЕХ объектов программисты поработали очень качественно — все движется, крутится, нет полной статики, как мы наблюдали это раньше. И самое главное, все эти детали очень осмысленно вписываются в общую картину. Можно с умилением наблюдать, как маленький кран разгружает на бегущий конвейер очередную порцию поступивших ресурсов, или с восхищением смотреть, как строится ваш очередной завод. Прочие эффекты тоже не отстают — взрывы, гибель техники (как это мило смотрится, когда вражеский трек подрывается на вашей мине и разлетается на мелкие части, а горящее колесо еще скачет по дороге, или как падает выброшенный взрывной волной боец!), проход самолетов между небоскребами на бреющем полете или, как венец разрушения — ядерный взрыв! Можно заметить, как одно здание, падая, разрушает соседнее и валит деревья — о них нужно сказать отдельно, наконец-то техника может проломиться через деревья или разрушить их взрывом. Странно, что это не было реализовано ранее. Осталось сказать только одно, но большое "НО" — это системные требования к компьютеру, который сможет воспроизвести всю эту красоту. При первом знакомстве с ними входишь в состояние легкого шока. Увы, вам потребуются только самые последние достижения компьютерных технологий, или красотой придется пожертвовать.

Часть вторая. Интерфейс и геймплей

Первый взгляд... Ой, а это действительно C&C, или это клон Старого Крафта? Интерфейс буквально идентичен близардовскому, что очень удивляет. Конечно, это удобно, но почему отказались от фирменного и не менее удачного собственного? Это удивляет, и опять же, три ра..., прошу прощения, воюющие стороны... К тому же, история этой игры абсолютно не пересекается с историей мира C&C, что тоже несколько озадачивает. Конечно, радует, что наконец-то вынесли на основную панель управляющие функции, которые раньше были доступны лишь через особые комбинации клавиш, известные только посвященным. Удобно, относительно привычно — большего нам от интерфейса и не надо.

Геймплей... Добавление некоторого РПГ-шного кусочка должно было повлиять на стратегию игрока, но на практике это не оказывает большого влияния. Основной по-прежнему является тактика "танкового кулака". И тут совершенно не радует AI противника. Он ничуть не изменился и не радует тактическими находками. Поэтому игра проходится очень легко. Играя на предпоследнем уровне сложности, мне пришлось воспользоваться "сохраненкой" только один раз. Как заметил один мой коллега, "когда игра начинает надоедать — она заканчивается". Дело в том, что за каждую сторону можно пройти не более 7 миссий. Многие возможности юнитов просто не используются. Возможно, это было рассчитано на сетевую игру, но большим набором сетевых карт игра, опять же, не может порадовать.

Часть третья. Вывод

Игра получилась слишком "сырой", с множеством недоработок, но с большой перспективой развития на этом движке. Революции не получилось, а вышла достаточно добротная и красивая стратегия, полностью укладывающаяся в канву серии C&C.

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

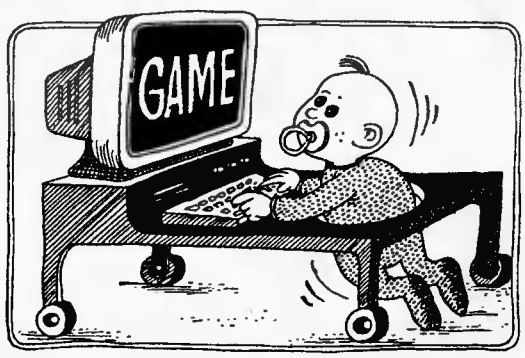


Рисунок О.Попова

КУПЛЮ, ПРОДАМ, ОБМЕНЯЮ

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

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

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

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

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

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

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

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

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

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

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

Продаю: С1-55, С1-112, Г3-36А, Ч3-32, М1109, М2051, Ц4313, Ц4353, Р40114, КТ817Г/КТ819Г (б/у), программы для "ZX-Spectrum" и др.
Куплю документацию на Ч3-32, Г3-36А.
Возможен обмен на тюнер ("Пионер", "Техникс" и др).
Тел. (095) 944-53-13, с 19.00 до 21.00, Михаил.

Продаю:
- компьютер "Scorpion 256" (расширенная клавиатура для "Scorpion"; контроллер для подключения IBM-клавиатуры и мыши; 2 дисковода "Robotron");
- монитор "Электроника 32ВТЦ-202;
- два принтера (без головок) "Электроника СМ6312";
- 160 дискет 5,25S (системные программы, электронные журналы и газеты);
- черно-белый монитор "Электроника МС6105.01";
- два дисковода TEAC на запчасти;
- дисковод "Электроника 5305";
- дисковод "Epson SD-600".

165300, Архангельская обл., г.Котлас, ул.Мартемьяновская, 44 — 7, Ятченко Н.Ф.

Ищу возможные схемы модернизации игровой приставки SUBOR (подключение к ПК, перапрошивка ПЗУ и т.д.). Приму в дар любой ЭЛТ-монитор.
Тел. в Минске 257-91-46.

Куплю: любые выпуски сборника "Радиодизайн", журналы "Телвспутник" за 2002-2003 гг.; панель к лампе ГУ-34Б: диод 6Д13Д; ПК, совместимый с "ZX-Spectrum" с контроллером дисковода и пакет программ к нему.
40000, Украина, г.Сумы, а/я 9, Александр.

Куплю запрограммированные ПЗУ для контроллера TR-DOS (версия 5.03 или 5.04) или готовый контроллер со схемой.
385765, Р. Адыгея, Майкопский р-н, ст.Кужорская, ул.Чапаева,1, Ветров И.П.

Ищу контроллер дисковода к ПК "Байт" (можно без ВГ-93) с дисководом 5,25" и принципиальную схему к ПК "Балтик".
223082, Минская обл, п/о Бровки, 31, Кривощеков Б.Г.
Тел. 504-35-16, Борис.

Куплю контроллер жесткого диска для ПК "Истра 4816.03" (1991 г.) и программное обеспечение для ПК "Искра-1030М".
453500, Башкортостан, г.Белорецк, а/я 21, Сафонов С.Н.
Тел. (34792) 4-45-67.

Уважаемые читатели!

Подписаться на наши журналы (индексы: **48996** —"Радиомир", **48924** — "Радиомир. КВ и УКВ", **48925** — "Радиомир. Ваш компьютер") во II полугодии 2003 г. можно по каталогу Агентства "Роспечать", **стр. 264** или по каталогу РО "Белпочта" "Газеты и журналы Республики Беларусь", **стр. 159** (раздел "Издания РФ, распространяемые по прямым договорам").

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

Год	Можно заказать следующие номера журналов			Расценки на 1 экз. (с учетом пересылки)		
	Радиолюбитель	Радиолюбитель. Ваш компьютер	Радиолюбитель. КВ и УКВ	По России и СНГ (рос. руб.)	По Беларуси (бел. руб.)	По Украине (гривны)
1998	4, 5	1 — 5, 8, 9, 11, 12	1	20	700	5
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	3, 5, 6	25	800	5
2000	2 — 6, 8 — 12	2 — 8, 10 — 12	4, 6, 7, 9, 11, 12	25	900	6
2001	2 — 6	1, 2, 4 — 6	2 — 6	30	1000	6
	Радиомир	Радиомир. Ваш компьютер	Радиомир. КВ и УКВ	По России и СНГ (рос. руб)	По Беларуси (бел. руб.)	По Украине (гривны)
2001	7 — 12	7, 9 — 12	7 — 12	35	1300	7
2002	1 — 12	1 — 12	1 — 12	40	1400	8
2003	1 — 6	1 — 6	1, 2, 4 — 6	45	1700	9
2003	7 — 12	7 — 12	7 — 12	45	1900	9

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

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 эти сведения необходимо указать в графе "назначение платежа".***

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

Приобретение отдельных номеров журналов

В РОССИИ:

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

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

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

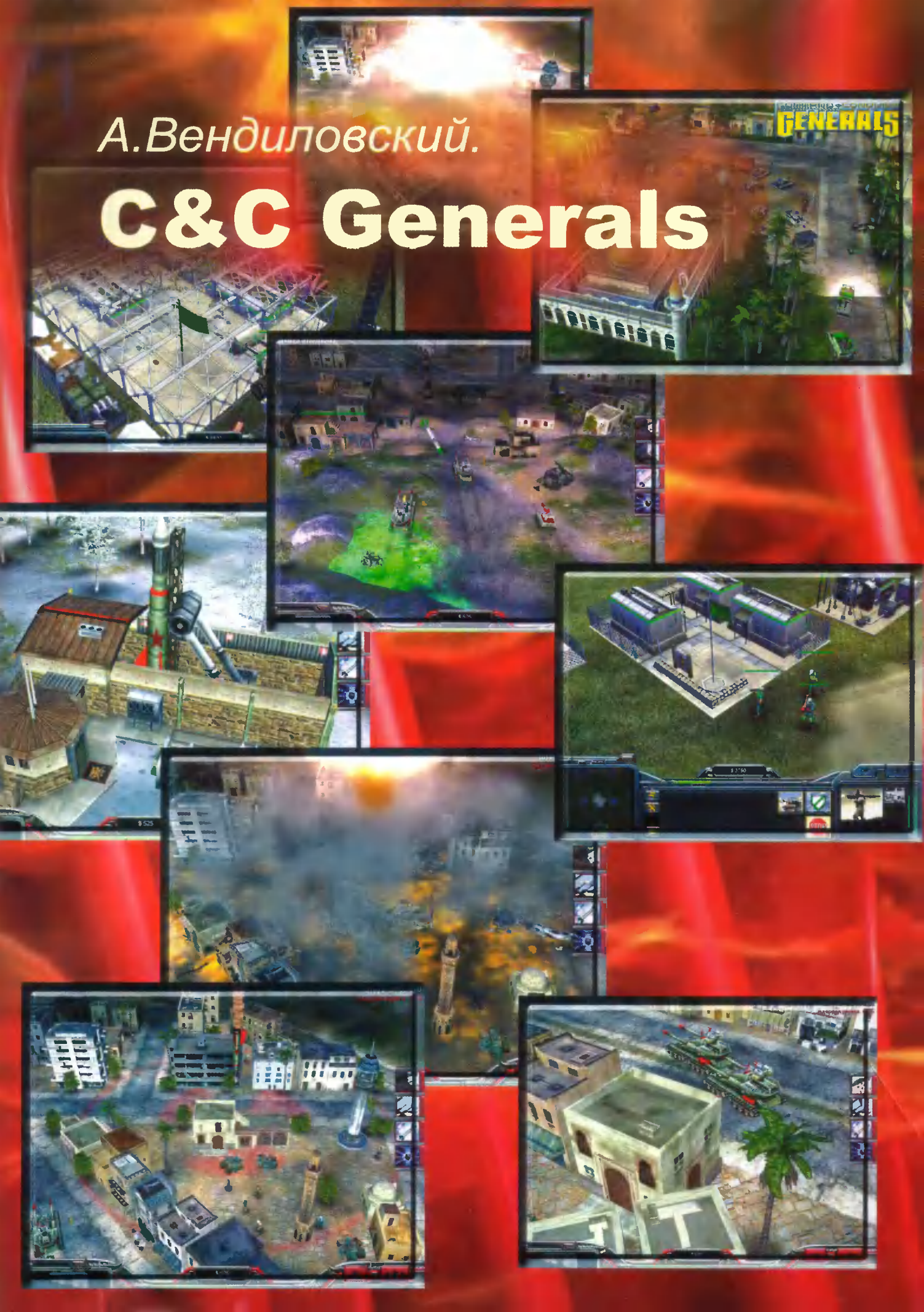
В **ООО "ТРЭНТЭКС"**: 111024, г.Москва, 4-я Кабельная, 2А (ст. метро "Авиамоторная"), тел. (095) 258-91-94, 258-91-95. E-mail: abook@mail.ru, abook@inbox.ru
На радиорынках: "Царицыно" (место 13/А), "Митино" (ряд 1, контейнер 17 и место Т-8); в ТК "Савеловский" (ст. метро "Савеловская", места А4, А5);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

На УКРАИНЕ:

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

В БЕЛАРУСИ:

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



А.Вендиловский.

C&C Generals


СУПЕРХИТ
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
MAX PAYNE
АБСОЛЮТНО ПОЛНАЯ РУССКАЯ ВЕРСИЯ
ОЗВУЧЕННАЯ ПРОФЕССИОНАЛЬНЫМИ АКТЁРАМИ
+КОДЫ
+СЕКРЕ
2CD
Unreal II
THE AWAKENING
КОЛЛЕКЦИЯ ИГРУШЕК
ЛИЦЕНЗИОННАЯ КОПИЯ ИГРЫ
ШТУРМОВИК
ЗАБЫТЫЕ СРАЖЕНИЯ
DELTA FORCE
BLACK HAWK DOWN
американская версия с
НА РУССКОМ ЯЗЫКЕ
CARMAGEDD
Автокалипсис Сегодня
grand theft auto
4в1
POLICE
Platinum
РУССКИЕ ИГРЫ
ТОМ 1
HERO X
PACIFIC WARRIORS
БЫКИ И КОРОВЫ v2.0
КАРТОЧНЫЕ ИГРЫ
РУССКИЕ ВЕРСИИ
2в1
РУССКАЯ + АНГЛИЙСКАЯ ВЕРСИЯ
AIRPORT TYCOON 2
3D MANAGEMENT SIMULATION
A TAKE-TWO COMPANY
I.G.I-2
СЕКРЕТНЫЙ УДАР