8•2002
http://radio-mir.com
Индексы: 48925, 45995

радиомир

В НОВЫЙ ВЕК —
С НОВЫМ НАЗВАНИЕМ!

Ваш компьютер

Ребенок и лекарство

Лекарственн
средства
в педиатри
2000-2001

Фармакотера
в педиатри

КОЛЛЕКЦИЯ АЛЬБОМОВ 1985 - 2000

UFO 2

11 АЛЬБОМОВ В НОВОМ ЗВУКОВОМ ФОРМАТЕ

1985-2000

а также
Phil Mog & Pete Way
& MSG

160 MP3 ~ 3 ГБ

ДОМАШНЯЯ КОЛЛЕКЦИЯ

VIDEO FOR P

ПОСЛЕДНЯЯ ФАНТАЗИ

ДУХИ

4 MPEG

ЧЕСТВО

VIDEO FOR PC

куда уж
хуже?

Мартин
Лоуренс
Дэнни
Де Вито

DivX VIDEO
DVD КАЧЕСТВ

4 EG

ГИГАНТЫ
CAKEWALK
v.2.0

Cakewalk Music Creator 2002 Edition
Cakewalk Guitar Tracks Pro v2
Cakewalk Pyro MP3 and CD Maker v1.5
Fxpansion VST-DX Adapter Standard Edition v3.302
Fxpansion DR-008 DXi v1.01

PYRO
MP3 & CD

ВСЕ
Windows

Коллекция операционных сис

Коллекция Windows
Антивирусы

Всё, что нужно
ХАКЕРУ

...же вы найдете все, что нужно хакеру для "взлома"...

СЕРИЯ ТЕМАТИЧЕСКИХ СБОРНИКОВ

Modern Electric
Blues

часть вторая

160/192

ДОМАШНЯЯ КОЛЛЕКЦИЯ


Полузакл
Полузакл

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

Главный редактор
Ольга СТРЫЖАНКОВА

Зам. гл. редактора
Иван БЕЛЬСКИЙ

Редколлегия:
Сергей ДРОЗДОВСКИЙ,
Алексей КОНДРАШОВ,
Анатолий КОРБИТ,
Владимир КУЦЕНКО,
Елена ЛЕВИТМАН,
Николай МЕДВЕДЬ,
Геннадий ПЕЧЕНЬ,
Геннадий ШУЛЬГИН
Контактные телефоны:
в Москве (095) 105-99-89,
в Минске (017) 249-41-47

Компьютерная верстка —
Янина БЕЛЬСКАЯ

Техническая графика —
Наталья БЕЛЬСКАЯ,
Александр ГОРАЮТИН,
Мария КАЛАБУХОВА

Оформление обложки —
Наталья БЕЛЬСКАЯ,
Надежда БОГОМОЛОВА

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

220095, РБ, г.Минск-95, а/я 199,
факс (017) 221-01-10

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

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

Материалы для публикации принимаются в рукописном, печатном и электронном вариантах

Требования к графическим материалам рекламного характера в электронном виде: CorelDRAW до 9.0, все шрифты в кривых; bitmaps 300 dpi; TIFF 300 dpi; CMYK. Приложить печатную копию.

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

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

Подписано к печати 25.06.2002 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная.

Отпечатано в типографии
ЗАО "Красногорская типография".
Заказ 1637.

радиомир
Ваш компьютер

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

№8/август/2002

ЧИТАЙТЕ В НОМЕРЕ

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

С. ШИРКО,
E-mail: S_Shirko@tut.by

Не все мы вынуждены быть богатыми, или системы на платформе Socket 7

Несмотря на то что компьютер сейчас значительно более доступен для рядового пользователя, чем, скажем, лет десять назад, дешевым его все равно не назовешь. Поэтому, наряду с производительностью, актуальным при выборе ПК является и вопрос его стоимости.

Среди процессоров безусловным лидером по критерию цена/производительность в настоящее время является уже хорошо известный в народе Duron. Сейчас он является практически идеальным вариантом для домашнего использования. Однако, как бы там ни было, цена за набор Duron 700+Socket A колеблется в районе 100 у.е., а для многих это не очень мало. Те, кто готов пожертвовать производительностью ради уменьшения стоимости, чаще всего обращают внимание на системы с процессором Celeron. Но и на такую систему не у всех хватает средств. Естественно, от этого желание иметь компьютер на рабочем столе нисколько не уменьшается, а потому ищется какой-то альтернативный вариант покупки. На мой взгляд, этим вариантом вполне может оказаться система на платформе Socket 7.

Естественно, Socket 7-платформа является устаревшей и малоперспективной, но в то же время, например, компьютер с процессором K6-2-450 вполне справится с любыми офисными приложениями, позволит с комфортом "посидеть" в Интернете, послушать музыку, посмотреть видеофильмы в формате MPEG4, да и в игры поиграть на нем можно — Quake III, Need for Speed 5, Worms Armageddon, Heroes III, Mech Warriors II и многие другие идут без особых проблем. Поэтому, вопреки сложившимся традициям делать обзоры по самым новым процессорам и материнским платам, попытаемся свести в одну статью основной материал по платам Super 7 и процессорам для них.

Как все начиналось

Первоначально платы Socket 7 использовались совместно с такими процессорами как Intel Pentium/Pentium MMX, AMD 5k86/K6, Cyrix 6x86/Cyrix M2 и IDT Winchip/Winchip 2. Ни один из них в настоящее время не пользуется спросом из-за низкой производительности. И, наверное, не вспоминали бы мы уже Socket 7, если бы не выпустила AMD в 1998 г. свой знаменитый K6-2, работающий на системной шине 100 МГц, а сама платформа не была усовершенствована до Super 7 — появилась поддержка шины 100 МГц и AGP. Из-за низкой стоимости как процессоров под эту платформу, так и соответствующих материнских плат, системы на базе платформы Super 7 (иногда ее называют SuperSocket 7) быстро получили признание у не слишком богатых пользователей.

Процессоры

AMD K6-2 (300...550 МГц) "родился" в мае 1998 г. и позиционировался как альтернатива популярному в то время Pentium II. Вот его основные технические характеристики.

Площадь кристалла — 81 мм^2, техпроцесс — 0,25 мкм, объем кеша 1-го уровня — 64 Кб, кеш L2 находится на материнской плате. Работает процессор на системной шине 100 МГц (кроме AMD-K6-2/300-66 — шина 66 МГц отражена в маркировке). Поддерживает технологии MMX и 3DNow! По скорости работы в офисных приложениях K6-2 сравним с аналогичным по частоте Pentium 2.

Однако совсем по-другому складывается расстановка сил в приложениях, интенсивно использующих операции с плавающей запятой (3D-игры, распознавание речи, сжатие видео) — здесь AMD проигрывает своему конкуренту. И проигрывает вовсе не потому, что процессор плохой. Дело в том, что инженеры AMD сделали ставку не на мощный FPU, как Intel, а на свое ноу-хау — технологию 3DNow! Теоретически она должна неплохо поднимать скорость выполнения операций с вещественными числами, только вот на практике это возможно лишь при ее поддержке со стороны разработчиков программного обеспечения. Вот тут-то и кроется проблема — несмотря на старания AMD, далеко не во все игры и мультимедийные приложения включена оптимизация под эту технологию.

AMD K6-III появился с выходом очередного процессора от Intel — Pentium-III. От K6-2 его отличает кеш 2-го уровня объемом 256 Кб, помещенный на ядро процессора и работающий на его полной частоте (кеш материнской платы работает как кеш третьего уровня). Кроме того, в новый процессор была добавлена функция пакетной записи в память Write Allocate, которая позволяет передавать данные по шине не как придется, а 8-байтовыми пакетами, что также дает небольшой выигрыш в производительности. Эти нововведения позволили K6-III даже обгонять аналогичный Pentium III (не Coppermine) в офисных программах. Однако в играх и других приложениях, требующих усиленных вычислений с плавающей запятой, процессор от AMD опять не радует высокими показателями. Причины все те же — медленный FPU и отсутствие нормальной поддержки 3DNow! со стороны производителей программного обеспечения.

Не так давно AMD представила усовершенствованный вариант K6-2 — K6-2+. Отличия — 0,18-микронная технология и кеш 2-го уровня 128 Кб на ядре процессора. Несмотря на то что K6-2+ создавался в первую очередь для мобильных систем, установить его можно в любую Socket 7-плату, поддерживающую K6-2/K6-3 и позволяющую вывести на ядро процессора напряжение 2,0 В. Единственное, может потребоваться обновление BIOS

материнской платы. Проверить этот момент можно при помощи пакета программ SiSoft Sandra — в подразделе Extended CPU Feature раздела CPU&BIOS следующие пункты должны быть отмечены зеленым:

- Data Prefetch;
- Speculative Write ordering;
- Write allocation;
- Write allocate 15...16Meg.

Тем, кому интересны конкретные данные по производительности названных выше процессоров, могу посоветовать зайти на www.iXBT.ru.

Чипсеты и материнские платы

Чипсетов, поддерживающих системную шину 100 МГц и AGP, немало. Наибольшее распространение у нас получили VIA Apollo MVP3 и ALI Aladdin V. Вот их некоторые характеристики.

MVP3:

- состоит из северного моста VT82C598 и южного — VT82C586;
- работает с L2-кешем 512 Кб, 1 Мб или 2 Мб;
- имеет асинхронную шину памяти (частота работы памяти не привязана к частоте FSB);
- поддерживает UDMA33, USB и ASPI;
- обслуживает до 5 слотов PCI, AGP 2x.

Aladdin V:

- включает северный мост M1541 и южный — M1543;
- работает с L2-кешем 256 Кб, 512 Кб или 1 Мб ;
- имеет синхронную шину памяти;
- поддерживает UDMA33, USB и ASPI;
- обслуживает до 5 слотов PCI, AGP 2x.

В целом оба чипсета эквивалентны друг другу по возможностям и имеют практически одинаковую производительность. Стоит лишь отметить, что в последних вариантах своего чипсета (MVP3+) VIA использовала более современный южный мост VT82C686A, поддерживающий UDMA66 и AMR.

Меньшей популярностью пользуется интегрированный чипсет SiS530, содержащий в своем составе 64-разрядное 2D/3D-видеоядро собственной разработки. Кстати, фирма SiS первой среди всех остальных внедрила поддержку режима UDMA66 в контроллере IDE.

Что касается более новых чипсетов, поддерживающих платформу Super 7, то нужно отметить их ориентирование исключительно на офисные системы. К ним можно отнести интегрированные чипсеты VIA MVP4, SiS540 и ALI Aladdin7.

VIA MVP4 представляет собой комбинацию MVP3 и 64-разрядного графического 2D/3D-контроллера Trident Blade 3D. Южный мост VT82C686A обеспечивает поддержку AMR, UDMA66, USB, ACPI.

SiS540 является вариантом SiS630 для систем Super 7. Он по своим функциональным особенностям похож на MVP4, разве что, имеет несколько более совершенное 128-разрядное графическое ядро, выполненное на базе SiS300, 4 разъема USB и интегрированный сетевой контроллер. Этот чипсет может послужить основой для создания дешевого, но вполне приличного решения для офиса.

ALi Aladdin 7 (M1561/M1535) является высокоинтегрированным решением для рынка недорогих ПК. Имея практически все современные "навороты", он не поддерживает кеш-память 2-го уровня на материнской плате, поэтому может работать только с процессорами AMD K6-3, K6-2+.

Что касается собственно плат под эти процессоры, то тут, конечно, "как повезет". Но могу порекомендовать **ASUS P5A** (AladinV), **FIC PA-2013** (MVP3), **Chaintech CT-5AGM2** (MVP3) и **LS 5MVP3** (MVP3) — о них слышал только хорошее, хотя раз на раз не приходится.

Особенности

Особенность первая. Бытует мнение, что Socket 7-процессоры от AMD слабо греются, поэтому заострять внимание на выборе радиатора с кулером не обязательно — чем дешевле, тем лучше. В какой-то степени так оно и есть. Однако если компьютеры на базе K6-2(III) в течение всего дня не выключаются в плохо проветриваемом офисе, то при недостаточном охлаждении процессор запросто может выйти из строя.

Вторую особенность я почерпнул из программы System Optimize, автор которой для повышения быстродействия предлагает прописать в реестре для K6-2 следующие строки:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager\MemoryManagement
"SecondLevelDataCache"=dword:00000200
("SecondLevelDataCache"=dword:00000100 – для K6-3).
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SessionManager\MemoryManagement
"SecondLevelDataCache"=dword:00000200;
("SecondLevelDataCache"=dword:00000100 – для K6-3).
```

Лично я особого прироста в производительности не почувствовал, но чем черт не шутит.

Особенность третья. Так как платы Super 7 изготавливаются на чипсетах VIA, Ali или SiS, а, например, в базе драйверов Windows 98 содержится информация в основном о продукции Intel, то, после инсталляции на ПК операционной системы, потребуется установить драйверы под соответствующую материнскую плату. В противном случае возможны проблемы с AGP-видеокартами, режимами UDMA и др.

Четвертая особенность касается недорогих видеокарт Vanta от Nvida — они часто отказываются нормально работать на системах Super 7. Поэтому будьте осторожны. Сама по себе Vanta является удачным приобретением для дешевого компьютера, и если она заработает — значит, вам повезло. Но в любом случае договаривайтесь о возможности возврата видеокарты при возникновении проблем.

В заключение, подчеркну, что если при приобретении компьютера решающим фактором является стоимость, то система на платформе Super 7 и с процессором от AMD вполне может войти в список претендентов на покупку.

Измышлизмы

Дисковод — средство для занесения вирусов!

В инструкции написано — "нажать", но нигде не сказано, когда отпустить. :-((

Любовный треугольник — два модема и помехи с АТС.

И по команде ShutDown село солнце...

Внимание, идет подготовка к последнему запуску Windows95.

Автомат Калашникова — это преобразователь стека в очередь.

Если завис компьютер, выдерни шнур, выдави стекло.

Компьютер позволяет решать все те проблемы, которые до изобретения компьютера не существовали.

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

Работа в системе Mathematica 3.0/4.0.
Символьные выражения

В предыдущей статье мы рассмотрели общие принципы программирования в системе Mathematica. Изученные приемы вполне пригодны для работы как с числовыми, так и с символьными выражениями. Однако эти два типа данных различаются достаточно сильно, как правило, символьное выражение — сложный и структурированный объект. Соответственно, возрастает как количество, так и сложность процедур обработки. В первую очередь ясным, как же представляются символьные выражения в Mathematica.

Разумеется, та форма записи, которая отображается на экране, далеко не всегда совпадает с той формой, с которой работает система. Забота о пользовательском интерфейсе приводит к тому, что эта разница от версии к версии все возрастает, и "рабочий" формат ввода-вывода приближается к стандартным математическим обозначениям. Поэтому в Mathematica предусмотрена встроенная команда **FullForm** (полная форма) для перехода ко внутренним обозначениям системы. Например, **FullForm**[a x^2 + b x + c] возвращает Plus[c, Times[b, x], Times[a, Power[x, 2]]]. Для более наглядного отображения той же информации существует также и древовидная форма **TreeForm** (**рис.1**). В полной форме строго соблюдаются следующие правила — встроенные команды начинаются с заглавной буквы, аргументы берутся в квадратные скобки, даже a + b будет выглядеть как Plus[a,b].

Такая жесткая структурированность выражения имеет свои очевидные преимущества. Система легко "раскладывает по полочкам" даже самые сложные выражения. У любого символьного выражения имеется заголовок (**Head**) — команда верхнего уровня, в рассмотренной ситуации (a x^2 + b x + c) — это Plus. Эта команда применяется к аргументам (a x^2, b x, c), в качестве которых могут выступать как числовые и символьные величины (c), так и сложные символьные выражения (a x^2, b x). Последние, в свою очередь, также разлагаются на более простые составляющие. Для извлечения подвыражения из сложного выражения применяется команда **Part**, или, как и при работе со списками, двойные квадратные скобки (**рис.2**).

Знание структуры выражения позволяет решать уже более сложные задачи. Можно, например, используя клавиатуру и мышь, заменить последнее слагаемое в выражении c + b x + a x^2 на a y^2. Но если речь идет об обработке большого количества информации, необходимо уметь производить такого рода операции программно. Так, в рассмотренном примере достаточно указать expr[[3, 2, 1]] = y (рис.2).

Более того, в выражении можно заменить заголовок **Apply**[Times, expr] (**рис.3**), в результате, от суммы некоторых величин (Plus) мы переходим к их произведению (Times). Любую функцию можно применить как ко всем подвыражениям первого уровня (команда **Map**), так и к подвыражениям определенного уровня (уровень указывается в фигурных скобках), к выражениям всех уровней (**MapAll**), а также к отдельному подвыражению (**MapAt**).

Рис. 1

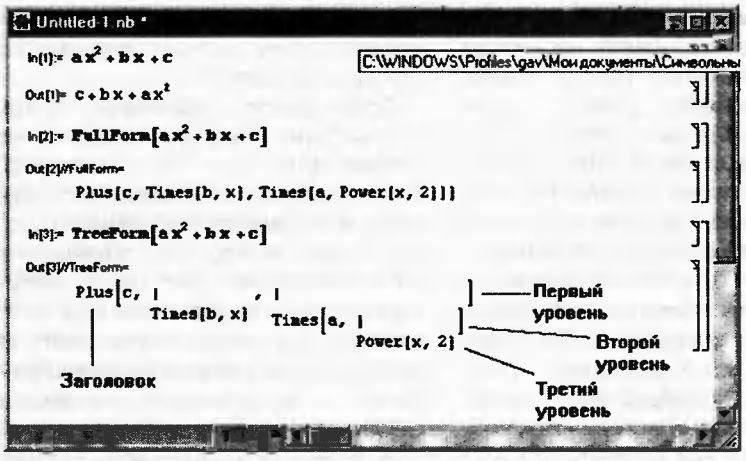


Рис. 2

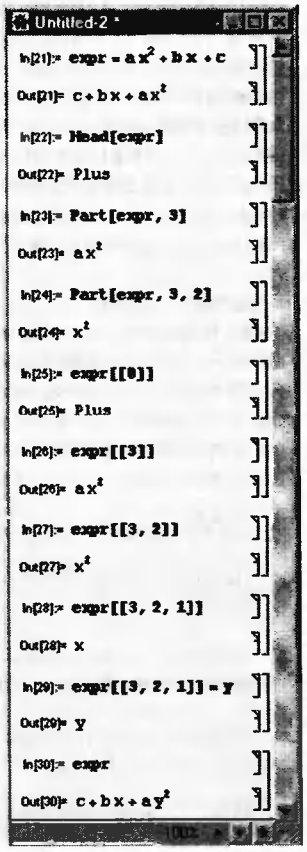


Рис. 3

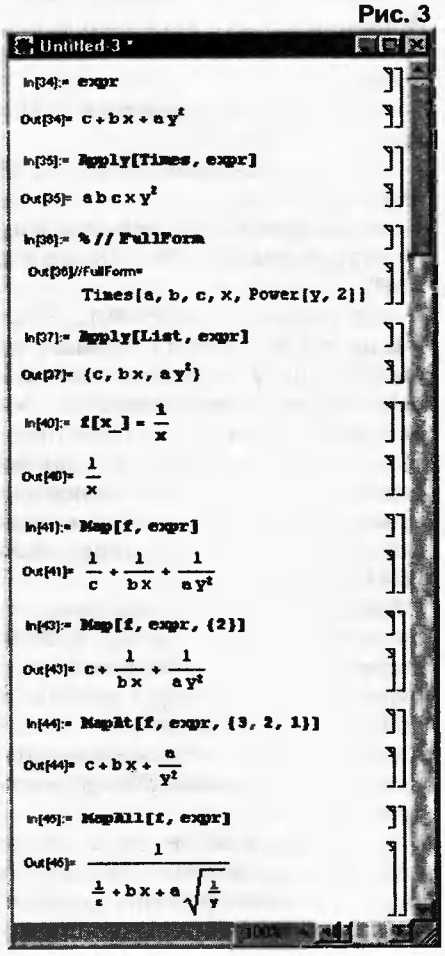


Рис. 4

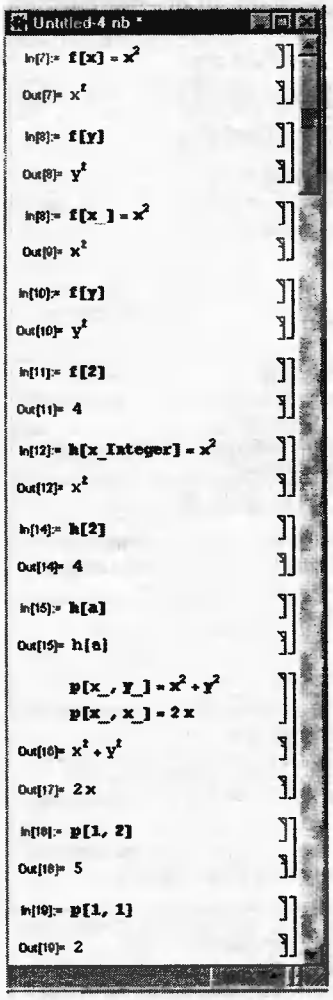
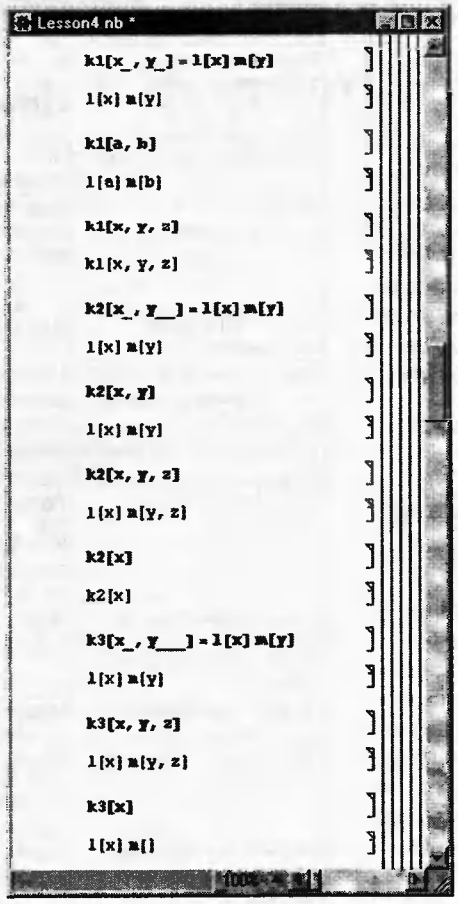


Рис. 5



таким образом, что она будет работать с разным числом аргументов.

Особенно эффективным является использование шаблонов вместе с подстановками. Подстановка (конструкция вида /.x→a) применяется, как следует из названия, для подстановки нового выражения (здесь — a) вместо прежнего (x). А вот подстановка с шаблоном (/.x_→x^2) позволяет возвести в квадрат все переменные (**рис.6**). Можно также указать последовательность подстановок:

{1, 2, 2.5, -1, -3.4 + I} /. x_Integer → -x /. x_Real → x^2 /. x_Complex → 1/x.

Однако в таком варианте подстановка срабатывает только однократно:

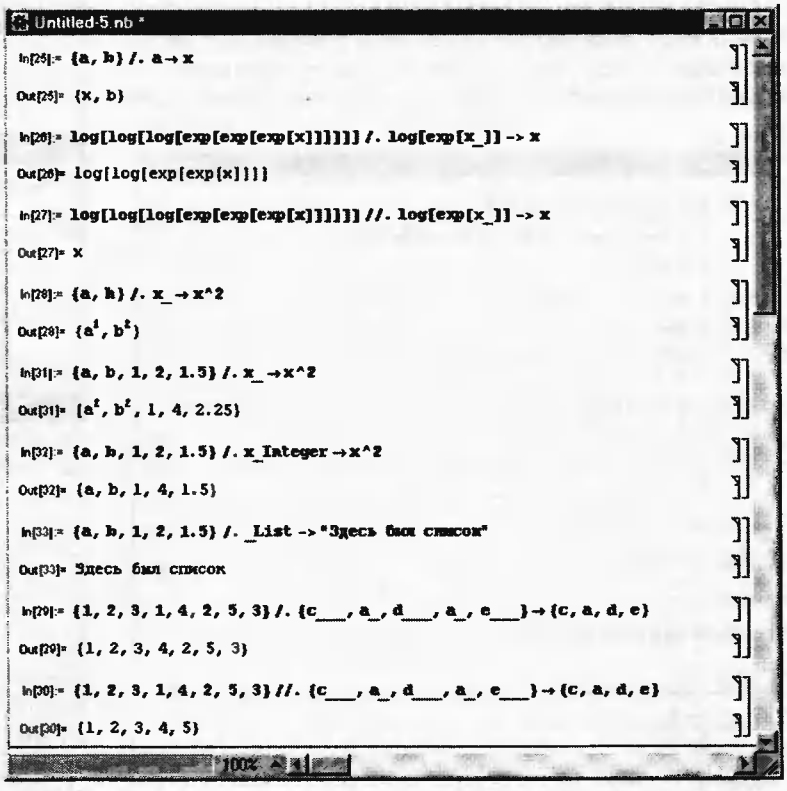
log[log[log[exp[exp[exp[x]]]]]] /. log[exp[x_]] → x возвращает log[log[exp[exp[x]]]].

Для того чтобы выполнить указанное преобразование до конца, используется команда повторной подстановки (//.):

log[log[log[exp[exp[exp[x]]]]]] //. log[exp[x_]] → x возвращает x.

С использованием подстановок и шаблонов можно достаточно эффективно обрабатывать списки. Так, замена {c___, a_, d___, a_, e___} → {c, a, d, e} позволяет выбросить повторяющийся элемент (обозначенный a) из списка. На месте c, d, и e может находиться произвольное число элементов, что задается при помощи тройного значка подчеркивания. Применение подстановок и шаблонов в работе со

Рис. 6



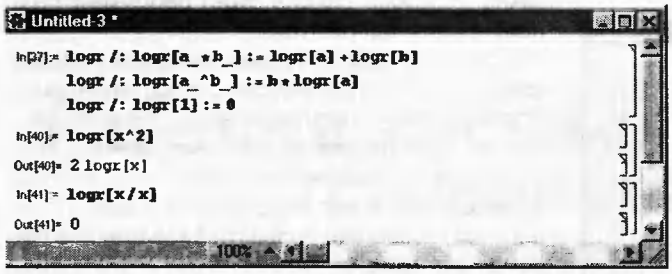
Для работы с символьными выражениями предусмотрена также достаточно гибкая и удобная система шаблонов. Один из вариантов шаблона уже встречался нам при определении функции. Так, функция, заданная в виде f[x]=x^2, срабатывает только для аргумента x, но при задании f[x_]=x^2 (со значком подчеркивания) работает с любым аргументом (f[y] возвращает y^2, f[2] возвращает 4 и т.д.). Таким образом, конструкцию x_ можно понимать как "любой x".

Более сложный вариант — шаблон вида x_Integer (переменная_тип данных), например, h[x_Integer]:=x^2. Здесь функция h возводит в квадрат только целые числа, для остальных возвращает h[x] (**рис.4**). Аналогично работает шаблон вида переменная_заголовок — функция f[x_List] определена только для выражений с заголовком List (т.е. для списков). Используются шаблоны с двумя и тремя подчеркиваниями:

- x___ — шаблон для последовательности одного или более выражений;
- x____ — шаблон для последовательности нуля или более выражений (**рис.5**).

Аналогично, шаблон x__List применяется для последовательности из одного или более списков, x___List — для последовательности нуля или более списков. В результате всех этих манипуляций можно определить функцию списками позволяет системе Mathematica сравняться в этой области с такими мощными языками как Lisp и Prolog. Интересные примеры можно найти на одном из сайтов компании Wolfram Research (http://www.mathsource.com).

Рис. 7



Встроенный язык программирования Mathematica по праву относится к языкам высокого уровня. Например, достаточно минимального описания свойств функции, для того чтобы система могла с ней достаточно успешно работать. Так, после ввода

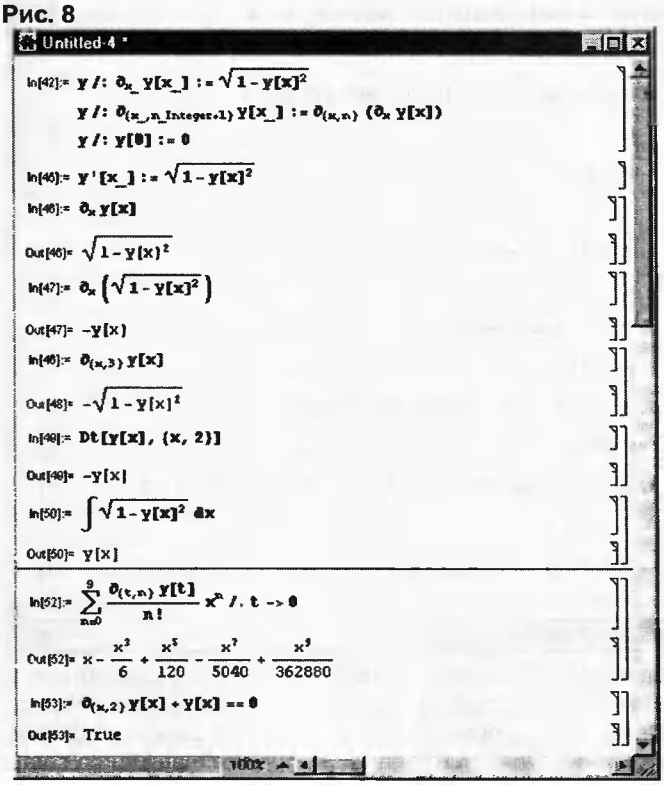
```
logr /: logr[a_*b_] := logr[a] + logr[b]
logr /: logr[a_^b_] := b*logr[a]
logr /: logr[1] := 0
```

Mathematica, используя указанные свойства функции logr, может производить с ней вычисления (**рис.7**). Для описания функции используются символы **/:**, но существует одно ограничение — в левой части функция должна быть либо заголовком, либо выражением первого уровня. Эту особенность нужно учитывать, например, при работе с производными — полной формой для t'[x] является достаточно сложное выражение Derivative[1][t][x]. Здесь приходится идти на небольшие хитрости (**рис.8**), но в результате вы можете производить с функцией как интегрирование, так и разложение в ряд. В этом примере за функцией y[x] скрывается обычный синус, для проверки функция подставляется в дифференциальное уравнение.

В таблице приведены основные команды для работы с символьными выражениями.

В этой статье мы ни разу не обращались к стандартным операторам программирования (циклам, условным переходам и т.д.). Но за рассмотренными командами на самом деле стоят огромные программные блоки. Для

Рис. 8



Действие	Реализация
Основные команды для работы со структурой выражений	
Вывод выражения в полной форме	a x2 + b x +c//**FullForm** или **FullForm**[a x2 + b x +c]
Вывод выражения в древовидной форме	a x2 + b x +c//**TreeForm** или **TreeForm**[a x2 + b x +c]
Выделение подвыражений	expr = a*x^2 + b*x + c — определение выражения **Part**[expr, 2] или expr[[2]] expr[[3, 2]]
Вывод заголовка выражения	**Head**[expr]
Выделение выражений, относящихся к определенному уровню	**Level**[expr, {2}] **Level**[expr, 2] — выражения до второго уровня включительно
Дополнительные команды для работы с заголовками и подвыражениями	
Замена заголовка выражения	**Apply**[Plus, List[a, b, c]] или Plus **@@** List[a, b, c] — заменяет заголовок List на Plus, выражение заменяется на Plus[a, b, c], т.е. a+b+c
Применение функции (оператора) к элементам первого уровня выражения	**Map**[f, {a, b, c}] f **/@** {a, b, c}
Применение функции (оператора) к элементам произвольного уровня выражения	**Map**[f, {{a, b}, {c, d}}, {2}] — применение ко второму уровню **Map**[f, {{a, b}, {c, d}}, 2] — применение до второго уровня включительно **MapAll**[f, {{a, b}, {c, d}}] — применение ко всем уровням
Применение функции к определенным подвыражениям	MapAt[f, m, {{1, 1}, {2, 3}}] — применение к элементам m[[1,1]] и m[[2,3]]
Многократное применение функции	**Nest**[f, x, 3] — троекратное применение функции f к выражению x: f[f[f[x]]] **NestList**[f, x, 3] — список многократных применений выражения: {x, f[x], f[f[x]], f[f[f[x]]]}
Работа с шаблонами	
Пример шаблона	**Пример применения**
k1[x_, y_]=l[x] m[y]	k1[x, y] возвращает l[x] m[y] k1[x, y, z] возвращает k1[x, y, z]
k2[x_, y__]=l[x] m[y]	k2[x, y] возвращает l[x] m[y] k2[x, y, z] возвращает l[x] m[y, z] k2[x] возвращает k2[x]
k3[x_, y___]=l[x] m[y]	k3[x, y] возвращает l[x] m[y] k3[x, y, z] возвращает l[x] m[y, z] k3[x] возвращавт l[x] m[ ]
Работа с подстановками	
Подстановка	**Результат**
{a, b} /. x_ → x^2	{a^2, b^2}
{a, b, 1, 2, 1.5} /. x_Integer → x^2	{a, b, 1, 4, 1.5}
{a, b, 1, 2, 1.5} /. _List→ "Здесь был список"	"Здесь был список"
a /. _List → "Здесь был список"	a — в последних примерах подстановок можно даже не указывать имя переменной (т.е. можно использовать как _List, так и x_List)

сравнения, попробуйте на одном из распространенных языков написать программу для удаления повторяющихся элементов из списка. Остается только порадоваться, что такой огромный труд на себя взяли разработчики системы. А в следующей статье мы используем рассмотренные здесь средства для решения конкретных задач.

Литература

1. А.Гринчук. Работа в системе Mathematica 3.0/4.0. — Радиомир. Ваш компьютер, 2000, №№8-12; 2001, №№1-7.

Преобразование файлов P-CAD в графический формат PCX

А.ГОРЯЧКИН,
г.Кыштым,
Челябинской области.

Несовместимость файловых форматов, создаваемых в системе автоматизированного проектирования печатных плат P-CAD, с графическими форматами, а также отсутствие специального программного обеспечения, с помощью которого можно было бы просто и удобно конвертировать файлы форматов P-CAD в графические форматы, порождает определенные трудности для радиолюбителей, инженеров-конструкторов и разработчиков печатных плат.

Необходимость преобразования файлов P-CAD в графический формат возникает при выводе чертежа платы на принтер или плоттер, не входящий в число поддерживаемых САПР P-CAD, либо для размещения чертежей на Web-страницах в Интернете. Конвертирование файлов P-CAD также требуется для редактирования их в распространенных графических редакторах, для изготовления фотошаблонов печатных плат, или для просмотра этих файлов на компьютере, в программном обеспечении которого отсутствует P-CAD.

Способы, описанные в [1], не решают все возникающие при конвертации проблемы. Они неудобны в использовании, вносят всевозможные дополнительные искажения, требуют обязательного последующего редактирования в программах AutoCAD и Corel PHOTO-PAINT, а также соответствующих аппаратных ресурсов. Вниманию читателей предлагается еще один, более простой и удобный способ преобразования файлов формата PCB. Тем более, что предлагаемый способ отличается от вышеупомянутых высоким профессиональным уровнем качества и верностью преобразования изображения при полном отсутствии каких-либо искажений. Этот способ предназначен для преобразования файлов формата PCB, созданных в САПР P-CAD v4.5, которая до последнего времени считалась одной из наиболее популярных программ для автоматизированного проектирования и разводки печатных плат на персональных компьютерах IBM PC.

Для решения проблем, связанных с преобразованием файлов P-CAD в графический формат, предлагаются три программы: BRD, PLTVIEW и PIPE. Все три программы являются DOS-приложениями.

О программе BRD

BRD представляет собой конвертор файлов формата PostScript (***.psc**) в растровый графический формат PCX. PostScript — это язык описания страниц, созданный фирмой Adobe Systems. Файлы формата PostScript создаются входящей в САПР P-CAD v4.5 программой PCPLOTS. BRD в процессе работы интерпретирует исходный файл формата PostScript, а результат помещает в битовую карту. Затем битовая карта выводится в растровый графический формат PCX.

Программа BRD должна обязательно находиться в корневом каталоге диска C: в директории C:\Brd.

Файлы, входящие в BRD, делятся на две группы — основные и вспомогательные. Основные файлы — **brd.exe** и **brdx.exe** — должны обязательно находиться в директории C:\Brd\Exe. Вспомогательные файлы должны быть расположены в директории C:\Brd. Состав и описание вспомогательных файлов программы BRD:

- **brdx.cfg** — конфигурационный файл программы;
- **courb.rfa** — файл масштабируемого шрифта Courier-Bold формата PostScript Type 1 фирмы Adobe Systems. Шрифты формата PostScript очень широко употребляются в издательском деле и полиграфии;
- **dos4gw.exe** — оболочка DOS Extender;
- **fontmap** — файл описания таблицы применяемых шрифтов формата PostScript Type 1.

Программа BRD была разработана в Новосибирском институте автоматики и электрометрии Сибирского отделения Российской Академии наук (http://www.iae.nsk.su). Дата разработки этой версии программы BRD — май 2000 г.

О программе PLTVIEW

Программа PLTVIEW представляет собой простой и удобный просмотрщик файлов формата PLT, созданных в САПР P-CAD v4.5. Программа PLTVIEW, состоящая из одного файла **pltview.exe**, допускает свое произвольное размещение на жестком диске и может быть расположена в любом месте по желанию пользователя. Это beta-версия программы, разработана она была еще в 1991 г. в Советском Союзе практически сразу после выхода "в свет" P-CAD v4.5, и "в ходу" она уже более десяти лет. К сожалению, авторы программы PLTVIEW мне неизвестны.

О программе PIPE

Программа PIPE предназначена для поддержки символов кириллицы в САПР P-CAD v4.5 при создании файлов формата PLT. Программа поддерживает файлы формата PLT не только версии P-CAD 4.5, но и версий P-CAD 2.0 и P-CAD 1.31. К сожалению, автор этой программы мне неизвестен.

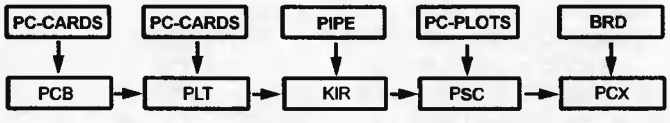
Системные требования

Предлагаемые программы очень компактны (все три программы вместе умещаются на одной дискете), просты в применении и нетребовательны к аппаратным ресурсам PC. Они вполне работоспособны при следующих минимальных системных требованиях:

- компьютер IBM PC/AT или совместимый;
- процессор Amd 386DX 40 МГц;
- операционная система MS-DOS v6.22;
- размер оперативной памяти 8 Мб;
- монитор VGA;
- манипулятор "мышь".

Все три программы нормально работают также из-под Windows 9x/Me и не требуют перезагрузки в режим эмуляции MS-DOS.

Ниже на рисунке схематично показан процесс конвертации файлов. Верхний ряд — это используемые программы, нижний — соответствующие расширения файлов.



Подготовка исходного файла формата PostScript в САПР P-CAD v4.5

Допустим, что файл описания топологии печатной платы в формате PCB уже создан. Далее, после вывода на экран монитора нужного слоя, нужно программой PC-CARDS сохранить в отдельном файле в формате PLT топологию каждого слоя. Т.е. если плата — двусторонняя, то, соответственно, нужно создавать два файла формата PLT. Эти файлы представляют собой двоичные файлы описания графики. Для просмотра полученных файлов формата PLT предлагается ис-

пользовать упомянутую выше программу PLTVIEW, указав ей при вызове только один параметр — имя просматриваемого файла. Далее нужно загрузить программу PC-PLOTS, которая предназначена для вывода графической информации. В меню программы необходимо выбрать опцию Configure PC-PLOTS и внести изменения в конфигурацию программы, указав Output device — disk, Default plotter — PostScript.

В результате внесенных изменений конфигурация PC-PLOTS должна иметь следующий вид:

```
PC-PLOTS Configuration
Default configuration                    PCPLOTS.CFG
Plotter port                                   PORT1
Output device                                   disk
Output record length                              —
Plot Flash-Apertures                             No
Aperture List                 Gerber Model 32(33, 41)
Default plotter                             Postscript
Paper Size Setting                              US
Default paper size                               A
Plot                                          slow
Zero-width line diameter                     0.020"
Polygon cross hatch aperture width           0.250"
Polygon cross hatch spacing                  0.250"
Text dimensions (% of text size):
Text character width (including space)           70%
Text character height                            80%
Space below text line                            10%
Space above text line                            10%
Space from top of text to bar                    20%
Enter configuration filename: PCPLOTS.CFG
```

Далее, сохранив изменения в файле конфигурации PCPLOTS.CFG, нужно загрузить полученные файлы формата PLT в программу PC-PLOTS и получить файлы в формате PostScript (PSC). Важным параметром в процессе задания генерации файла формата PostScript является масштаб. Программа PC-PLOTS автоматически подбирает масштаб прорисовки так, чтобы область прорисовки вписывалась в лист жестко заданного размера. Необходимо управлять этим параметром так, чтобы при конвертации файла из формата PSC в формат PCX можно было восстановить первоначальный масштаб. При задании масштаба прорисовки (параметр Scale Factor) его нужно округлять в меньшую сторону. Например, если программа PC-PLOTS предлагает масштаб 1,059, то его нужно задать равным 1, если же программа PC-PLOTS предлагает масштаб 0,683, то его нужно задать равным 0,5, и т.д. Задаваемый масштаб желательно запомнить или записать на бумаге. Эти данные потом понадобятся при внесении изменений в конфигурационный файл программы BRD C:\BRD\brdx.cfg.

И вот, наконец, наступил заключительный этап преобразования файлов. Полученные файлы формата PSC нужно скопировать в директорию C:\BRD, которая обязательно должна находиться в корневом каталоге, и внести некоторые изменения в конфигурационный файл C:\BRD\brdx.cfg — в блоке [plot] в строке PCPlotScale нужно указать масштаб, использованный при создании файла в формате PSC. В строке OutputFile указывается имя файла, в который будет записан результат конвертации. Содержимое конфигурационного файла можно корректировать с помощью любого текстового редактора.

В результате внесенных изменений файл brdx.cfg должен иметь следующий вид:

```
[brdx]
Resolution:     40
BandHeight:     0
MemFactor:      0.25
OutputFile:     x.pcx

[plot]
PCPlotPage:     250.0 250.0
PCIndent:       0.0
PCPlotScale:    1.0
PCSwapAxis:     0

[pcad]
PCScaleX:       1.0
PCScaleY:       1.0
PCNegative:     0
PCMirror:       0
```

Версия PCAD 4.5 требует значения PCSwapAxis равного 0, для других версий — 1. Остальные параметры в конфигурационном файле также можно изменять. Программа универсальна, и позволяет при необходимости создавать негативное (строка PCNegative: 1) и зеркальное (строка PCMirror: 1) изображения, а также увеличивать или уменьшать масштаб по каждой из осей в отдельности (строки PCScaleX и PCScaleY).

Для более удобной работы с программами BRD и PLTVIEW лучше всего использовать файловую оболочку Dos Navigator v1.50 или выше. Предварительно перед началом работы нужно внести следующие изменения в файл обработки по расширению **dn.ext**, который должен располагаться в том же каталоге, где находится и сама программа Dos Navigator:

```
psc {C:\Brd\Exe\brdx.exe -p !.!
       ren x.pcx !.pcx}
plt {C:\Brd\Exe\pltview !.!}
```

Наличие ключа -p при запуске файла **brdx.exe** обязательно, его отсутствие лишает возможности работать с этой программой. После окончания редактирования файла **dn.ext** нужно перезагрузить Dos Navigator для принятия внесенных изменений. Далее указываем курсором на нужный файл формата PSC и нажимаем на клавишу Enter, программа запускается, и на экране монитора должно появиться следующее сообщение:

```
File:           File_name.psc
Plot Scale:     0.5
Output Scale:   243 151
Scale:          1.000000
Mirror:         0
Negative        0
Continue (Y/N) [Y]?
```

Если нажать клавишу Y или Enter, то программа начнет преобразовывать файл формата PSC в графический формат PCX и сохранит его на диске под именем **file_name.pcx** в каталоге C:\BRD. Нажав клавишу N, завершим работу программы без создания файла. Для удобства полученные файлы формата PCX желательно сразу переносить в специально отведенную директорию, например, C:\PCX, для дальнейшего просмотра, редактирования и хранения.

Файлы формата PCB желательно создавать в метрической системе единиц измерения, а не в дюймовой. Иначе, после распечатки файлов формата PCX на принтере, можно обнаружить, что габаритные размеры печатных плат получились немного больше фактических размеров платы. Для решения этой проблемы можно применить масштабирование, изменив маштабирующие коэффициэнты в конфигурационном файле **brdx.cfg** в блоке [pcad] — PCScaleX: 0.983196 и PCScaleY: 0.983196.

Коэффициент 0,983196 был получен в результате деления значений реальных габаритных размеров в метрической системе для печатной платы, которая была создана в дюймовой системе, на величины, полученные после распечатки этой печатной платы в формате PCX. Указанный коэффициент 0,983196 не является какой-либо констан-

той, и наиболее правильным будет тот коэффициент, который наиболее точно отобразит габаритные размеры печатной платы после распечатки.

Программу BRD крайне нежелательно использовать в режиме командной строки. Это внесет такие неудобства, что пользователи просто откажутся от применения данной программы.

Файлы формата PCX, преобразованные с помощью программы BRD, получаются всегда монохромными, т.е. черно-белыми. При желании, полученные файлы формата PCX можно конвертировать в наиболее распространенные графические форматы GIF или JPEG для дальнейшей публикации в сети Интернет, например, с помощью программы-просмотрщика графических файлов ACDSee v3.0 и выше. В этой же программе есть возможность создавать Web-страницы с уменьшенными копиями рисунков (меню Plug-ins, вкладка HTML Album Generator). Для просмотра на экране монитора печатных плат в формате PCX лучше всего воспользоваться Imaging Preview — штатной программой Windows 9x/Me. При необходимости файлы формата PCX можно редактировать, удобнее всего это делать в графическом редакторе Corel PHOTO-PAINT из пакета CorelDRAW v9.0, в этой же программе лучше и распечатывать на принтере эти файлы.

О промежуточном файловом формате KIR

Дополнительно хотелось бы рассказать о проблемах, связанных с отсутствием поддержки русского языка (символов кириллицы) в САПР P-CAD v4.5 при создании файлов формата PSC.

Промежуточный формат KIR нужен только в случае применения символов кириллицы при создании файлов формата PCB. Дело в том, что если при создании файлов формата PCB пользователь использовал символы кириллицы, например, для обозначения названия печатной платы, децимального номера конструкторского документа или обозначения выводов радиоэлементов, то в файлах формата PSC, и соответственно PCX, после их получения и последующего просмотра и распечатки нарушается правильная кодировка кириллических символов. Можно, конечно, обойти эту проблему, применив так называемый способ транслитерации, используя для написания русского текста латинские символы, например — Plata priemnika. Но этот способ не очень удобен для чтения. Можно также решать эту проблему, редактируя графические файлы формата PCX, заменяя нарушенные буквенные символы русскими буквами. Но идеальным решением проблем, связанных с отсутствием поддержки символов кириллицы при создании файлов формата PSC, является специально разработанная для этого программа PIPE. Правила пользования этой программой очень просты. Для этого нужно полученный файл формата PLT скопировать в директорию, где находится файл **pipe.exe**, в командной строке набрать

```
pipe file_name.plt,
```

и нажать клавишу Enter. После этого программа запустится, и по окончании работы в этой директории должен появиться файл с тем же именем, но уже с другим расширением (**file_name.kir**). Далее нужно загрузить программу PCPLOTS и использовать полученный файл **file_name.kir** как обычно в случае с файлом формата PLT.

Программы BRD и PLTVIEW можно найти в Интернете по адресу http://www.geocities.com/cvttograph/soft.zip, файл pipe.exe выложен на web-страницу журнала http://radio-mir.com

Литература

1. В.Лузянин. Преобразование файлов P-CAD в другие форматы. — Радио, 2001, №6, С.30.

2. А.Горячкин. Преобразование файлов P-CAD в графические форматы. — Радио, 2002, №1, С.24

С.РЮМИК,
г.Чернигов.

С Интернета по нитке...

Поскольку мы получаем большие выгоды от изобретений других людей, мы должны быть рады послужить остальным каким-либо своим изобретением.
Бенджамин Франклин

Глобальная сеть Интернет в своей основе является бесплатной. В это утверждение трудно поверить, глядя на цены услуг Интернет-провайдеров или на стоимость Интернет-карточек. Тем не менее, факт остается фактом. Разобраться с ребусом поможет **рис.1**, на котором изображена упрощенная схема подключения абонентов ("пойнтов") к условной информационной магистрали Интернет.

Рис. 1

Как следует из приводимых на **рис.2** "формул", функционирование сети Интернет в конечном итоге оплачивают сами абоненты. Получается, что каждый пользователь опосредованно платит за все, в том числе за обслуживание Интернет-шлюзов и за аренду дорогостоящих спутниковых каналов связи. К счастью, из-за большого числа абонентов по всему земному шару, доля глобальных затрат оказывается незначительной. Существует четкая зависимость — чем больше будет абонентов (N, P), провайдеров (M) и шлюзов (K), тем меньше придется платить рядовому пользователю за работу в Интернете.

Рис. 2

Поскольку ни одно из государств мира еще не додумалось брать специальный налог за право доступа к сети Интернет, сам Интернет можно считать бесплатным. Логично, что финансовые расходы, которые несет абонент, хотелось бы чем-то скомпенсировать. Самая большая награда — это доступ к безбрежному морю информации, все остальное зависит только от умения человека ею разумно распорядиться.

Разновидности программ

Информация в Интернете делится на три большие группы — бесплатная, условно-бесплатная и платная. В частности, для компьютерных программ существует следующая классификация.

Freeware (сокращение от free software) — абсолютно бесплатные программы с возможностью многократного тиражирования и с неограниченным сроком действия. Компьютерные остряки называют их "фривольными". Правила хорошего тона предписывают не изменять ни байта в программах

при тиражировании, а также указывать фамилии и имена авторов при ссылке на программу. При внесении своих модификаций в программу не лишним будет оставить упоминание об авторах первоисточника или авторах идеи.

Shareware (сокращение от share software) — условно-бесплатные программы. На компьютерном жаргоне их называют "шароварными", что созвучно с англоязычной транскрипцией перевода. Авторы подобных программ вводят определенные ограничения в интерфейс обслуживания. Например, действует лимит на количество дней работы, на объем вводимой или выводимой информации, на число запусков. В некоторых случаях при начальной загрузке появляется предупреждающая надпись о необходимости регистрации. Бывает так, что к файлу результатов работы дописывается ремарка "Не зарегистрировано!". В shareware-программе могут отсутствовать некоторые важные функции, в том числе может блокироваться запись результатов работы на диск, вывод их на печать и т.д. Чтобы получить полную версию, необходимо выслать автору денежное вознаграждение.

Commercial — коммерческие платные программы, распространяемые за деньги, и, как правило, немалые. За примерами далеко ходить не надо — это все Windows, MS Office, AutoCAD, Adobe Photoshop и многое другое. Коммерческие продукты защищены авторскими правами, и, по идее, их нелегальные копии нельзя использовать при производстве других изделий или для оказания платных услуг.

Перечисленные группы программ являются базовыми, и хотя в последнее время появилось много новых названий, все они так или иначе вписываются в приведенную классификацию. Для того чтобы ориентироваться, что есть что, рассмотрим наиболее типичные случаи, относящиеся к бесплатному программному обеспечению.

Adware (ad-sponsored software, bannerware) — разновидность freeware, в которой содержится баннерная реклама. Обычно это программы, активно использующие Интернет, например, WWW-просмотрщики, Download-клиенты, серферы. При получении такими программами доступа в глобальную сеть, происходит автоматическая загрузка из Интернет новых баннерных роликов. Adware-программы, с точки зрения пользователя, являются абсолютно бесплатными, без каких-либо ограничений. В свою очередь, автор программы получает определенные дивиденды от рекламодателей.

Donationware — занимают промежуточное положение между freeware и shareware. От первого типа они отличаются включением в текст программы просьбы автора прислать ему "кто сколько может", а от второго типа — отсутствием реально действующих ограничений. Такие программы были распространены лет 15...20 назад, в эпоху становления DOS и разработки множества вспомогательных утилит. Программы были небольшими, легко читаемыми, и ставить какую-то защиту на них обходилось "себе дороже".

Spyware (trackware, Big Brotherware) — программа-шпион, которая по запросу извне может отсылать ее автору информацию о параметрах компьютера, адресах E-mail, паролях. Часто такие программы автоматически попадают в компьютер при посещении Интернет-магазинов, Интернет-базаров, рекламных сайтов. Суть простая — подобным способом коммерсанты пытаются выяснить вкусы своих посетителей, их количество и географическое распределение потенциальных покупателей. Если дело ограничивается только этим, то spyware абсолютно безвредны. Неприятность грозит со стороны хакеров, когда какая-нибудь простенькая бесплатная программка, обеспечивающая дозвон до провайдера, может коварно утаскивать ваши пароли и отсылать по Сети своему создателю. Такими свойствами обладают и некоторые вирусы, распространяемые по электронной почте.

Postcardware — разновидность freeware. Доступ к программе пользователь получает только после обязательной регистрации. Для этого требуется отправить автору программы почтовую карточку, открытку или запрос по E-mail, а взамен получить пароль. Цель автора — узнать, насколько популярна его программа, и какому контингенту пользователей она интересна.

Semi-free software (полусвободное программное обеспечение) — разрешает частным лицам использовать, копировать, распространять и модифицировать программы (включая распространение модификаций), но только в неприбыльных целях. Это так называемые программы "для дома, для семьи".

Demoware (demo) — демонстрационная версия программы, предназначенная для рекламы или для предварительного ознакомления с функциональными возможностями продукта. Для пользователя она бесплатна, хотя по большому счету бесполезна. Полная версия программы, как правило, распространяется на коммерческой основе.

К какой из перечисленных категорий отнести собственноручно разработанную программу — решать только ее автору. Единых рекомендаций не существует. Более того, лексика программистов периодически пополняется свежими названиями "...ware"-программ, авторы которых изощряются в придумывании все новых и новых условий их распространения.

Мотивация распространения бесплатных программ

Для рядового Интернет-пользователя интерес представляют, в первую очередь, бесплатные программы freeware и их разновидности. Осторожный читатель вправе заметить, что "бесплатный сыр бывает только в мышеловке". Попробуем проанализировать, какие причины побуждают авторов "раздавать" бесплатный софт.

1. Программа получилась небольшой по объему (выполняемым функциям) или потребовала на ее создание мало времени. Уважающий себя программист не станет подражать "собаке на сене" и отдаст свой скромный труд на всеобщее пользование.

2. Бесплатная программа может стать хорошей рекламой потенциальных возможностей автора. К примеру, открыл страничку в Интернете, поместил на ней описание программы, ввел функцию свободной загрузки. Предположим, что программа вызвала интерес. Ссылки на страничку ставят все ведущие поисковые серверы, автору даются призы на конкурсах, количество пользователей программы исчисляется тысячами. В будущем на автора, как на перспективного программиста, могут обратить внимание солидные фирмы-разработчики. Часто таким путем в Интернет попадают программы из курсовых и дипломных проектов студентов вузов, а также работы учеников средних школ.

3. С помощью бесплатно распространяемой программы автор может создать виртуальный клуб по интересам. Появление новых знакомств, интересный форум, свежие идеи, консультации, взаимовыручка — вот "капитал" автора. Очень часто организуются интернациональные группы программистов, которые поочередно совершенствуют программный продукт. В качестве примеров могут служить многочисленные программы-эмуляторы игровых приставок Dendy, Sega Mega Drive, PlayStation, Nintendo-64.

4. Попытка распространять вирусы под видом бесплатных программ. К сожалению, такие случаи встречаются на практике, поэтому каждую новую скачанную через Интернет программу следует проверять "антивирусом" до ее запуска на компьютере, а не после...

5. Бесплатная программа может являться пробной бета-версией коммерческого продукта, но с еще не до конца исправленными ошибками. Хорошо, если они не фатальные и серьезно не мешают в работе. Для уверенности желательно перепроверить получаемые программой ре-

зультаты каким-нибудь другим способом.

6. Создание бесплатной программы может служить альтернативным ответом любителей-программистов своим профессиональным коллегам. Принцип простой — "...и мы умеем не хуже". Классический пример — оболочка Norton Commander, по подобию которой было выпущено большое количество аналогичных по функциям файловых менеджеров. Более свежий пример — операционная система Linux, задуманная как некоммерческая альтернатива Windows.

7. Бесплатное распространение программы через Интернет может служить действенной защитой от плагиата. Обычно в "свободное плавание" отправляют одну из первых версий программы, в которой четко фиксируется основная идея алгоритма и указывается авторство. Программа выполняет все возложенные на нее функции, но без второстепенного сервиса (видеозаставок, музыки, красивой графики). Дальнейшее совершенствование программы может перевести ее в разряд shareware или commercial, но в любом случае авторские права на идею всегда можно восстановить через многочисленные копии начальной программы на сайтах Интернет-зеркал. "Плагиат" легко обнаруживается и физически удаляется со всех поисковых серверов и сайтов.

8. В разряд бесплатных могут переводиться некоторые нормально работающие, но устаревшие версии коммерческих продуктов. Цель психологически простая — если пользователь заинтересуется бесплатной версией, то с большой степенью вероятности он может купить ее обновление с расширенными возможностями.

9. Существует целый класс бесплатных программ, авторы которых сознательно распространяют их без ограничений, руководствуясь принципами GNU. Поскольку это направление включает в себя целое философское учение, то рассмотрим его подробнее.

Принципы GNU

В 1984 году группа дипломированных специалистов во главе с Ричардом Столлманом (Richard Stallman) основала Проект GNU (GNU Project). Задумана была разработка полностью свободной в распространении Unix-подобной операционной системы, включающей в себя множество свободных программ. Название "GNU" расшифровывается как рекурсивный акроним словосочетания "GNU's Not Unix" ("GNU — это не Unix"). Основные компоненты системы были готовы к 1996 г., когда состоялся ее первый тестовый выпуск. В дальнейшем различные варианты GNU стали базироваться на ядре Linux, что позволило им называться GNU/Linux системами.

Базовым понятием Проекта GNU является "свобода программного обеспечения", которая означает право пользователя свободно запускать, копировать, распространять, изучать, изменять и улучшать его. В английском языке слово "free" переводится двояко — "бесплатный" и "свободный". В Проекте GNU используется второе понятие, которое имеет четыре разновидности:

- свобода 0 — возможность запускать программу на выполнение в любых целях;
- свобода 1 — возможность изучения логики работы программы и адаптации ее к своим нуждам;
- свобода 2 — возможность свободного распространения копий программы;
- свобода 3 — возможность улучшать программу и публиковать свои улучшения без уведомления кого-либо.

Программа считается свободной от ограничений при выполнении всех четырех свобод. Необходимым условием осуществления свободы 1 и свободы 3 является доступ к исходным текстам программы, которые бесплатно предоставляются по запросу. При выполнении свободы 2 пользователь вправе взимать плату за копирование программы (и только!), хотя может распространять ее бесплатно.

Сторонники Проекта GNU выработали своеобразный "кодекс чести", в котором нет места "пиратству" и "плагиату". Существует даже список слов и выражений, не приемлемых в сообществе. Принципам GNU может соответствовать любая программа любого жанра любой операционной системы. Знак авторского права "копирайт" рекомендуется заменять "копилефтом", единственным ограничением которого является запрет накладывать какие-либо ограничения при дальнейшем распространении программы (т. е. свободная программа не должна превращаться в shareware даже после внесения в нее модификаций).

Если внимательно проанализировать философию рассматриваемого течения, то ее основная идея заключается в коллективном накоплении банка программных продуктов, которым мог бы воспользоваться любой желающий. Например, кто-то разработал интересный алгоритм, кто-то — программу, кто-то — драйвер, кто-то — библиотеку функций. Имея в наличии листинг исходного кода с комментариями, можно улучшить программу (алгоритм, драйвер, библиотеку) и опять-таки сделать ее доступной для всех и каждого. В итоге автор программы через некоторое время может бесплатно получить обратно свой продукт, но уже наделенный новыми возможностями. Циклы улучшений повторяются многократно и напоминают восходящее развитие по спирали.

А как же быть с оплатой труда программистов? Внесем ясность. Проект GNU не запрещает брать деньги за услуги по тиражированию свободных программ или за услуги по сопровождению (адаптации) программного обеспечения у конкретного потребителя. Принципы GNU не запрещают разрабатывать коммерческие продукты "на сторону", без этого в реальной жизни не обойтись. Не секрет, что в большинстве случаев оплачивают программы те люди, которые не умеют их писать самостоятельно. Если человек с трудом отличает "фортран" от "ситроена", то ему не поможет знание исходного кода. А вот от программистов у разработчика программ не должно быть секретов. И это основное положение системы GNU.

С изложенной теорией можно соглашаться, не соглашаться, спорить, но она реально существует и поддерживается Free Software Foundation, Inc. (http://www.gnu.org, российское отделение — http://www-ru.gnu.org.ru). Более того, наблюдается тенденция к увеличению числа приверженцев GNU, особенно после появления широкого доступа в Интернет программистов из стран с невысоким средним достатком.

Поиск бесплатных программ

В Интернете существует целый ряд специальных сайтов, на которых собрано множество бесплатных и условно-бесплатных программ с их описаниями. Среди англоязычных сайтов известна большая коллекция программ Tucows (http://tucows.com) с копиями-"зеркалами" по всему миру, в том числе и в российской части Сети (http://tucows.online.ru/). На сервере ZDNet (http://downloads-zdnet.com) программ не так много, но они "отборные", в том числе пробные версии коммерческих продуктов. Следующий сервер — CNET (http://www.cnet.com), содержащий богатую и хорошо структурированную коллекцию. Получая программу, вы точно знаете, что она не перестанет работать через две недели.

Из русскоязычных сайтов наиболее известны:

- http://freeware.ru — коллекция бесплатных программ для Windows, большей частью поддерживающих русский язык;
- http://www.download.ru — собрание программ с их краткой аннотацией;
- http://www.listsoft.ru — этот сайт получил популярность, благодаря листу рассылки по электронной почте, на кото-

рую можно легко подписаться;

- http://soft.zp.ua — коллекция бесплатного софта SuperSOFT (на русском языке), размещенная в украинской части Интернет.

Часто пользователь испытывает чувство разочарования и досады, когда указанная в журнальной статье ссылка на местоположение файла "не работает". Винить автора статьи не следует, скорее всего, ссылка была действительной на момент написания статьи, но затем исчезла. Причин исчезновения множество, и на выяснение подробностей в каждом конкретном случае не хватит таланта Шерлока Холмса. Быть может, проще смириться с объективной реальностью и попытаться найти нужный вам файл в другом месте.

Технология поиска предлагается следующей.

Во-первых, через поисковые машины типа Yahoo!, Google, Yandex попробовать найти ссылки на файл по его полному (частичному) имени, а затем по ключевым словам названия программы. Поиск может привести на новый сайт автора, на сайты коллекционеров программ, на сайты фанов-поклонников.

Во-вторых, на крупных порталах существуют специальные файловые службы поиска, например, на Rambler — это http://ftpsearch.rambler.ru. Поиск необходимо проводить не только по полному названию файла, но и по его части без расширения и последних цифр в названии. Типичный случай — файл "vercus2_3.zip" не найден, зато файл "vercus2_3.rar" имеется. Если автор выпустил более свежую версию программы, то называться она может "vercus2_4.zip", что также необходимо проверить.

В-третьих, попытаться заглянуть в "прошлое" Интернета, побывав на сайте http://web.archive.org, на котором хранится 11 миллиардов (!) html-страниц, начиная с 1996 г. Информация с многих закрытых ныне сайтов хранится только там.

Рис. 3

Рис. 4

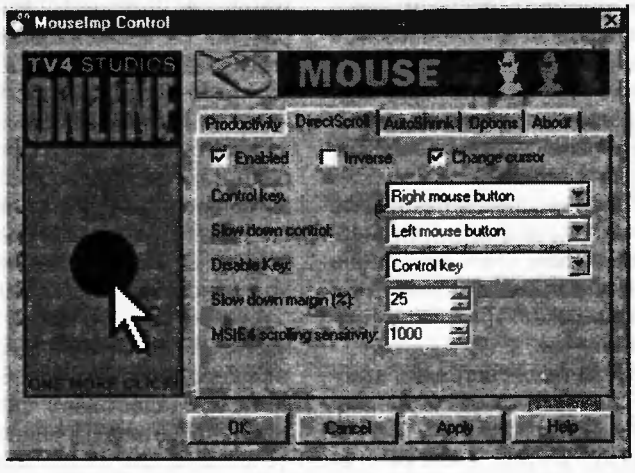
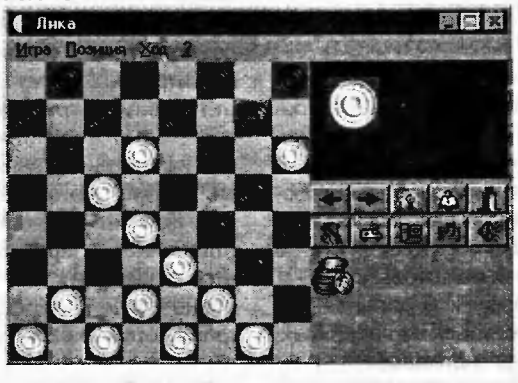


Рис. 5



Подборка описаний бесплатных программ

В качестве примера возможностей и разнообразия бесплатных программ рассмотрим три из них.

KEGA (версия 0.04b, freeware, http://www.eidolons-inn.de/kega/kega_0.04b.zip, 186 Кб) — эмулятор игровых приставок Sega Mega Drive-II, Sega Master System и Sega CD на IBM-совместимом компьютере. Автором является талантливый программист из Великобритании Steve Snake (это его псевдоним), который хорошо известен любителям эмуляции по программе Kgen98. Первая версия KEGA была выпущена 12 января 2002 г. и очень быстро завоевала популярность, благодаря высокому быстродействию, широким возможностям и впечатляющему проценту работающих игр. В рейтинге сайта http://emu-russia.km.ru эмулятор KEGA неизменно получает высокие оценки. Кстати, на этом же сайте можно бесплатно скачать практически все известные в мире игры для Sega Mega Drive-II и Sega Master System. Экранная картинка KEGA-0.04b при запуске игры "Sonic-3" показана на **рис.3**. Цифры "60,3" в левом нижнем углу экрана указывают на количество эмулируемых кадров в секунду, что соответствует работе телевизора в стандарте NTSC без каких-либо замедлений.

MouseImp (версия 1.0.3.6, freeware, http://www.tv4studios. com/downloads/mouseimp.zip, 387 Кб) — программа фирмы TV4 STUDIOS, улучшающая сервис работы со стандартной мышью. Многим известна ситуация, когда, при работе в Microsoft Internet Explorer или Microsoft Word для просмотра большого по объему текста приходится постоянно мышью передвигать бегунок прокрутки вверх-вниз, что весьма неудобно. Программа MouseImp (**рис.4**) позволяет облегчить работу с мышью и обеспечить интерфейс, напоминающий Adobe Acrobat Reader, то есть с помощью "искусственной руки", которая вызывается нажатием и удержанием правой кнопки мыши. Перетаскивание текста вверх-вниз или вправо-влево не составит особого труда, при этом все остальные функции мыши остаются без изменений. Программа MouseImp работает в прозрачном для операционной системы режиме и совместима с Windows 95/98/NT/2000. Число программ, поддерживающих "мышиную искусственную руку", довольно велико (в том числе Adobe Photoshop), но если необходимо расширить возможности, то на сайте http://www.tv4studios.com имеется платная версия MouseImp PRO, распространяемая на условиях "try-before-you-buy".

Шашечная картотека (версия 2.0, freeware, http://chat.ru/~olvn/mc20rsetup.exe, 795 Кб) — сборник логических игр, автор — Олег Адибеков. Фантазия автора сборника позволила создать ему игру "Лика" — гибрид реверси и шашек (**рис.5**). Правила аналогичны обычным русским шашкам, за исключением того, что назад не бьют и дамку не ставят. Сбитая шашка не снимается с доски, а заменяется шашкой противоположного цвета, как в реверси. Бой в игре обязателен. Шашки, дошедшие до края доски, останавливаются. Проигрывает тот, кто не может сделать очередной ход, то есть все его шашки сбиты или заперты. Предусмотрена игра против компьютера или человека, белыми или черными, имеется режим подсказки, отмены хода, установки размеров доски, включение и выключение музыки. Текст помощи и все надписи выполнены на русском языке.

От редакции. Уважаемые читатели, предлагаем продолжить описания интересных программ из Интернета и сделать рубрику "Из Интернета по нитке..." постоянной. Присылайте нам ссылки на Интернет-адреса бесплатного программного обеспечения с краткой характеристикой возможностей и картинкой внешнего вида программы. Темы программ могут быть самыми разными (кому что нравится).

ЛИСТАЯ СТРАНИЦЫ

О том, как редакции журнала "CHIP" удалось **разогнать процессор Pentium 4 2,2 ГГц до частоты 3 ГГц**, рассказано в статье "За гранью возможного" **(CHIP, №3/2002, С.18...21).**

Задача стояла следующая — собрать компьютер из компонентов, имеющихся в свободной продаже, не делать из него оверклокинговую конфигурацию, не применять "экстремальных" методов охлаждения и добиться стабильной работы Windows XP. Эксперимент выполнялся в два этапа. Сначала подняли тактовую частоту системной шины до 133 МГц. Умножитель частоты в процессоре имеет коэффициент умножения 22, благодаря чему тактовая частота процессора составила 2926 МГц. Далее, увеличили напряжение питания микропроцессора до 1,8 В (вместо 1,5 В), поставили второй кулер и установили частоту шины 137 МГц. Частота процессора достигла величины 3014 МГц.

В статье подчеркивается, что ставилась задача разогнать только процессор и системную шину. Это возможно лишь с материнской платой Gigabyte GA8IRXP, у которой шины AGP и PCI можно отстыковать от FSB. Обычно эти три шины работают на частотах 100 МГц (FSB), 66 МГц (AGP) и 33 МГц (PCI). Это соответствует коэффициентам деления 2/3 и 1/3. Но плата GA8IRXP обеспечивает еще и коэффициенты 1/2 и 1/4. Таким образом, при частоте системной шины 133 МГц, AGP- и PCI-шины работают нормально на своих частотах. Это позволяет разгонять только процессор и чипсет, не рискуя вывести из строя остальные компоненты.

При частоте FSB 137 Мгц, частоты AGP и PCI составили 68 и 34 МГц соответственно, при этом система работала устойчиво. Последняя включала:

- процессор Pentium 4 Northword 2,2 ГГц;
- материнскую плату Gigabyte GA8IRXP;
- модуль оперативной памяти PC2100 256 Мб (CL2) от Infineon;
- видеокарту на базе чипа GeForce 3 (драйвер 23.11);
- винчестер IBM 305020 DTLA (UDMA 100);
- звуковую карту SoundBlaster Live! 5.1 Platinum;
- ОС Windows XP Home.

Для тестирования использовались: кодировщик Windows Media Encoder 7.1 (видеофайл размером 128 Мб перекодировался из AVI-формата в WMV), 3D игра Expendable (16 битный цвет, разрешение 640x480), которая сильно нагружает процессор, но не затрагивает ресурсы видеокарты, и DVD-кодирование (преобразовывался VOB-файл длиной 181 Мб в формат DivX с помощью FlaskMPEG 0,6 и кодека DivX версии 4.12). Результаты сравнения с тремя другими вариантами процессора Pentium 4 приведены в **таблице**.

Процессор	Pentium 4 2 ГГц	Pentium 4A 2 ГГц	Pentium 4 2,2 ГГц	Pentium 4 3ГГц
Media Encoder 7.1, сек	231	217	203	159
Expendable, кадров в сек	92	111	119	153
DVD-кодирование, сек	204	200	191	154

По мнению редакции журнала "CHIP", процессор Pentium 4 Northwood заслуживает корону короля оверклокинга. Увеличение частоты системной шины на 37% дало прирост производительности более 20% — результат просто замечательный. Поэтому нет ничего удивительного в том, что уже в середине этого года Intel намерена увеличить частоту шины Pentium 4 до 133 МГц. Pentium 4 мог бы неплохо прибавить в скорости и осложнить тем самым жизнь AMD. Особенно большим потенциалом обладают будущие A-модификации процессора Northwood (2.0A и т. п.). Это демонстрирует всему миру, что Intel опять делает очень хорошие процессоры.

Всем, кто любит слушать громкую музыку, работая на компьютере, или играть в "громкие" игрушки и при этом не мешать окружающим, можно порекомендовать использовать **головные телефоны**. В статье *И.Голубцова* "Имеющий "уши" да услышит" **(ПЛ: компьютеры, №3/2002, С.68...74)** рассмотрены результаты тестирования 11 моделей наушников.

Цель данного теста, подчеркивает автор — определить, какие головные телефоны лучше всего подходят пользователям компьютеров. Ведь в отличие от обычных моделей, такие наушники должны качественно воспроизводить не только CD, но и музыку формата MP3, маскируя побочные эффекты сжатия, а также гармонично работать совместно со звуковыми картами, расширяющими стереобазу за счет реализации эффектов объемного звучания.

По внешнему виду и конструкции многие модели существенно отличаются друг от друга. Почти все наушники состоят из одних и тех же частей: чашек, оголовья и амбушюр. Чашка — это та деталь конструкции, в которую помещается акустический излучвтель. Она имеет непосредственный контакт с ушной раковиной. Часто наушники оснащаются амбушюрами — мягкими валиками, которые располагаются с внутренней стороны чашек и прилегают к голове слушателя. Они, во-первых, обеспечивают комфорт, особенно при долгом использовании, и, во-вторых — частичную изоляцию слушателя от внешних источников звука. Оголовьем называется упругая дуга, соединяющая чашки. Оно может быть пластиковым или металлическим и предназначено для четкой фиксации наушников на голове.

В головных телефонах используется два типа акустического оформления — открытое и закрытое. В первом случае внешняя сторона чашки прозрачна для заука, а во втором — нет. При использовании закрытого акустического оформления легче добиться объемного и мощного звучания, но страдает точность воспроизведения из-за влияния собственных резонансов корпуса чашек. С другой стороны, в открытых наушниках даже при большой громкости слышны внешние шумы.

Амбушюры наушников имеют разную конструкцию. Профессиональные, а также многие домашние модели оснащаются амбушюрами circum-aural. Они представляют собой круглые валики, полностью охватывающие ушную раковину. Такие амбушюры достаточно эффективно препятствуют распространению звука, поэтому они часто используются в моделях с закрытым акустическим оформлением.

Более простые амбушюры supra-aural применяются в дешевых наушниках. Амбушюры данного типа представляют собой мягкие накладные подушечки, которые, в отличие от валиков, не охватывают ушную раковину, а прилегают к ней. Накладные амбушюры используются исключительно вместе с открытыми малогабаритными чашками. Поэтому хорошей акустической изоляции и высокого качества от таких наушников ждать не приходится.

Очень важным элементом конструкции наушников является крепление оголовья и чашки. В дорогих моделях оно выполнено в виде шарниров, позволяющих чашкам вращаться одновременно и в горизонтальной, и в вертикальной плоскостях. Такая конструкция очень удобна, т.к. настройка положения амбушюр происходит автоматически. В дешевых моделях чашки жестко крепятся к оголовью, поэтому изменение их положения может производиться только за счет общей гибкости конструкции.

Оголовье — один из ключевых элементов конструкции наушников. Если оно сильно давит на теменную область или, наоборот, слишком свободно, то даже идеальные амбушюры не смогут создать у слушателя ощущение комфорта.

В наушниках могут применяться электродинамические или пьезоэлектрические звуковые излучатели. Головные телефоны с электродинамическими излучателями устроены так же, как и обычные полноразмерные динамики. Звук в них генерируется с помощью конусного диффузора и прикрепленной к нему магнитной катушки. Такая конструкция имеет свои достоинства и недостатки. Электродинамические излучатели отличаются высоким уровнем звукового давления, они способны воспроизводить широкий диапазон частот и относительно дешевы в производстве. Но, с другой стороны, для того чтобы динамики звучали правдоподобно, их размер должен быть достаточно велик. Поэтому, например, профессиональные наушники всегда очень большие.

Пьезоэлектрические излучатели не способны создать высокий уровень звукового давления. Их главное достоинство заключается в высокой точности воспроизведе-

ния. Поэтому, несмотря на огромную стоимость, пьезоэлектрические наушники иногда используются специалистами, профессионально работающими со звуком. Любительские и большинство профессиональных моделей оснащены электродинамическими излучателями.

Тестировались наушники AKG K66, Koss KTX-PRO, Nady QH 560, Panasonic RP-HT447, SBC HP550 и SBC HP840 ф.Philips, Sennheiser HD200, Sony MDR-201TV, RP-F300 и RP-F400 ф.Technics, Thomson HED615. Все модели имеют электродинамический тип излучателей и стереоадаптер. Остальные параметры приведены в **табл.1**, результаты тестирования — в **табл.2**.

Подводя итоги, автор отмечает, что, несмотря на разнообразие моделей начального и среднего уровня, практически без колебаний удалось выявить победителей трех номинаций. Критерий успеха весьма прост. Наушники должны быть удобными, а звучание — обладать высоким качеством при воспроизведении записей с компакт-диска и MP3-файлов. "Выбором редакции" были признаны наушники AKG K66. Эта модель одинаково хорошо подойдет и любителям компьютерных игр, и тем, кто просто хочет расслабиться под любимую композицию. Высокая мощность, сбалансированное звучание при воспроизведении звуковых файлов различных форматов и удобная конструкция — вот основные достоинства модели. Победителем в номинации "Лучшая покупка" стали наушники Koss KTX-PRO. Возможно, они не выделяются своим внешним видом. Зато качество их звучания — одно из лучших в тесте, несмотря на компактность применяемых акустических систем. Может быть, поначалу некоторым пользователям не понравится непритязательная конструкция данной модели. Но зато, взглянув на цену, они наверняка изменят точку зрения.

Наконец, приз за "Техническое совершенство" завоевала модель Sennheiser HD200. Помимо продуманной конструкции, она выделяется отличным качеством звучания при воспроизведении записей с компакт-диска. В этом ей не было равных.

Табл. 1

Модель	Тип амбушюр	Акустическое оформление	Номинальная выходная мощность, мВт	Пиковая выходная мощность, мВт	Диапазон воспроизводимых частот, Гц	Чувствительность, дБ/мВт	Коэффициент нелинейных искажений, %	Импеданс, Ом	Длина кабеля, м	Вес, г	Регулятор громкости	Цена, USD
K66	Circum-aural	Открытое	200	-	18...22000	96	<1	32	3	210	Нет	45
KTX-PRO	Supra-aural	Открытое	-	-	15...25000	103	<0.2	60	1,5	68	Есть	25
QH 560	Circum-aural	Открытое	-	-	20...20000	110	-	40	3	-	Нет	45
RP-HT447	Circum-aural	Закрытое	-	1000	10...27000	102	-	22	5	290	Есть	25
SBC HP550	Circum-aural	Открытое	-	500	10...28000	102	-	32	1,5	-	Нет	30
SBC HP840	Circum-aural	Открытое	-	1000	8...29000	105	-	40	3	296	Нет	40
HD200	Circum-aural	Закрытое	-	-	12...22000	106	<0.3	64	3	140	Нет	50
MDR-201TV	Supra-aural	Открытое	100	-	14...20000	100	-	24	5	75	Есть	25
RP-F300	Circum-aural	Закрытое	-	1000	10...27000	102	-	22	4	275	Нет	35
RP-F400	Circum-aural	Открытое	-	1000	8...27000	102	-	40	3	205	Нет	45
HED615	Circum-aural	Закрытое	100	-	20...20000	103	-	32	5	-	Есть	21

Табл. 2

Модель	Дизайн	Конструкция	Удобство испсльзования	Качество звучания	Общая оценка
K66	4	4	5	5	5
KTX-PRO	4	4	3	4	4
QH 560	5	3	3	3	3
RP-HT447	4	4	5	4	4
SBC HP550	5	5	4	4	4
SBC HP840	4	3	4	3	3
HD200	3	4	4	5	4
MDR-201TV	2	2	2	2	2
RP-F300	4	4	4	4	4
RP-F400	3	5	4	3	3
HED615	4	4	4	4	4

Игры — один из главных двигателей компьютерного прогресса. После покупки нескольких новых игр нередко приходится приобретать новый процессор или видеокарту. Но настоящий игроман не меньшее внимание уделяет и манипулятору, с помощью которого можно управлять игрой. На них и рассчитан **новый гнущийся геймпад фирмы Maxxtro**, рассмотренный *С.Маленковичем* в статье "FlexiPad: о гибкости управления и радостях игрока" **(ПЛ: компьютеры, №3/2002, С.58...59)**.

Настоящие игроки используют клавиатуру лишь в случае крайней необходимости, замечает автор. Для них созданы мощнейшие арсеналы специальных манипуляторов. Некоторые ухитряются собрать у себя дома два-три джойстика, руль для автогонок и, конечно, игровой планшет.

Очевидно, что эти манипуляторы упрощают игроку жизнь. Здесь есть только нужные кнопки и рычажки, их форма удобна, а запас прочности таков, что даже очень азартный игрок не сможет сломать манипулятор на самом интересном месте любимой игрушки. Но есть и недостатки. Джойстик, например, следует ставить на ровную поверхность и надежно фиксировать — иначе он может перевернуться. Руль тоже жестко крепится к столу, ему требуется очень много места, да и сфера его применения ограничена. Единственный манипулятор, который специально предназначен для постоянного использования на весу, — это геймпад. Его странная форма, немного напоминающая крылья летучей мыши или, скорее, бумеранг, выбрана так, что большие пальцы зависают над основными кнопками, а указательные касаются дополнительных клавиш и переключателей. Теперь с этой игрушкой можно даже скакать по комнате, если, конечно, провод будет достаточно длинным. К столу вы больше не прикованы — уклоняйтесь от выстрелов и поворачивайте свой болид, сколько душе угодно.

Но нет в жизни идеальных вещей, даже геймпад ограничивает игрока: руки находятся на фиксированном расстоянии под определенным углом и, порой, устают от этого. Этот досадный факт заметила фирма Maxxtro, известная разнообразной и весьма доступной по цене периферией — мышками, акустикой, игровыми манипуляторами. Так на рынке появился FlexiPad — самый гибкий геймпад, который мы когда-либо видели. Его форма обычна для этого типа манипуляторов, разве что края "бумеранга" выделяются чуть резче чем обычно. Это не удивительно, ведь геймпад разделен пополам специальной вставкой. На левой части манипулятора имеется традиционная кнопка-джойстик, которая нажимается в восьми направлениях. Справа — шесть кнопок, расположенных в два ряда, а еще две длинные клавиши спрятаны на

торце игрового планшета в передней части. Все кнопки выполнены из гладкого красного пластика.

Двухметровый провод заканчивается стандартным аналоговым разъемом для игрового порта. С подключением и эксплуатацией FlexiPad проблем не возникает. Он прекрасно распознается как в Win9x/Me/XP, так и во всех новых и старых, работающих под DOS играх.

Упомянутая вставка в центральной части геймпада резиновая и потому очень гибкая. Благодаря этому можно "мять" манипулятор в руках, разворачивать его половинки друг относительно друга, располагать руки почти в любом комфортном положении. Эта гибкость действительно пригодится тем, кто управляет гоночной машиной "всем телом".

Этот геймпад проверен и в другой экстремальной ситуации. Использовалась трехмерная игра X-Tension, идеи которой навеяны знаменитой в начале 90-х серией Elite. Чтобы управлять кораблем космического торговца через FlexiPad, сначала требуется установить в Windows игровой манипулятор. К сожалению, крохотное руководство по эксплуатации не содержало никакой полезной информации по установке, дискеты с драйвером тоже не было. Пришлось посетить сайт maxxtro.ru, где и обнаружилась инструкция. Оказалось, что драйверы не нужны, достаточно установить требуемые параметры через "Панель управления", раздел "Игровые манипуляторы". Правда, Windows не позволяет установить больше четырех кнопок. Пришлось схитрить и назвать четыре дополнительных кнопки осями джойстика. Управление оказалось довольно удобным, и то, что геймпад немного сгибался, когда его напряженно сжимали во время полета, действительно помогало рукам расслабиться и сохранить контроль над игровой ситуацией.

С учетом крайне невысокой цены (около 10 USD), С. Маленкович советует всем азартным игрокам запастись гибким геймпадом, перед тем как начать прохождение очередной головокружительной бродилки или трехмерной стрелялки.

Несколько новых антивирусных программ исследовал *Д.Петлин* и рассказал об этом в статье "Мягкий заслон на пути вирусов" **(Hard'n'Soft, №2/2002, С. 88...92)**.

Похоже, что от вирусов уже никуда не деться, констатирует автор. Их разработчики постоянно находят "дыры" в программном обеспечении, минимизируют код, изыскивают новые способы проникновения на компьютеры, все более тщательно скрывают свои "творения" от антивирусов. Немалую помощь в распространении вирусов оказывают и сами пользователи, из-за своей беспечности и любопытства запускающие вложенные в письма программы.

В такой ситуации как на домашний, так и на офисный ПК, помимо традиционного набора программ, полезно устанавливать антивирус. Какой именно? Ответ на этот вопрос Д. Петлин пытался найти, исследуя доступные в России антивирусные пакеты: Norton Antivirus 2001, Stop! 3.2, "Антивирус Касперского" (AVP), McAfee VirusScan 6.0, DrWeb и Norton Antivirus 2002. Все они (за редким исключением) вполне успешно справились с поиском вирусов и лечением от них, предоставив множество сервисных функций. Кроме универсальных пакетов, существует множество небольших утилит, способных защитить от отдельных вирусов. Такие программы, как правило, появляются буквально на следующий день после того, как стало известно о вирусной эпидемии, распространяются бесплатно и доступны в крупнейших файловых архивах Сети (например, на www.download.com).

Исследование антивирусных пакетов проводилось на компьютере под управлением Windows Me (с использованием загруженных с сайта Microsoft обновлений) и с почтовым клиентом Outlook Express 6.0 — эта программа намного аккуратнее обращается с вложениями в письма, чем предыдущие версии. Для проверки антивирусных пакетов использовались вирусы SirCam, Aliz, Nirnda, модификации червя Magistr, а также другие, "кочующие" по электронной почте и Сети. Антивирусные пакеты справлялись и с устранением из ПК троянцев, отмечает автор, однако для этой разновидности вредоносных прогрвмм одних антивирусов явно мало — необходимо использовать еще и брандмауэры.

Пакет Norton Antivirus 2001 (версия 7.0) имеет очень простой, понятный и удобный интерфейс. Программа помещает свою пиктограмму в Tray-область панели задач. При получении почты там же появляется еще один значок в виде конвертика. Если почта успешно получена, он исчезает. Главное окно программы вызывается двойным щелчком по иконке в Tray-области или стандартным образом из меню "Пуск". Вообще, Norton Antivirus 2001 отличает наглядность значков. Сразу можно классифицировать системные сообщения программы — когда возникают серьезные проблемы, а когда все в порядке. Перед началом сканирования системы указываются область проверки и типы файлов, причем сканировать можно и жесткие диски, и сменные носители (дискеты, ZIP, CD). Проверка оперативной памяти происходит автоматически.

Norton Antivirus позволяет формировать отчеты несколько типов, в том числе об обнаруженных вирусах, зараженных и излеченных файлах, а также файлах, находящихся на т.н. карантине — в эту секцию попадают файлы для последующей передачи их Symantec. Программа умеет работать по расписанию. Проверка системы на наличие вирусов происходит достаточно быстро. Встроенный в антивирус почтовый прогу-клиент проверяет почту на наличие вирусов во вложенных в письма файлах (в том числе архивных). Если вирус все-таки проник в систему, включается наивысшая степень защиты — при этом специальную проверку можно провести в режиме DOS.

Пользователь получает предупреждение, даже когда пытается открыть папку с вирусом в файповой оболочке. Такая многоуровневая защита обеспечивает относительную безопасность, а большое количество настроек антивирусного пакета — развитые механизмы защиты. Для пополнения антивирусной базы используется механизм Live Update, но при желании дополнения и обновления к антивирусу можно скачать и вручную.

Norton Antivirus 2002 (версия 8.0) является очередной версией известного антивируса Symantec. Она имеет совершенно новый интерфейс: дизайн программы стал похож на оформление Web-сайтов. Вид кнопок, ссылок и других элементов интерфейса наводит на мысль, что Symantec планирует интегрировать продукт со своим корпоративным сайтом. В Norton Antivirus 2002 появилась возможность проверять исходящие почтовые сообщения, хотя антивирус при этом изрядно докучает отображением небольшого окна с индикатором процесса. По сравнению с предыдущими версиями программы, число настроек Norton Antivirus 2002 уменьшилось. Антивирусный пакет стал еще проще в работе, при этом его функциональность осталась на высоком уровне — Norton Antivirus 2002 не ударил в грязь лицом при поиске и лечении зараженных файлов. Так, в случае обнаружения вируса (а для этого достаточно просто открыть папку, содержащую зараженный файл) антивирусный пакет информирует о зараженных файлах и предлагает их вылечить, удалить или сдать в карантин.

Norton Antivirus 2002 требует больше системных ресурсов, чем предыдущая версия, но зато прекрасно совместим с Windows Me/2000/XP. Обработка почты происходит корректно: сначала сообщения проверяются собственным прогу-сервером, а только потом передаются почтовому клиенту.

Программа Stop! 3.2 — новичок в мире антивирусного ПО и пока не может считаться комплексным антивирусным пакетом, что связано с ее ограниченными возможностями. В частности, в ней не предусмотрена функция автоматической защиты — есть лишь сканер. Вирусы могут сколь угодно долго бесноваться на вашем компьютере, но пока не запущен Stop! (вернее, пока вы не активировали в нем функцию проверки на вирусы), вы никак не заметите их присутствия, если, конечно, подобно вирусу Magistr, они не проявляют себя внешне. Как обычно, перед сканированием можно выбрать область проверки — программа проверяет память, файлы и папки (в т.ч. на сетевых дисках), а также список программ, которые запускаются при старте системы. Антивирусные базы обновляются автоматически при вызове соответствующей команды меню.

Наиболее интересная возможность Stop! связана с мощными эвристическими алгоритмами анализа кодов с целью обнаружения вирусов. К примеру, даже без обновления антивирусных баз, Stop! обнаружил новые варианты уже известных вирусов, например,

Nimda, SirCam и Magistr (впрочем, Stop! наилучшим образом подходит для поиска вирусов-нетроянцев). Это тем более примечательно, что ни "Антивирус Касперского", ни Norton Antivirus 2001 без обновлений вирусных баз их не распознавали. К сожалению, за все приходится платить — такая проверка, будучи включенной, длится довольно долго. В антивирусную базу Stop! включено всего 5 тыс. вирусов, что, конечно же, ничтожно мало по сравнению с 50 с лишним тысячами вирусов Norton Antivirus. Тем не менее, этого количества вполне достаточно для "отлова" вирусов, появившихся за последние несколько лет. Разработчики постоянно улучшают свой продукт — объявлена версия 4.0, кроме того, создается монитор для постоянного наблюдения за системой.

Антивирусный пакет "Антивирус Касперского" 3.5.1 считается лидером отечественного "вирусоборства". Правда, использовать его не так уж и просто. Пакет довольно громоздкий и состоит из множества слабо интегрированных между собой компонентов — их интерфейсы разительно отличаются. К тому же, каждая программа "Антивируса Касперского" так и норовит добавить свой значок в Tray-область панели задач, да и стартует антивирус мучительно: при его запуске вся система на несколько секунд замирает.

Основной компонент антивируса — это "Антивирусный монитор", постоянно находящийся в оперативной памяти компьютера и контролирующий обращения к файлам и секторам диска (главному загрузочному сектору и загрузочным секторам). При обнаружении вируса монитор предлагает вылечить зараженный объект, удалить или заблокировать его. Для лечения вируса сканер приходится запускать вручную. Компонент для сканирования неудобен — например, папка, в которой "Антивирусный монитор" обнаружил вирус, зачастую проверяется в последнюю очередь (или в обычном порядке, но все равно долго), тогда как с нее следовало бы начинать. Процесс сканирования проходит неторопливо, гораздо медленнее, чем в других программах. Остается добавить, что "Сканер" съедает изрядную долю системных ресурсов и заметно замедляет работу системы.

Доступ ко всем компонентам пакета осуществляется из центра управления AVP Control Center. Эта программа — неудачная попытка создать объединенный интерфейс управления. Польза от такого центра управления, по мнению автора, невелика — его можно использовать лишь для работы по расписанию. AVP Control Center добавляет еще одну пиктограмму в Tray-область панели задач и требователен к ресурсам ПК. Кроме того, из центра управления производится обновление антивирусной базы через Интернет в автоматическом или ручном режиме.

В "Антивирус Касперского" входит отдельная программа-инспектор, проверяющая диски на наличие изменений содержимого файлов и директорий, хотя инспектирование можно было бы совместить со сканированием. Еще один дополнительный компонент антивирусного пакета под названием Kaspersky Mail Checker "заботится" о пользователях почтовых клиентов и серверов Exchange/Outlook. Впрочем, обнаружив вирус, он не лечит его, а предлагает сохранить вложение на диске и проверить его сканером. В целом перед почтовыми вирусами "Антивирус Касперского" продемонстрировал свою слабость — проверка осуществляется уже после того как вложенный в письмо вирус попал в почтовый ящик и, возможно, даже успел стартовать (по крайней мере, так происходило в версии Outlook Express 5.0).

В вышедшей недавно версии 4.0 "Антивируса Касперского" есть ряд новых возможностей по поиску и лечению зараженных файлов. Выполненный в стиле Outlook интерфейс четвертой версии антивируса предоставляет доступ ко всем компонентам пакета и средства его настройки. В программе упростилась процедура создания расписания работы компонентов, на отдельной странице центра управления (Control Center) отображаются файлы, изолированные в режиме карантина. "Антивирус Касперского 4.0" создан на базе нового ядра, которое практически всегда распознавало вирусы и их модификации. Инструменты мониторинга блокируют доступ к файлам, предлагая пользователю различные варианты решения проблем, включая удаление объекта с вирусом или помещение его в карантин. Программа тесно интегрируется с ОС, в том числе с Windows XP, и отслеживает попытки запуска зараженных файлов, архивов, а также электронных сообщений Microsoft Outlook Express. Даже если выгрузить из памяти все компоненты антивируса (из Tray-области и из диспетчера задач), попытки обратиться к зараженным файлам все равно будут пресекаться. Эта особенность "Антивируса Касперского 4.0" делает его намного более полезным средством, чем предыдущая версия.

Несколько лет назад антивирус McAfee VirusScan 6.0 был очень популярен, но потом затерялся среди более именитых собратьев. Тем не менее, он продолжил свое развитие и, хотя сегодня не является лучшим продуктом в своей области, не так уж и плох, а его вирусные базы обновляются достаточно оперативно.

Этот антивирусный пакет состоит из двух частей — монитора и сканера. Монитор постоянно находится в памяти и выполняет проверку системы в фоновом режиме. С его помощью можно узнать, сколько вирусов найдено, какие файлы инфицированы и т.д. Впрочем, не в пользу McAfee VirusScan говорит тот факт, что программа не смогла обнаружить вирус SirCam и некоторые другие при входе в папку. А при прямом запуске SirCam он был обнаружен только после заражения системы, и антивирус не смог должным образом с ним справиться (несмотря на то что специально для этого программа и вирусные базы были обновлены). Впрочем, последняя версия антивируса (6.02), появившаяся в декабре прошлого года, без проблем обнаруживала и удаляла вирус SirCam.

С помощью центра управления McAfee VirusScan Center можно сканировать диски — логические/физические, сетевые/локальные, съемные и прочие устройства хранения данных, доступные в системе, составлять в планировщике расписание работы антивирусного пакета, перевести в карантин обнаруженные ранее инфицированные файлы и добавить новые — отсюда файлы отправляются в компанию McAfee для препарирования. Сканирование файлов при помощи этого антивируса производится быстро, однако утверждать, что он способен обеспечить стопроцентную защиту, нельзя.

DrWeb for Windows 95-2000 4.26 — один из самых компактных антивирусов. Свою историю DrWeb начал очень давно, оставшись в памяти благодарных людей как шустрая, качественная и хорошая защита от вирусов. Сейчас DrWeb немного сдал свои позиции. Программа очень мала (дистрибутив занимает приблизительно 3,5 Мб) и состоит из двух интегрированных частей. Монитор отслеживает запуск программ, запись на жесткий диск и действия приложений. В него также встроен планировщик, с помощью которого можно указать, какие операции и когда следует выполнять антивирусу. Сканер либо лечит найденные монитором вирусы, либо проверяет диск на наличие вирусов.

При входе в папку, содержащую зараженные файлы, антивирус никак на них не отреагировал, но при запуске этих файлов сразу обнаруживал вирус. К сожалению, монитор не пресекает загрузку зараженных программ — вирус успевает стартовать и начать свою деструктивную деятельность. При сканировании папки с вирусами DrWeb порой пропускает зараженные файлы, поэтому целиком и полностью доверять этому пакету не стоит.

Подводя итоги, автор отмечает, что McAfee VirusScan проявил себя не очень надежной системой, пропускающей и не замечающей вирусы. DrWeb может быть надежным при тонкой настройке, но ему свойственна та же проблема — пропуск зараженных файлов при сканировании диска. В качестве сканера Stop! показал себя с хорошей стороны, но в настоящее время он не может гарантировать полную защиту от вирусных атак, поскольку не выполняет постоянный мониторинг системы. Хороши и эвристические алгоритмы этого антивируса. "Антивирус Касперского" оказался достаточно надежен и отыскал все вирусы, но у него есть существенный недостаток — он их не лечит сразу, необходимо запускать сканер. Кроме того, программа потребляет много системных ресурсов и имеет неудобный интерфейс; ее можно рекомендовать опытным пользователям. Неплохим решением для дома и небольших офисов является Norton Antivirus 2001, который эффективен при обнаружении почтовых вирусов. А на более мощных ПК можно попробовать Norton Antivirus 2002, который совместим с Windows XP, проверяет входящую и исходящую почту и, кроме того, очень прост в использовании.

М.ДОЛИНСКИЙ,
г.Гомель.

Решение задач на графах

(Продолжение. Начало в №7/2002)

2.3. Задача "Эх, дороги"

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

Имеется N городов, которые пронумерованы от 1 до N (где N — натуральное число, 1<N<100). Некоторые из них соединены двусторонними дорогами, которые пересекаются только в городах. Имеется два типа дорог — шоссейные и железные. Для каждой дороги известна базовая стоимость проезда по ней.

Необходимо проехать из города A в город B, уплатив за проезд минимальную сумму. Стоимость проезда зависит от набора проезжаемых дорог и от способа проезда. Так, если вы подъехали к городу C по шоссейной (железной) дороге X—C и хотите ехать дальше по дороге C—Y того же типа, то вы должны уплатить только базовую стоимость проезда по дороге C—Y. Если тип дороги C—Y отличен от типа дороги X—C, то вы должны уплатить базовую стоимость проезда по дороге C—Y плюс 10% от базовой стоимости проезда по этой дороге (страховой взнос). При выезде из города A страховой взнос платится всегда. Написать программу, которая находит самый дешевый маршрут проезда в виде последовательности городов и вычисляет стоимость проезда по этому маршруту.

Спецификация входных данных

Входные данные находятся в текстовом файле с именем TOUR.IN и имеют следующий формат:

- в первой строке находится число N;
- во второй строке — число M (M — количество дорог, натуральное число, M≤1000);
- в каждой из следующих M строк находятся 4 числа (x, y, t, p), разделенные пробелом, где x и y — номера городов, которые соединяет дорога, t — тип дороги (0 — шоссейная, 1 — железная), p — базовая стоимость проезда по ней (p — вещественное число, 0<p≤1000);
- в последней строке задаются номера начального и конечного городов A и B.

Спецификация выходных данных

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

- в первой строке находится число S — стоимость проезда по самому дешевому маршруту, с точностью два знака после запятой;
- в каждой из последующих строк, кроме последней, находится два числа — номер очередного города в маршруте (начиная с города A) и тип дороги, по которой выезжают из этого города (0 или 1), разделенные пробелом.
- в последней строке находится единственное число — номер города B.

Пример входного файла TOUR.IN

```
3
2
1 2 0 10.00
2 3 1 10.00
1 3
```

Пример выходного файла TOUR.OUT

```
22.0
1 0
2 1
3
```

Метод Дейкстра (города — вершины) не может быть применен непосредственно, поскольку оптимальный маршрут может потребовать заезда в один и тот же город два раза с целью уменьшения оплаты страхового сбора за выезд из этого города.

Вводим 2*N вершин (где N — число городов). Для каждого города — две вершины. Первая — если мы въехали в город по дороге типа 0, вторая — если въехали в город по дороге типа 1.

Теперь можно применять непосредственно метод Дейкстра для вычисления кратчайших расстояний от заданного города A до всех введенных вершин.

Для конечного города B мы выбираем лучший из вариантов — заехать в B по дороге типа 0 или по дороге типа 1.

Метод Дейкстра позволяет сохранять предшественников вершин на оптимальном маршруте, что и используется для вывода полного ответа в заданном формате.

Рассмотрим решение задачи более подробно, основываясь на тексте программы-решения.

Ввод исходных данных осуществляется следующим образом:

```
read(N);                                        {N - число городов}
read(M);                                        {M - число дорог}
for x:=1 to N do                                {Устанавливаем}
  for y:=1 to N do                              {между всеми городами}
    for t:=0 to 1 do p[x,y,t]:=MAXR; {максимальную стоимость}
for i:=1 to M do                          {x - номер города "откуда"}
  begin                                         y - номер города "куда"}
    readln (x,y,t,p[x,y,t]);                    {t - тип дороги}
    p[y,x,t]:=p[x,y,t];                         {P[x,y,t] - стоимость}
  end;                              {проезда от x к y по дороге типа t}
read(A,B);                    {Начальный и конечный пункты маршрута}
```

Подготовка к выполнению алгорима Дейкстра:

```
for i:=1 to N do                                {При выезде из города}
  for t:=0 to 1 do                              {страховой взнос}
    p[a,i,t]:=1.1*p[a,i,t];                     {платится всегда}

for i:=1 to N do                                {Для всех городов}
  begin                          {кратчайшие стоимости  до первого}
    D[i,0]:=P[a,i,0];                           {по шоссейной дороге}
    D[i,1]:=P[a,i,1];                           {по железной дороге}
  end;

for i:=1 to N do                                {Для всех городов}
  begin
    Labl[i,0]:=FREE;                                {Свободны}
    Labl[i,1]:=FREE;                            {по обоим типам дорог}
    pred[i,0]:=A;                               {Предыдущий город - A}
    pred[i,1]:=A;                               {для обоих типов дорог}
  end;
Labl[A,0]:=DONE;                                {Начальный город обработан}
Labl[A,1]:=DONE;                                {для обоих типов дорог}
pred[A,0]:=0;                                   {Предыдущий у начального - 0}
pred[A,1]:=0;                                   {для обоих типов дорог}
```

Основная часть алгоритма Дейкстра представляет собой цикл по количеству вершин, оставшихся необработанными:

```
for i:=1 to 2*N-2 do
  begin
    Поиск ближайшей к первой вершине из необработанных
    Пересчет наименьших стоимостей через ближайшую вершину
          до вершин, смежных с ближайшей
  end;
```

Первый этап — "Поиск ближайшей к первой вершине из необработанных":

```
MinD:=MaxR;                                     {MinD - максимум}
for J:=1 to N do                                {Для всех городов}
  for t:=0 to 1 do                              {Для дорог обоих типов}
    if (Labl[j,t]=FREE){Если въезд в город j по дороге типа t}
        and                                     {еще не обрабатывался и}
       (minD>d[j,t])      {стоимость  от города A до города j}
      then                                  дороге типа t меньше чем MinD, то}
        begin
          Bliz:=j;                                  {Ближайшая - j}
```

```
        bt:=t;                                  {ее тип - bt}
        MinD:=d[j,t                    {минимальную стоимость в MinD}
      end;
Labl[Bliz,bt]:=DONE;          {Пометить как обработанную вершину}
                          {с минимальной стоимостью от города A}
                                  {из еще необработанных вершин}
```

Второй этап — "Пересчет наименьших стоимостей через ближайшую вершину до вершин, смежных с ближайшей":

```
for j:=1 to N do                                  {Для всех городов}
  for t1:=0 to 1 do                         {Для дорог обоих типов}
    if (Labl[j,t1]=FREE)
                  {Если дорога до города j типа t необработана}
      and (d[j,t1]           {и стоимость  до нее от вершины A}
                  >                                   {больше чем}
      d[Bliz,bt] + C(t1,bt)*P[Bliz,j,t1])
                                  {стоимость  через ближайшую}
      then                                                {то}
        begin
          d[j,t1]:=d[Bliz,bt]+C(t1,bt)*P[Bliz,j,t1];
                                                {заменяем стоимость}
          Pred[j,t1]:=Bliz; Typp[j,t1]:=bt;       {запоминаем}
        end;                    {предыдущие город и тип дороги}
```

При вычислении стоимости проезда через ближайшую вершину учитывается страховой сбор при переходе с дороги одного типа на дорогу другого типа с помощью функции C(t1,t2):

```
function C(t1,t2:byte):real;
  begin
      if t1=t2 then C:=1.0 else C:=1.1
  end;
```

Вывод результатов осуществляется так:

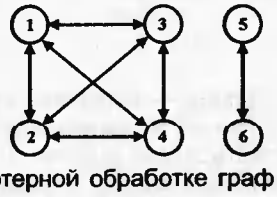
```
G:=B; is:=0;                    {Текущий город - конечный город}
if d[B,0]<d[B,1]                                  {Тип въезда -}
  then t:=0 else t:=1;                 и(лучший из двух возможных)
stt[1]:=t;                      {Текущий тип - выбранный лучший}


while (G<>0) do                    {Пока текущий город не равен 0}
      begin
          inc(is);           {Инкрементируем количество городов}
          stg[is]:=pred[G,t];
                                 {сохраняем текущий город в путь}
          stt[is+1]:=typp[G,t];                      {и его тип}
          NG:=pred[G,t];      {Переустанавливаем текущие город}
          NT:=typp[G,T];            {и тип  на предыдущие город}
          G:=NG; t:=NT;                                 {и тип}
      end;
   writeln(min(d[B,0],d[B,1]):0:2);
                                  {выводим минимальную стоимость}
   for i:=is-1 downto 1 do writeln(stg[i],' ',stt[i]); {и путь}
   writeln(B);                  {добавляем в путь последний город}
```

Далее приводится полный текст решения задачи.

```
{$R+,E+,N+}
program by98d1m2;
    const
        FREE = 0;
        DONE = 1;
        MAXR = 1e4;
var
   p                          : array [1..80,1..80,0..1] of
single;
   D                          : array [1..80,0..1] of single;
   pred                       : array [1..80,0..1] of 0..100;
   typp,Labl                  : array [1..80,0..1] of 0..1;
   stg                        : array [1..100] of 0..100;
   stt                        : array [1..100] of 0..1;
   M                          : 0..1000;
   N,x,y,A,B,i,k,j,G,NG         : 0..100;
   t,t1,t2,bt,NT                : 0..1;
   is,Bliz                      : 0..100;
   Mind                         : single;
   Mint,MT                      : byte;

function C(t1,t2:byte):real;
    begin
        if t1=t2 then C:=1.0 else C:=1.1
    end;
function min(x,y:single):single;
    begin
        if x<=y then min:=x else min:=y
    end;

begin
  assign(input,'tour.in'); reset(input);
  assign(output,'tour.out'); rewrite(output);
  read(N);
  read(M);
  for x:=1 to N do
     for y:=1 to N do
        for t:=0 to 1 do p[x,y,t]:=MAXR;
  for i:=1 to M do
     begin
         readln (x,y,t,p[x,y,t]); p[y,x,t]:=p[x,y,t];
     end;
  read(A,B);
  for i:=1 to N do
     for t:=0 to 1 do p[a,i,t]:=1.1*p[a,i,t];
  for i:=1 to N do
      begin D[i,0]:=P[a,i,0]; D[i,1] :=P[a,i,1]; end;
  for i:=1 to N do
      begin
          Labl[i,0]:=FREE; Labl[i,1]:=FREE;
          pred[i,0]:=A; pred[i,1]:=A;
      end;
  Labl[A,0]:=DONE; Labl[A,1]:=DONE; pred[A,0]:=0; Pred[A,1]:=0;
  for i:=1 to 2*N-2 do
      begin
          MinD:=MaxR;
          for J:=1 to N do
              for t:=0 to 1 do
                  if (Labl[j,t]=FREE) and (minD>d[j,t])
                     then begin
                                Bliz:=j; bt:=t; MinD:=d[j,t]
                          end;
              Labl[Bliz,bt]:=DONE;
              for j:=1 to N do
                 for t1:=0 to 1 do
                    if (Labl[j,t1]=FREE) and
                       (d[j,t1]>d[Bliz,bt] + C(t1,bt)*P[Bliz,j,t1])
                       then begin
                            d[j,t1]:=d[Bliz,bt]+C(t1,bt)*P[Bliz,j,t1];
                           Pred[j,t1]:=Bliz; Typp[j,t1]:=bt;
                              end;
      end;
G:=B; is:=0;
if d[B,0]<d[B,1] then t:=0 else t:=1;
stt[1]:=t;
while (G<>0) do
      begin
          inc(is);
          stg[is]:=pred[G,t]; stt[is+1]:=typp[G,t];
          NG:=pred[G,t]; NT:=typp[G,T];
          G:=NG; t:=NT;
      end;
writeln (min(d[B,0],d[B,1]):0:2);
for i:=is-1 downto 1 do writeln(stg[i],' ',stt[i]);
writeln(B);
close(input); close(output);
end.
```

3. Поиск в глубину

Ниже на рисунке приведен еще один пример неориентированного графа с шестью вершинами.



При компьютерной обработке граф может задаваться списком ребер (дуг) для каждой вершины. Например, для

графа, приведенного на рисунке, этот список выглядит так:
- 1: 2, 3, 4 — первая вершина соединена со второй, третьей и четвертой;
- 2: 1, 3, 4;
- 3: 2, 3, 4;
- 4: 1, 2, 3;
- 5: 6;
- 6: 5.

Для хранения этой информации в памяти компьютера можно воспользоваться двумерным массивом G[1..N,1..M], где N — число вершин в графе, M — максимально возможное число ребер (дуг) у одной вершины графа. Для удобства обработки этой информации можно также иметь одномерный массив kG[1..N] — количества ребер (дуг) вершин графа.

Тогда для обработки списка связей текущей вершины U можно написать:

```
for i:=1 to kG[U]
  begin
    V:=G[U,i];
     ...
  end;
```

Тем самым, мы получаем обработку дуги, связывающей вершины U и V для всех вершин, непосредственно связанных с U.

Для обработки ***всех*** связей всех вершин можно использовать поиск в глубину (DFS — Depth-First Search):

```
...
for U:=1 to N do Color[U]:=WHITE;
for U:=1 to N do
    if Color[U] = WHITE then DFS(U);
...
...
Procedure DFS(U:longint);
   var
       j : longint;
   begin
       Color[U]:= GRAY;
       for j:=1 to kG[U] do DFS(G[U,j]);
   end;
 ...
```

Здесь:
- Color[U] = цвет вершины (WHITE (константа=1) — белый, если мы еще не посещали эту вершину; GRAY (константа=2) — серый, если мы посещали данную вершину;
- DFS(U) — рекурсивная процедура, которая вызывает себя для всех вершин, потомков данной вершины.

Т.е. для обеспечения поиска в глубину на заданном графе G[1..N,1..M] с количествами дуг из вершин G[1..N], вначале мы устанавливаем всем вершинам цвет WHITE, а затем для всех вершин, которые еще не посещались, (Color[G[U,j]]=WHITE) вызываем рекурсивную процедуру DFS.

Таким способом формируются все возможные маршруты в графе. При этом нет ограничений на количество раз использования дуг и заходов в вершины.

Если же по условиям задачи потребуется посещать каждую вершину не более одного раза, чтобы формировать маршруты из несовпадающих вершин, это может быть обеспечено в процедуре DFS условным вызовом:

```
Procedure DFS(U:longint);
    var
        j : longint;
    begin
        color[U]:=GRAY;
        for j:=1 to kG[U] do
            if color[G[U,j]]=WHITE then DFS(G[U,j]);
    end;
```

Если по условиям задачи требуется каждую дугу использовать не более одного раза, то можно ввести расцветку дуг:

Color[1..N,1..M] — со значениями FREE или DONE, где, как и ранее, N — число вершин в графе, M — максимально возможное число ребер (дуг) у одной вершины графа.

А процедура DFS формирования маршрутов таким образом, что бы каждая дуга использовалась в маршруте не более одного раза, будет выглядеть следующим образом:

```
procedure DFS(v:byte);
   var j : byte;
   begin
       for j:=1 to kG[v] do
           if (color[v,j]=FREE)
              then
                  begin
                      ...
                      color[v,j]:=DONE;
                      DFS(G[v,j]);
                      color[v,j]:=FREE;
                       ...
                  end;
   end;
```

Здесь вводятся пометки на дуги Color[v,j] = FREE — если дуга еще не обработана, и DONE — если она включена в текущий путь.

Если в задаче требуется вывести найденный путь, то для его хранения заводится специальный массив SLED [1..Q], где Q — максимальное количество ребер в пути, вершины текущего пути заносятся в этот массив и удаляются из него в процессе обхода графа:

```
procedure DFS(v:byte);
   var   j : byte;
   begin
      for j:=1 to kG[v] do
          begin
              ...
              inc(ks); sled[ks]:=G[v,j];
              DFS(G[v,j]);
             dec(ks);
            ...
          end;
   end;
```

Приведенная теоретическая информация далее иллюстрируется примерами решения конкретных задач.

3.1. Задача "Дороги"

(Республиканская олимпиада по информатике 1997 г.)

Дана система односторонних дорог, определяемая набором пар городов. Каждая такая пара (i,j) указывает, что из города i можно проехать в город j, но это не значит, что можно проехать в обратном направлении.

Необходимо определить, можно ли проехать из заданного города A в заданный город B таким образом, чтобы посетить город C и не проезжать ни по какой дороге более одного раза.

Входные данные задаются в файле с именем PATH.IN следующим образом. В первой строке находится натуральное число N (N=50) — количество городов (города нумеруются от 1 до N). Во второй строке находится натуральное число M (M≤100) — количество дорог. Далее в каждой строке находится пара номеров городов, которые связывает дорога. В последней (M+3)-й строке находятся номера городов A, B и C.

Ответом является последовательность городов, начинающаяся городом A и заканчивающаяся городом B, удовлетворяющая условиям задачи. Ответ должен быть записан в файл PATH.OUT в виде последовательности номеров городов, по одному номеру в строке. Первая

строка файла должна содержать количество городов в последовательности. При отсутствии пути следует записать в первую строку файла число -1.

Пример	Ответ
3	3
2	1
1 2	2
2 3	3
1 3 2	

Кратко идея решения может быть изложена следующим образом:

- поиск в глубину;
- если встречаем вершину B, устанавливаем соответствующий флаг;
- если встречаем вершину C, и флаг B установлен — выводим результат и завершаем работу;
- после завершения поиска (требуемый маршрут не найден) выводим -1.

Изложим решение более подробно. Ввод исходных данных осуществляется следующим образом:

```
read(N,M);
for i:=1 to M do
  begin
     read(x,y); inc(kG[x]); G[x,kG[x]]:=y;
     color[x,kG[x]]:=FREE;
  end;
read(A,B,C);
```

Здесь, как и прежде:

- kG[i] — количество дуг из вершины i;
- G[i,j] (для j от 1 до kG[j]) — вершины, с которыми вершина i связана соответствующей дугой

Кроме того, введен цвет дуги: FREE — свободна, DONE — занята (FREE/DONE — константы).

Главная программа фактически включает только следующие операторы:

```
LabC:=0;        {Установить метку – вершину C еще не посещали}
DFS(A);                          {Поиск в глубину от вершины A}
writeln(-1);       {Вывод признака отсутствия требуемого пути}
```

Рекурсивная процедура поиска в глубину от вершины V выглядит следующим образом:

```
procedure DFS(v:byte);
  var   j : byte;
  begin
    for j:=1 to kG[v] do
      if (color[v,j]=FREE)
         then
           begin
             if (G[v,j]=B) and (LabC=1)
                then begin OutRes; halt; end;
             if G[v,j]=C then LabC:=1;
             color[v,j]:=DONE; inc(ks); sled[ks]:=G[v,j];
             DFS(G[v,j]);
             color[v,j]:=FREE; sled[ks]:=0; dec(ks);
             if G[v,j]=C then LabC:=0;
           end;
  end;
```

Т.е. для всех еще не обработанных (color[v,j]=FREE) дуг от текущей вершины выясняем:

- если она ведет в конечный пункт, а город C уже проходили — вызов процедуры вывода результата и останов;
- если текущая дуга ведет в город C, устанавливаем соответствующую метку (LabC=1).

Помечаем дугу как обработанную, заносим ее в массив SLED, который содержит текущий обрабатываемый путь, KS — количество элементов в нем.

Вызываем процедуру DFS от вершины (G[v,j]), в которую ведет текущая дуга.

Перед выходом из процедуры DFS восстанавливаем состояние, "исходное" перед ее вызовом — снимаем отметку обработки с дуги (color[v,j]:=FREE), удаляем ее из массива SLED (dec(ks)). Оператор sled[ks]:=0; выполнять не обязательно, но он предоставляет удобства отладки — чтобы в массиве SLED ненулевыми были только реально входящие в путь элементы.

И, наконец, процедура вывода результата:

```
procedure OutRes;
  begin
      writeln(ks+2);
      writeln(A); for i:=1 to ks do writeln(sled[i]); writeln(B);
  end;
```

Поскольку по алгоритму построения начальный (A) и конечный (B) города не заносились в массив SLED, они выводятся отдельно, а количество городов в пути (KS) перед выводом увеличивается на 2.

Ниже приводится полный текст программы:

```
program by97d2s3;
const
    FREE=1;
    DONE=2;
var
  G,color                    : array[1..50,1..100] of byte;
  D                            : array[1..50] of longint;
  kG,sled                    : array[1..50] of byte;
  path                         : array[1..100] of byte;
  MinD,is                    : longint;
  i,j,x,y,A,B,C,N,M,ks,LabC  : byte;
  Yes                          : boolean;

procedure OutRes;
  begin
    writeln(ks+2);
    writeln(A); for i:=1 to ks do writeln(sled[i]); writeln(B);
  end;

procedure DFS(v:byte);
  var j : byte;
  begin
    for j:=1 to kG[v] do
      if (color[v,j]=FREE)
        then
          begin
            if (G[v,j]=B) and (LabC=1)
                then begin OutRes; halt; end;
            if G[v,j]=C then LabC:=1;
            color[v,j]:=DONE; inc(ks); sled[ks]:=G[v,j];
            DFS(G[v,j]);
            color[v,j]:=FREE; sled[ks]:=0; dec(ks);
            if G[v,j]=C then LabC:=0;
          end;
  end;

begin
  assign(input,'path.in'); reset(input);
  assign(output,'path.out'); rewrite(output);
  read(N,M);
  for i:=1 to M do
    begin
     read(x,y); inc(kG[x]); G[x,kG[x]]:=y; color[x,kG[x]]:=FREE;
    end;
  read(A,B,C);
  LabC:=0;
  DFS(A);
  writeln(-1);
  close(input); close(output);
end.
```

(Продолжение следует)

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

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

(Продолжение. Начало в №№9-12/2001, №№1-7/2002)

Программирование под WIN32

4. Консольный ввод и знакомство с Идой

#1. В прошлой главе мы разобрались с консольным выводом. Сегодня — разберемся с консольным вводом. Для этого напишем простую программу, запрашивающую строку символов, а затем ее же (строку) и выводящую. Это будет очередной маленький шажок, который для вас вполне может стать решающим, так как именно эта программа впоследствии послужит нам жертвой для негуманных экспериментов, в которых мы впервые используем два хирургических инструмента — отладчик SoftIce (на правах скальпеля) и дизассемблер IDA Pro (на правах томографа). И если при виде окровавленных внутренностей программы вам не поплохеет, более того — если вам *это* понравится, значит, вы имеете неплохие шансы стать либо серийным маньяком (крэкером), либо исследователем программ (реверсером). Все же остальные профессии отныне перестанут для вас существовать ;).

Итак, открываем ASM Editor и набиваем в нем следующий текст:

```
      .386
      .model flat, stdcall
      option casemap :none   ; case sensitive

; ########################################################

      include \tools\masm32\include\windows.inc
      include \tools\masm32\include\kernel32.inc
      includelib \tools\masm32\lib\kernel32.lib

; ########################################################

    .data

      Msg1         db "Type something > "
      Msg2         db "You typed > "
      ConsoleTitle db 'Input & Output',0


; ########################################################

    .code

; ########################################################

Main proc
  LOCAL InputBuffer[128] :BYTE  ;буффер для ввода
  LOCAL hOutPut          :DWORD ;хэндл для вывода
  LOCAL hInput           :DWORD ;хэндл для ввода
  LOCAL lpszBuffer       :DWORD ;адрес буфера
  LOCAL nRead            :DWORD ;прочитано байт
  LOCAL nWriten          :DWORD ;напечатано байт

  ;устанавливаем титл окна
  invoke SetConsoleTitle, addr ConsoleTitle

  ;получаем хэндл для вывода
  invoke GetStdHandle,STD_OUTPUT_HANDLE
  mov hOutPut, eax

  ;печатаем "Type something > "
  invoke WriteConsole,hOutPut,addr(Msg1),17,addr nWriten,NULL

  ;получаем хэндл для ввода
  invoke GetStdHandle,STD_INPUT_HANDLE
  mov hInput, eax

  ;вводим
  invoke ReadConsole,hInput,addr InputBuffer,128,ADDR nRead,NULL

  ;печатаем "You typed > "
  invoke WriteConsole,hOutPut,addr(Msg2),12,addr nWriten,NULL

  ;печатаем то, что ввели
  invoke WriteConsole,hOutPut,addr(InputBuffer),nRead,addr nWriten,NULL

  ;задержка, чтобы полюбоваться
  invoke Sleep, 2000d

  ;выход
  invoke ExitProcess,0
Main endp

; ########################################################

  end Main
```

Строка **LOCAL InputBuffer[128] :BYTE** резервирует 128 байтов памяти под строку символов, которую мы будем запрашивать при помощи апишной функции **ReadConsole**. Вот ее описание:

```
BOOL ReadConsole(
  HANDLE hConsoleInput,         // handle to console input buffer
  LPVOID lpBuffer,              // data buffer
  DWORD nNumberOfCharsToRead,   // number of characters to read
  LPDWORD lpNumberOfCharsRead,  // number of characters read
  LPVOID lpReserved             // reserved
);
```

Попутно дам урок английского языка, который сам знаю не очень: "*number of characters to read*" переводится как "*число символов, подлежащих чтению*", а "*number of characters read*" — как "*число прочитанных символов*". Согласитесь, это существенная разница ;).

Теперь смотрим на следующие строки:

```
;вводим
invoke ReadConsole,hInput,addr InputBuffer,128,ADDR nRead,NULL
...
;печатаем то, что ввели
invoke WriteConsole,hOutPut,addr(InputBuffer),nRead,addr nWriten,NULL
```

В первой мы запросили строку максимум из 128 символов. Введено же, естественно, будет намного меньше. А напечатать нам нужно именно столько, сколько было введено. Помедитируйте над параметром **nRead**, который я выделил жирным. Думаю, вы без труда уловите тайный философский смысл **lpNumberOfCharsRead** ;)

#2. Далее обратите внимание, что адрес переменной мы получаем не при помощи **offset**, а при помощи **addr**. Все их различие заключается в том, что **addr** может работать с локальными переменными, а вот **offset** — нет.

Открою вам страшную тайну! На самом деле локальная переменная — это всего лишь зарезервированное место в стеке. Когда компилятор встречает **addr**, он сначала проверяет, *локальная* это переменная или *глобальная*. Если глобальная, он помещает адрес этой переменной в объектный файл, то есть работает аналогично **offset**. Если же это локальная переменная, то перед вызовом функции генерируется следующая последовательность инструкций:

```
lea eax, LocalVar
push eax
```

Учитывая, что **lea** может определить адрес метки в "рантайме", все работает прекрасно! (c) Iczelion.

Как обычно, я надеюсь на то, что вы не поверили мне на слово. Мы обязательно разберемся с тем, как происходит резервирование места в стеке и обращение к локальным переменным, но сделаем это немного позже. Для начала нам нужно хотя бы чуть-чуть ознакомиться с инструментарием. Начнем мы, пожалуй, с IDA — The Interactive Disassembler.

#3. IDA относится к интенсивно развивающимся продуктам, то есть, вследствие постоянного совершенствования, даже *близкие версии могут вести себя по-разному*. А посему оговорюсь — описываемая мной последовательность действий рассчитана на версию **4.1.5.520**, самым честным образом купленную. Если у вас другая версия, и мои советы "не проходят", то, во-первых, я не виноват, а во-вторых — для разнообразия попробуйте не только тупо следовать руководству, но еще и головой думать (сорри за грубость).

Что касается дизассемблирования вообще, то тут четко необходимо уяснить для себя одну вещь: вследствие того что ***ассемблирование — это однонаправленный процесс с потерями***, автоматическое восстановление исходного текста **невозможно**. Хотя, казалось бы, чего тут сложного — перевод двоичного кода процессора в удобночитаемые мнемоники... А фиг вам, задачка еще та!

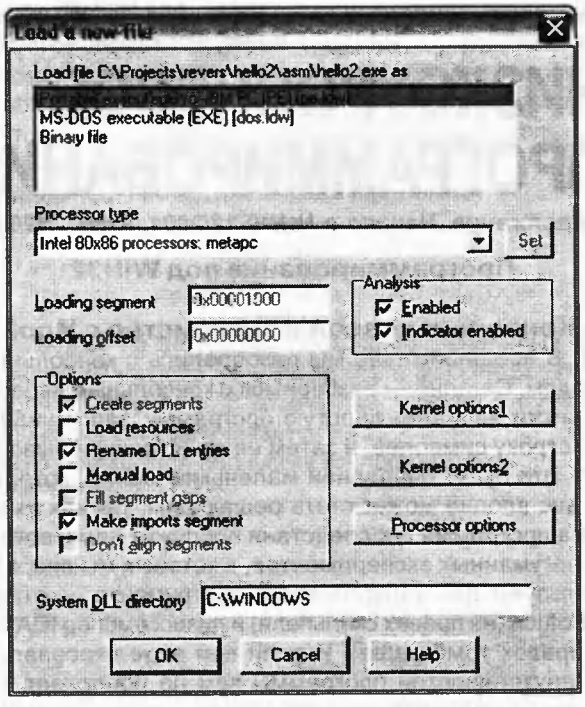
Существуют две категории дизассемблеров — **автономные** и **интерактивные**. Автономные требуют у юзверя все необходимые им указания до начала процесса дизассемблирования и не позволяют вмешиваться в сам процесс. Соответственно, если результат нас не устраивает, или мы желаем попробовать какую-либо дополнительную фичу из предоставляемых дизассемблером, то весь процесс (а для больших программ он может длиться часами!) придется повторять, и, скорее всего, не один раз.

А вот интерактивные позволяют "вручную" управлять процессом "препарирования" программы. В любой момент мы можем сказать дизассемблеру: "Парень, ты гонишь", и помочь этому парню, например, отличить адреса от констант либо определить границы инструкций и т.д. Соответственно, интерактивные дизассемблеры имеют хорошо развитый пользовательский интерфейс, а некоторые (не буду показывать пальцем, все наверняка уже догадались, какой дизассемблер я имею в виду) имеют даже собственный Си-подобный язык скриптов! И даже более того — являются **виртуальной программируемой машиной**!

#4. Итак, давайте проведем первое знакомство с этим плодом человеческого гения...

Запускаем ИДУ и озадачиваем ее нашим исполняемым файлом (File > Open).

Вот что мы увидим:



Я не менял настройки, оставил все default. Но посмотрите, например, на список "Processor type", разве он не впечатляет? Жмем на OK и получаем дизассемблированный листинг нашей программы.

Наверняка вы некоторое время потягаете вверх-вниз вертикальный скроллбар и помедитируете над полученным результатом ;). И только потом начнете читать дальше... Что ж, совершенно правильное поведение ;)) Именно это я называю "медитацией" ;).

Теперь пойдем дальше...

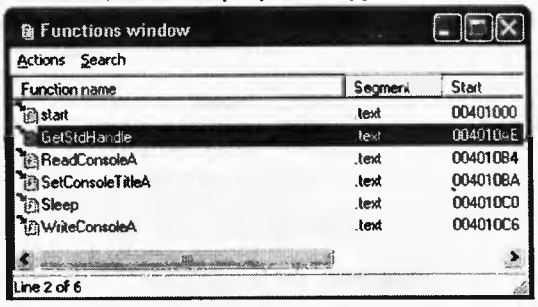
Лезем в пункт меню Views и знакомимся с *некоторыми* его подпунктами. Сразу же совет — запоминайте горячие клавиши того или иного пункта меню. Это сэкономит вам кучу времени ;).

Итак, *View > Toggle dump view* переключит режим отображения из *дизассемблированного листинга* в режим *дампа*, то есть мы увидим простыню шестнадцатеричных циферек:

```
.text:00401000  55 8B EC 81 C4 6C FF FF-FF 68 1D 30 40 00 E8 A7 "UЛьБ-l   h_0@.шз"
.text:00401010  00 00 00 6A F5 E8 94 00-00 00 89 85 7C FF FF FF "...jїшФ...ЙЕ|   "
.text:00401020  6A 00 8D 85 6C FF FF FF-50 6A 11 68 00 30 40 00 "j.НЕl   Pj_h.0@."
.text:00401030  FF B5 7C FF FF FF E8 8B-00 00 00 6A F6 E8 6C 00 " µ|   шЛ...jЎшl."
...
```

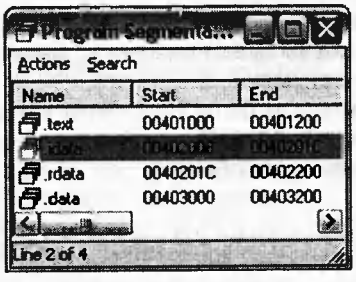
Не правда ли, до боли знакомо? Переключаемся назад в режим листинга нажатием клавиши F4.

View > Open subview > Functions выдаст нам окошко со всеми имеющимися в программе функциями:



Как видим, здесь в одну "простыню" собраны как наши собственные функции (в нашем примере это одна-единственная, с именем **start**), так и обращения к внешним, **апишным** функциям. К каждой из функций прилагается еще и дополнительная информация, подробнее об этом мы еще поговорим.

Кликаем *View* > *Open subviews* > *Segments* и видим следующую картинку:



Как вы уже, должно быть, догадались, это окошко показывает нам, из каких секций состоит дизассемблированная программа.

.text — эта секция содержит исполняемый код. Благодаря 32-битной плоской адресации содержимое аналогичных секций всех объектных файлов, подаваемых на вход линкера, собирается в одной секции .text исполняемого PE-файла.

.Idata — содержит данные об импортируемых приложением функциях, т.е. это **таблица импорта**. Эта таблица состоит из: 1) массива с описанием используемых DLL'ок, 2) двух массивов с адресами импортируемых функций и 3) массива имен импортируемых функций. Страшно?! Не переживайте, эту секцию мы еще рассмотрим детально, а пока что просто знайте, что такая есть ;).

.rdata — содержит данные, доступные только для чтения, как-то: литерные строки, константы, отладочную информацию... Это тоже не берите в голову, попозже мы копнем глубже и эту секцию ;).

.data — содержит инициализированные и глобальные переменные. Как и для секции **.text**, содержимое **.data**-секций всех объектных файлов, подаваемых на вход линкера, собирается в одной секции **.data** исполняемого PE. На всякий случай напомню, что локальных переменных в этой секции вы не найдете.

#5. Двойной щелчок по той или иной секции переместит указатель на то место дизассемблированного листинга, где эта секция начинается. Кликаем *.data* и перемещаемся вот в это место нашего листинга:

```
.data:00403000 ; Segment type: Pure data
.data:00403000 _data           segment para public 'DATA' use32
.data:00403000                 assume cs:_data
.data:00403000                 ;org 403000h
.data:00403000 unk_403000      db  54h ; T  ; DATA XREF: start+2B↑o
.data:00403001                 db  79h ; y
.data:00403002                 db  70h ; p
.data:00403003                 db  65h ; e
.data:00403004                 db  20h ;
.data:00403005                 db  73h ; s
.data:00403006                 db  6Fh ; o
.data:00403007                 db  6Dh ; m
.data:00403008                 db  65h ; e
.data:00403009                 db  74h ; t
.data:0040300A                 db  68h ; h
.data:0040300B                 db  69h ; i
.data:0040300C                 db  6Eh ; n
.data:0040300D                 db  67h ; g
.data:0040300E                 db  20h ;
.data:0040300F                 db  3Eh ; >
.data:00403010                 db  20h ;
.data:00403011 unk_403011      db  59h ; Y  ; DATA XREF: start+6D↑o
.data:00403012                 db  6Fh ; o
.data:00403013                 db  75h ; u
.data:00403014                 db  20h ;
.data:00403015                 db  74h ; t
.data:00403016                 db  79h ; y
.data:00403017                 db  70h ; p
.data:00403018                 db  65h ; e
.data:00403019                 db  64h ; d
.data:0040301A                 db  20h ;
.data:0040301B                 db  3Eh ; >
.data:0040301C                 db  20h ;
.data:0040301D unk_40301D      db  49h ; I  ; DATA XREF: start+9↑o
.data:0040301E                 db  6Eh ; n
.data:0040301F                 db  70h ; p
.data:00403020                 db  75h ; u
.data:00403021                 db  74h ; t
.data:00403022                 db  20h ;
.data:00403023                 db  26h ; &
.data:00403024                 db  20h ;
.data:00403025                 db  4Fh ; O
.data:00403026                 db  75h ; u
.data:00403027                 db  74h ; t
.data:00403028                 db  70h ; p
.data:00403029                 db  75h ; u
.data:0040302A                 db  74h ; t
.data:0040302B                 db    0 ;
.data:0040302C                 align 200h
.data:0040302C _data           ends
.data:0040302C
.data:0040302C
.data:0040302C                 end start...
```

Что мы видим? Длиннющую "простыню" из байтов! Ида нам подсказывает, какой символ соответствует тому или иному шестнадцатеричному значению. Не нужно быть семи пядей во лбу, чтобы догадаться, что эта простыня соответствует следующему куску нашего исходника:

```
.data

      Msg1         db "Type something > "
      Msg2         db "You typed > "
      ConsoleTitle db 'Input & Output',0
```

То есть **unk_403000** (смотрим на дизассемблированный листинг) — это глобальная переменная **Msg1** (смотрим на исходник нашей программы), **unk_403011** — это **Msg2**, а **unk_40301D** — это **ConsoleTitle**.

Теперь посмотрим на нашу дизассемблированную секцию данных. Напротив каждой метки имеется комментарий наподобие **DATA XREF: start+2B↑o**. Но это не просто комментарий — это перекрестная ссылка, которая свидетельствует о том, что к текущему адресу произошло обращение из процедуры **start**. Более того, указывается и адрес, *по которому* происходит обращение — смещение в **2Bh** от начала процедуры **start**. Стрелка указывает на относительное расположение источника перекрестной ссылки, а буква "o" свидетельствует о том, что обращение произошло по смещению (*offset*).

Теперь о главном. Если есть ссылка, то по ней можно (и нужно!) куда-нибудь проследовать, как по обычной html-гиперссылке. Итак, перемещаем указатель на слово **start+2B↑o** в комментарии и жмем на **Enter**!

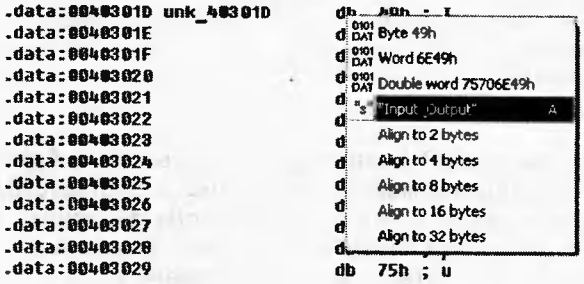
Перепрыгиваем на следующую строчку:

```
.text:0040102B  push    offset unk_403000 ; lpBuffer
```

И в самом деле, мы видим, что обращение к переменной ("пихание" оной в стек) происходит по смещению **2B** при помощи префикса **offset**. И тут же видим ну вообще потрясающую вещь — IDA смекнула, что череда пушей перед **call WriteConsoleA** — это передача параметров соответствующей апишной функции, проанализировала там чего-то... и решила, что эта переменная — **lpBuffer**, совсем как в MSDN'овском описании функции **WriteConsole**! А ниже и хорошо знакомые нам переменные **lpReserved**, **lpNumberOfCharsWritten**, **nNumberOfCharsToWrite**, **hConsoleOutput**. Не правда ли, впечатляет? Сравните с листингами, генерируемыми другими дизассемблерами, и вы поймете, что Иду не зря называют седьмым чудом света ;).

Перемещаем указатель на **unk_403000**, жмем на Enter и перепрыгиваем назад в секцию данных.

#6. Честно говоря, мне не нравится то, какой простыней Ида дизассемблировала блок данных. Хотелось бы лицезреть его в таком виде, каким он был в исходном тексте ;). Для этого ставим указатель на **последний** db в секции данных и жмем на правую кнопку мыши, а в вывалившейся контекстной менюшке



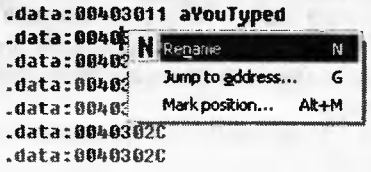
выбираем "s", т.е. "переопределить в строку". В итоге получим крАсивую, и, что самое главное, сходу понятную строчку:

```
.data:0040301D aInputOutput db 'Input & Output',0 ; DATA XREF: start+9↑o
```

Проделаем такую же процедуру и с двумя "вышележащими" метками, получив в итоге следующее отображение секции данных:

```
.data:00403000 _data            segment para public 'DATA' use32
.data:00403000                  assume cs:_data
.data:00403000                  ;org 403000h
.data:00403000 aTypeSomething   db 'Type something > '  ; DATA XREF: start+2B_o
.data:00403011 aYouTyped        db 'You typed > '       ; DATA XREF: start+6D_o
.data:0040301D aInputOutput     db 'Input & Output',0   ; DATA XREF: start+9_o
.data:0040302C                  align 200h
.data:0040302C _data            ends
```

Желающие могут обозвать метки по-своему. Для этого кликните правой кнопкой по адресу и выберите пункт Rename.



И можно вводить любые ругательные слова, которые только придут на ум ;).

#7. Обратите внимание вот на что.

Когда переменные были помечены как **unk_403000**, **unk_403011** и **unk_40301D**, то обращение к ним осуществлялось следующим образом:

```
...
.text:00401009 push offset unk_40301D ; lpConsoleTitle
...
.text:0040102B push offset unk_403000 ; lpBuffer
...
.text:0040106D push offset unk_403011 ; lpBuffer
...
```

Когда мы переопределили данные из байтовых в строковые, то Ида переобозвала переменные в **aTypeSomething**, **aYouTyped** и **aInputOutput**, а обращение стало производиться уже к совершенно другим "именам собственным".

А вот если бы мы переопределили данные начиная не с последней метки, а с первой, то получилась бы строка:

```
.data:00403000 aTypeSomethingY db 'Type something > You
typed > Input & Output',0
```

А обращение к ней осуществлялось бы вот как:

```
...
.text:00401009  push   (offset aTypeSomethingY+1Dh) ; lpConsoleTitle
...
.text:0040102B  push   offset aTypeSomethingY ; lpBuffer
...
.text:0040106D  push   (offset aTypeSomethingY+11h) ; lpBuffer
...
```

Как я уже говорил, на сегодняшний день не существует ни одного полностью автоматического дизассемблера, способного генерировать безупречно работоспособный листинг (вырожденые примеры наподобие нашего — не в счет), поэтому в той или иной мере, но доводить его до готовности приходится человеку. Что мы помаленьку и будем учиться делать.

Засим будем считать, что первое знакомство с Идой состоялось, а в качестве домашнего задания — попробуйте диассемблировать нашу программу Sourcer'ом и WinDASM'ом, чтобы, как говорится, почувствовать разницу ;).

И да пребудет с вами сила, братья! В следующем номере журнала мы с вами будем устанавливать SoftIce...

(Продолжение следует)

Программирование алгоритмов с использованием динамических структур данных

А.ШАКИРИН,
г.Минск, БАТУ, каф.ВТ.

Цель этой статьи — освоить методику создания приложений, в которых используются динамические структуры данных, и рассмотреть примеры создания приложений.

1. Использование динамических массивов

Необходимо создать приложение для вычисления наименьшего и наибольшего из всех значений элементов целочисленной матрицы $A=\{a_{ij}\}$, где $i=1, 2, \ldots, m$; $j=1, 2, \ldots, n$. Значения m и n должны задаваться пользователем на панели интерфейса, а элементы матрицы A — генерироваться с помощью датчика случайных чисел и размещаться в памяти динамически.

Один из возможных вариантов панели интерфейса создаваемого приложения показан на **рис.1**.

Рис. 1

1.1 Размещение компонентов на Форме

Разместим на Форме компоненты Label, SpinEdit, Button и StringGrid.

Сохраним модуль под именем UnDinMas (текст модуля приведен в п.1.3).

1.2 Создание процедур обработки событий FormCreate и Button1Click

Двойным нажатием клавиши мыши на Форме и кнопке Button1 создайте соответствующие процедуры обработки событий. Пользуясь текстом модуля UnDinMas, внимательно наберите операторы этих процедур.

При желании можно создать процедуру, которая будет выделять заданным цветом границы ячеек с наименьшим и наибольшим значениями в компоненте StringGrid. Для создания такой процедуры сделайте активным компонент StringGrid, и на странице Events(события) Инспектора Объектов дважды щелкните мышью в правой части события OnDrawCell. В ответ Delphi создаст обработчик этого события — процедуру **procedure TForm1.StringGrid1DrawCell** и установит курсор между операторами **begin** и **end** этой процедуры. Используя текст модуля UnDinMas, внимательно наберите операторы процедуры TForm1.StringGrid1DrawCell.

1.3 Текст модуля UnDinMas

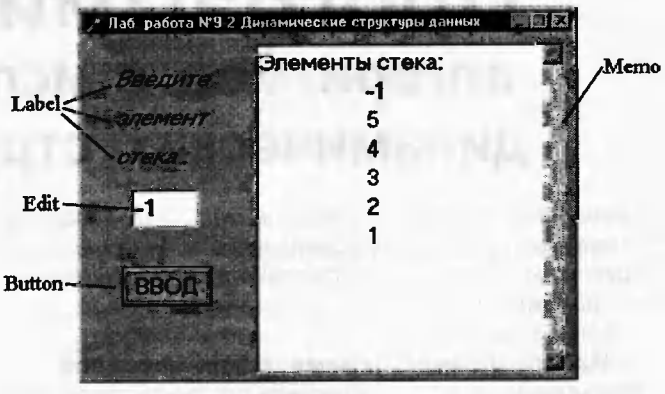
```
Unit UnDinMas;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, Grids, Buttons;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    Label8: TLabel;
    StringGrid1: TStringGrid;
    Label2: TLabel;
    Label5: TLabel;
    Label3: TLabel;
    Button1: TButton;
    Label4: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label9: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure SpinEdit2Change(Sender: TObject);
    procedure StringGrid1DrawCell(Sender: TObject; Col, Row: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}
Type
 Mas=array[1..1] of integer; // массив целочисленных значений
 pMas=array[1..1] of ^mas;   // массив указателей
var                          // объявление глобальных переменных
 pA:^pMas;                   // указатель на массив указателей
 m,n,max,min:integer;
procedure TForm1.FormCreate(Sender: TObject);
begin
 m:=6;                       // начальное значение m
 n:=8;                       // начальное значение n
 SpinEdit1.Text:='6';
 SpinEdit2.Text:='8';
 StringGrid1.RowCount:=m;  // количество строк
 StringGrid1.ColCount:=n;  // количество столбцов
end;
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
 m:=StrToInt(SpinEdit1.Text);// m присваивается содержимое поля редактора
 StringGrid1.RowCount:=m;
end;
procedure TForm1.SpinEdit2Change(Sender: TObject);
begin
```

```
 n:=StrToInt(SpinEdit2.Text);// n присваивается содержимое поля редактора
 StringGrid1.ColCount:=n;
end;
procedure TForm1.Button1Click(Sender: TObject);
label 1;
var
 i,j,k,l,r:integer;
begin
 Randomize;        // инициализация датчика случайных чисел
 GetMem(pA,4*m);  // выделение памяти для массива из m указателей
 for i :=1 to m do
  begin            // формирование i-й строки массива
   { Выделение памяти для n элементов i-й строки}
   GetMem(pA^[i],SizeOf(integer)*n);
   pA^[1]^[1]:=Random(1000);// случайное целое число занести в массив
   for j:=1 to n do
    begin          // формирование j-го элемента строки
     1: r:=Random(1000);   // генерирование случайного числа
     for k:=1 to i do
      for l:=1 to j do
       if r=pA^[k]^[l] then // если такое число уже есть в массиве тогда...
                        goto 1;
     pA^[i]^[j]:=r; // случайное число занести в массив
    end;
  end;
  for i:=1 to m do  // элементы массива занести в ячейки
   for j:=1 to n do // компонента StringGrid1
    StringGrid1.Cells[j-1,i-1]:=IntToStr(pA^[i]^[j]);
 { Поиск min и max значений среди элементов массива}
 max:=pA^[1]^[1];
 min:=max;
 for i:=1 to m do
  for j:=1 to n do
   if pA^[i]^[j]<min then
                    min:=pA^[i]^[j]
                         else
                    if pA^[i]^[j]>max then
                                    max:=pA^[i]^[j];
 Label7.Caption:=IntToStr(min); // вывод min значения
 Label9.Caption:=IntToStr(max); // вывод max значения
 for i:=1 to m do
  { Освобожение памяти, занимаемой n элементами i-й строки)
  FreeMem(pA^[i],SizeOf(integer)*n);
 { Освобождение памяти, занимаемой массивом из m указателей}
 FreeMem(pA,4*m);
end;
procedure TForm1.StringGrid1DrawCell(Sender: TObject; Col,
begin
 with StringGrid1.Canvas do
  if StringGrid1.Cells[Col,Row]=IntToStr(min) then // если элемент ячейки
   begin                                           // равен min тогда...
   Brush.Color:=clGreen; // установить цвет кисти зеленый
   FrameRect(Rect);   // выделить границы ячейки заданным цветом
   end
   else
   if StringGrid1.Cells[Col,Row]=IntToStr(max) then // если элемент ячейки
    begin                                           // равен max тогда...
    Brush.Color:=clRed;  // установить цвет кисти красный
    FrameRect(Rect);  // выделить границы ячейки заданным цветом
    end
end;
end.
```

2. Использование динамических списков

Создадим приложение для формирования стека, который заполняется путем ввода целых положительных чисел с клавиатуры. Как только будет введено первое отрицательное число, содержимое стека должно выводиться на панель интерфейса, а память, занимаемая его элементами — освобождаться.

Рис. 2



Один из возможных вариантов панели интерфейса создаваемого приложения показан на **рис.2**.

2.1. Размещение компонентов на Форме

Разместим на Форме компоненты Label, Edit, Button и Memo.

Сохраним модуль под именем UnStek (текст модуля приведен в п.2.3).

2.2 Создание процедур обработки событий FormCreate и Button1Click

Двойным нажатием клавиши мыши на Форме и кнопке Button1 создайте соответствующие процедуры обработки событий. Используя текст модуля UnStek, внимательно наберите операторы этих процедур.

2.3 Текст модуля UnStek

```
Unit UnStek;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
Type
 PSt=^Zap;
 Zap=record
      inf:integer;
      adr:PSt
     end;
Var            // объявление глобальных переменных:
 PVer,         // указатель вершины стека
 PTek:PSt;     // текущий указатель
 ElSt:integer; // элемент стека
procedure TForm1.Button1Click(Sender: TObject);
begin
 New(PTek);                  // выделить память
 ElSt:=StrToInt(Edit1.Text);// в ElSt занести значение из Edit1
```

```
PTek^.inf:=ElSt;    // в информационную часть стека занести ElSt
PTek^.adr:=PVer;    // в адресную часть занести указатель на вершину
PVer:=PTek;// указатель вершины должен указывать на последний элемент
if ElSt>=0 then     // если элемент стека неотрицательный тогда...
        begin
        Edit1.Text:='';// очистить окно редактора Edit1
        Edit1.SetFocus;// передать фокус ввода редактору Edit1
       end
      else
      begin
       Memo1.Lines.Add('Элементы стека:'); // вывести заголовок
      repeat
       Memo1.Lines.Add(#9+IntToStr(PTek^.inf));// вывод элементов
       PVer:=PTek^.adr;
       Dispose(PTek);  // освободить память
       PTek:=PVer
      until PTek=nil;
     end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
 PVer:=nil;     // инициализировать указатель вершины
 ElSt:=0;       // инициализировать элемент стека
end;
end.
```

3. Индивидуальное задание

В заданиях №№1...15 необходимо использовать динамические массивы, а в заданиях №№16...21 — динамические списки. Во всех заданиях необходимо предусмотреть контрольный вывод исходных данных.

1. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры, меняет местами элементы с наибольшим и наименьшим значениями среди четных и выводит полученный массив.

2. Создать приложение, которое осуществляет ввод ***m*** строк и ***n*** столбцов двумерного массива с клавиатуры и выводит номер строки и номер столбца наименьшего из всех значений его элементов.

3. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры и выводит порядковый номер элемента с наименьшим значением среди нечетных.

4. Создать приложение, которое осуществляет ввод значений элементов двумерного массива ***n***-го порядка с клавиатуры и выводит значение наибольшего из элементов главной диагонали.

5. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры, изменяет порядок следования элементов на противоположный и выводит полученный массив.

6. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры и выводит порядковый номер элемента с наибольшим значением среди четных.

7. Создать приложение, которое осуществляет ввод значений элементов двумерного массива ***n***-го порядка с клавиатуры и выводит значение суммы элементов главной диагонали.

8. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры, меняет местами элементы с минимальным и максимальным значениями и выводит полученный массив.

9. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры и выводит порядковый номер элемента с наименьшим значением среди положительных.

10. Создать приложение, которое осуществляет ввод значений элементов двумерного массива ***n***-го порядка с клавиатуры, изменяет порядок следования элементов главной диагонали на противоположный и выводит преобразованный массив.

11. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры, меняет местами элементы с минимальным и максимальным значениями среди положительных и выводит полученный массив.

12. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры и выводит порядковый номер элемента с наибольшим значением среди отрицательных.

13. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры, меняет местами элементы с наибольшим значением среди отрицательных и наименьшим среди положительных и выводит полученный массив.

14. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры и выводит среднее арифметическое элементов массива.

15. Создать приложение, которое осуществляет ввод ***k*** значений элементов одномерного массива с клавиатуры, меняет местами элементы с наименьшим значением среди четных и наибольшим среди нечетных и выводит полученный массив.

16. Создать приложение, которое заносит в стек целые положительные числа с клавиатуры, выводит содержимое стека и среднее арифметическое его элементов.

17. Создать приложение, которое заносит в стек символы с клавиатуры, выводит содержимое стека и сообщение о том, содержится или нет в стеке заданный символ.

18. Создать приложение, которое заносит в каждый элемент стека английское слово с клавиатуры и, как только будет введено слово "end", выводит содержимое стека.

19. Создать приложение, которое заносит в стек произвольные целые числа с клавиатуры, выводит содержимое стека и сообщение о том, содержится или нет в стеке заданное число.

20. Создать приложение, которое заносит в стек символы с клавиатуры, выводит содержимое стека и сообщение о том, упорядочены ли элементы стека по алфавиту или нет.

21. Создать приложение, которое заносит в стек положительные целые числа с клавиатуры и, как только будет введено число, равное сумме введенных чисел, выводит содержимое стека.

Литература

1. В.В.Феофанов. Delphi 3. Учебный курс. — М.: Нолидж, 2001.

2. Э.Возневич. Delphi. Освой самостоятельно. — М.: Восточная книжная компания, 1996.

3. Дж.Матчо, Д.Р.Фолкнер. Delphi. — М.: БИНОМ, 1995.

4. М.Канту. Delphi 2 для Windows 95/NT. — М.: ООО "Малип", 1997.

Pentium глазами программиста

(Окончание. Начало в №7/2002)

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

Вторая система команд

С точки зрения программиста наиболее важным достижением разработчиков МП семейства Pentium явилось создание второй системы команд. Она состоит преимущественно из SIMD-команд, хорошо приспособленных к особенностям конвейерной обработки. SIMD является аббревиатурой от слов "Single Instruction Multiple Data" — "одна инструкция много данных". В отличие от двухадресных команд общего назначения, оперирующих с двумя числами, SIMD-команды обрабатывают сразу две группы чисел, поэтому мы будем называть их групповыми командами. Новые инструкции позволяют программировать параллельные вычислительные процедуры, а это существенно расширяет сферу возможных применений персональных компьютеров.

Вторая система команд формировалась постепенно, в три приема. Сначала появился МП Pentium MMX, выполнявший группу, состоящую из 57 новых инструкций. Эту же группу инструкций, без каких-либо изменений, поддерживал Pentium II. Название MMX (Multi Media eXtention) разработчики объясняли тем, что при выборе состава команд были проанализированы алгоритмы, применяемые в различных мультимедийных приложениях. Но именно ориентация на эти приложения и неудачно проведенная рекламная кампания сыграли злую шутку с новыми командами. О них поговорили и забыли.

Команды MMX были первой попыткой реализации групповых операций для работы с целыми числами. Следующим шагом были потоковые SIMD-инструкции (SSE1 — Streaming SIMD Extention). Группа, включающая 70 таких инструкций, появилась у Pentium III. В ее состав вошли новые целочисленные групповые операции, а главное — групповые операции с 32-разрядными вещественными числами (обычная точность). Это значительно расширяло круг задач, при программировании которых можно использовать групповые операции.

Наиболее существенное увеличение состава второй системы команд, сразу на 144 инструкции, произошло с выпуском Pentium 4. При этом почти удвоилось количество целочисленных групповых операций, и появились групповые операции с 64-разрядными вещественными числами (двойная точность).

В полном объеме вторая система команд доступна только на МП Pentium 4, она содержит 271 новую инструкцию, а при их записи на языке ассемблера используется 176 новых имен, т.е. одному имени может соответствовать несколько инструкций, различающихся кодами операций. В этом нет ничего нового по сравнению с командами общего назначения.

Вторая система команд позволяет выполнять арифметические и логические операции, сдвиги и сравнения чисел, преобразования форматов, перегруппировку и извлечение отдельных чисел, различные варианты пересылок. Операнды команд могут располагаться в памяти или в **специальных регистрах mmx и xmm**. Следует подчеркнуть, что эти регистры недоступны командам общего назначения.

Регистры **mmx** — 64-разрядные, это просто другое название числовых регистров FPU (процессора, выполняющего операции с плавающей точкой) и другой способ доступа к ним, минуя стек. В них могут находиться **только группы целых чисел**.

Регистры **xmm** — 128-разрядные, они впервые появились у Pentium III. Изначально в них могли находиться только вещественные числа или их группы, а затем (Pentium 4) стало возможно размещение в них групп целых чисел.

В зависимости от типа чисел (целые или вещественные), команды делятся на три категории:

- работа с однородными группами целых чисел, которые могут иметь размер байта (byte), слова (word), двойного слова (dword) или квадраслова (qword). Количество чисел в группе зависит от их разрядности и от разрядности всей группы (64 или 128);
- работа с одной парой 32-разрядных или 64-разрядных вещественных чисел (обычная или двойная точность вычислений FPU);
- работа с четырьмя парами вещественных чисел обычной точности или с двумя парами вещественных чисел двойной точности.

Кроме того, как уже говорилось ранее, есть группа команд, предназначенных для управления процессом кеширования и доступа к памяти минуя кеш.

Ориентация на выполнение групповых операций влечет за собой следующие специфические особенности новых команд.

1. В большинстве случаев отсутствуют признаки, характеризующие результат выполнения операции, которые обычно хранятся в регистре флагов (EFLAGS), исключение описано в п.3. Это делает невозможным использование условных ветвлений по результатам выполнения операций. В частности, при работе с целыми числами невозможно контролировать переполнение результата, и надо изменять алгоритм вычислений так, чтобы исключить переполнение.

2. При сравнении целых или вещественных чисел, разряды, занимаемые каждым числом в группе первого операнда (приемника), либо заполняются единицами, либо очищаются. Все единицы являются признаком выполнения указанного в команде условия для данной пары чисел; все нули означают, что для данной пары условие не выполнено.

3. Исключением являются специальные команды сравнения **двух** вещественных чисел, представленных с обычной и двойной точностью. После их выполнения формируются и помещаются в EFLAGS признаки, характеризующие результат операции, и можно использовать условные ветвления.

Следует отметить два основных недостатка второй системы команд. Во-первых, это отсутствие операции деления целых чисел, и во-вторых, отсутствие команд для вычисления элементарных функций от вещественных операндов — прямых и обратных тригонометрических

функций, логарифмов, степеней и пр. Поэтому программисту придется либо вспоминать способы их вычисления, либо обращаться к FPU для выполнения соответствующих операций.

Групповые операции позволяют максимально использовать вычислительные возможности МП и ускорить выполнение задачи за счет одновременной обработки нескольких чисел. Наибольший выигрыш по сравнению с командами общего назначения получается при использовании групповых команд в циклах, выполняющих обработку больших массивов чисел, различные манипуляции с матрицами и т.п.

Если у вас есть доступ к Internet, то исчерпывающее описание всех команд и рекомендации по их применению вы найдете в технической документации по программированию Pentium 4 на сайте www.intel.com/design/Pentium4/manuals. Документация состоит из трех томов (файлов формата pdf) с общим названием IA-32 Intel Architecture Software Developer's Manual.

Языки программирования и компиляторы

Автор данной статьи убежден в том, что профессиональный программист должен быть полиглотом и уметь работать на таком языке программирования, который лучше всего подходит для решения стоящих перед ним задач. Но когда речь идет об использовании новых возможностей Pentium 4, то выбор оказывается не столь уж большим. В технической документации предлагается использовать компиляторы Intel Fortran Compiler и Intel C++ Compiler.

Подробными сведениями о новых возможностях Intel Fortran Compiler я не располагаю, поэтому мы не будем обсуждать этот вопрос. О том, как используются новые команды при работе с компилятором Intel C++, мы поговорим во второй части данного раздела. А начнем с обсуждения возможности компиляции программ, составленных на языке ассемблера.

Доводы противников программирования на ассемблере автору хорошо известны, и надо сказать, что некоторые из них вполне справедливы, но давайте посмотрим на вещи реально. Современные компиляторы, как правило, позволяют включать ассемблерные коды в текст программы и/или использовать процедуры (подпрограммы), составленные на языке ассемблера. Именно в таких вставках или подпрограммах и можно использовать новые команды, выполняющие групповые операции. Правда, для этого программист должен хорошо знать особенности любимого им компилятора и уметь программировать на ассемблере. Мне могут возразить — не лучше ли использовать новые версии компилятора? Лучше, но существуют ли такие компиляторы, и многие ли имеют доступ к современным лицензионным программным продуктам?

Существует несколько разновидностей макроассемблеров, из них наиболее популярными являются MASM и TASM. Учитывая, что компания Microsoft является основным производителем программных продуктов для IBM PC, речь пойдет о MASM, но все сказанное ниже в равной степени относится и к TASM.

Парадокс! В технической документации, распространяемой Intel, описываются команды ассемблера, приводятся примеры фрагментов программ или целые подпрограммы, состоящие из этих команд, но нигде вы не найдете указаний на то, какая версия MASM (или TASM) "понимает" эти команды и способна компилировать приводимые примеры. Мы вновь возвращаемся к вопросу о доступности компилятора с нужного вам языка.

MASM 6.11 был последней версией, официально распространяемой компанией Microsoft. Был, и все! Но эта версия еще не компилировала инструкции MMX, а тем более, SSE1 или SSE2. Даже если вы готовы заплатить деньги за приобретение новой версии лицензионного компилятора — она просто не продается!

Сказанное не означает, что Microsoft прекратила разработку новых версий MASM — она не может это сделать хотя бы потому, что он нужен для работы других компиляторов, выпускаемых той же Microsoft. Новые версии входят в состав новых компиляторов, например, Си, но отдельно они не продаются.

Однако, "мир не без добрых людей". В Internet можно найти дистрибутив версии 6.14, он называется MASM32 v7 и дополнение к нему до версии 6.15 (ml.exe). Версия 6.14 компилирует все инструкции Pentium III, а 6.15 — все инструкции Pentium 4. Отметим, что MASM32 предназначен для выполнения в 32-разрядной среде, его дистрибутив содержит макроопределения, примеры программ и библиотеки, упрощающие программирование для среды Windows.

Если новые версии MASM вам недоступны, или по каким-то причинам их нельзя использовать, то альтернативным решением являются макроопределения новых команд. Они преобразуют мнемонические записи команд в объектные коды, т.е. выполняют компиляцию тех команд, которые не известны компилятору.

В качестве примера приведем простейшее макроопределение команды CPUID, оно нужно в том случае, если MASM не компилирует команды группы P5 (Pentium I). Как уже говорилось, команда CPUID позволяет программно определить основные характеристики любого МП семейства Pentium.

Пример 4. Макроопределение команды CPUID

```
cpuid  macro          ; заголовок макроопределения
       db 0Fh, 0A2h   ; код команды cpuid
       endm           ; конец макроопределения
```

Если макроопределение из примера 4 расположить в начале исходного текста программы, то в процессе компиляции макроассемблер, обнаружив в теле программы команду CPUID, выполнит это макроопределение, в результате чего в объектный модуль будет включен код 0Fh, 0A2h.

Готовые макроопределения команд MMX, SSE1 и некоторых других можно взять на авторском сайте Агнера Фога — www.agner.com/assem. Они рассчитаны на компиляцию с помощью старых версии MASM 5.10 или TASM 3.0 и, разумеется, более поздних. В каждом файле есть преамбула, поясняющая назначение и способы использования макроопределений.

Таким образом, при программировании на ассемблере существуют реальные возможности компиляции второй системы команд. Лучше, конечно, использовать новые версии компиляторов MASM32, но, в крайнем случае, можно обойтись и макроопределениями.

Замечание: в MASM32 отсутствует директива, предназначенная для описания 128-разрядных слов или групп данных, поэтому при компиляции новых команд не производится проверка соответствия типов операндов. Один из двух операндов, чаще всего первый, обязатель-

но является 128-разрядным регистром (xmm), а второй может содержать имя переменной любого типа (db, dw, dd, dq).

Вернемся к компилятору Си. В технической документации рекомендуется несколько способов использования новых команд, мы отметим только три из них — прямые вставки ассемблерного кода, процедуры (подпрограммы) на языке ассемблера и Intrinsic (встроенные функции или определения новых команд). Два первых варианта не требуют особых разъяснений, а вот встроенные определения команд — это нечто новое.

Во втором томе руководства по программированию МП Pentium 4 (Instruction Set Reference) при описании инструкций MMX, SSE1 и SSE2, кроме записи обозначения команды на языке ассемблера, приводится Intrinsic. Например, в конце описания команды ADDPD, выполняющей сложение двух пар 64-разрядных вещественных чисел, приведены следующие две строчки:

```
Intel C/C++ Compiler Intrinsic Equivalent
__m128d _mm_add_pd (m128d a, m128d b)
```

На сайтах компании Intel можно найти много примеров, иллюстрирующих применение новых команд при программировании вычислительных процедур. В некоторых из них показаны два варианта одной и той же процедуры, составленные с использованием Intrinsic и ассемблерных вставок. Вот строчка из варианта примера, составленного с применением Intrinsic:

```
mx0 = _mm_mul_pd(tx, WM->dm30);
```

В варианте с ассемблерной вставкой ей соответствуют следующие команды:

```
movapd xmm1, [esi+192] ; загрузка двух вещественных чисел в xmm1
mulpd  xmm1, xmm0      ; умножение двух пар вещественных чисел
```

Сравнение этих вариантов показывает, что запись с применением Intrinsic короче, но, на мой взгляд, она менее наглядна. Чему отдать предпочтение — решать вам, уважаемые читатели. Но не забывайте, что Intrinsic можно использовать только при работе с новейшими версиями компиляторов Си.

Intrinsic являются попыткой использования стандартных языковых средств для описания групповых операций. Насколько она удачна — это другой вопрос, важен сам факт этой попытки. Разработчикам компиляторов с языков высокого уровня для МП семейства Pentium еще предстоит пройти путь, который проходили разработчики компиляторов для языков программирования, ориентированных на параллельные вычисления. Но последние имели дело с другой категорией компьютеров, специально приспособленных для параллельных вычислений. Поэтому далеко не все найденные ими решения можно автоматически перенести на компиляторы для IBM PC.

Заключение

Пожалуй, наиболее важным результатом эволюции семейства Pentium явилось появление второй системы команд, большинство из которых позволяет обрабатывать сразу несколько пар целых или вещественных чисел. Кроме того, они лучше чем традиционные команды группы x86 приспособлены к особенностям конвейерной обработки. Поэтому при программировании любых вычислений с вещественными числами, независимо от того, возможно их параллельное выполнение или нет, предпочтение следует отдавать новым командам. При работе с целыми числами новые команды можно использовать только если реализуемый алгоритм допускает параллельные вычисления, поскольку операции с парами целых чисел они не выполняют.

Сделай сам:
Установка и удаление программ

И.ШИРКО,
E-mail: FDC@tut.by

Эта статья открывает цикл статей, посвященных практическому применению Delphi. В этой среде разработки можно создавать приложения любого типа — от простеньких "Hello, World" до сложнейших Internet-проектов. А это значит, что практически любую программу, которую можно отыскать на ПК, можно создать самому. И не только создать, но при этом и улучшить. Ведь, как известно, идеальных программ не бывает. Если у вас есть вопросы, как реализовать ту или иную функцию понравившейся программы в Delphi, либо иные вопросы о Delphi, то пишите мне на E-mail, с радостью постараюсь вам помочь.

В Панели управления (Control panel Windows) находится апплет "Установка и удаление программ". Название говорит само за себя — с помощью этого апплета мы удаляем и устанавливаем программы. Давайте посмотрим, как он работает. При загрузке апплет читает все ключи из раздела реестра *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall* (именно в этот раздел прописываются сведения о деинсталлировании программ). Из каждого ключа читается параметр *DisplayName* (для системных приложений — *QuietDisplayName*) и отображается его значение. Если пользователь выбрал операцию "Добавить/удалить...", то из параметра *UninstallString* считывается и выполняется командная строка.

А теперь реализуем нечто подобное в Delphi, но с некоторыми отличиями — добавим возможность удаления сведений о программе из реестра (если она была удалена "вручную") и удаления компонентов Windows (в "Удаление и установка программ" для этого нужно перейти на закладку "Установка Windows").

Создайте новый проект и разместите на форме три кнопки (Tbutton) и ListBox:TListBox, как показано на **рис.1**.

Рис. 1

В разделе ***Var*** определяются несколько глобальных переменных:

```
var
 i:integer;
```

```
 reg:TRegistry;      //для работы с реестром
 StrList:TStrings;   //список ключей раздела .../Uninstall
 PathList:TStrings;  //командные строки
 DirList:TStrings;   //"отфильтрованные" ключи реестра
 a, b:string;
```

Для события формы OnShow запишите следующую процедуру:

```
procedure TForm1.FormShow(Sender: TObject);
begin
 DirList:=tstringlist.Create; {создаем объекты для хранения строк}
 StrList:=tstringlist.Create;
 PathList:=tstringlist.Create;
 ListBox1.Clear; {очищаем ListBox}
 reg:=Tregistry.Create;   {создаем объект для работы с реестром}
 reg.RootKey:=windows.HKEY_LOCAL_MACHINE;       {будем работать
                          в разделе реестра HKEY_LOCAL_MACHINE}
{открываем нужный ключ}
 reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Uninstall',False);
 reg.GetKeyNames(strlist); {получаем все подразделы этого ключа}
 for i:=0 to strlist.Count-1 do
  begin
   reg.CloseKey;
   reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Uninstall\'+strlist[i],
   False); {перебираем все подразделы}
{читаем из каждого раздела параметр DisplayName}
   a:=reg.ReadString('DisplayName');
{если a - пустая строка, то читаем параметр QuietDisplayName}
   if a='' then
    a:=reg.ReadString('QuietDisplayName');
{если a снова пуста, то записываем в нее имя подраздела}
   if a='' then a:=strlist[i];
{читаем командную строку}
   b:=reg.ReadString('UninstallString');
{если b - пустая строка, то читаем параметр QuietUninstallString}
   if b='' then b:=reg.ReadString('QuietUninstallString');
{если строка b не пустая, то добавляем параметры в списки: имя
ключа, название программы, командная строка}
   if b<>'' then
    begin
     ListBox1.Items.add(a);
     PathList.add(b);
     DirList.Add(StrList[i]);
    end;
  end;
{уничтожаем объект reg}
 reg.Free;
end;
```

При показе формы в ListBox'е отобразятся доступные для удаления\установки программы. Теперь для кнопки "Удалить программу" запишите процедуру для события OnClick:

```
procedure TForm1.Button1Click(Sender: TObject);
var
 si:Tstartupinfo;
 p:Tprocessinformation;
 exe:string; {командная строка}
begin
{если в ListBox'е выделена строка, то продолжаем}
 if ListBox1.ItemIndex<>-1 then
  begin
{получаем командную строку для выбранного элемента}
   exe:=PathList.Strings[listbox1.ItemIndex];
{сворачиваем окно нашей программы, запускаем командную строку,
дожидаемся завершения удаления\установки приложения и
восстанавливаем окно нашей программы}
   FillChar( Si, SizeOf( Si ) , 0 );
   with Si do
    begin
     cb:=SizeOf( Si);
     dwFlags:=startf_UseShowWindow;
     wShowWindow:=4;
    end;
   Application.Minimize;
  Createprocess(nil,pchar(copy(exe,pos(';',exe)+1,length(exe))),nil,nil,false,
 Create_default_error_mode,nil,nil,si,p);
   Waitforsingleobject(p.hProcess,infinite);
   Application.Restore;
{обновляем данные об установке\удалении программ}
   Form1.OnShow(self);
  end;
end;

Следующая процедура обрабатывает событие OnClick кнопки
'Удалить из списка':
procedure TForm1.Button2Click(Sender: TObject);
var
 Dir:string; {имя ключа}
begin
{Если в ListBox'е выбран какой-нибудь элемент, то продолжаем}
if ListBox1.ItemIndex<>-1 then
 begin
{получаем имя ключа, который нужно удалить}
  Dir:=DirList[listbox1.ItemIndex];
{удаляем нужный ключ}
  reg:=Tregistry.Create;
  reg.RootKey:=windows.HKEY_LOCAL_MACHINE;
  reg.OpenKey('Software\Microsoft\Windows\CurrentVersion\Uninstall',False);
  reg.DeleteKey(dir);
  reg.Free;
{удаляем из ListBox'а выбранный элемент}
  listbox1.items.Delete(ListBox1.ItemIndex);
 end;
end;
```

Основная часть на этом завершена. Теперь осталось добавить несколько мелочей для повышения удобства при работе с программой:

1. Выделите кнопку "Обновить", и на вкладке Events Инспектора объектов, напротив события OnShow, из раскрывающегося списка выберите процедуру FormShow.

2. Таким же образом для события OnDblClick ListBox'а выберите процедуру Button1Click. Теперь при двойном щелчке мыши по выбранной программе, для нее запустится Uninstall.

3. Для события OnKeyPress ListBox'а запишите следующую процедуру:

```
procedure TForm1.ListBox1KeyDown(Sender: TObject; var Key:
Word; Shift: TShiftState);
begin
{если нажата клавиша Delete, то удаляем информацию о выбранной
программе из реестра}
 if key=VK_DELETE then form1.Button2.OnClick(self);
end;
```

Вот и все. Предложения по темам следующих статей этого цикла присылайте на мой E-mail.

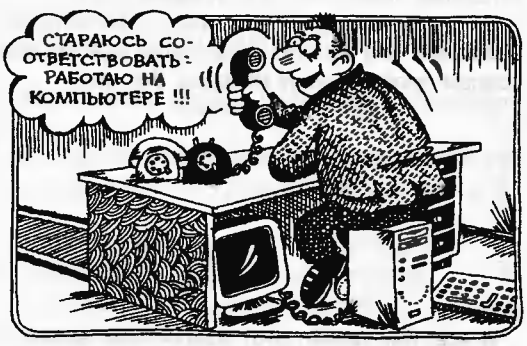


Рисунок О.Попова

Svet(r)off /HI-TECH,
http://hi-tech.nsys.by/

Как писать на MASM в строчку

В этой маленькой статье речь идет совсем не о том, что TASM лучше MASM'а, а о том, как записывать последовательности команд в исходном тексте в одну строку.

Настоящим ассемблерщикам рекомендуется непосредственно перед прочтением нижележащего крамольного еретического текста выполнить следующие операции:

1. Взять 20 капель спиртовой настойки валерианового корня (Tinctura Valerianae).

2. Влить упомянутые в п.1 20 капель в высокую узкую стеклянную емкость емкостью 0,5 л.

3. Дополнить упомянутую в п.2 емкость до уровня золотого ободка пивом (beer, пиво), по вкусу.

4. По рецепту Д.Бонда взболтать, но не размешивать.

5. Употребить полученную суспензию внутрь большими глотками в три приема: сначала 0,25 л + 10 капель, затем 0,125 л + 5 капель, и затем оставшиеся 0,125 л + 5 капель.

6. Кликнуть на ссылку <http://192.168.1.10/hi-tech/forum/>, найти реквизиты подписки на конференцию RTFM_Helpers и активно участвовать в ней, выбросив из головы кошмарные идеи, изложенные в этой статье.

Данный совет продиктован исключительно заботой о здоровье настоящего ассемблерщика, который, как известно, испытывает непереносимые мучения от любого комфорта и просто-таки титаническим усилием воли заставляет себя писать программы с помощью мнемокодов команд, а не одной только директивы db. Ибо звучат еще в наших ушах бессмертные слова: "Ну вы, блин, заставьте нас еще горизонтально исходник скроллить, как в вижуал васике!".

Ну вот, продолжаем для оставшихся ренегатов и предателей.

Ассемблер — величайший язык, царь языков, практически лишенный недостатков, достойный всенародной любви и поклонения — почти таких же, как заместитель руководителя Администрации Президента Российской Федерации. Но есть у него (у ассемблера, а не, упаси боже, заместителя) один не то чтобы недостаток, а так, мелкая мелочь.

Неприятность этой мелкой мелочи мы ощутили во всем ее необозримом масштабе, когда спустя три дня после окончания гарантийного срока "сдох" наш любимый Sony GDM-200, и временно перекочевал под стол вместе со своими 1280x1024. А вместо него на столе оказался старенький SyncMaster 500s, с натугой выдающий 800x600. С тех пор до 80% нашего рабочего времени уходило на переключение между окнами и на, чтоб он пропал совсем, скроллинг исходного текста вверх-вниз.

Существенно сократить число ежесекундно сгорающих нейронов нам удалось благодаря одному-единственному совсем простенькому макросу. Вот он:

```
@ MACRO p0,p1,p2,p3,p4,p5,p6,p7
p0
p1
p2
p3
p4
p5
p6
p7
ENDM
```

И все! Как только этот макрос был включен в нашу постоянную макробиблиотеку, мы получили счастливую и давно желаемую возможность записывать в одну строчку несколько (а именно — от 2 до 8) команд.

Возьмем, к примеру, самую что ни на есть типичную ситуацию. Нужно скопировать фрагмент памяти в новое место. Что запишет в исходный текст кто-нибудь из покинувших нас настоящих ассемблерщиков? Ну, например, нечто вроде этого:

```
;Copy from one buffer to another
mov esi,one_buffer
mov edi,another_buffer
mov ecx,how_many_bytes
cld
rep movsb
```

А что запишем мы, ренегаты и предатели? А вот что:

```
;Copy from one buffer to another
@<mov esi,one_buffer>,<mov edi,another_buffer>,<mov
ecx,how_many_bytes>,<cld>,<rep movsb>
```

Вот какие преимущества, по нашему скромному IMHO, мы получаем в результате этого маневра:

1. Про уменьшение трудозатрат на вертикальный скроллинг уже было сказано.

2. Кроме того, наше подсознание перестает возиться с понятием "команда" и переходит на более высокий уровень — "блок". Результатом чего, очевидно, должно явиться повышение производительности труда.

3. У нас появляется дополнительный стимул к хорошему комментированию исходного текста.

Может оно, казалось бы, ассемблерщику и ни к чему. Но это только до тех пор, пока не соберешься разобраться в собственном коде, а то и, не дай Бог, кто-нибудь не захочет купить наши гениальные исходники за 100000 "зелени". Вот тогда-то и скажешь себе — вот дурак-то был, когда комментарии не писал!

4. С повышением потребности в комментировании возрастают наша грамотность, культура, любовь к языку, родному и английскому, а также умение кратко выражать свои мысли, причем не только в троллейбусе.

5. Облегчается копирование текста. Допустим, за год работы мы копируем текст с места на место 50000 раз, в среднем по 10 строк. Так вот, сравните, каково: во втором случае мы копируем 50000 строк, а в первом — 500000! В десять раз больше, но за ту же зарплату!

Освоив предлагаемый метод, вы можете смело идти к своему руководителю и рекомендовать ему снизить вам зарплату в 10 раз. Гарантируем — с этого момента вы будете его любимым сотрудником.

6. Появляется возможность наконец-то изучить клавишные макросы рабочей среды. Щелк на горячую клавишу — и группа строк свернулась в одну строку. Щелк на другую — и строка развернулась в группу строк. Отлаживай, дорогой.

7. При желании можно считать, что повышается читабельность исходного текста.

Недостаток у предлагаемого метода только один — вот сиди теперь и думай, чего это я, дурак, все с ассемблером вожусь? Может, на Visual Basic перейти?

Дальнейшее совершенствование предложенного метода видится нам на пути увеличения числа параметров макроса. Предварительные оценки показывают, что добавление только одного параметра p8 может привести к увеличению числа обрабатываемых макросом строк примерно на 12,5%. Желающие могут провести соответствующие эксперименты с целью уточнения этой оценки.

Желающие полюбоваться новым видом исходника и почерпнуть кое-какие стилевые идеи, могут кликнуть на hi-tech/pub/coding/win32/files/servicedx.zip либо зайти на страницу журнала http://radio-mir.com.

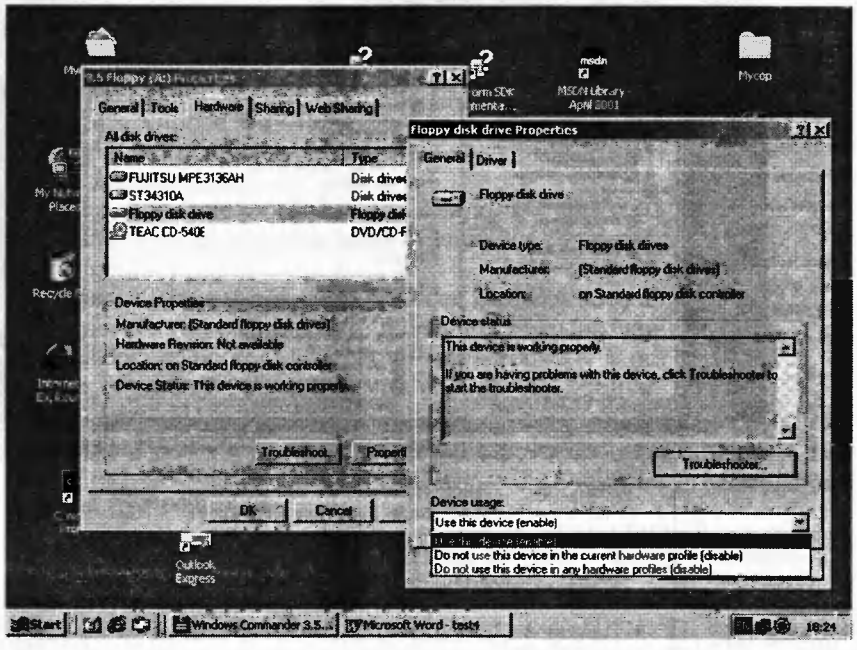
Программное управление доступностью устройств в среде Windows 2000

М.РЕВОТЮК,
г.Минск, БГУИР, кафедра ИТАС,
E-mail: rmp@gw.bsuir.unibel.by

Ряд практических ситуаций порождает необходимость исключения из списка доступных некоторых установленных на компьютере устройств. Например, при работе компьютера в ответственных системах, учебных классах или даже в домашних условиях может потребоваться отключение CD-ROM, floppy-диска, модема или сетевой карты. В среде Windows 2000 пользователь с достаточными привилегиями для таких действий может воспользоваться стандартным интерфейсом, используя опции "Enable/Disable" MMC [1].

На **рис.1** приведен пример управления доступностью floppy-диска.

Рис. 1



Однако ручное манипулирование устройствами характеризуется низкой оперативностью и не всегда приемлемо. Предмет обсуждения в этой статье — легитимное управление доступностью устройств из прикладной программы или пакетных командных файлов.

В среде Windows 2000 имеется по меньшей мере два пути организации корректного доступа к управлению устройствами [1, 2]:

- PnP Configuration Manager;
- Setup API.

Соответствующие интерфейсные функции определены в файлах cfg.h и cfgmgr32.h, однако они документированы не полностью. Их назначение по отношению к рассматриваемой задаче пришлось выявлять посредством систематизации разрозненных сведений из фирменной документации, анализа заголовочных файлов и примеров исходных текстов.

Описание установленных в среде Windows NT/2000 устройств имеет древовидную структуру, представляя сервисы уровня драйверов физических устройств и более высокие уровни, например, файловой системы или протоколов обмена. Отдельное устройство, в конце концов, представлено конкретным листом дерева. Однако, т.к. взаимозависимость компонентов вычислительной среды автоматически учитывается, путь к устройству часто можно задать указанием промежуточного узла дерева.

Одним из ключевых признаков узла дерева установленных устройств на логическом уровне является так называемый "класс" (*CLASS*) устройства.

Важнейшими стандартными классами вычислительной среды Windows 2000 [1, 2] являются:

```
Class = CDROM
Class = DiskDrive
Class = Display
Class = FDC
Class = FloppyDisk
Class = HDC
Class = Image
Class = Keyboard
Class = MediumChanger
Class = Media
Class = Modem
Class = Monitor
Class = Mouse
Class = Multifunction
Class = MultiportSerial
Class = Net
Class = NetClient
Class = NetService
Class = NetTrans
Class = Ports
Class = Printer
Class = SCSIAdapter
Class = System
Class = TapeDrive
Class = USB
Class = LegacyDriver
Class = Unknown
Class = Printer Upgrade
```

Другими признаками, однозначно соответствующими признаку *CLASS*, но конкретизирующими представление физических устройств, являются:

- имя сервиса (*SERVICE*), обслуживающего устройство (драйвера);
- дружественное имя (*FRIENDLYNAME*);
- описание устройства (*DEVICEDESC*);
- глобальный уникальный идентификатор класса (*CLASSGUID*).

На **рис.2** представлено дерево устройств и компонентов системы.

Виды запрашиваемой информации об узлах дерева устройств перечислены в заголовочном файле cfgmgr32.h [2], например:

- `CM_DRP_SERVICE` — имя сервиса устройства;
- `CM_DRP_FRIENDLYNAME` — дружественное имя;
- `CM_DRP_DEVICEDESC` — описание устройства;
- `CM_DRP_CLASS` — класс;
- `CM_DRP_CLASSGUID` — глобальный уникальный иден-

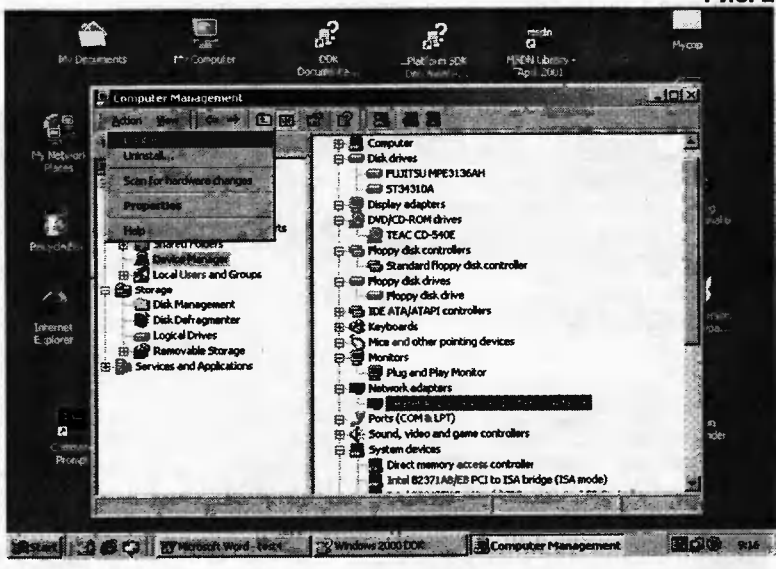
тификатор класса.

Можно заметить, что признаки узла отличаются степенью абстрагирования.

Ниже представлен исходный текст иерархии классов, проверенный в системе программирования MS Visual C++ 6.0:

```
//////////////////////////////////////////////////////////////
// Классы управления доступностью устройств
//   Файл DevNodeCheck.h
//////////////////////////////////////////////////////////////

#ifndef DISABLE_ENABLE_DEVICE_NODE#define
DISABLE_ENABLE_DEVICE_NODE#define _WIN32_WINNT 0x0500#include
<windows.h>#include <cfg.h>#include <cfgmgr32.h>#include
<stdio.h>#include <conio.h>#pragma comment(lib,»cfgmgr32.lib»)

//////////////////////////////////////////////////////////////
// Сканирование дерева устройств
//////////////////////////////////////////////////////////////

class DevNodesScaner {
 static ULONG QuestModes[];

protected:

 DEVINST dnDevInst;
 CONFIGRET config;

 ULONG GetStatus();

public:

 DevNodesScaner() {}
 void LookFor(PTCHAR Node);
 virtual void NodeOperation(PTCHAR Node);
 virtual ~DevNodesScaner() {}
};

// Список анализируемых параметров узла

ULONG DevNodesScaner::QuestModes[] = {
 CM_DRP_SERVICE,      // имя сервиса
 CM_DRP_FRIENDLYNAME, // дружественное имя
 CM_DRP_DEVICEDESC,   // описание устройства
 CM_DRP_CLASS,        // класс устройства
 CM_DRP_CLASSGUID     // глобальный уникальный идентификатор
};

// Поиск узла дерева устройств

void DevNodesScaner::LookFor(PTCHAR Node) {
 for (ULONG EnumIndex=0; ;EnumIndex++) {
  TCHAR EnumBuffer[MAX_DEVICE_ID_LEN];
  ULONG Length = sizeof(EnumBuffer);
  config = CM_Enumerate_Enumerators(
              EnumIndex,EnumBuffer,&Length,0);
  if (config==CR_NO_SUCH_VALUE) break;
  Length=0;
  config = CM_Get_Device_ID_List_Size(
           &Length,EnumBuffer,CM_GETIDLIST_FILTER_ENUMERATOR);
  if (config==CR_SUCCESS) {
   PTCHAR ListBuffer = new TCHAR[Length+1];
   if (ListBuffer) __try {
    config = CM_Get_Device_ID_List(
                EnumBuffer,ListBuffer,Length,
                CM_GETIDLIST_FILTER_ENUMERATOR);
    if (config==CR_SUCCESS) {
     for (PTCHAR Temp=ListBuffer, Item=0; *Temp; Temp++) {
      for (Item = Temp++; *Temp; Temp++);
        config = CM_Locate_DevNode
       (&dnDevInst,Item,CM_LOCATE_DEVNODE_PHANTOM);
      if (config==CR_SUCCESS) {
       TCHAR buf[MAX_DEVICE_ID_LEN];
       for (int i=0; i<sizeof(QuestModes)/sizeof(*QuestModes); i++) {
        ULONG len = sizeof(buf);
        config = CM_Get_DevNode_Registry_Property(
                    dnDevInst,QuestModes[i],0,buf,&len,0);
        if (config==CR_SUCCESS) {
         if (!strcmp(Node,buf)) {
          NodeOperation(Node);
          return;
         }
        }
       }
      }
     }
    } __finally {
     delete ListBuffer;
    }
   }
  }
 }
}

// Обработка узла дерева устройств

void DevNodesScaner::NodeOperation(PTCHAR Node) {
 for (int i=0; i<sizeof(QuestModes)/sizeof(*QuestModes); i++)
{
  TCHAR buf[MAX_DEVICE_ID_LEN];
  ULONG len = sizeof(buf);
  config = CM_Get_DevNode_Registry_Property(
              dnDevInst,QuestModes[i],0,buf,&len,0);
  if (config==CR_SUCCESS) {
   printf("\nNode(%d) = \"%s\"",i,buf);
  }
 }
}

// Выборка состояния узла устройств

ULONG DevNodesScaner::GetStatus() {
 ULONG Status = 0, Problem = 0;
 config = CM_Get_DevNode_Status(&Status,&Problem,dnDevInst,0);
 return Problem;
}

//////////////////////////////////////////////////////////////
// Изменение состояния узла дерева устройств
//////////////////////////////////////////////////////////////

class Changer: public DevNodesScaner {

public:

 Changer() {}
 ~Changer() {}
 virtual void NodeOperation(PTCHAR Node);
};

// Обработка узла дерева устройств

void Changer::NodeOperation(PTCHAR Node) {
 ULONG Problem=GetStatus();
```

Рис. 2



```
 if (config==CR_SUCCESS) {
  if (Problem==CM_PROB_DISABLED) {
   config = CM_Enable_DevNode(dnDevInst,0);
  } else {
   config = CM_Disable_DevNode(dnDevInst,CM_DISABLE_UI_NOT_OK);
  }
  if (config==CR_SUCCESS) {
   config = CM_Reenumerate_DevNode(
              dnDevInst,CM_REENUMERATE_SYNCHRONOUS);
  }
 }
}

#endif //DISABLE_ENABLE_DEVICE_NODE
```

Здесь базовый класс DevNodesScaner предоставляет возможность функцией void DevNodesScaner::LookFor(PTCHAR Node) организовать поиск узла, адресуемого признаком Node. Поиск ведется посредством последовательного рассмотрения значения Node в качестве одного из видов признаков, перечисленных в массиве DevNodesScaner::QuestModes[]. В случае успешного поиска, над узлом выполняется операция, определенная виртуальной функцией void DevNodesScaner::NodeOperation(PTCHAR Node). Имеющаяся в классе DevNodesScaner версия функции NodeOperation отображает значения признаков устройства всех анализируемых видов.

В производном классе Changer функция NodeOperation выполняет инвертирование состояния доступности устройства.

Очевидно, что реализуемая в классе DevNodesScaner схема поиска устройства может приводить к неоднозначному выбору из нескольких однотипных устройств. В подобных ситуациях придется понижать уровень абстракции признака.

Следующая тестовая программа (файл DevNodeCheck.exe) позволяет проверить работоспособность интерфейса доступа к устройствам:

```
//////////////////////////////////////////////////////////
// Диспетчер альтернатив вызова
//////////////////////////////////////////////////////////

#include "DevNodeCheck.h"

void syntax(char *module) {
 printf("\nDevNodeCheck\n\n %s switch node\n",module);
 printf("\n Examples:");
 printf("\n Look Node Params:\n  %s %s",module,"l node");
 printf("\n Change Node State:\n  %s %s",module,"e node");
 getch();
}

int main(int argc, char *argv[]) {
  if (argc==1) {
  syntax(argv[0]);
  return 0;
 }
 DevNodesScaner *Switch = 0;
 switch (argv[1][0]) {

  case 'l': // Просмотр состояния узла
  case 'L':
   Switch = new DevNodesScaner;
   break;

  case 'e': // Изменение состояния узла
  case 'E':
   Switch = new Changer;
   break;

  default:
   syntax(argv[0]);
   return 1;
 }
 if (Switch) {
  Switch->LookFor(argv[2]);
  delete Switch;
  Switch = 0;
 }
 return 0;
}
```

Очевидно, что представленные классы и тестовая программа позволяют легко организовать функциональное замыкание интервала работы прикладной программы, который является критическим по отношению к активности устройства.

Например, пусть необходимо отключить интерфейсы сетевых адаптеров, обеспечивающих связь с другими компьютерами локальной вычислительной сети, на время выполнения криптографических операций.

Предварительно целесообразно проверить однозначность адресации сетевых карт, используя в качестве идентификатора узла устройств имя его класса "Net":

```
>DevNodeCheck L Net
Node(0) =  "rtl8029"
Node(2) = "Realtek RTL8029(AS) PCI Ethernet Adapter"
Node(3) =  "Net"
Node(4) =  "{4D36E972-E325-11CE-BFC1-08002BE10318}"
```

Далее командный файл с вызовом защищаемой программы, например SecretWork, необходимо дополнить командами изменения состояния узла:

```
>DevNodeCheck E Net
>SecretWork
>DevNodeCheck E Net
```

Адресация узла дерева устройств именем его класса безопасна, если управляемое устройство имеется в одном экземпляре. Список имеющихся на конкретном компьютере устройств можно получить различными путями, в частности, посредством входящей в состав Windows 2000 DDK [2] демонстрационной утилиты enable.exe (исходный текст размещен в подкаталоге src\general\enable).

Следует отметить, что приведенная программа, как показывают эксперименты, не разделяет процесс управления устройствами со стандартным интефейсом MMC — выключенное устройство можно включить тем же инструментом. Однако последнее, скорее, следует отнести к достоинствам среды Windows 2000. Вместе с тем, стандартные средства просмотра состояния устройств компьютера остаются работоспособными.

Литература

1. MSDN Library (http://msdn.microsoft.com)
2. Windows 2000 DDK (http://msdn.microsoft.com)
3. Microsoft SDK (http://msdn.microsoft.com)

Восстановление и обновление BIOS

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

В последнее время практически все фирмы-производители системных (материнских) плат используют для хранения BIOS (Basic Input/Output System — Базовая система ввода/вывода) микросхемы EEPROM (Electrically Erasable Programmable Read Only Memory — электрически стираемая программируемая постоянная память). Память EEPROM еще называется Flash Memory или flash-память, а система BIOS, запрограммированная в микросхеме Flash Memory — Flash BIOS. Главное достоинство микросхем Flash Memory заключается в том, что их содержимое можно стирать электрическим сигналом, не применяя ультрафиолетовое или рентгеновское излучение. Нельзя обойти вниманием и основные преимущества микросхем Flash Memory по сравнению с памятью EPROM — малое время доступа и малую длительность процесса стирания информации. Микросхемы Flash Memory можно программировать и перепрограммировать без помощи специального программатора, непосредственно в PC, не вынимая их из панели на материнской плате.

Учитывая, что происходит постоянное совершенствование оборудования и программного обеспечения, в новых моделях материнских плат производителями предусмотрена возможность перепрограммирования пользователем системы BIOS с целью ее обновления. Теперь производители периодически выпускают обновленные версии BIOS и размещают их на своих сайтах в Интернете. Пользователь, скачав нужную ему информацию, может обновить свою систему BIOS. Для установки новой версии BIOS необходимы специальная программа-прошивальщик, например, Flash Memory Writer, которая, как правило, поставляется вместе с материнской платой на компакт-диске, и файл с обновленной системой BIOS. Процедуру обновления BIOS проводят тогда, когда возникают проблемы после установки в компьютере тех или иных новых дополнительных устройств.

Но, к сожалению, никто не застрахован от ситуации, когда в процессе эксплуатации компьютера микросхема Flash BIOS по каким-либо причинам выходит из строя. Тогда полностью нарушается работоспособность персонального компьютера. В этой статье даются рекомендации по восстановлению и обновлению Flash BIOS.

Для того чтобы убедиться в том, что причиной отказа компьютера является неисправность Flash BIOS, нужно взамен микросхемы, которая установлена на материнской плате, временно установить другую, заведомо исправную микросхему Flash BIOS. Взять ее придется с другого компьютера, например, у своего товарища. Причем совершенно не обязательно, чтобы chipset'ы обеих материнских плат были полностью идентичными, однако желательно, чтобы они были от одной и той же фирмы. Если компьютер после установки заведомо исправной микросхемы Flash BIOS загружается нормально — причина отказа действительно заключается в неисправной BIOS. Остается только приобрести подходящую микросхему и записать в нее ту версию BIOS, которая прилагается к материнской плате на компакт-диске. Если же такой компакт-диск отсутствует, то нужную версию BIOS и программу-прошивальщик можно найти в Интернете. Приобретать лучше новую микросхему, потому что число циклов стирания/программирования у flash-памяти хотя и большое, но все же не бесконечное.

Итак, для восстановления работоспособности компьютера с поврежденной микросхемой Flash BIOS потребуется:

– найти заведомо исправную микросхему Flash BIOS, временно снятую с работоспособного компьютера;

– приобрести новую микросхему Flash Memory, в которую надо запрограммировать BIOS;

– снять неисправную микросхему Flash BIOS с панельки на материнской плате (при отключенном питании компьютера);

– вместо неисправной микросхемы Flash BIOS установить заведомо исправную микросхему, снятую с работоспособного компьютера;

– включить питание компьютера и добиться нормальной загрузки операционной системы с установленной заведомо исправной микросхемой Flash BIOS.

Найти на материнской плате микросхему Flash BIOS несложно. Она выполнена в DIP-корпусе, установлена в панельке и имеет 32 вывода. Сверху на корпусе микросхемы, возможно, будет находиться наклейка с обозначением фирмы-изготовителя системы BIOS. Под наклейкой указана маркировка, которая содержит информацию о фирме-изготовителе микросхемы flash-памяти, технических характеристиках и другую служебную информацию. В настоящее время микросхемы flash-памяти выпускаются практически всеми крупными фирмами-производителями микросхем памяти. Наиболее известные из них — Winbond, ATMEL, AMD, SST, MXIC, Intel. Микросхемы flash-памяти различаются емкостью и напряжением программирования. При замене микросхемы надо обращать на эти параметры внимание, т.е. если "родная" flash-память была объемом 128 Кб, то заменять надо ее аналогичной по объему. Емкость применяемой flash-памяти указывается в инструкции к материнской плате (User's manual). Если инструкция отсутствует, определить ее емкость можно без особого труда по последним цифрам маркировки, нанесенной на микросхему. Например:

– Winbond W29C020 — 2 Мбит или 256 Кб;

– AMD Am29F010 — 1 Мбит или 128 Кб.

Вполне возможно, что компьютер после перестановки микросхем загружаться не будет. Это означает, что или не подошла "чужая" микросхема BIOS, или причина не в BIOS'е, а в чем-то другом. Если же загрузка произошла нормально, то выключаем компьютер и вынимаем микросхему Flash BIOS. Затем делаем кольцо из одного витка изоляционного материала (например, полихлорвиниловой трубки диаметром 10 мм) вокруг микросхемы вдоль длины корпуса. Потом при помощи этой "ручки" можно легко извлечь микросхему из панельки. Стык трубки должен быть прочно соединен.

Далее, эту же микросхему неплотно вставляем в панельку, включаем компьютер, загружаемся в "чистый" DOS, выдергиваем за "ручку" микросхему и устанавливаем в панельку новую микросхему Flash Memory, которую надо программировать. Все это надо проделать, не выключая питания компьютера. Программа-про-

шивальщик и файл с BIOS для прошивки flash-памяти обязательно должны находиться в одной директории на жестком диске.

Запускаем программу-прошивальщик, указываем ей имя файла с находящейся в нем системой BIOS и ждем успешного завершения программы. После этого производим перезагрузку компьютера (обязательно через Reset) с целью проверки работоспособности микросхемы и компьютера в целом. Потом выключаем питание компьютера, вынимаем микросхему, снимаем кольцо и теперь уже плотно устанавливаем Flash BIOS на свое постоянное место.

При использовании этого "горячего" метода нужна особая внимательность и аккуратность при извлечении и установке микросхемы в панельку. Желательно это проделать таким образом, чтобы контакт с общим проводом (последним выводом в первом ряду выводов микросхемы) отключался последним, а подключался — первым. Добиться этого можно, наклонив микросхему перед установкой в сторону контакта с общим проводом. Не лишним также будет предварительно немного поупражняться в установке и извлечении микросхемы из разъема при отключенном питании компьютера. При прошивке и обновлении Flash BIOS ни в коем случае нельзя выключать питание компьютера, поэтому все эти операции желательно выполнять при подключенном источнике бесперебойного питания. После окончания прошивки или обновления можно считать содержимое Flash BIOS и сохранить его в файл на жестком диске в отдельной директории. Потом рекомендуется произвести побитное сравнение с эталонным файлом с BIOS для контроля верности произведенной записи.

Где и как искать обновленные версии BIOS

Искать обновленные версии BIOS нужно в Интернете на сайте производителя материнской платы. Пробовать, подойдет ли BIOS от другой платы или не подойдет — лучше не стоит. Адреса сайтов технической поддержки производителей указываются в инструкции к материнской плате. Найдя сайт производителя, нужно зайти в раздел "Support" или подобный ему и скачать соответствующий файл (чаще всего это самораспаковывающийся архив, который может содержать внутри программу-прошивальщик и обновленную версию BIOS). Но перед этим нужно будет точно узнать модель и ревизию материнской платы. Если на сайте производителя нет инструкций о том, где искать маркировку, посмотрите на самый крайний слот вашей материнской платы. На нем должны быть наклейка со штрих-кодом, серийный номер платы (он уже может помочь), а где-то рядом будет указана и модель платы. Иногда ревизия платы пишется на наклейке микросхемы Flash BIOS. Если этой информации недостаточно, попробуйте обратиться в службу технической поддержки производителя, например, написав электронное письмо. После того как модель определена, нужно найти раздел со списками новых BIOS'ов для этой платы. Почитайте список, в котором указано, что добавлено или исправлено в новых версиях. Если ничто из перечисленного вас не касается — BIOS лучше не обновлять.

Вот некоторые адреса сайтов технической поддержки, где можно найти обновления BIOS:

- для материнских плат ASUS:

```
www.asus.com
www.asus.com.tw
www.asus.com.de
```

- для материнских плат Gigabyte:

```
www.gigabyte.com.tw
```

Если Винда "упала"

А.УВАРОВ,
г.Белгород,
E-mail: uvandre@yandex.ru

Парашют выполнил недопустимую операцию и будет закрыт...
Юмор пользователей

Общеизвестно, что ничего вечного не существует, ОС Windows не исключение. Когда эта самая Windows внезапно "падает", для пользователя наступает "черный" день. И начинает он в панике метаться по друзьям и соседям в поисках спасательной дискеты, диска с дистрибутивом, и т.д., и т.п. Завершается эта эпопея переустановкой Окон и "срубанием" под горячую руку ценных наработок. И очень хорошо, если это окажутся записи к "халве", а не квартальный отчет для вашего шефа. А чего только стоит переустановка софта, если под этим у вас подразумевается не только Half-Life и Winamp. Во многих случаях целесообразно сначала попробовать реанимировать Windows — как правило, очень часто это удается. А если все-таки выяснится, что Винда безнадежно "больна", то можно спокойно сохранить все нужное, "грохнуть" ненужное и на досуге переустановить ее [1]. Спешу заверить, что при выполнении всех изложенных ниже рекомендаций вероятность последующей нормальной работы Windows весьма велика, так что лучше потратить вечер на "реанимацию", чем выходные на переустановку.

Мой опыт показывает, что сама по себе Винда падает очень редко, чаще всего причина имеет внешнее происхождение. Как правило, она — в системном реестре, системных файлах и библиотеках (*.dll, *.vxd), либо в вирусной атаке. Сбои в системном реестре могут и не приводить к падению ОС, но иметь не менее неприятные последствия.

Для того чтобы во всеоружии встретить возможные проблемы, потребуется немного времени, тем более, что большинство инструментов для этого уже под рукой. Прежде всего, надо сделать копию реестра с учетом последних изменений в системе. Для этого надо запустить стандартный виндовский редактор реестра Regedit и экспортировать реестр в файл, который затем сохранить в надежном месте. На всякий случай сохраните там же файлы system.dat и user.dat. Пользователям Win98 и WinMe следует также отключить автозагрузку утилиты Scanreg (лучше всего — при помощи утилиты Msconfig.exe.) ScanReg сканирует реестр при каждом запуске Windows, а так как безглючных окон еще не придумали, то она изредка сбоит. Последствием сбоя является восстановление реестра. И очень хорошо, если реестр восстановится по последней резервной копии, очень часто реестр восста-

навливается по состоянию на момент установки, что приводит к отказу 90% софта. А если вы любитель покопаться в реестре, то ее отключение просто необходимо. Часто бывает, что стоит только закрыть Regedit сразу после внесенных изменений, как появляется окно, предлагающее перезагрузить ОС в связи с ошибкой в реестре. В WinMe появилась штатная утилита восстановления системы, однако, очень скоро она создает резервные данные чудовищного размера (до 1...1,5 Гб), посему ее также лучше не использовать. Как она работает, я, к сожалению, не проверял.

Для успешной реанимации нам также не помешает спасательная дискета, антивирус для DOS и дистрибутив, с которого устанавливалась операционка.

От редакции: *описанная методика реанимации полностью применима только для "младших" версий Windows98 (не Second Edition).*

Теперь, имея все, что нужно, дабы не пропасть в тяжелой ситуации, можно перейти к рассмотрению наиболее типичных ситуаций.

Если, включив компьютер, вы обнаружили, что ярлыки поменяли свои места, а свежеустановленный (да и не только) софт отказал — налицо сбой в реестре. Не стоит отчаиваться, перезагрузите компьютер кнопкой Reset в режим MS-DOS (при загрузке удерживайте клавишу F8), а затем выберите в меню строчку "Command prompt only". Пользователям WinMe придется загрузиться с дискеты. Затем надо найти и удалить файлы system.dat и user.dat. А у файлов User.da0 и system.da0 поменять расширение на .dat. В файлах с таким расширением Windows сохраняет копию реестра при последней удачной загрузке, поэтому я и советовал перезагружаться через Reset, иначе велика вероятность подмены этих файлов на текущие. Затем снова перегружаем, и, если надо, импортируем в реестр сделанную нами копию (файл экспорта с расширением .reg). Импорт лучше проводить из редактора Regedit, так как при двойном щелчке на файле экспорта очень часто появляется окошко с надписью "Файл *.reg поврежден или не является файлом реестра", а в Regedit таких проблем не возникает. Если вы уверены, что сбой реестра вызван изменением текущих настроек (т.е. новых ключей не добавилось), то можно обойтись просто импортом файла копии реестра. В любом случае восстановить реестр файлами .da0 мы всегда успеем.

Намного серьезнее ситуация, когда файл реестра поврежден, либо ошибка весьма серьезна, и сразу же после загрузки Scanreg (еще раз повторю, что его лучше отключить) предлагает перезагрузиться и восстановить реестр. Если реестр восстановился, то следует импортировать в него копию, чтобы вернуть все текущие настройки. Иначе придется восстанавливать реестр вручную. То же следует предпринять, если "Винда" вдруг перестает грузиться без видимых причин (обычно просто зависает на каком-то этапе загрузки). Вначале следует попытаться восстановить реестр файлами .da0, однако причина сбоя могла появиться довольно давно, и через некоторое время Windows "упадет". Намного надежней удалить как .dat- так и .da0-файлы, заменив их на сохраненные нами при установке (или позже) system.dat и user.dat. Если все пройдет успешно, потребуется импортировать файл экспорта для восстановления работоспособности софта.

Не стоит также жалеть времени на проверку системы антивирусом, лучше это сделать до восстановления, чтобы возможно "затаившийся" вирус не свел на нет все наши старания. Недаром я выше писал о необходимости держать на своем компьютере антивирус для DOS. Загрузившись со спасательной дискеты, в первую очередь запустите его. Я долгое время пользуюсь DrWeb'ом — звезд с неба не хватает, но и не подводит.

Отдельным вопросом является восстановление реестра после некорректной работы утилиты Scanreg. В этом случае следует отключить эту утилиту, восстановить реестр импортом файла копии реестра и раза два - три перезагрузиться, после чего утилиту Scanreg можно снова включить.

Кроме сбоев в реестре, причиной "падения" Windows может быть сбой в библиотеке .dll либо драйвере виртуального устройства .vxd (сюда можно отнести еще ряд типов файлов, но эти сбоят чаще всего). Тогда Windows "радует" нас "синим окном смерти" либо сообщением об ошибке, едва прорисовав поверхность рабочего стола. В этом случае запишите на листочек название сбоящего компонента. Затем загрузитесь в режим MS-DOS и удалите этот компонент. При повторной загрузке появится сообщение типа "Не найден компонент, необходимый для запуска Windows. Для продолжения нажмите любую клавишу" — как правило, загрузиться удается в 90% случаев. После этого запускаем программу восстановления системных файлов (меню "Сервис" в "Сведениях о системе") и вставляем диск с дистрибутивом. Если сбоящий файл принадлежал какой-то программе, то никаких проблем с запуском Windows не будет, однако программу, которой ранее принадлежал файл, придется переустановить. Пользователям Win95 придется поискать удаленный файл в cab-архивах дистрибутива, сходив к товарищу, на чьей машине установлен WinRAR.

Самый тяжелый случай — когда после удаления сбоящего компонента Windows "умирает" вообще. Это говорит о том, что удаленный файл скорее всего входил в состав ядра ОС. Но не стоит отчаиваться (восстановите его вручную, извлечь его из cab-архива следует где-нибудь у товарища WinRAR-ом). После такого восстановления все-таки следует, завершив текущие дела, переустановить ОС.

Еще одна, самая "родная" причина отказа Windows — ошибки и неисправные кластеры на жестком диске. Причин тут может быть несколько — либо старость винчестера (как ни прискорбно, но придется смириться, и нечего валить на неизвестный науке вирус), либо профилактика винчестера проводилась так давно, что пользователь уже успел забыть это слово; также не исключена и вирусная атака. Если винчестер читается, то запускаем Norton Disk Doctor под DOS и ждем результата. Как правило, данные удается спасти в 90% случаев. Однако реанимация Windows — вопрос спорный. В любом случае выясните причину "осыпания" винчестера и проведите соответствующий "курс лечения" (если сами поставить диагноз затрудняетесь, соберите "консилиум" из друзей и знакомых — винт того стоит).

Спасение данных с "умершего" винчестера выходит за пределы данной статьи, и пользователю следует обратиться к иным источникам, в частности, к [2].

В заключение хочется отметить, что многие крайне неприятные глюки вызываются резидентными модулями антивирусов. Я бы не писал об этом, но мне знакомы несколько пользова-

С.РЮМИК,
г.Чернигов.

Простейший заменитель Sega-джойстика

Как известно, существуют игры, рассчитанные на трех- и шестикнопочные джойстики для игровой приставки "Sega Mega Drive-II". В случае отсутствия трехкнопочного джойстика можно воспользоваться схемой замещения, содержащей микросхему К561ЛС2 и транзистор КТ315Б, а в случае отсутствия шестикнопочного — схемой замещения на 9 микросхемах серии К561 [1].

Вниманию читателей предлагается еще одно решение, разработанное по принципу "проще некуда". Основой заменителя трехкнопочного джойстика (**рис.1**) является микросхема DD1 К555КП11. Она содержит четыре двухвходовых мультиплексора с общим управлением. К ее входам подключаются контакты кнопок джойстика "LEFT", "RIGHT", "A", "B", "C", "START". Управление производится импульсами отрицательной полярности по линии "SYN" от игровой приставки. С выходов микросхемы сигналы поступают на 9-контактную розетку XS1, внешний вид которой с лицевой стороны показан на **рис.2**.

Нагрузочные резисторы для кнопок SB1...SB6, как правило, не требуются. Однако при неустойчивой работе устройства их можно установить между цепью +5 В и соответствующими контактами кнопок. Номинал резисторов — 10...20 кОм. Аналогичная ситуация с кнопками SB7, SB8, для которых нагрузочные резисторы могут потребоваться в случае нестандартной конфигурации входных цепей игровой приставки.

Особая ценность предлагаемого джойстика состоит в том, что он практически "вечный". Действительно, средняя наработка на отказ микросхемы К555КП11 составляет не менее 50000 часов. В случае выхода ее из строя, ремонт потребует минимальных финансовых вложений. Возможные замены микросхемы DD1 — КР1533КП11А, К555КП16, КР1533КП16.

Конструктивно микросхему располагают внутри корпуса джойстика с внешней стороны печатной платы. Неисправный чип-"капельку" джойстика следует удалить. Выводы микросхемы DD1 должны быть предварительно отформованы в разные стороны для придания конструкции максимально плоской формы.

Монтаж ведется тонким проводом прямо на печатные дорожки, подводящие сигналы к "гребенке" кнопочных контактов. Для переброса жгута проводов с внутренней стороны платы на внешнюю, в последней сверлят отверстие диаметром 5...8 мм. Джойстик соединяют с розеткой XS1 шнуром длиной 1,5 м.

Литература

1. Рюмик С. Восстановление работоспособности Sega-джойстика. — Радіоаматор, 2000, №2, С.42...44.

Рис. 1

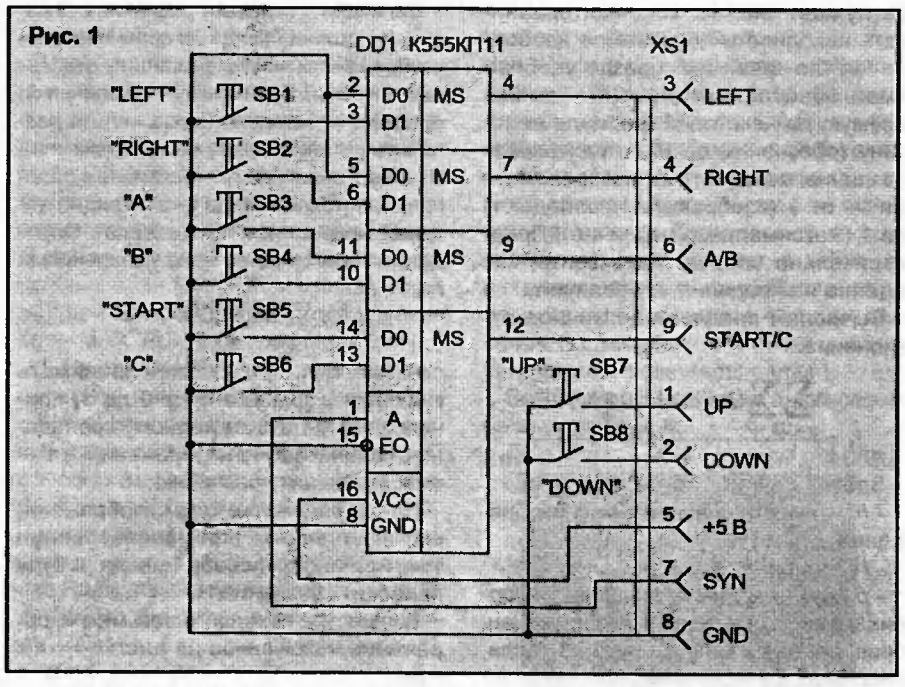
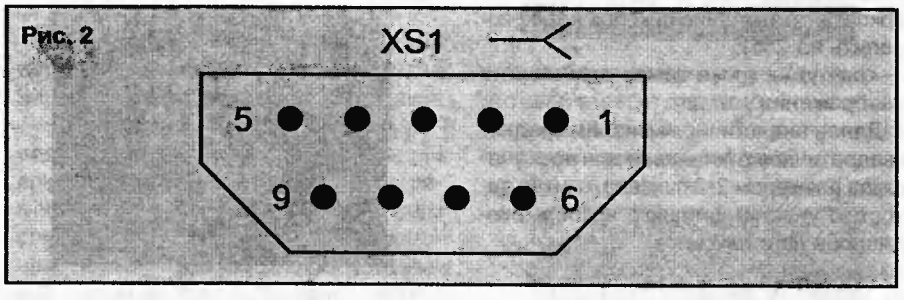


Рис. 2



телей, переустановивших по этой причине Windows, причем с нулевым результатом, т.к. после переустановки антивирусный софт старательно восстанавливался. Особенно известен своими проказами модуль spider.exe из DrWeb. У меня с ним компьютер глухо вис на стадии завершения работы, предварительно зачем-то обратившись к дисководу. В любом случае почаще заглядывайте в пункт автозагрузки и отключайте все лишнее — избежите ряда потенциальных проблем.

Литература

1. Уваров А.С. Установи Windows сам. — Радиомир. Ваш компьютер, 2001, N12, С.30.

2. Поляк-Брагинский А. Как спасти информацию на жестком диске. — Мир ПК, 2001, N9, С.88.

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

Вывод черно-белых изображений с градациями яркости

Как вы, наверно, уже догадались :), речь пойдет о различных способах вывода на экран "ZX-Spectrum" черно-белых изображений с градациями яркости. Для определенности положим, что размер изображения — 256*192, что соответствует разрешению экрана "ZX-Spectrum", а количество градаций яркости — 256.

Понятно, что о реальных 256 градациях яркости на "ZX-Spectrum" можно только мечтать, так что вывести мы сможем лишь что-то похожее на исходное изображение.

Как определить, какой способ вывода лучше? Можно, конечно, сравнивать исходное и получаемое изображения "на глаз". Но гораздо удобнее иметь объективный критерий — вычисляемую по некоторой формуле величину (обозначим ее R), отражающую различие между этими изображениями — от 0 (изображения совпадают) до 1 (максимальное различие). Предварительно следует рассказать, как именно я определил эту величину.

Вычислим среднеквадратичное отклонение:

$$r_1 = \frac{\sum_{y=0}^{h-1}\sum_{x=0}^{l-1}(p_1(x,y)-p_2(x,y))^2}{l\cdot h}.$$

Здесь:

- h — высота сравниваемых изображений;
- l — ширина;
- $p_1(x,y)$ — яркость пиксела с координатами (x,y) в первом изображении, приведенная к диапазону 0...1 (если, например, в изображении 256 градаций яркости, то значение пиксела надо разделить на 255);
- $p_2(x,y)$ — то же самое для второго изображения.

Далее аналогично вычислим среднеквадратичное отклонение для всех участков размером 2*2 пиксела, считая яркостью участка среднюю яркость входящих в него пикселов:

$$r_2 = \frac{\sum_{y=0}^{h-2}\sum_{x=0}^{l-2}(m_1(x,y)-m_2(x,y))^2}{(l-1)\cdot(h-1)},$$

где:

$$m_1(x,y) = \frac{\sum_{j=0}^{1}\sum_{i=0}^{1}p_1(x+i,y+j)}{4},$$

$$m_2(x,y) = \frac{\sum_{j=0}^{1}\sum_{i=0}^{1}p_2(x+i,y+j)}{4}.$$

И для участков 4*4 пиксела:

$$r_3 = \frac{\sum_{y=0}^{h-4}\sum_{x=0}^{l-4}(n_1(x,y)-n_2(x,y))^2}{(l-3)\cdot(y-3)},$$

где:

$$n_1(x,y) = \frac{\sum_{j=0}^{3}\sum_{i=0}^{3}p_1(x+i),y+j)}{16},$$

$$n_2(x,y) = \frac{\sum_{j=0}^{3}\sum_{i=0}^{3}(p_2(x+i,y+j)}{16}.$$

Величина r_1 отражает различие с близкого расстояния (когда на сравниваемых изображениях можно различить отдельные пикселы). Величина r_2 — различие со среднего расстояния (когда нельзя различить детали меньше чем 2*2 пиксела). Наконец, величина r_3 — различие с дальнего расстояния (когда уже не видно деталей меньше чем 4*4 пиксела). Определим R как среднее между значениями r_1, r_2 и r_3:

$$R = \frac{r_1 + r_2 + r_3}{3}.$$

R (как и r_1, r_2, r_3) может принимать значения в диапазоне от 0 до 1, причем 0 соответствует полному совпадению сравниваемых изображений, а 1 — максимальному различию.

Далее рядом с каждым изображением, полученным при использовании того или иного способа вывода, я буду приводить значение R.

В качестве исходного возьмем изображение, показанное на **рис.1**.

Рис. 1

Итак, первый способ вывода. Все пикселы, имеющие яркость меньше 128, выводим черными, а яркость 128 и больше — белыми (**рис.2**).

Следующий способ — с использованием так называемых "чанков". В каждом квадрате размером 4*4 пиксела выводим одну из 17 текстур с наиболее близким уровнем яркости (в первой текстуре все 16 пикселов черные, во второй — 15 черных и 1 белый, ..., в последней — все 16 пикселов белые). Таким образом, получается 17 псевдоградаций яркости (**рис.3**).

Рис. 2 *R = 0,0906*

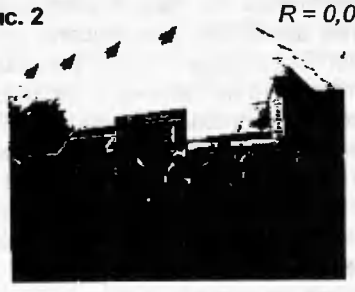
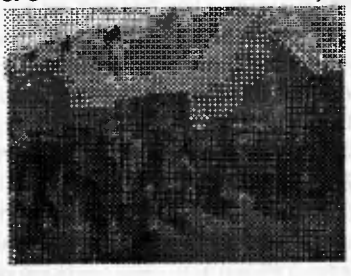


Рис. 3 *R = 0,0708*



Рассматривать такое изображение (как и изображения, получаемые с использованием описанных далее способов) желательно не с близкого расстояния, когда оно воспринимается как мешанина пикселов, а издалека, когда яркость соседних пикселов усредняется. Если нет возможности отойти от монитора на достаточное расстояние, или с такого расстояния изображение кажется очень маленьким, то можно добиться того же эффекта, "размыв" изображение — для этого достаточно установить перед экраном фильтр из полупрозрачной полиэтиленовой пленки.

Еще один способ вывода. Для каждого пиксела (а не для участка 4*4, как в предыдущем способе) определяем наиболее близкую по яркости текстуру и выводим на экран пиксел из этой текстуры. Координаты пиксела, который берется из текстуры, определяются как остатки от деления координат пиксела в изображении на размер текстуры.

На **рис.4а** приведено изображение, получающееся при использовании 17 текстур 4*4, а на **рис.4б** — при использовании 256 текстур 16*16 (вообще-то их 257, а не 256, но, так как в исходном изображении лишь 256 градаций яркости, одна из текстур не используется).

Это было, так сказать, вступление :). Рассматривались уже известные спосо-

Рис. 4а *R = 0,0683*

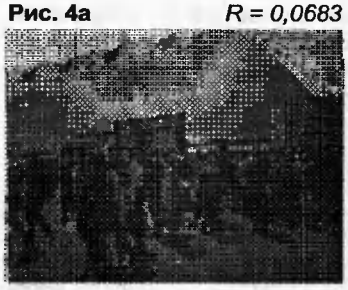
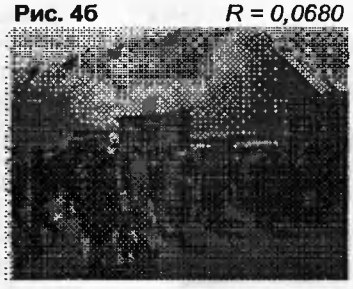


Рис. 4б *R = 0,0680*



бы вывода, которые, как вы могли заметить, не отличаются достаточным качеством. А вот теперь речь пойдет о том, как "выжать" из "ZX-Spectrum" все, на что он способен. Я расскажу о способах, позволяющих на порядок повысить качество выводимого изображения!

Смотрите, на "ZX-Spectrum" нам доступны 8 цветов с двумя градациями яркости (bright=0 и bright=1). Так как черный цвет при bright=1 остается черным, всего получается не 16 цветов, а 15. При черно-белом режиме отображения эти 15 цветов превращаются в 15 градаций яркости. За счет их использования и можно повысить качество изображения!

Естественно, для этого нужно, чтобы "ZX-Spectrum" был подключен к черно-белому монитору или телевизору (или к цветному, но с возможностью переключения в черно-белый режим). Но даже если в вашем случае это не так, все равно советую дочитать статью до конца — вдруг что-то окажется полезным.

Сразу же отмечу, что основанный на этом эффекте способ вывода черно-белых изображений с градациями яркости может быть использован и на отличных от "ZX-Spectrum" компьютерах, на которых изображение может быть цветным, но нет градаций серого.

Кстати, как мне кажется (на практике не проверял), чтобы получить черно-белое изображение на цветном мониторе, достаточно соорудить простейший переходник: три сигнала R, G, B, идущие от компьютера, преобразуем в один (черно-белый) и подаем его на входы R, G, B монитора, вот и все. При этом из деталей понадобятся лишь несколько резисторов.

Итак, как выводить изображение? Очевидно, сначала надо определить яркость каждой из имеющихся 15 градаций — число в диапазоне 0...255. Рассказ о том, как это сделать, получился довольно объемным, и я решил вынести его в Приложение 1. А сейчас будем считать, что яркости вычислены, и мы получили значения a0...a7 для цветов 0...7 при bright=0, и значения b0...b7 для тех же цветов при bright=1.

Так как атрибуты задаются для знакоместа (8*8 пикселов), изображение будем выводить познакоместно. Рассмотрим процесс вывода одного знакоместа.

Пусть MIN_PIX и MAX_PIX — соответственно минимальный и максимальный уровни яркости в текущем знакоместе исходного изображения. Пусть MIN и MAX — соответственно яркости paper и ink (при выбранном значении bright) в текущем знакоместе выводимого изображения. Зная MIN_PIX и MAX_PIX, надо определить атрибуты знакоместа (ink, paper и bright) так, чтобы выполнялись условия — MIN ≤ MIN_PIX, MAX ≥ MAX_PIX. Очевидно, в большинстве случаев это можно сделать не единственным способом. Тогда из всех возможных пар (MIN, MAX) выбираем такую, чтобы разность между MAX и MIN была наименьшей.

Чтобы не перебирать все 128 возможных сочетаний ink, paper и bright, удобно поступить следующим образом. Сначала проверяем, выполняется ли условие MAX_PIX > a(7)? Если да, то bright не может быть равно 0, т.е. bright=1. Среди значений b0...b7 отыскиваем максимальное b(i), при котором выполняется условие b(i) ≤ MIN_PIX, и минимальное b(j), при котором выполняется условие b(j) ≥ MAX_PIX. Тогда paper=i, а ink=j.

Если же MAX_PIX ≤ a(7), придется рассмотреть два варианта — bright=0 и bright=1. Находим b(i) и b(j), как было описано выше. Затем среди значений a0...a7 отыскиваем максимальное a(k), при котором выполняется условие a(k) ≤ MIN_PIX, и минимальное a(l), при котором выполняется условие a(l) ≥ MAX_PIX. Теперь надо выбрать из двух пар (b(i), b(j)) и (a(k), a(l)) такую, где разница между значениями меньше. Если это первая пара, то bright=1, ink=j, paper=i. Если вторая пара — bright=0, ink=l, paper=k.

После того как атрибуты знакоместа определены, остается узнать, какие пикселы в нем будут цвета ink, а какие — paper. В предыдущем способе вывода мы получали 257 псевдоградаций яркости между черным и белым с помощью 257 текстур 16*16. Ну а в этом способе мы можем получить с помощью тех же текстур 257 псевдоградаций между MIN и MAX. Для каждого пиксела определяем наиболее близкую по яркости текстуру и выводим на экран пиксел из этой текстуры.

Вот что получается при таком способе вывода (**рис.5**). Не правда ли, качество изображения значительно повысилось?

Процедура, выводящая изображение этим способом, приведена в Приложении 2. Там же вы найдете процедуры для способов, которые описаны далее.

Рис. 5 *R = 0,0106*

Для дальнейшего повышения качества изображения можно использовать так называемый multicolor-эффект. Сущность его в следующем — если изменить атрибут знакоместа, когда луч уже прорисовал на экране его часть, то оставшаяся часть знакоместа будет прорисована уже с новым значением атрибута. Так можно увеличить атрибутное разрешение по вертикали.

Если надо увеличить разрешение вдвое (т.е. задать свой атрибут для каждой половины знакоместа — области размером 8*4 пиксела), удобнее не перезаписывать значения атрибутов при прорисовке изображения, а сделать так — построить верхние половины знакомест со своими атрибутами на одном экране, нижние половины со своими атрибутами — на другом, а затем переключать экраны каждый раз, когда прорисуются очередные 4 строки пикселов (т.е. верхняя или нижняя половина строки знакомест).

Вот что получается при таком способе вывода (**рис 6**).

Рис. 6 *R = 0,0090*

Очевидно, максимум возможного — это свой атрибут для каждой линии знакоместа (8 пикселов). Вот получаемое при этом изображение (**рис.7**).

Рис. 7 *R = 0,0071*

При практической реализации максимального атрибутного разрешения, однако, возникает проблема. Дело в том, что за 224 такта (время прорисовки од-

ной строки) нужно обновить 32 байта атрибутов строки, а сделать это не так-то просто: обычные способы пересылки данных требуют гораздо больше времени. В процедуре, приведенной в Приложении 2, я решил эту проблему так: выводится не все изображение, а окно шириной 11 знакомест, которое можно перемещать вправо-влево. Вы можете попробовать увеличить ширину выводимой области изображения, используя способы быстрой пересылки данных — например, описанные в [5].

Есть еще один способ улучшения качества. Можно сформировать такие два изображения, чтобы среднее между ними было наиболее похоже на оригинал, а затем быстро чередовать их — при этом зритель как раз и увидит среднее изображение.

Пусть атрибуты обоих изображений одинаковы. Если раньше яркость пиксела в знакоместе могла быть одной из двух — MIN или MAX, то при быстром чередовании двух изображений она может быть уже одной из трех — MIN, MAX или (MIN+MAX)/2 (когда на одном изображении яркость пиксела — MIN, а на другом — MAX). За счет этого и повышается качество. Если раньше мы получали 257 псевдоградаций яркости от MIN до MAX с помощью 257 различных текстур 16*16, то теперь получим 257 псевдоградаций от MIN до (MIN+MAX)/2 и еще 257 — от (MIN+MAX)/2 до MAX, а всего, таким образом, будет 513 псевдоградаций от MIN до MAX.

При формировании таких двух изображений их атрибуты (они одинаковы) определяются так же, как и раньше. Для каждого пиксела определяем, какая из 513 псевдоградаций (от 0 до 512) наиболее близка к нему по яркости. Если номер псевдоградации меньше 256, то на первом изображении соответствующий пиксел выключаем, а на втором — берем из текстуры с номером, равным номеру псевдоградации. Если же номер псевдоградации больше или равен 256, то на первом изображении пиксел включаем, а на втором — берем из текстуры с номером, равным номеру псевдоградации, уменьшенному на 256.

Используя этот способ для вывода исходного изображения, получим такие два изображения (**рис.8а, 8б**).

Рис. 8а

Рис. 8б

Рис. 9 *R = 0,0026*

При их быстром чередовании мы увидим среднее изображение (**рис.9**).

Однако, рассматривая его, вы будете разочарованы из-за довольно заметного мерцания с частотой 25 Гц. Можно ли его уменьшить, и если да, то как? Попробую объяснить на примере. Пусть надо получить в некоторой области экрана серый цвет за счет быстрой смены белого и черного. Можно сделать так — в одном кадре показывать всю эту область белой, а в другом — черной (**рис.10а**).

Рис. 10а

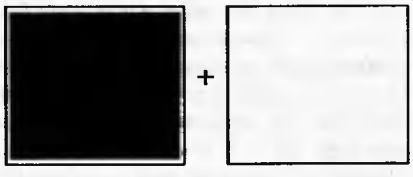
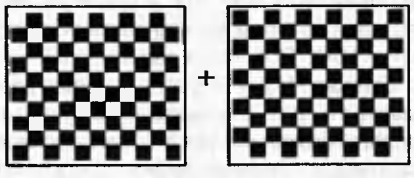


Рис. 10b



Мерцание при этом будет очень заметно. А можно в одном кадре выводить в этой области "шахматную" текстуру, а в другом кадре — такую же текстуру, но в которой вместо белых пикселов — черные, и наоборот (**рис.10б**). Тогда мерцание будет гораздо меньше, а издалека — вообще не будет заметно.

Почему так получается? При рассматривании изображения издалека яркость соседних пикселов усредняется, а так как во втором случае соседние пикселы мерцают в противофазе, то их средняя яркость и в одном, и в другом кадре будет одинаковой — в результате, изображение не будет мерцать.

Иначе говоря, средняя яркость достаточно малых участков на первом и втором изображениях должна быть примерно одинаковой, а достигнуть этого можно, если соседние мерцающие пикселы будут в противофазе.

В нашем случае (рис.8) изображения сильно отличаются друг от друга, и поэтому мерцание при их смене очень заметно. Чтобы изображения стали более похожими друг на друга, сделаем следующее. Просматривая пикселы (для определенности, слева направо и сверху вниз), определим мерцающие (т.е. различающиеся на первом и втором изображениях). Первый такой пиксел сделаем темным на первом изображении и светлым на втором, следующий наоборот — светлым на первом изображении и темным на втором (т.е. он будет в противофазе с предыдущим), и так далее.

Во что превращаются приведенные на рис.8 изображения после такой обработки, показано на **рис.11а, 11б**.

Рис. 11а

Рис. 11б

Сразу видно, что теперь они гораздо больше похожи друг на друга. Мерцание при их быстрой смене становится значительно меньше, а издалека — вообще не заметно. Результирующее изображение будет, разумеется, тем же самым (рис.9) — ведь от перемены мест слагаемых сумма не меняется.

Кстати, вот совет, как не видеть мерцания даже с близкого расстояния. Возьмите лист бумаги, проткните в нем отверстие и смотрите на экран сквозь это отверстие. Количество света, попадающего в глаз, будет меньше, и мерцание сильно уменьшится.

Рис. 12 *R = 0,0022*

Рис. 13 *R = 0,0017*

Быстрое чередование двух изображений можно совместить с multicolor, что еще повысит качество. На **рис.12** показано, что получится, когда для каждой половины знакоместа задается свой атрибут, на **рис.13** — когда свой атрибут задается для каждой линии знакоместа.

Описанные выше способы иллюстрировались на примере одного исходного изображения. На **рис.14 а...з** приведены примеры для еще нескольких изображений (слева — исходное, справа — получающееся при выводе). Использовался способ с быстрым чередованием двух изображений и multicolor'ом, атрибуты задавались для каждой линии знакоместа (кстати, сканирование фотографий, послуживших исходными изображениями, выполнил по моей просьбе Алексей Летаев. Спасибо!)

Рис. 14а *R = 0,0020* Рис. 14б

Рис. 14в *R = 0,0021* Рис. 14г

Рис. 14д *R = 0,0029* Рис. 14е

Рис. 14ж *R = 0,0023* Рис. 14з

Чтобы увидеть на изображении еще больше деталей, можно, во-первых, использовать вывод с увеличением, и, во-вторых, повысить контрастность выводимого участка изображения.

При выводе с увеличением мы не сможем увидеть сразу всю картинку, но за счет этого на видимом фрагменте сможем различить больше деталей. Вот пример— на **рис.15а** показана центральная часть рис.5, для удобства сравнения увеличенная вдвое, а на **рис.15б** — то, что получится, если увеличивать вдвое центральную часть непосредственно в процессе вывода.

Приведенные в Приложении 2 процедуры вывода могут выводить указанную область изображения с увеличением в 2, 4, 8... раз. Думаю, вам не составит труда догадаться, как увеличивать изображение в количество раз, не являющееся степенью двойки.

Теперь о повышении контрастности. Если в выводимом изображении (или в выводимой части изображения) самая темная точка светлее чем 0 и/или самая светлая точка темнее чем 255, то можно "растянуть" диапазон яркости до 0...255 по следующей формуле:

$$A' = 255\frac{A - A_{min}}{A_{max} - A_{min}},$$

где:

- A' — новая яркость;
- A — исходная яркость;
- A_{min} — яркость самой темной точки;
- A_{max} — яркость самой светлой точки.

После такого преобразования мы увидим темные участки изображения темнее, чем надо, а светлые — светлее, но при этом можно будет различить больше деталей. На **рис.16** вы можете увидеть изображение, полученное, как и рис.15б, но с повышением контрастности.

Рис. 15а Рис. 15б

Рис. 16

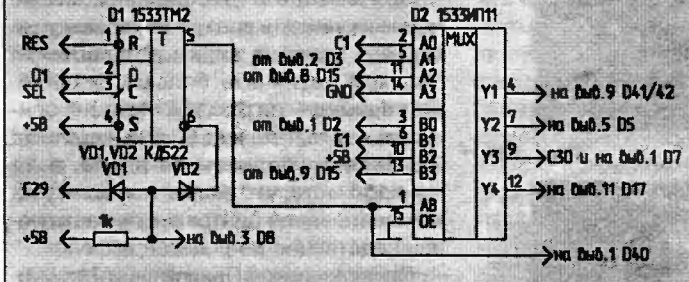
(Продолжение следует)

По страницам электронных изданий

Дополнительный графический режим — 512x192

(c)V.M.G. KHARKOV,
Ukraine.

Предлагаемый режим позволяет расположить на экране 512 точек в строке или 85 символов матрицей 6x8. Это дает возможность разместить значительно больше информации на экране монитора и сделать более удобной работу с текстом, монохромной графикой, терминальными программами и т.д. На рисунке приведена схема доработки.



Сигнал	Контакт	Примечания
A12	1	
A13	2	
A14	3	
A3	4	
DIRQ	5	Сигнал IORQ, пропущенный через контроллер FDD
A15	6	
-SEL	9	Требуемый сигнал, который нужно инвертировать

Также на м/с D38 сверху необходимо напаять D38' (1533ИР23), всеми выходами соединенную с D38, а ее входы соединить с соответствующими выходами D37. На выводы 1 микросхем D38 и D38' подается сигнал С3 и сигнал с вывода 2 м/с D3; на выводы 11...9, 12, 15, 16, 19 микросхемы D40 необходимо через резисторы номиналом 1 кОм подать сигнал GND.

Доработка включается установкой бита 1 порта #EFF7 и выключается по RESET сбросом этого же бита. После доработки в памяти компьютера образуется две области: #4000...#57FF — четные байты на экране, и #6000...#77FF — нечетные байты на экране. "Второй" экран адресуется аналогично (#C000...#C800 и #E000...#E800).

Расположение байтов на экране можно представить следующим образом:

4000	6000	4001	6001	4002	6002	...	401F	601F

В двух словах как работает схема. При включении расширенного экрана, выходные регистры сдвига начинают тактироваться с удвоенной частотой. При выполнении цикла (чтение байта изображения/чтение байта атрибута) вместо байта атрибута читается байт изображения из второй половины экрана с адреса #6000. Он читается в дополнительный регистр, который подключается параллельно основному регистру для байта данных изображения (в описании это D38 и D38'). Регистр хранения байта атрибута переводится в Z-состояние, а чтобы он не выдавал вместо цветов RND-последовательность, его выходы через резисторы "подпираются" к земле и к +5 В.

Нумерация элементов на схеме дана относительно схемы "Пентагона" (by V.M.G.) и может не совпадать с вашей схемой (адресные сигналы пронумерованы с A0 по A15).

Перед подачей на соответствующий вывод микросхем нового сигнала, необходимо "оторвать" от вывода старый.

Сигнал SEL — это проинвертированный сигнал с вывода 9 дешифратора 1533 ИД7, в таблице приведена информация о подключении дешифратора.

КОМПЬЮТЕРНЫЕ ХРОНИКИ

FOREVER 2002

В середине марта этого года вот уже в третий раз прошло demoparty Forever (Trencin, Словакия). Состоялись конкурсы на платформе "ZX-Spectrum" в номинациях Graphics, Music, 1k Intro и Demo. Итак, кто же победил на этот раз?

Graphics (всего 20 работ): 1 — "Springtime Feelings Forever" by Diver/4D (394), 2 — "In Time" by Starily Stall (344), 3 — "Walter" by LCD (330).

Music (всего 21 работа): 1 — "Exonique" by Gasman/Hooy Program (266), 2 — "Improvisation #8" by Ivan Roshin (263) :-), 3 — "Theme from Navstevnici" by Factor6/PHT+K3L (256).

1k Intro (всего 3 работы): 1 — "Intro that goes ping" by Gasman/Hooy Program (285), 2 — "Galatrax" by SerzhSoft (268), 3 — "Mahnung" by Key-Jee/TBL (256).

Demo (всего 7 работ): 1 — "AlterEgo" by Hooy Program (295), 2 — "Reject" by Factor6/PHT+K3L (256), 3 — "Laya" by 3SC (240).

Официальный сайт demoparty — http://4everparty.host.sk/

Хроники вел И.Рощин.

ВОЗВРАЩАЯСЬ К НАПЕЧАТАННОМУ
("РМ. ВК", №3/2002, С.39)

При подготовке к печати статьи И.Рощина **"Программирование задержек на ассемблере Z80"** были допущены ошибки.

1.В описании процедуры WAIT_LONG (с.41, первая колонка) должно быть:

"Ниже приведен текст еще одной процедуры, для реализации более продолжительных задержек (от 469 до 2^{32}-1 тактов)".

2. В листинге процедуры WAIT_LONG (с.41, вторая колонка) соответствующие строки должны иметь вид:

```
WAIT_L1     ADD   HL,BC
            INC   L             ;Проверяем: L=0?
            DEC   L             ;(не изменяя флаг C).
```

Редакция приносит извинения автору и читателям.

Полузэки

К.КЛИЩЕНКО,
г.Минск.

Если сильно захотеть, можно в космос полететь.
Народная мудрость

Одна половина у мистера Крококота была котом, другая — крокодилом.
"Дональд Биссет"

Давайте сразу договоримся, что раз игра вторична, то говорить об этом в статье больше 34245456 раз мы не будем. Совершив чудеса эквилибристики, я упомяну об этом во много раз меньше. Правильнее было бы сказать, что вторична не сама игра, а две ее основные составляющие — Heroes of Might & Magic и Magic: The Gathering.

О чем это я? Ах да, о новой русской (по счастью, не ново-русской) игре "Демиурги". Чего хотели добиться авторы, понятно — сочетания исследований и набегов на залежи ресурсов и разнообразных, изящных и практически неограниченных варинтов тактических решений в бою. А вот что получилось, сейчас и разберем.

Можно написать сколько угодно раз, что подобной графики не было ни в одной пошаговой стратегии. Я не буду этого отрицать. Гораздо хуже то, что если судить по основной массе отзывов, это чуть ли не единственное достоинство игры. К счастью, на мой взгляд, это неправда.

Игра вполне захватывает. Захват происходит в два этапа (кто бы догадался?). Обилие ресурсов и "оригинальность" передвижений составляют основу действа на глобальной карте. Человек, сочувствующий моей псевдоубогости по этому вопросу, поведай, зачем здесь столько ресурсов? Понятно, что команде авторов повезло с художниками и специалистами по графике. Но зачем демонстрировать свой талант таким количеством всяких источников-шахт. К тому же, пространства, не покрытые монстрами, являются вам подобно участкам кожи меж лат средневекового немца (скорее всего — тевтона, идущего громить Великое Княжество Литовское). А система "зарядки" героя "могуществом, в сраженьи твердым" даст фору любому прожженному бюрократу. Вполне преодолимо, скажете вы. Тогда я (хотя, к сожалению, и не автор этой игры) подам вам еще одну свинью, и не надейтесь, что во рту у нее будет яблоко.

Благими намерениями куда вымощена дорога? Увы, не к волшебнику Гудвину. Туда же попадут и намерения программистов. Они очень хотели улучшить несовершенную систему любой пошаговой стратегии — ее пошаговость. Синхронизация ходов для уничтожения всяких приоритетов — это хорошо. Но, к сожалению, в этой игре найдено не лучшее решение. Перемещения героя ограничиваются не только его умением бороздить просторы, но и его действиями. Так, за один ход положено всего одно осмысленное действие, и неважно, что вам до цели оставался всего один лишь шаг, а на полную зарядку героя уедет по меньшей мере два-три хода. Точнее, игре неважно, а вот по вас это порядочно бьет. И несомненно комкает впечатление от игры.

Ну да шут (и король) с ними, главное в этой игре — сражения. В борьбе за власть никто не будет петь песни и строить пирамиды (хм-хе) для победы над врагом ("прозрачный" намек на "Цивилизацию"). Все будут творить заклинания: глобальные — для воздействия на мир, боевые — для приведения основного механизма игры в движение. И как бы ни старались авторы подчеркнуть усилия, затраченные на попытки разнообразить действо на глобальной карте, главной частью игры несомненно являются бои — честное соперничество один на один в попытке доказать сопернику и себе состоятельность притязаний на этот мир. Ради этих непродолжительных моментов растраты заклинаний и ведется сбор ресурсов и этих самых заклинаний. Конечно, ни разнообразие, ни количество карт не может сравниться с Magic: The Gathering, но их достаточно, для того чтобы увлечь не на один час.

Так как авторам не хотелось превращать свою игру во второе "Собрание", то, естественно, многое будет определяться вашими умелыми действиями на "большой" карте. Так, для создания сильной "колоды" вам придется искать магазины по продаже заклинаний. Это привносит в бой новые моменты. Невозможно будет использовать заранее просчитанные и опробованные ходы для построения смертоносного сочетания заклинаний. Вы будете строить из того, что есть. Несомненно это является плюсом игры.

Минусом боя, но, вполне вероятно, плюсом игры в целом является ограничение количества "карт в колоде" тридцатью. Иначе слишком продолжительные сражения добавили бы скуки. Описывать бой в подобной игре вообще дело неблагодарное (равно как и расписывать настольные РПГ). Поэтому я не стану этим заниматься (объяснение "из-за лени" — куда ближе, но бьет по самолюбию).

А впрочем, попротиворечу сам себе и набросаю основные черты. Нужно убить соперника с помощью магии. После выбора нужных (или кажущихся таковыми) заклинаний вы идете в бой. Но "злобные" авторы не дают воспользоваться любым из заклинаний. Как и в любой карточной игре, у вас будет возможность использовать только те заклинания, которые у вас "на руках". Каждый ход к вам "на руку" приходит заклинание "из колоды". У этого простого факта есть несколько особенностей — их проще почерпнуть из игры.

Вы бросаете взгляд "на руку". И что же? Купленные вами за бешенные...мммм.... ресурсы заклинания недоступны (а иногда и просто "незаряжены" — очередная причуда авторов). Так вот, кроме всех затрат на получение этих самых чар, существует еще и цена в единицах эфира. Эфир можно получить из источников эфира (поразительное, но вполне логичное и обоснованное умозаключение). А источники эфира в бою появляются спустя некоторое время (в отличие от The Gathering).

До сих пор от нас ничего не зависело, но спешу обрадовать — скорость поступления источников (как и количество вашего здоровья) зависит от вашего уровня. А этот самый уровень повышается за счет посещения некоторых объектов и побед в боях, что опять же способствует исследованию карты. Уровень также дает развитие вашим умениям (ну, здесь чистые "Герои"). В общем, хватит об уровне. Так вот, получили вы возможность наколдовать что-либо. А возможностей тут хватает: заклинания вызова, лечения, повреждения, усиления, защиты, черт знает чего — все это становится на вашу сторону и наступает на ваше горло. И так далее...

Четыре противоборствующие расы можно условно поделить на хороших и плохих. А можно условно и не делить.

Хорошие Кинеты не обладают большим выбором сильных существ — Авиаков и Ламий за таковых можно не считать. Зато у них самые интересные и оригинальные способы достижения победы.

Не менее хорошие Виталы положились на силы природы (насекомо-непонятные по большей части), а потому обнаружить в их стане что-нибудь пчело-клеще-древоорбо-подобное — не проблема.

Злобные Хаоты концентрируются на сражении в его чистой форме, то есть в защите и нападении.

А Синтеты привлекли (точнее, собрали) под свои знамена механических червей, механических зверей, механических кузнечиков и механического Тр-тр Митю.

Хотя, как всегда, на пути к власти нет чистого зла и нет чистого добра. В этот раз состязаются медиа-магнаты (проще говоря — за власть над Эфиром). Хотя в данной игре это и не столь важно.

Честно говоря, мне игра не понравилась, но настоятельно ее всем рекомендую. Просто игра не понравилась именно мне, а так, она, при всей своей вторичности, очень хороша.

P.S. В игре есть миссии для "пробы пера" в глобальном соперничестае и дуэль для чистых "поединщиков".

C&C Renegade

А.ВЕНДИЛОВСКИЙ,
г.Минск.

Ох уж эти старые времена... Помните старый добрый ДОСовский Command&Conquer, ставший очередной вехой в развитии RTS-ок? Сколько было времени на него потрачено, сколько бессонных ночей!.. Спрайтовые фигурки, на зеленом фоне маленькие прямоугольники, изображающие солдат. Ты помнишь те миссии за GDI, когда тебе давали лишь бравого коммандоса с чудо-снайперской винтовкой и полным мешком С4? Да, тяжелые были миссии. Помнишь ту, когда твоя база была практически в руинах, и ты последними силами бросился атаковать базу NOD? Твой небольшой отряд, едва вошел в каньон, как тут же оказался под перекрестным огнем — бестолково мечущиеся солдаты, пытающиеся отстреливаться, беспомощно застывшая техника, едва успевавшая сделать несколько выстрелов, прежде чем быть уничтоженной. У тебя оставалась только одна надежда — из-за края экрана медленно появился транспортный вертолет...

Из-за экрана появился транспортный вертолет, и в этот миг весь этот спрайтовый мир содрогнулся, словно заваливаясь назад. Лопасти вертолета стали рассекать появившийся воздух, и, словно навстречу стальной птице, вырастали горы, здания обретали объем, наполняясь мелкими деталями, будь то лесенка на крышу или витражное стекло в церкви. Этот мир, такой знакомый тебе, обретал новый ракурс, становился все больше и больше, заполняя все вокруг себя, пока ты сам не стал частью этого мира, еще одним его участником.

Вертолет на мгновение завис над полем боя. И еще через мгновение по сброшенной вниз веревке заскользил боец. Шрам, пересекавший его лицо, не делал его уродливым, но каждый, кто заглядывал ему в глаза, чувствовал ту неукротимую энергию, которая готова была вырваться и смести все на своем пути.

Опытным глазом он быстро оценил обстановку. Мгновенно в его руках оказалась снайперская винтовка. Сквозь оптический прицел был очень четко виден нодовский гранатометчик, укрытый тяжелой броней, спрятавший свое лицо под темным, ничего не отражающим шлемом. Пальцы, как и тысячи раз до этого, мягко нажали на спусковой крючок, и на этом темном шлеме появилось большое, но очень аккуратное отверстие — гранатометчик нелепо взмахнул руками и упал. В следующее мгновение в перекрестье прицела оказался еще один вражеский солдат, и все повторилось заново. Шесть резких выстрелов, и левая часть каньона была очищена. Отбросив уже ненужную винтовку, он схватил автомат и короткими перебежками от укрытия к укрытию стал подбираться к самому сердцу боя. К нему со всех ног бросился командир отряда.

— Я лейтенант Ка.

— Заткнитесь, лейтенант.

Коммандос явно был не склонен разводить на поле боя китайские церемонии, и при одном виде отутюженной формы молодого лейтенанта его просто тошнило. "Кадет, так его раз так!" — раздраженно подумал он. Рядом оглушительно разорвался вражеский снаряд, и их всех порядком засыпало землей.

— Ты в полном дерьме, лейтенант, и будешь делать то, что я тебе прикажу! — он ткнул пальцем в слегка подбитый "Мамонт", — эта зверушка должна быть через десять минут на ходу, она мне очень потребуется. И никаких "но"! — оглушительно рявкнул он, — пусть твои инженеры покажут, что не зазря здесь паек жрут!

Капитан Хавок, элитарный боец-коммандос вооруженных сил GDI, с профессиональной непринужденностью короткой очередью срезал группу огнеметчиков, которые явно надеялись зайти с тыла его людям. Но плотность огня еще была очень велика. Схватив пробегающего сержанта за рукав, он прокричал ему в ухо: "Если твои люди сейчас не задавят огневые точки, то нам всем каюк! Поднимай людей!"

— Вперед засранцы! Или вы хотите жить вечно? — заорал сержант, заставляя людей бросаться вперед, подталкивая их не только словом, но иногда и внушительным пинком, доказывая, что сержанты во всех армиях одинаковы. Шевелите ногами, каракатицы!

В тот же миг вскочил и Хавок — его автомат словно зажил собственной жизнью, посылая смертоносные очереди одну за другой. Не ожидавшие такого поворота событий нодовские силы в растерянности отступили. Этим замешательством и воспользовался капитан, юркнувший в уже отремонтированный танк "Мамонт" .

— Станцуем рок-н-ролл, девочки? — усмехнулся он, берясь за рычаги управления. Многотонная махина зарычала и, набирая скорость, понеслась по дороге, не забывая стрелять из обоих орудий. Вырвавшись из плена ущелья, танк превратился в стальную Немезиду, с легкостью расшвыривавшую легкие укрепления противника, а пулеметная точка вместе с расчетом была просто вдавлена в землю. От тяжелого удара ворота, ведущие на базу противника, были сорваны и отброшены. Танк остановился перед бараками NOD. Здание, выполненное в виде руки, сжимавшей земной шар, должно было наводить на мысль о могуществе братства НОД и их предводителя Кейна. Вслед за танком на базу ворвалась пехота, проводя полномасштабную зачистку местности. Ворота барака только стали открываться, когда танк глухо рявкнул из всех своих орудий, превращая как сами ворота, так и тех, кто стоял за ними, в клочья.

Еще не улеглась пыль от взрыва, когда Хавок нырнул в пролом, посылая снаряд за снарядом туда, где по его мнению должны были находиться автоматические турели. Не дожидаясь подмоги, он стал прорываться к центральному пульту управления. В главном зале его поджидал элитарный взвод штурмовиков НОДа, но удача была сегодня не на их стороне. Лазерная винтовка с легкостью пробила не только огнеметчика, но и баллоны с горючей смесью, висевшие у него за спиной. Вырвавшееся пламя пожирало все на своем пути, выжигая плоть и испаряя кровь, крики ужаса и боли заглушил взрыв баков с тибериумом. И те, кто пережил огненный шторм, пожалели, что остались в живых, когда ядовитый воздух тибериумных испарений вошел в их легкие.

Словно танцуя со смертью, Хавок ворвался в аппаратную, сея разрушение. Легким движением руки брусок С4 был прикреплен к пульту главного компьютера.

— Все не можешь успокоиться, Хавок? — голос за спиной был полной неожиданностью для него. Перекатившись, он рефлекторно направил автомат — посреди комнаты возвышалась голограммная фигура Кейна.

— Неужели ты веришь в эти сказки, что GDI приведут мир к процветанию? Очнись, друг мой! Только наше братство поможет человечеству подняться на новую ступеньку! А человек с такими способностями, как у тебя, может сделать у нас очень хорошую карьеру — тут Кейн хитро улыбнулся — некоторые из твоих бывших людей уже оценили мое предложение.

Ни говоря ни слова, Хавок нажал на кнопку детонатора. Здание содрогнулось, Кейн исчез, и повсюду загорелось аварийное освещение. Под рев сирен он выбрался на крышу. Открывшаяся панорама захватывала дух —

впереди огромной глыбой стоял Храм, окруженный лазерными обелисками. Атака на эту последнюю линию обороны захлебнулась. Хавок спокойно навел на него гранатомет и послал в его сторону небольшой серебристый снаряд. В полете у снаряда "выросли" усики антенн, и на дворик перед храмом приземлился сигнальный бакен. Не спеша, Хавок активировал программу на своем персональном компьютере, который вечно таскал на запястье, и представил, как где-то очень высоко, в холодном и пустынном космосе, получив его сигнал, спутник Ion Cannon сейчас наводится на сигнал бакена, как яркий поток энергии срывается к Земле... Небо над Храмом потемнело, и каскад молний, шипя и извиваясь, ударил с небес в Храм, раскалывая здание до самого его основания. Коммандос улыбнулся. В этот момент заработала связь со Штабом...

Итак, перед нами шуттер от первого лица во вселенной C&C. Сразу оговорюсь, за основу взята первая часть этой стратегии. Поэтому ветераны будут в восторге от новой встречи с главными персонажами той игры. Снова доктор Мебиус, Хавок, Кейн... Вообще, узнавание окружающего мира — это один из основных козырей этой игры. Точность воспроизведения моделей зданий и техники просто поражает. Бывало, выйдешь из танка, и давай рассматривать Обелиск Света (конечно, предварительно расстреляв его из главного калибра) — так вот ты какой, большой и страшный... Честное слово, я очень хотел придраться к моделькам, но не смог, все действительно совпадает с тем, что было в оригинальном C&C. Качество прорисовки моделек тоже достойно похвалы — очень качественно сделано.

Если говорить о качестве графики, то сразу перейдем к движку игры. Сделан он на хорошую, крепкую четверку — конечно, в нем отсутствуют динамические источники света, открытая местность не может порадовать большим разнообразием деталей, но вот внутренности зданий проработаны по полной программе. Начиная от фресок в церкви и заканчивая обустройством самой маленькой казармы, все очень грамотно и настолько обширно, что на первых порах легко заблудиться. Но что мне действительно понравилось, так это мелочи, так оживляющие картину, будь то разбитый самолет второй мировой войны, водопад или очень красиво проработанные осветительные мачты. Ну а ночной бой — это просто незабываемое зрелище (особенно в городе).

Для каждого любителя хорошей стрелялки много значит и вооружение бойца. И на этот раз разработчики не подкачали — арсенал достоин любого Рембо, оружие — на любой вкус. В первый момент, как это часто бывает, я выбрал наиболее эффективное орудие убийства и стал проходить уровни только с его помощью, привычно считая весь остальной арсенал ненужной экзотикой. Но вскоре мне пришлось кардинально изменить свое мнение, когда я столкнулся с таким количеством врагов, что даже все стрейфы и прыжки не могли мне уже помочь. Я прыгал как заяц из угла в угол, но все было бесполезно. Вот тогда-то мне и открылась суровая правда — с одним автоматом игру не пройдешь, и нужно очень четко чувствовать момент, чтобы вовремя сменить оружие.

Так часто задаваемый вопрос об AI на этот раз особо интересен. Когда я в первый раз запустил игру, то интеллект противника меня весьма не порадовал. Враги брали числом, но никак не умом. Уже приготовившись поставить клеймо "спинномозговой шуттер", я совершенно случайно заглянул на сайт Westwood, и поверьте мне, то, что я там прочитал, вызвало у меня гомерический хохот. Если коротко изложить то послание, оно сводится к одному — в спешке перед выпуском игры разработчики ЗАБЫЛИ установить скрипты, отвечающие за AI, так что качайте патч для установки мозгов! После установки патча сложность игры резко изменилась, миссии стали очень тяжелыми, и обычно без предварительного планирования просто проваливались. Говорить о том, что наскоком и нахрапом уровни больше не проходились, наверно, не стоит.

Но что действительно создает качественый геймплей, как это хорошая атмосфера боя в игре. Действие происходит и помимо вас — танки атакуют базу, их бомбит авиация, корабли сражаются с береговой артиллерией, а десантные группы так и норовят спуститься на парашютах у вас за спиной. Бой живет своей жизнью, и вы лишь один из его участников. Это придает очень мощный динамизм игре, подстегиваемый боевой и не надоедающей музыкой, с аккомпанементом из оружейной стрельбы и воплей противника, время от времени сменяющихся предсмертными стонами. Большего и желать не стоит. Эта игра захватит вас на очень много часов, пока последний враг не будет уничтожен, пока последняя турель не будет разбита, пока... пока игра не будет пройдена до конца.

Мультиплеер заслуживает отдельного разговора. Наверное, не только меня временами посещали мысли — вот бы сделали стратегию, в которой за каждого отдельного юнита играл бы один человек, и все это с видом "из глаз"! Можно сказать, что первая попытка уже состоялась. Поиграв на начальном, обучающем уровне, я оценил всю масштабность данного проекта. Играть в него втроем или даже впятером было несколько скучно. Нужно было, по меньшей мере, человек десять за каждую команду. После продолжительных бесед за бутылочкой пива с админами соседних офисов и демонстрации им возможностей игры, был разработан коварный план по сращиванию локальных сетей четырех фирм. В нерабочее время, вдали от начальства, из загашников доставались провода, хабы, долбились стенки, настраивались протоколы — короче, велась активная подрывная деятельность. Запуск игры парализовал деятельность этих фирм на трое суток. От начальственного разгрома нас спасло лишь то, что само начальство самозабвенно не отрывалось от игрушки. Первое впечатление — игра командная и только командная, одиночка ничего не может там сделать. Приходится координировать силы: кто будет охранять харвестер, который приносит доход в казну базы; кто будет инженером, и станет ремонтировать укрепления базы; кто будет охранять подступы к самой базе. Про денежки я не оговорился, после каждого рейса харвестер приносит на счет каждого игрока энную суму денег, на которые можно купить амуницию, оружие, танки... В общем, планирование атаки на базу противника (в которой тоже ворон не считают, да еще турелей натыкано как сорняков в огороде!) ведется не хуже, чем в Генштабе. Прорыв противника на базу может лишить вас стратегических заводов по производству техники или (упаси Боже!) тиберимного хранилища. Как вы сами понимаете, это придает игре определенный азарт. А как радуется душа, когда на базе противника вырастает небольшой, но очень аккуратненький ядерный грибок или Ion Cannon, и чувствуешь, что не зря так старался, прорываясь через заслоны врага. Это даже вызвало споры, не будет ли правильней считать мультиплеер командной RTS-стратегией от первого лица, чем обычным шуттером от первого лица.

Вывод — очень добротная игра, которая не должна оставить вас равнодушными.

КУПЛЮ, ПРОДАМ, ОБМЕНЯЮ

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Недорого **продам или обменяю** струйный принтер EPSON COLOR STILUS-600 с засохшей головкой.
Тел. в Воронеже (0732) 72-46-62. Виталий.
E-mail: ra3qq@mail.ru

Продаю недорого (можно по частям) коллекцию схем и мануалов по телефонии и видеотехнике, на бумаге и CD. Перечень вышлю.
426028, г.Ижевск, а/я 1502.
E-mail: danna100@chat.ru

Уважаемые читатели!

Подписаться на наши журналы (индексы: **48996** — "Радиомир", **48924** — "Радиомир. КВ и УКВ", **48925** — "Радиомир. Ваш компьютер") во II полугодии 2002 г. можно по каталогу Агентства "Роспечать", **стр. 235**, или по каталогу РО "Белпочта" "Газеты и журналы Республики Беларусь", **стр. 124** (раздел "Издания РФ, распространяемые по прямым договорам").

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

Год	Имеющиеся в наличии номера			Расценки на 1 экз. (с учетом пересылки)		
	Радиолюбитель	Радиолюбитель. Ваш компьютер	Радиолюбитель. КВ и УКВ	По России и СНГ (рос. руб.)	По Беларуси (бел. руб.)	По Украине (гривны)
1998	3, 5, 8	1 — 5, 8 — 12	1, 6, 7, 11, 12	25	800	6
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	1 — 3, 5, 6	30	1000	7
2000	2 — 5, 7 — 12	1 — 8, 10 — 12	4 — 7, 9 — 12	35	1200	8
2001	2 — 6	1 — 6	2 — 6	40	1400	8,5
	Радиомир	Радиомир. Ваш компьютер	Радиомир. КВ и УКВ	По России и СНГ (рос. руб)	По Беларуси (бел. руб.)	По Украине (гривны)
2001	7 — 12	7 — 12	7 — 12	40	1400	9
2002	1 — 12	1 — 12	1 — 12	45	1500	10

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

На бланке почтового перевода необходимо очень четко написать свой почтовый индекс, полный адрес, фамилию, имя и отчество полностью. В графе "Для письма" необходимо точно перечислить, какие конкретно номера какого из журналов Вы заказываете.

При оплате платежным поручением нужно предварительно выписать счет.

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

Приобретение отдельных номеров журналов

В РОССИИ:

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

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

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

В Москве в издательстве **"Солон-Р"**: ул.Садовая-Кудринская, д.11 (ст. метро Баррикадная, ст. метро Маяковская).
Время работы: с 09.00 до 18.00, кроме субботы, воскресенья.
Телефон: (095) 254-44-10.
Радиорынки: Царицынский (место 13/А), Митинский (ряд 8, контейнер 12, место Z25).
E-mail: Solon-R@coba.ru

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

На УКРАИНЕ:

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

В БЕЛАРУСИ:

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


C&C Renegade


НА РУССКОМ ЯЗЫКЕ
ФАРГУС
НА РУССКОМ ЯЗЫКЕ
EA SPORTS
NHL
2002
2 IN 1
РУССКАЯ И АНГЛИЙСКАЯ ВЕРСИИ
PC CD-ROM
ПИРАМИДА
ВИРТУАЛЬНЫЙ БИЛЬЯРД
1С
КОЛЛЕКЦИЯ ИГРУШЕК
ИЛ-2
ШТУРМОВИК
ОННАЯ КОПИЯ ИГРЫ ОТ "1С"
2в1
РУССКАЯ + АНГЛИЙСКАЯ ВЕРСИЯ
STAR TREK
BRIDGE COMMANDER
• ПЯТНАШКИ • 2D АРКАДА • МОРСКОЙ БОЙ • ДОМИНО
270
НАСТОЛЬНЫХ ИГР
на русском языке
НАСТОЛЬНЫЙ СПОРТ • О, СЧАСЛИВЧИК! • НАРДЫ
КРЕСТИКИ-НОЛИКИ • ШАШКИ • ША
АНЕКДОТЫ •
РУССКАЯ И АНГЛИЙСКАЯ ВЕРСИИ
Harry Potter
and the Sorcerer's Stone
Гарри Поттер
и Философский камень
PC CD-ROM
НА РУССКОМ ЯЗЫКЕ
ФАРГУС
2 IN 1
РУССКАЯ И АНГЛИЙСКАЯ ВЕРСИИ
6в1
РЫБКА ФРЕДДИ
(6в1)
ПОЛНОСТЬЮ НА РУССКОМ ЯЗЫКЕ
Русский Проект
Лучшие Гонки
ElectonicArts
4 в 1
RUS
NFS-5
Formula1-2000
SuperBike-2000
GrandPrix World
PC
НА РУССКОМ ЯЗЫКЕ
Русский Проект
DARK PLANET
BATTLE FOR NATROLIS
RUS